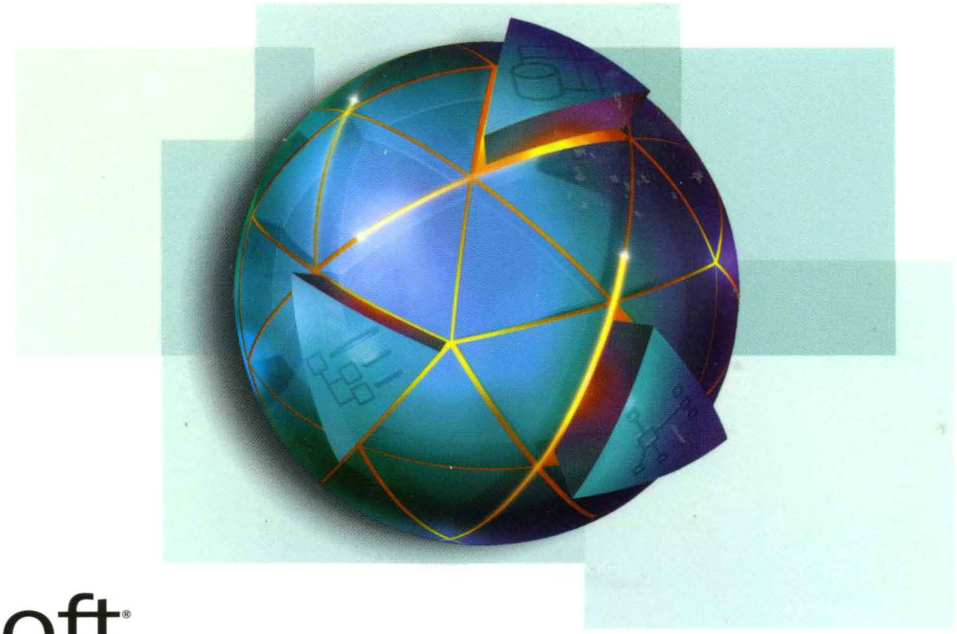
Designed for Microsoft® Windows NT® Windows®98

The Essential Guide to
Microsoft Visual InterDev 6.0



Microsoft® Visual InterDev™ 6.0 Programmer's Guide

Microsoft® Press

Microsoft® Visual InterDev™ 6.0 Programmer's Guide

Microsoft® Press

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Visual Studio Core Reference Set / Microsoft Corporation.
p. cm.
Includes index.
ISBN 1-57231-884-8
1. Microsoft Visual BASIC. 2. BASIC (Computer program language)
I. Microsoft Corporation.
QA76.73.B3M5598 1998
005.26'8--dc21 98-6655
CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 WCWC 3 2 1 0 9 8

Distributed in Canada by ITP Nelson, a division of Thomson Canada Limited.

A CIP catalogue record for this book is available from the British Library.

Acquisitions Editor: Eric Stroo
Project Editor: Anne Taussig

Part No. 097-0001982

Contents

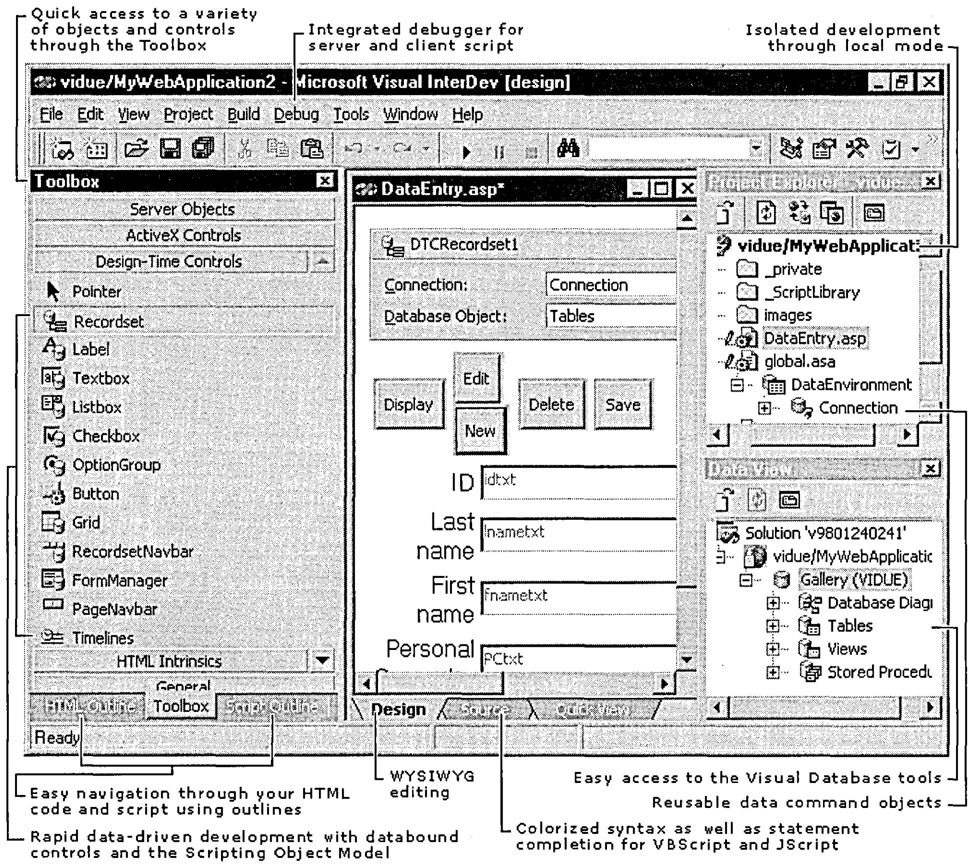
INTRODUCTION

Introducing Visual InterDev

Microsoft Visual InterDev 6.0 is a Web development tool designed for programmers who want to create:

- **Data-driven Web applications** using a data source supported by ODBC or OLE DB, such as the database management systems from Microsoft.
- **Broad-reach Web pages using HTML and script** in Web applications that take advantage of the latest advances in browser technology, such as Microsoft Internet Explorer 4.0, Dynamic HTML and multimedia features.
- **Robust development environment** with a Scripting Object Model, design-time controls (DTCs), and an extensible toolbox for rapid design, testing, and debugging of your pages.
- **Web teams that can develop pages in isolation** and maintain ready access to a master version, or teams that include nonprogrammers who work on the master version through Microsoft FrontPage.
- **Integrated solutions** that can include applets or components created in Microsoft Visual Basic, Visual C++, Visual J++, and Visual FoxPro.

The following figure provides a summary of features and tools to try inVisual InterDev.



This figure shows an .asp file open in the Design view of the HTML editor. The toolbox, Project Explorer, and Data View window have been resized so you can see their contents easily. You can customize your work area by closing, resizing, or rearranging any of the toolbars, toolboxes, or windows.

Prototype with Site Diagrams, Themes, and Layouts

Why type everything into a text file when you can use tools that let you concentrate on your content and functionality? Leave the details of file management, link repair, and navigation to the tools provided in Visual InterDev 6.0.

Visual InterDev 6.0 includes site design tools that help you easily plan pages, organize their links, and apply a consistent theme to your Web site.

Site diagrams You can use site diagrams to plan the overall structure of the Web site, to specify navigation between pages, and to take advantage of general visual design elements quickly and easily.

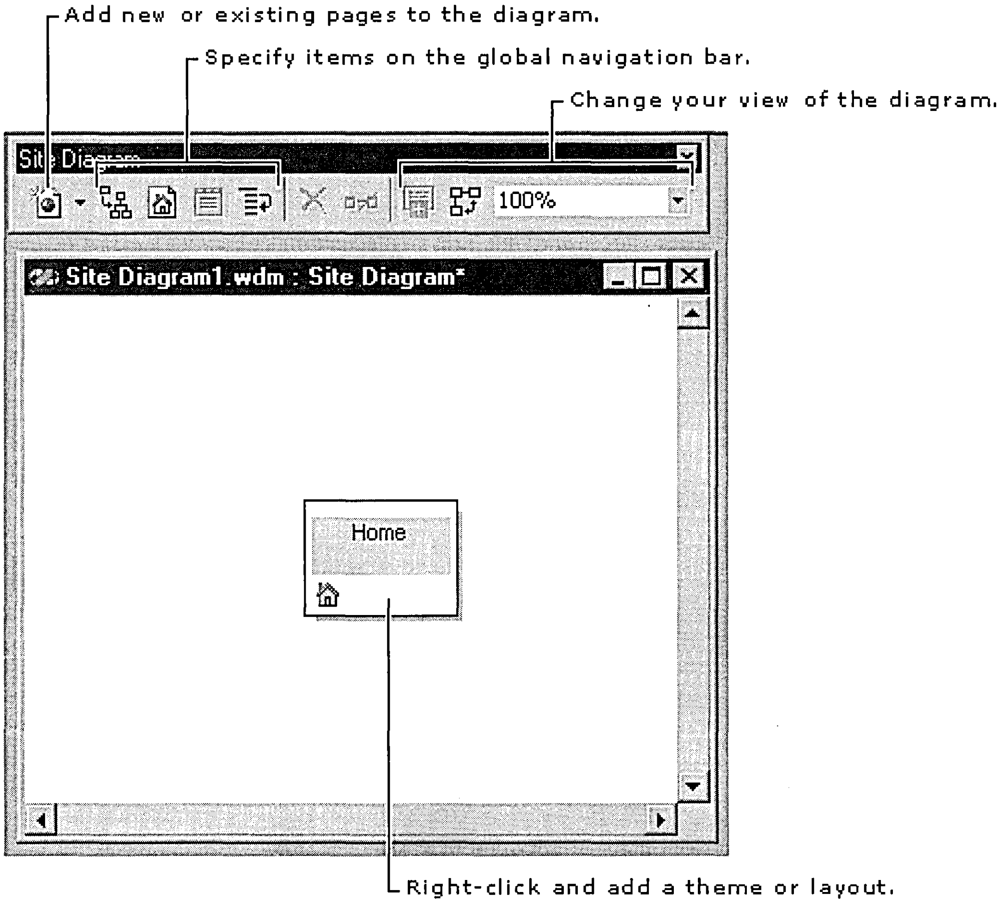
In a site diagram, you can create a prototype site containing multiple files and, at the same time, identify the hierarchical relationships between the files. It is this hierarchy that is used to define the site navigation structure. For example, your home page is considered a parent file. You can "attach" other pages below it to create children files.

Layouts Once you have established your navigation structure, you can add navigation bars to your Web pages. Using a layout, you can quickly define navigation bars that include combinations of parent, children and sibling files. For example, the home page can link to several children that can link to siblings and so on.

Themes Easily add a consistent visual impact to your Web pages through themes.

The themes and layouts are extensible and customizable so you can create different styles for all of the pages in your Web application or apply them to parts of your site.

When you use site diagrams, layouts, and themes to develop your Web site, the actual file structure and navigation bars are created automatically. To simplify maintenance once you've developed your Web application, site diagrams allow you to keep your navigation bars current when you update the site diagram.


Add new or existing pages to the diagram.
Specify items on the global navigation bar.
Change your view of the diagram.
Site Diagram
100%
Site Diagram1.wdm : Site Diagram*
Home
Right-click and add a theme or layout.

In the figure above, a new site diagram has just been opened. Any pages added to this diagram are also added to the project when you save the diagram.

Try It!

- Create a site diagram by right-clicking your project in the Project Explorer and choosing **Add** and then **Site Diagram**.
- Add some files to this diagram by using the first button on the Site Diagram toolbar. Save the diagram and notice that files appear in your project.
- Preview the new pages in your Web browser and test the navigation automatically supplied. If you see "[vinavbar]" on your page, you need to install the NavBar FrontPage extension. You can do this by running Setup and installing the server component.

- Change the relationships of files in the site diagram, save the diagram, and preview it. Notice that the navigation links have been automatically updated.
- Apply a theme and layout to your files by selecting files in the site diagram, choosing **Apply Theme and Layout** from the shortcut menu, and selecting the theme or layout you want.

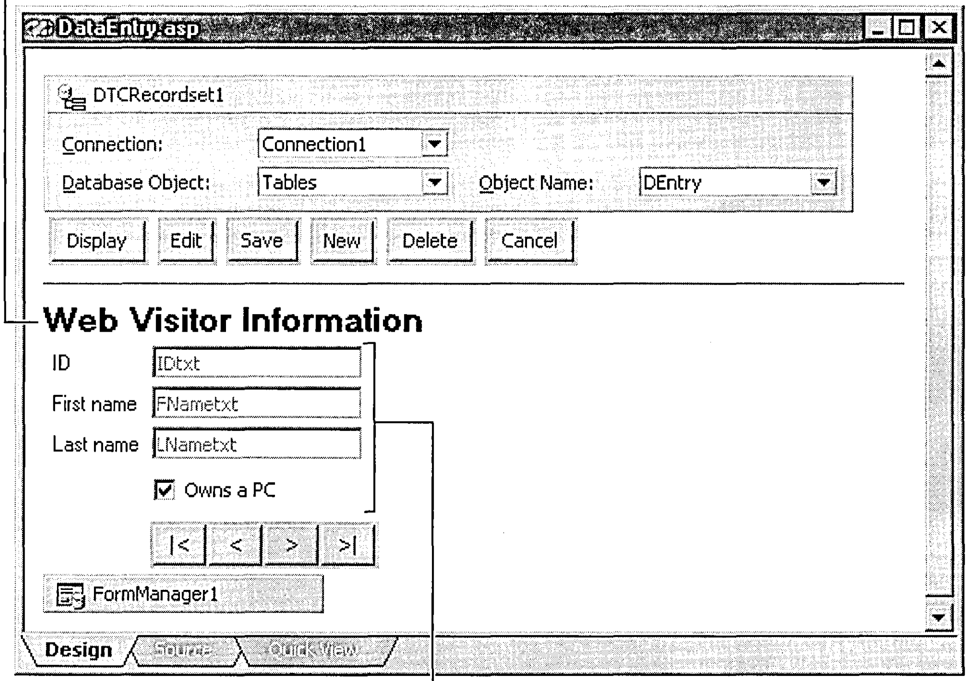
Develop in WYSIWYG View or Colorized Code

Visual InterDev 6.0 includes three ways to view your HTML and ASP pages.

These three views are the cornerstone of Visual InterDev 6.0. They replace the simple source code editor included with Visual InterDev 1.0 and support design-time controls (DTCs), debugging, statement completion, and object browsing.

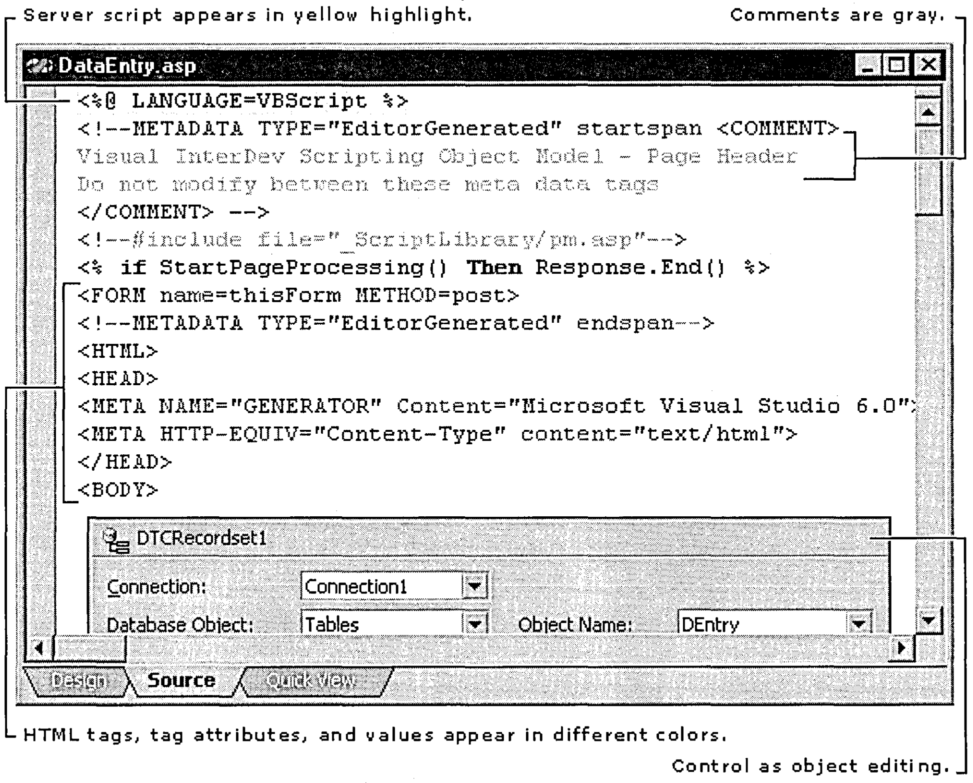
Design view Creates your page in WYSIWYG view. You enter content or drag items from the toolbox or data environment directly to your page. Use the toolbox, toolbars, and menus to build your page.

Format text using the Format Menu or toolbar.



Insert a table and then add controls to its cells to position the controls on the page.

Source view Shows the HTML or ASP source code. Like Design view, you can enter content or drag items from the toolbox or data environment directly to your page. Use the toolbox, toolbars, and menus to build your page.


Server script appears in yellow highlight.
Comments are gray.
DataEntry.asp
DTCRecordset1
Connection:
Connection1
Database Object:
Tables
Object Name:
DEntry
Design
Source
Quick View
HTML tags, tag attributes, and values appear in different colors.
Control as object editing.

```
<%@ LANGUAGE=VBScript %>
<!--METADATA TYPE="EditorGenerated" startspan <COMMENT>
Visual InterDev Scripting Object Model - Page Header
Do not modify between these meta data tags
</COMMENT> -->
<!--#include file="_ScriptLibrary/pm.asp"-->
<% if StartPageProcessing() Then Response.End() %>
<FORM name=thisForm METHOD=post>
<!--METADATA TYPE="EditorGenerated" endspan-->
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<META HTTP-EQUIV="Content-Type" content="text/html">
</HEAD>
<BODY>
```

Quick view Displays the results of HTML code and client script before the page is saved. If you want to view your page in a browser, you need to save the page. This view does not use a Web server so does not process server script.

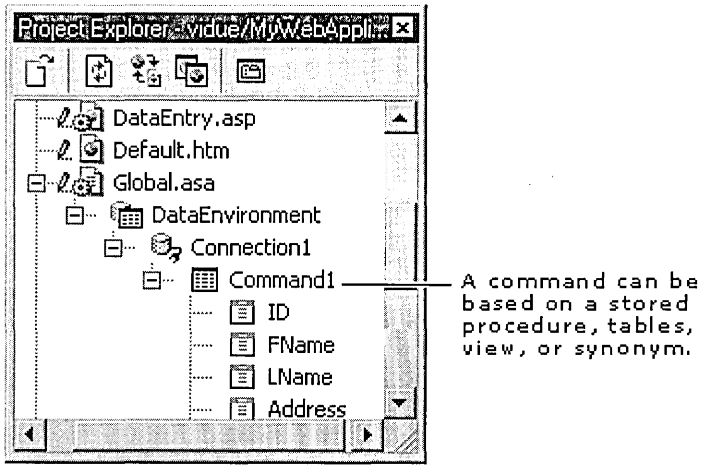
Try It!

- Experiment with the Design view by opening a file and adding some text. Select the text and use the HTML toolbar to change the font and size of your text.
- Insert an event by browsing a list of events in the HTML Outline window. Double-click an event and notice that the event prototype is inserted into the Source view of the document.
- Add a control and modify its properties. Select the control and choose Properties from the shortcut menu.
- Explore statement completion in the Source view. Add a script block to the file and then type “window” to display a list of properties and methods.

- When in Design or Source view, notice the properties on HTML elements in the Properties window. Notice that changing properties here can affect tag attributes without your having to edit the tags directly.
- Edit and create cascading style sheets in the dedicated CSS editor. Open any CSS file, such as those in the Themes directories of the project. If your project doesn't have a theme directory, you need to apply a theme.

Connect to Data and Create Reusable Data Commands

The new data environment provides easy commands for making your Web application data-driven. Instead of burying complex SQL statements deep within an .asp file, the statements are now exposed, maintained, and reused at the application level through the data environment under the Global.asa file. Instead of modifying the query within each page, you can modify the data command and your changes are incorporated into files that reference that data command. Also, you can drag fields from the command directly onto your HTML or ASP page.


Project Explorer - vidue/MyWebAppli...
DataEntry.asp
Default.htm
Global.asa
DataEnvironment
Connection1
Command1
ID
FName
LName
Address
A command can be based on a stored procedure, tables, view, or synonym.

Try It!

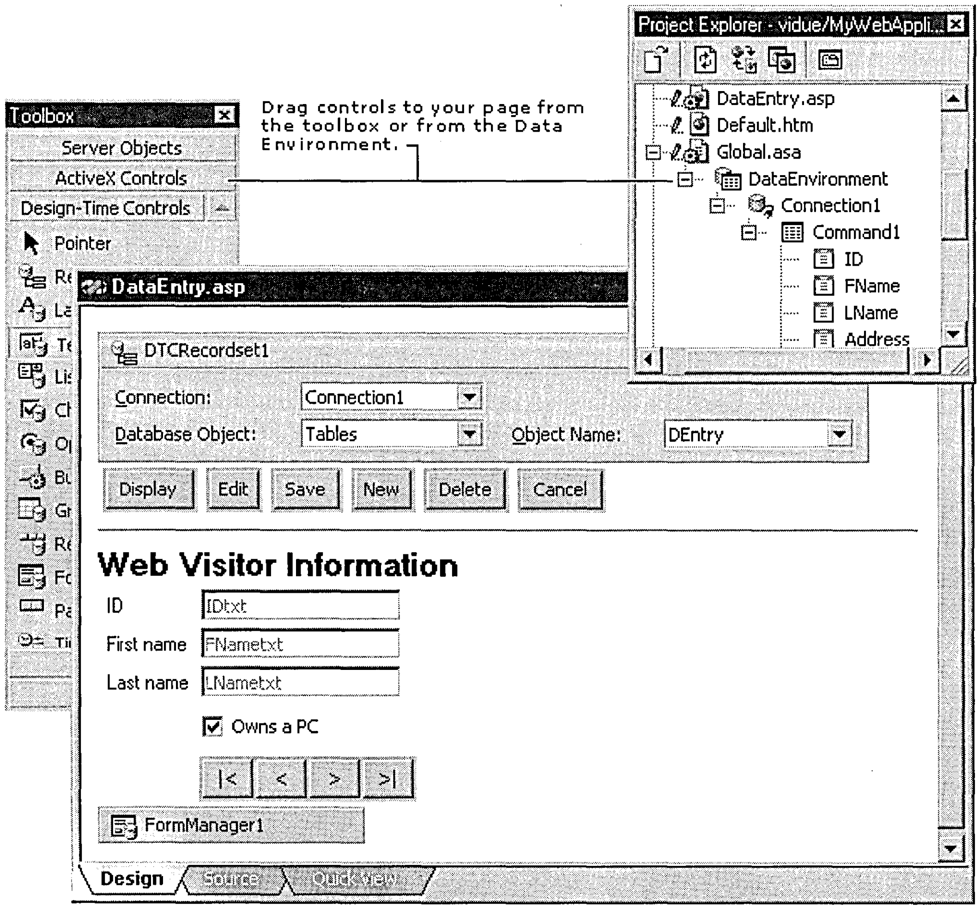
- Create a new data connection by right-clicking the project name in the Project Explorer and selecting Add Data Connection.
- After creating the data connection, notice that a Data Environment object appears under the Global.asa node. Under the Data Environment, you can find the connection.
- Using the connection, you can add a data command to create a reusable SQL statement in the data environment. Right-click on the project and select Add Data Command. Fill out the property pages to specify the data source for the data command.
- Try dragging fields directly out of the data environment onto the page. Notice that DTCs are inserted on the page for each field copied.

Display Data with Data-bound Controls

Creating an interactive Web page with data is as simple as dragging and dropping, setting some properties, and saving the page. No coding is required.

However, for those so inclined, Visual InterDev exposes a full object model that allows you to fine-tune your application, perform client validation, and have full control of your Web application. Visual InterDev 6.0 supports not only full-reach applications, using the ASP engine to produce simple HTML pages for the client, but also DHTML and Microsoft Internet Explorer 4.0 data binding for a richer client experience.

For example, this figure shows a simple data entry page that was created using data-bound controls.



Connection: Connection1
Database Object: Tables
Object Name: DEntry

ID: IDtxt
First name: FNametxt
Last name: LNametxt
[✓] Owns a PC

Try It!

- After creating a new data connection, drag a Recordset control from the Design-Time Control toolbox tab onto a page. Set the control's properties to bind this recordset to the data connection of your choice. You can also drag a data command onto a page to add a recordset bound to that data command.
- Drag a Textbox control onto the page. Open its properties and bind it to the Recordset control. You can also drag fields from the data environment directly.
- Drag a RecordsetNavbar control on to the page. Again, open its properties and bind it to the Recordset control.
- Make sure the PageObject control is still the first control on the page and publish the page. Navigate through the records at will.
- Switch the type of HTML used by the control. Open the Properties window for the recordset and go to the Implementation tab.
- Select either Generic HTML (ASP-based) or Internet Explorer 4.0 HTML (DHTML-based). Republish the page.

  Notice that for Internet Explorer 4.0, the page does not make a round trip to the server for each new record; instead, the record is replaced in line.
- Go to Source View and notice the object model exposed by each of the Design-Time Controls when the outline tab is displayed.

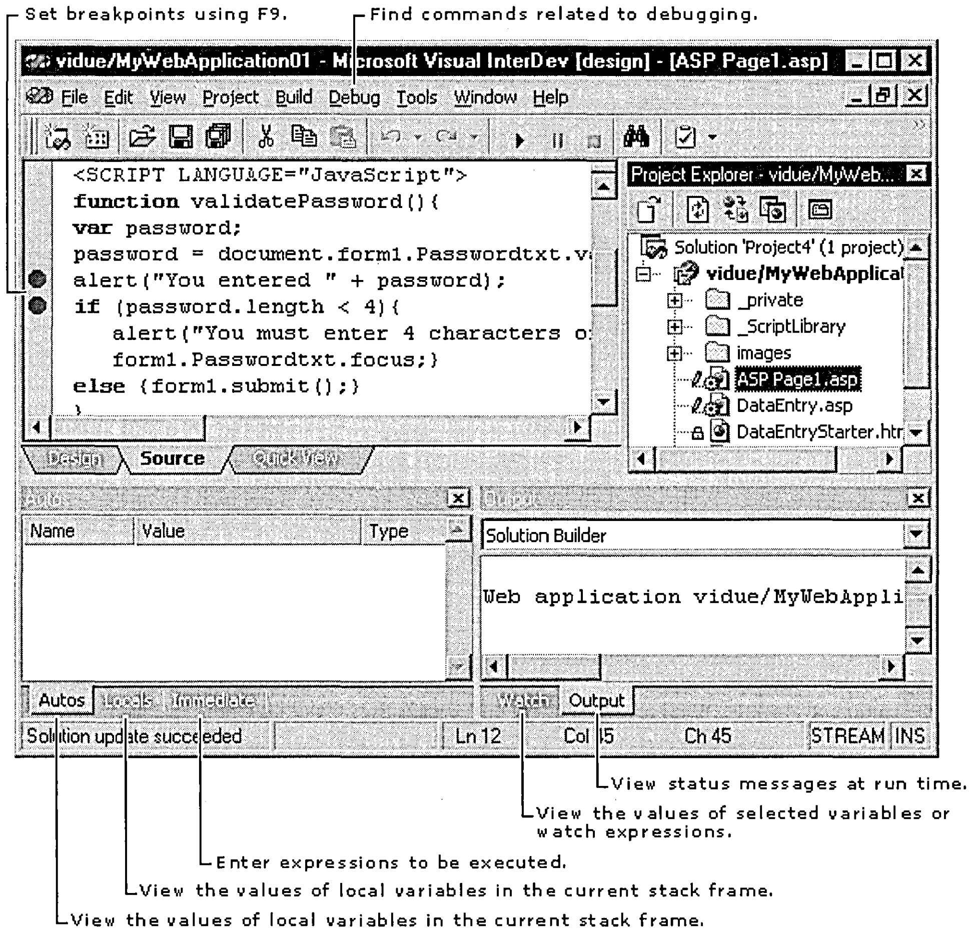
Debug Server and Client Script within Visual InterDev

To debug script, you can use Visual InterDev installed on the Web server or you can use Visual InterDev on a separate machine to debug script remotely.

> **Note** In this version, remote debugging is supported only with Microsoft Windows NT systems. Using a Microsoft Windows 95 client is not yet supported.

Visual InterDev 6.0 supports full client and server script debugging using everything you expect from a full-featured debugger.

This figure shows an ASP page open in the HTML editor and the debugger active.


Set breakpoints using F9.
Find commands related to debugging.
vidue/MyWebApplication01 - Microsoft Visual InterDev [design] - [ASP Page1.asp]
File Edit View Project Build Debug Tools Window Help
Project Explorer - vidue/MyWeb...
Solution 'Project4' (1 project)
vidue/MyWebApplicat
_private
_ScriptLibrary
images
ASP Page1.asp
DataEntry.asp
DataEntryStarter.htr
Design
Source
Quick View
Auto
Name
Value
Type
Output
Solution Builder
Autos
Locals
Immediate
Watch
Output
Solution update succeeded
Ln 12
Col 15
Ch 45
STREAM
INS
View status messages at run time.
View the values of selected variables or watch expressions.
Enter expressions to be executed.
View the values of local variables in the current stack frame.
View the values of local variables in the current stack frame.

```
<SCRIPT LANGUAGE="JavaScript">
function validatePassword(){
var password;
password = document.form1.Passwordtxt.v
alert("You entered " + password);
if (password.length < 4){
   alert("You must enter 4 characters o
   form1.Passwordtxt.focus;}
else {form1.submit();}
}
```

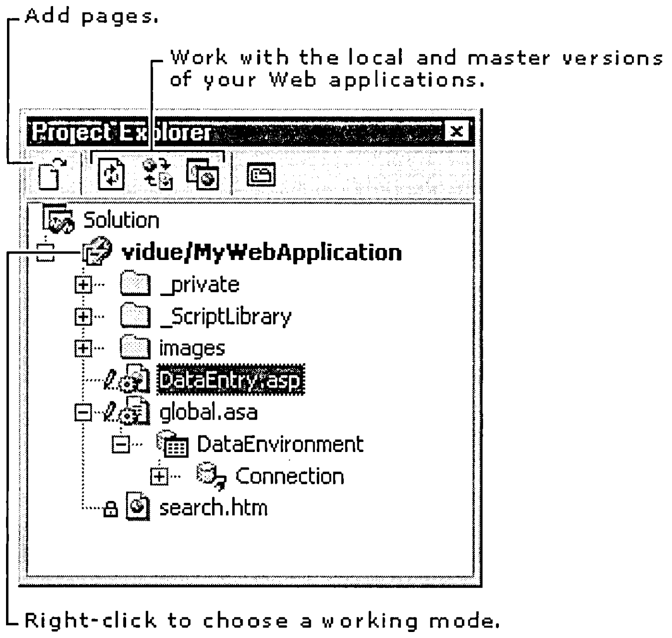
```
Web application vidue/MyWebAppli
```

Try It!

- Specify a start page for debugging. Right-click a file in the Project Explorer and choose **Set as Start Page.**
- View an .asp or .htm file in the browser. In Visual InterDev, choose Processes from the Project menu. After viewing the file, connect Visual Studio to the Internet Explorer and Microsoft Internet Information Server (IIS) processes.
- Debug the file just like you would debug any other form or function. View your running documents, open documents to debug, set breakpoints, and then preview the files again. Breakpoints on the client or server will occur and you can single-step through your script and check the process state.

Develop Web Applications on a Team

Visual InterDev 6.0 is specifically designed to meet the unique challenges of team-based Web development. Visual InterDev Web projects are connections to Web applications on a Web server. With Visual InterDev Web projects in local mode, you can take advantage of developer isolation to change and test application files locally before they are committed to the master Web server.



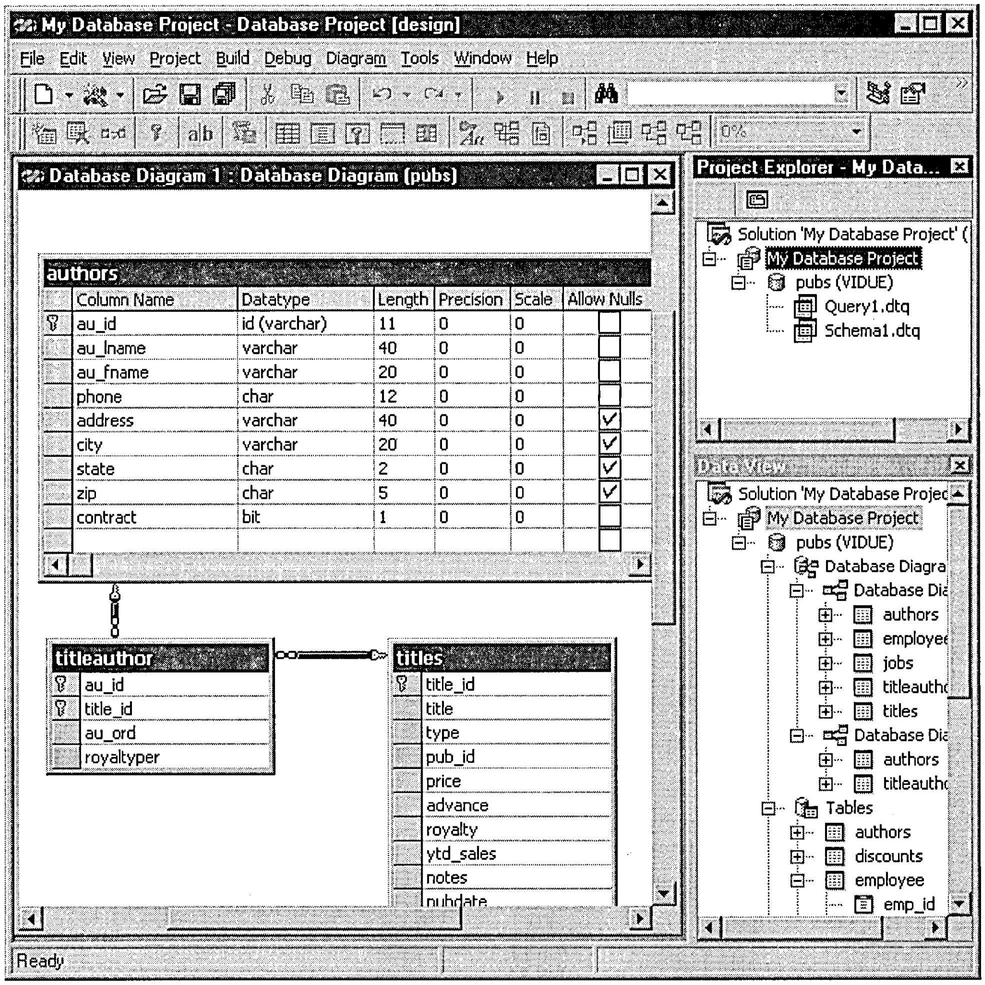
Try It!

- When creating a Web project, select **local mode**. You can change this later by choosing from the **Working Mode** options available from the **Project** menu.
- With a local copy of a file, save the file, and preview it in the browser.

  Notice that the file is being served by the local system, not the master Web server. If IIS is installed on the client system, .asp files can also be previewed.
- If you are working with team members on the application, try refreshing the project to view files they have been added to the master Web application by other developers. To refresh the project, use the Refresh button on the Project Explorer toolbar or right-click the project and select **Refresh**.
- Use Microsoft Visual SourceSafe to add version control to your Web application.

Creating and Modifying Database Objects

If you are using Microsoft SQL Server, you can also work on your database using the Microsoft Visual Database tools. After you create a connection in your project, you can work on database diagrams, database objects, and queries.


My Database Project - Database Project [design]
File Edit View Project Build Debug Diagram Tools Window Help
0%
Database Diagram 1 : Database Diagram (pubs)
Project Explorer - My Data...
Solution 'My Database Project' (
My Database Project
pubs (VIDUE)
Query1.dtq
Schema1.dtq
Data View
Solution 'My Database Projec
My Database Project
pubs (VIDUE)
Database Diagra
Database Dia
authors
employee
jobs
titleauth
titles
Database Dia
authors
titleauth
Tables
authors
discounts
employee
emp_id
Ready

authors

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
au_id	id (varchar)	11	0	0	☐
au_lname	varchar	40	0	0	☐
au_fname	varchar	20	0	0	☐
phone	char	12	0	0	☐
address	varchar	40	0	0	✓
city	varchar	20	0	0	✓
state	char	2	0	0	✓
zip	char	5	0	0	✓
contract	bit	1	0	0	☐

titleauthor

au_id
title_id
au_ord
royaltyper

titles

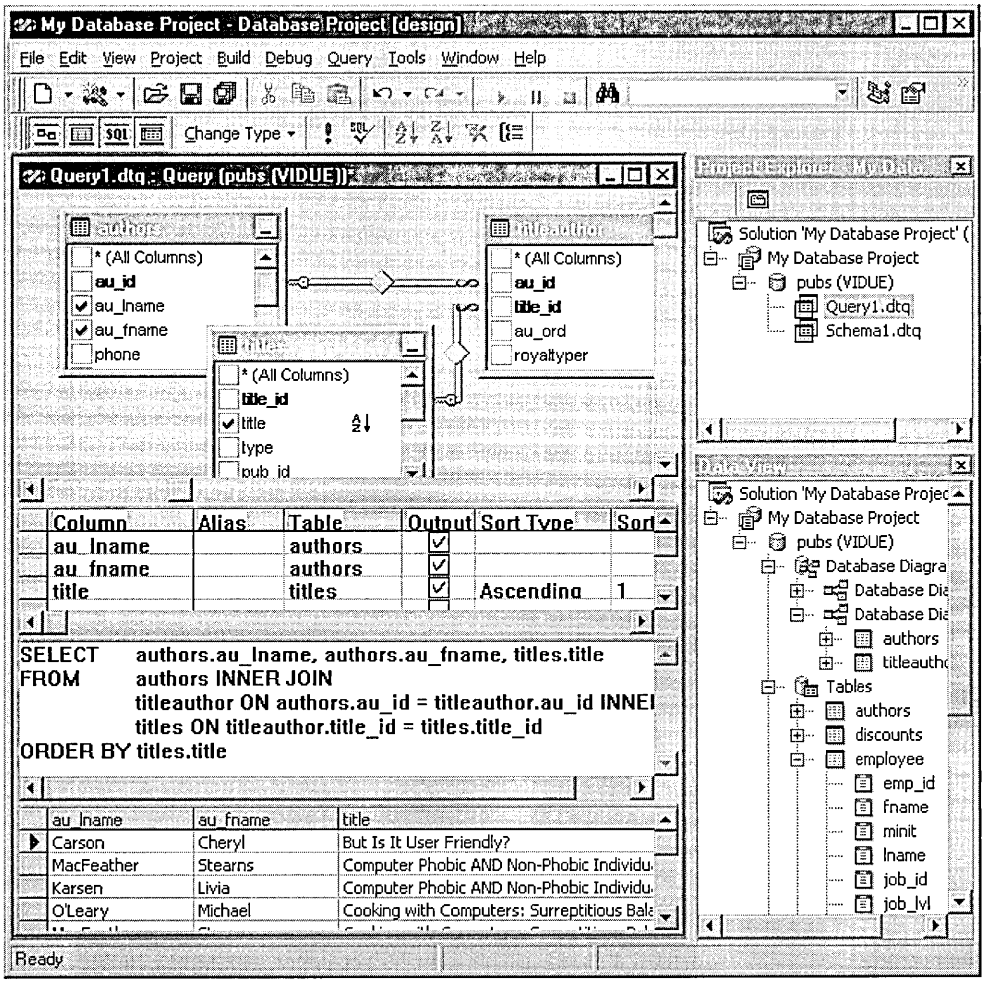
title_id
title
type
pub_id
price
advance
royalty
ytd_sales
notes
pubdate

Try It!

- Explore the database by expanding the list of database objects in the Data View window.
- Experiment with the design of your database without affecting the database until you choose to save your new design by creating a new database diagram.

- Create, modify, or delete database objects such as tables, views, and stored procedures using Data View.
- Open a table and add data or choose to design the table and complete complex DDL operations by changing a column's data type.
- View and save change scripts of the SQL code for the changes you made in a database diagram. You can submit change scripts to database administrators for review and execution in controlled database environments.

Using the Query Designer, you can choose from four different ways to construct and execute queries against any ODBC-compliant database. The figure below shows a query open in the Query Designer.



Column	Alias	Table	Output	Sort Type	Sort
au_lname		authors	✓		
au_fname		authors	✓		
title		titles	✓	Ascending	1

```
SELECT     authors.au_lname, authors.au_fname, titles.title
FROM       authors INNER JOIN
           titleauthor ON authors.au_id = titleauthor.au_id INNEI
           titles ON titleauthor.title_id = titles.title_id
ORDER BY titles.title
```

au_lname	au_fname	title
Carson	Cheryl	But Is It User Friendly?
MacFeather	Stearns	Computer Phobic AND Non-Phobic Individu
Karsen	Livia	Computer Phobic AND Non-Phobic Individu
O'Leary	Michael	Cooking with Computers: Surreptitious Bal

Try It!

- Create a new view using the Query Designer. In the Data View window, right-click the Views node and select New View.
- Drag tables from the Data View tables section to the query and use graphical controls to manipulate the query definition. Drag fields between the table to specify a relationship.
- Specify search criteria, sort order, and output columns in the criteria grid.
- Create a variety of query types: Select, Insert, Update, and Delete. Use an SQL pane to type ANSI-SQL statements — or let the Query Designer generate the SQL for you.
- Browse and edit live views of data in your database tables.

PART 1

Basics

The chapters in the Basics section provide you with the core background and "how to" information for Microsoft Visual InterDev. Each chapter lists the explicit procedural steps to accomplish common Web application tasks.

Chapter 1 Web Project Management

A Web project contains the files and information needed to create and publish a single Web application within Microsoft Visual Studio. This chapter takes you through the steps to create a Web project and add new or existing files to it.

Chapter 2 Web Basics

This chapter steps you through creating pages, modifying them, and previewing them, in the browser of your choice.

Chapter 3 Database Basics

Database Basics describes how to create a connection to a database, query the database, display data from the database on your Web pages, and create a form that allows your users to edit and update database records.

Chapter 4 Editing Basics

This chapter covers basic HTML editing, adding components to your Web pages, adding script to make your Web pages more dynamic, and choosing an alternate editor.

Chapter 5 Walkthroughs

The Walkthroughs take you through common Web application scenarios with instructions and illustrations. Scenarios covered in this chapter include "Creating a Home Page," "Debugging Script," "Working with Multiple Developers," "Deploying a Web Application," and "Simplifying Data Entry Pages."

CHAPTER 1

Web Project Management

In Microsoft Visual InterDev, you first create a Web project so that you can:

- Manage working files locally.
- Maintain master files on a server.

A Web project contains the files and information needed to create and publish a single Web application within Microsoft Visual Studio. The files within a Web application can consist of several different file formats that you modify during design time: HTML pages, Active Server Pages (ASP), image files, layouts, themes, and so on.

A Web project manages two copies of the Web application: local and master. All the master Web application files are stored on the master Web server. Before editing files, you retrieve files from the server so that working copies of the files are placed locally into your local Web application.

In a multiple-developer scenario, each member of the development team has his or her own project, which can refer to the same master Web application. For a walkthrough scenario, see "Working with Multiple Developers" in Chapter 5, "Walkthroughs." For more information about projects, servers, and Web applications, see "Project Architecture" in Chapter 6, "Web Project Concepts."

> **Note** You can create two types of projects with Visual InterDev — Web projects and database projects. For information on database projects, see Chapter 22, "Managing Database Projects."

Creating a Web Project

A Microsoft Visual InterDev Web project is stored locally on your machine. The files on your machine make up the local Web application. In Microsoft Visual Studio, Web projects and other project types are accessed from a solution.

To create a new Web project

- Add a new project to a solution if you have one already open in the Project Explorer.

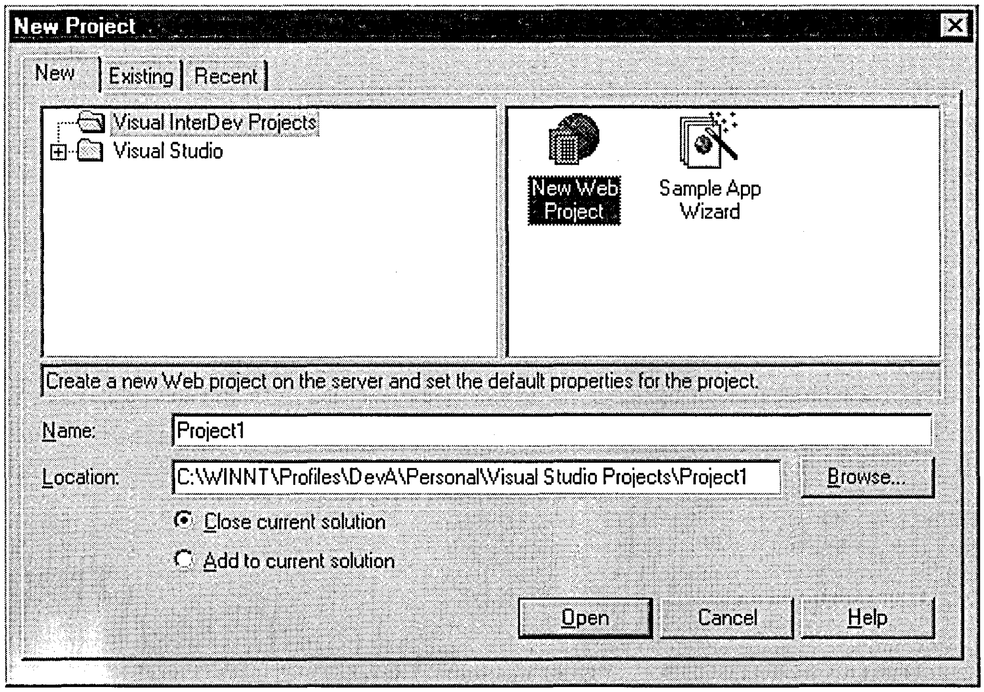
  – or –

  Create a new solution and a new project simultaneously.

If you are creating your first Web application, then it's easiest to create a solution and a project at the same time.

To create a solution and a Web project at the same time

1. From the **File** menu, select **New Project.**
2. From the **New** tab, select **Visual InterDev Projects** in the left pane and **New Web Project** in the right pane.



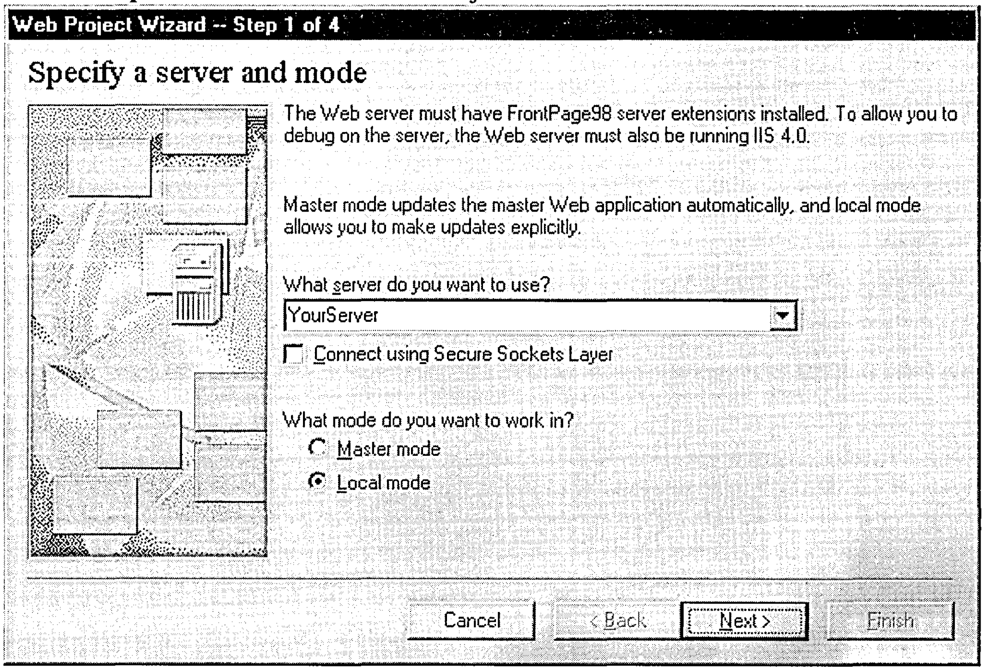
Create a new Web project on the server and set the default properties for the project.

Name: Project1

Location: C:\WINNT\Profiles\DevA\Personal\Visual Studio Projects\Project1

[✓ Close current solution] Add to current solution

3. In the **Name** text box, type a name for the new solution.
   If you already have a solution open, select the **Close current solution** option.
4. Click **Open**. This launches the Web Project Wizard.



Web Project Wizard -- Step 1 of 4

Specify a server and mode

The Web server must have FrontPage98 server extensions installed. To allow you to debug on the server, the Web server must also be running IIS 4.0.

Master mode updates the master Web application automatically, and local mode allows you to make updates explicitly.

What server do you want to use?
YourServer

☐ Connect using Secure Sockets Layer

What mode do you want to work in?
○ Master mode
● Local mode

Step 1: Specify a server and mode

Specify the name of your Web server, whether you want to connect using Secure Sockets Layer, and specify Master or Local mode.

Step 2: Specify your Web

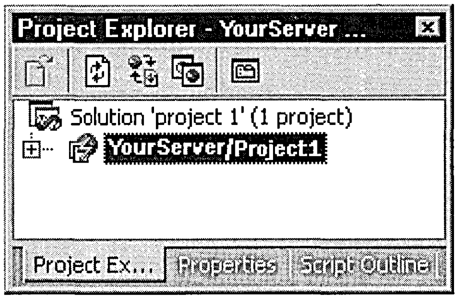
You can choose to have Visual InterDev create a new Web application on your Web server, or you can connect to an exisiting Web application.

Step 3: Apply a layout

It is not necessary to use a layout. If you want, you can select **None** now and apply a layout at a later time.

Step 4: Apply a theme

It is not necessary to use a theme. If you want, you can select **None** now and apply a layout at a later time.


Project Explorer - YourServer ...
Solution 'project 1' (1 project)
YourServer/Project1
Project Ex...
Properties
Script Outline

The new solution appears in the Project Explorer. If the Project Explorer is not visible, choose **Project Explorer** from the **View** menu. Expand the solution to see the new project and its files.

Adding Files

Once you have created a Web project, you can add any type of file to the project.

To insert a file into a project

1. In the Project Explorer, select the project or subfolder where you want to insert the file.
2. From the **Project** menu, choose **Add Item**.

   The **Add File** dialog box appears.
3. In the **New** tab, choose a file type in the right pane and provide a name in the **Name** box. For more information, see "Creating Pages" in Chapter 2, "Web Basics."

   – or –

   In the **Existing** tab, browse to the file or files to be inserted. Be sure to select the appropriate file type in the **Files of Type** box.
4. Click **Open** to add the file to the Web project.

   Note You can also add files or folders to a project by dragging them from Windows Explorer onto a folder in the Project Explorer.

Deleting a Web Project

You can delete a Web Project and its associated Web application or just remove the project from a solution without deleting its files.

A Web application has two copies of the Web files that are managed by the Web project: local and master. In addition to deleting the project file (.vip), you can delete one or both of these copies of the application. For more information about local and master files, see "Web File Processing" in Chapter 6, "Web Project Concepts."

To delete a Web project

1. In the Project Explorer, select the Web project you want to delete and from the **Edit** menu, choose **Delete Project**.
2. In the **Delete Project** dialog box, choose which copy of the files you want to delete.

To delete	Choose
Your local Web project (.vip) file and the local copy of the Web application	**Local Web project and all associated files**
Your local Web project (.vip) file and the master copy of the Web application	**Master Web project and all associated files**
Your local Web project (.vip) file and both the local and master copies of the Web application	**Both the master and local Web projects and all files**

3. Choose **OK**.

Note If you choose to delete only the local Web project and files, you can later create a project that points to the master copy of the Web application that remains on the server.

If you choose to delete only the master copies, you also delete the local project file (.vip), but the local directory and Web application files remain on your machine.

You might want to keep your project and its associated files but remove it from the solution.

To remove a project from a solution

1. In the Project Explorer, select the Web project you want to remove.
2. From the **File** menu, choose **Remove Project**.
3. In the message, choose **Yes**.

The project reference in the solution file is removed but the project file and the local and master Web files remain.

CHAPTER 2

Web Basics

Once you've started working in a Web project, you typically want to experiment with your HTML pages and Active Server Pages before updating the Web server with new files. Microsoft Visual InterDev makes it easy to create pages, modify them, and preview them, in the browser of your choice, before actually updating the server.

Creating Pages

Once you have a Web project, you can add HTML pages and active server pages (.asp files) to make your Web site functional.

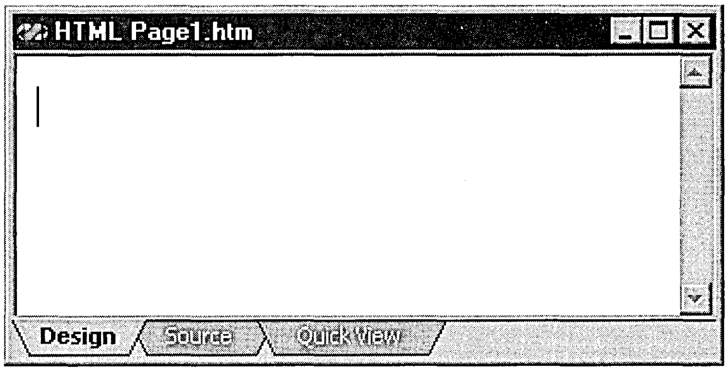
Note Client-side script is added to HTML pages. Server-side script is added to active server pages.

To create a new page

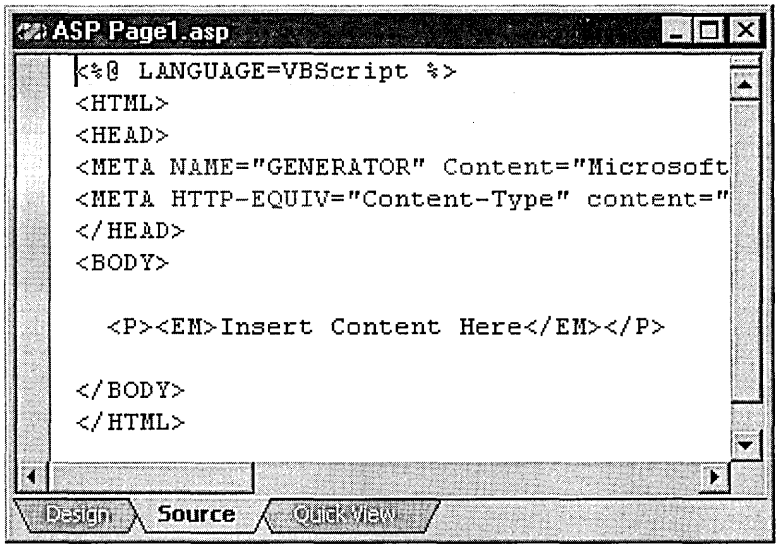
1. In the Project Explorer, select the project or subfolder where you want to add the new page.
2. From the **Project** menu, choose **Add Item**.
3. In the **Add Item** dialog box, click the **New** tab.
4. In the right pane, select **HTML Page** or **ASP Page**.
5. Type a new file name in the **Name** box.
6. Click **Open**.

Microsoft Visual InterDev opens the new page in the editor.

- The default view of the editor for HTML pages is Design View. As you edit your page in Design View, your text appears with all the formatting applied, the way you see documents in a word processor. For more information, see "Editing HTML" in Chapter 4, "Editing Basics."



- The default view for ASP pages is Source View. As you edit in Source View, your tags and script are color-coded, making them easier to read. For more information, see "Adding Scripts" in Chapter 4, "Editing Basics."



```
<%@ LANGUAGE=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft
<META HTTP-EQUIV="Content-Type" content="
</HEAD>
<BODY>

  <P><EM>Insert Content Here</EM></P>

</BODY>
</HTML>
```

Saving Pages

It is always a good idea to save your pages regularly to avoid losing your work.

To save the current page

1. Click the page to make sure it has the focus.
2. From the **File** menu, choose **Save**.

The file is saved to your local machine. Depending on your network connection, the file might also be updated in the master Web project:

Network	Local mode	Master mode
Online	The page is only saved locally.	The page is saved locally and updated on the master Web server.
Offline	The page is only saved locally.	The page is only saved locally.

For more information about the differences between local mode and master mode, see "Project Architecture" in Chapter 6, "Web Project Concepts."

After you save the page, you can preview the page in Quick View or any browser. For more information, see the next section, "Previewing Pages."

Previewing Pages

While editing HTML or ASP pages, you can easily check your progress by previewing the page.

To preview a page you are editing

- In the editor, click the **Quick View** tab at the bottom of the window.

  Note It is not necessary to save a file before viewing it in Quick View.

When you preview a page, you see how the page would appear in Microsoft Internet Explorer 4.0. However, because you are previewing the page locally, there are certain limitations:

- Server-side script is not processed.
- Data-bound design-time controls do not display data.

To get around these limitations, view the page directly in a browser, such as Microsoft Internet Explorer 4.0.

To preview a page in a browser

1. In the Project Explorer, select the file.
2. From the **View** menu, choose **View in Browser**.

If you are working locally, a local copy of the file is opened by the default browser. You can also preview the page in a different browser, or change the default browser.

To preview a page in a different browser

1. In the Project Explorer, select the file.
2. From the **View** menu, choose **Browse With.**
3. In the **Browse With** dialog box, select a browser.

   If you have a browser installed on your machine that is not listed in the dialog box, you can add it to the list by clicking **Add**.

   Note You can change the default browser in the **Browse With** dialog box by selecting a browser from the list and then choosing **Set as Default**.

CHAPTER 3

Database Basics

Microsoft Visual InterDev gives you the capability to connect to data in most databases. For example, you can connect to SQL Server, Microsoft Access, or Oracle databases. Once you've established a connection to a database, you can select a particular set of records from the database, and display this set of records on your Web page.

To make displaying and editing data on your Web page easy, Microsoft Visual InterDev supplies a wide variety of data-bound design-time controls:

- The Recordset control references a database and allows you to extract a set of records.
- The RecordsetNavbar control lets you move from record to record within the database from your Web page.
- Additional individual controls, such as labels, text boxes, and list boxes, display the data from the database.
- The Grid control makes it easy to display the data from your database on the Web page. You can display multiple records in a grid, sort the records, and control the formatting and layout of the data.

When you add a design-time control to an ASP or HTML page, the control automatically places script on the page, which displays the data and enables functionality such as editing the data or navigating through the records. You can also extend the functionality of the Web page (for example, add validation or custom navigation) by writing event handlers which can modify the state of the controls.

> **Note** You can also use the data environment to add data connections and create and manage data-bound controls in one location. This enables you to create powerful custom solutions directly, using the Microsoft Visual InterDev editor and debugger to modify the script that displays and manipulates the data on the Web page. For more information, see "The Data Environment" in Chapter 18, "Database Concepts."

Connecting to a Database

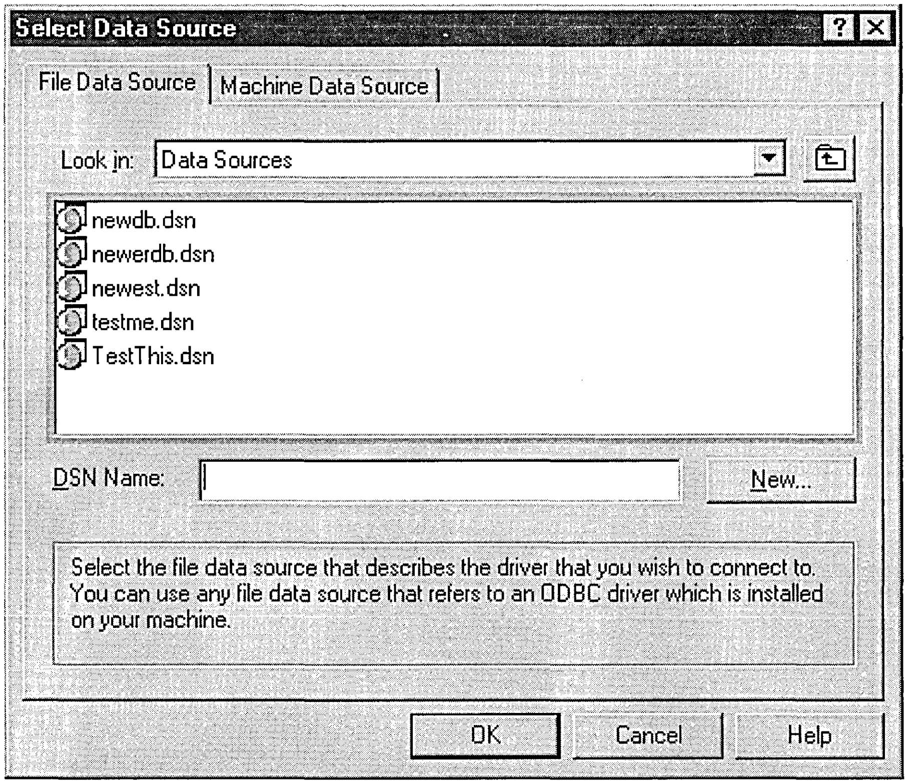
Before you can display or edit data on your Web page in Microsoft Visual InterDev, you must connect to a database. A data connection provides your Visual InterDev project with access to a particular database. You can then use this data connection to connect to the database and display its data on your Web page.

When you connect to a database, you first create a data source name (DSN) for the database or choose an existing one. Then, you use the DSN to create a data connection and add it to your project.

To create a data source name

1. Open your Visual InterDev project and select it in the Project Explorer.
2. From the **Project** menu, choose **Add Data Connection**.

   The **Select Data Source** dialog box appears.



Select the file data source that describes the driver that you wish to connect to. You can use any file data source that refers to an ODBC driver which is installed on your machine.

3. On the **File Data Source** tab, choose **New**.
4. Select the database driver you want and choose **Next**. The database driver must match the type of database you are connecting to. For example, if your database is a SQL Server database, select the SQL Server driver.
5. Type the name for the connection file and choose **Next**.

   The extension .dsn is automatically added to the file name.
6. Choose **Finish**.
7. In the dialog box or boxes that appear, fill in the driver-specific information, such as the name of the database to access or the file to open.

   > **Note** When you specify a database, use the relative path to the database, not its location on your development computer. For example, if the database is located on a Web server, use the UNC path to the database. This ensures the database will be available from your Web server.

   You are returned to the **Select Data Source** dialog box, and the file data source name you created is displayed in the list.

For more information on the types of data sources you can create, see "Choosing a Data Source Name" in the Microsoft Visual Database Tools online documentation.

> **Tip** You can also use the ODBC Data Source Administrator (located in the ODBC folder of your Control Panel) to create and name data sources. These data sources will appear in the **Select Data Source** dialog box, and you can use them to create data connections in Microsoft Visual InterDev.

Once you've created a data source name for a database, you can use it to establish a data connection to that database.

To add a data connection to your project

1. On the **File Data Source** tab of the **Select Data Source** dialog box, select the data source name you created and choose **OK**.

   A dialog box showing parameters for the data connection is displayed. The dialog box you see depends on the type of data source you have selected. For example, if you select a SQL Server data source, you'll see a **SQL Server Login** dialog box.
2. Enter the data connection parameters. For example, for a SQL Server database, enter the login ID and password.

   The data connection is displayed under the **DataEnvironment** folder in your project, underneath the **global.asa** folder. You can also browse and edit the data from this database in the **Data View** window.

You are now ready to retrieve and view the data you want from the database. For more information, see the next section, "Querying the Database."

Querying the Database

Once you've added a data connection to a database, you can query the database to specify a set of records that you want to use with a particular Web page. Microsoft Visual InterDev makes sets of database records available through the Recordset design-time control.

To add a Recordset control to an ASP or HTML page

1. Make sure that you have set options to view controls graphically. From the **View** menu, choose **View Controls Graphically**. To set this option as default, use the **HTML** node of the Options dialog box.
2. Open the ASP or HTML page in the editor. For information on creating ASP or HTML pages, see "Creating Pages," in Chapter 2, "Web Basics."
3. Drag the Recordset control from the **Design-Time Controls** tab of the **Toolbox** onto the page.

   Tip If the Recordset control is not shown in the **Toolbox**, right-click on the **Toolbox**, choose **Customize Toolbox**, and add the Recordset control.

You can now specify a set of records for the Recordset control.

To specify a set of records

1. In the Recordset control on the ASP or HTML page, set the **Connection** property to the name of the data connection for the database whose records you want to see.
2. Set the **Database Object** property to **Table** to display all the records in a table in the database. (You can also set this property to other sets of records in the database, such as a view or a stored procedure.)
3. Set the **Object Name** property to the name of the table or other database object whose records you want to use.

You can display data from this set of records by adding a data-bound control to your ASP or HTML page and setting the Recordset control as the control's data source. The control is then bound to the records in that table. For more information, see the next section, "Displaying Records."

Note You can also create data connections and add Recordset controls to your Visual InterDev project using the data environment. The data environment is especially valuable if you want the ability to programmatically manage your data and recordsets in one location. For more information, see Chapter 19, "Viewing Data," and "The Data Environment" in Chapter 18, "Database Concepts."

Visual InterDev allows you to take advantage of Microsoft Internet Explorer 4.0's client-side data binding, as well as the more traditional server-side data binding. For a discussion of client-side data binding, see "Data Binding" in Chapter 18, "Database Concepts."

Displaying Records

You can display data on your Web page by using data-bound design-time controls, such as labels, text boxes, and list boxes, and provide navigation among the records by using a navigation control.

To add a data-bound control to an ASP or HTML page

1. Make sure that you have set options to view controls graphically. From the **View** menu, choose **View Controls Graphically**. To set this option as default, use the **HTML** node of the Options dialog box.
2. Open an ASP or HTML page that contains a Recordset control in the editor.
3. Drag a data-bound control from the **Design-Time Controls** tab of the **Toolbox** onto the page. For example, you can drag a Textbox control onto the page to display the contents of a particular field.
4. Right-click the control and choose **Properties**.
5. Set the **Recordset** property of the control to the name of a Recordset control on the current page.
6. If the control has a **Field** property, set it to the name of the field from the recordset you want the control to display.
7. Set other control properties as desired.

   For more information on data-bound controls and setting their properties, see "Design-Time Controls" and "Dialog Boxes and Windows" in the Visual InterDev online documentation.
8. Close the properties window, save the .asp or .htm file, and preview the file in the browser. The text box will display the field you selected for the first record in the connected database.

   For information on previewing ASP or HTML pages in a browser, see "Previewing a Page in a Browser," in Chapter 14, "Managing a Site Diagram."

You can easily provide navigation among the records you display on your Web page by using the RecordsetNavbar control.

To provide navigation among records

1. Open the ASP or HTML page in the editor.
2. Drag the RecordsetNavbar control from the **Design-Time Controls** tab of the **Toolbox** onto the page.
3. Right-click the control and choose **Properties**.

4. Set the **Recordset** property of the control to the name of a Recordset control on the ASP or HTML page.
5. If you want, close the properties window, save the .asp or .htm file, and preview the file in the browser. You can use the RecordsetNavbar control to move among the records in the underlying recordset.

By default, the RecordsetNavbar control provides **Move to First**, **Next**, **Previous**, and **Move to Last** buttons. You can use these buttons to display the different records in the underlying recordset on the page.

You can also customize navigation behavior by writing event handlers and using the object model exposed by this control.

You can also use the Grid design-time control to display multiple records from a database on your page. For more information, see the "Data-bound Grid Sample" and "Grid Properties Dialog Box" in the Visual InterDev online documentation.

For more information about displaying your data on Web pages, see Chapter 19, "Viewing Data"; Chapter 20, "Modifying Data"; and the next section, "Creating Event-Driven Forms."

Creating Event-Driven Forms

You can easily create a form with different modes on your Web page using the FormManager control. For example, you can create a data-entry form with Insert, Update, and Delete modes. When specified control events, such as clicking a button, occur, the form moves from one mode to another.

You can also specify property settings and methods to run both when a particular form mode is established, and during the transitions from mode to mode. All of this is accomplished without scripting, using the FormManager control's property pages.

For detailed examples of how to create a data-entry form using the FormManager control, see "Simplifying Data Entry Pages" in Chapter 5, "Walkthroughs."

The first steps in creating a Web page on which you'll use the FormManager control are to add a Recordset control and any other design-time controls you'll want to display on the page in the various form modes. For example, you may want text boxes to display data, and buttons to use to switch between form modes. For information on how to add data-bound controls to ASP or HTML pages, see "Querying the Database," and "Displaying Records," earlier in this chapter and Chapter 19, "Viewing Data."

When you have the controls you want to use in your form modes on the page, you can add a FormManager control, and then define its modes and transitions.

Note Although it's common, you don't have to display data on forms created using the FormManager control. If you aren't displaying data on your forms, you don't need to use a Recordset control.

To add a FormManager control to a page

1. Make sure that you have set options to view controls graphically. From the **View** menu, choose **View Controls Graphically**. To set this option as default, use the **HTML** node of the Options dialog box.
2. Open the ASP or HTML page in the editor.
3. Drag the FormManager control from the **Design-Time Controls** tab of the **Toolbox** onto the page.

   Note If you're using DHTML on your page, be sure to place the FormManager control on the page below any of the controls you want to use with the form. This ensures that the FormManager will have access to all objects on the page.

To add a form mode

1. Right-click the FormManager control on the page and select the **Properties** command from the shortcut menu.
2. On the **Form Mode** tab of the **Property Pages**, enter a name for the FormManager control if you want. This name is used to identify the control in the script that it generates. If you don't supply a name, a default name is used.
3. In the **States** group, enter the name you want to use for the mode in the **New Mode** box.
4. Choose the arrow key to the right of the **New Mode** box.
5. The new mode's name is displayed in the **Form Mode** list.

   For information on how to define the mode, see "To define a mode" later in this chapter.

To specify a default mode

1. Right-click the FormManager control on the page and select the **Properties** command from the shortcut menu.
2. On the **Form Mode** tab of the **Property Pages**, select the mode you want to use for the default mode in the **Default Mode** list.

   This mode must be one of the modes you've named and placed in the **Form Mode** list.

To define a mode

1. Right-click the FormManager control on the page and select the **Properties** command from the shortcut menu.
2. On the **Form Mode** tab of the **Property Pages**, select the mode you want to define in the **Form Mode** list.
3. In the **Actions Performed for Mode** table, select a control whose property you want to set or for which you want to run a method in the **Object** field. For example, you can set a Textbox control's disabled property to true, or use the addItem method to add a value to a Listbox control.

4. In the **Member** field, select the property you want to set or the method you want to run.

5. In the **Value** field enter the value you want to set the property to, or any parameters for the method.

   If you're setting a property, the **Value** field will show <value>. Replace this with the property setting. If the setting is a string, use quotation marks.

   If you're running a method, the **Value** field will show empty parentheses. Put any parameters for the method inside the parentheses. In the drop-down list in the **Member** field, the methods are listed with their parameters.

6. Repeat steps 3 through 5 until you've defined all the property settings and methods for this form mode.

   For more information on defining form modes, see "Simplifying Data Entry Pages" in Chapter 5, "Walkthroughs."

To specify when mode transitions occur

1. Right-click the FormManager control on the page and select the **Properties** command from the shortcut menu.
2. On the **Action** tab of the **Property Pages**, select the mode whose transition you want to define in the **Current Mode** field of the **Form Mode Transitions** table.
3. In the **Object** field, select the object you want to use to trigger a change from this mode to another mode. For example, you may want the user to click the Insert button to move from Update mode to Insert mode.
4. In the **Event** field, select the control event that will trigger the mode transition. For example, the onclick event of the Insert button.
5. In the **Next Mode** field, select the mode you want to move to when the control event occurs. For example, you might move to Insert mode.
6. Repeat steps 2 through 5 until you've defined transitions among all the modes on your form.
7. In the **Actions Performed Before Transition** table, you can also set properties and define methods that will take effect when a mode transition occurs, but before you move to the new mode. For information on how to make these settings, see "To define a mode," earlier in this chapter.

In addition to the methods provided with Microsoft Visual InterDev and the design-time controls' Scripting Object Model, you can also reference methods you've defined in the page's script in a mode or a mode transition. For example, you might want to run a method that validates data after you press a Save button, but before you move to a new mode.

To associate user-defined methods with a mode

1. Follow the instructions in "To define a mode" or "To specify when mode transitions occur" above.
2. In the **Member** field of the **Actions Performed for Mode** table on the **Form Mode** tab, or the **Member** field of the **Actions Performed Before Transition** table on the **Action** tab, enter your user-defined method.
3. Enter the method's parameters (if any) within parentheses in the **Value** field.

   Your method will run when the mode is activated, or when the mode transition occurs. If you have more than one user-defined method, the methods will run in the order in which they appear in the table on the property pages.

For an example of a user-defined method on a data-entry form, see the Data Entry Form Sample in the online Visual InterDev documentation.

CHAPTER 4

Editing Basics

Microsoft Visual InterDev application development is centered on creating Web pages. To help you, Visual InterDev provides a rich set of tools for editing and scripting HTML pages (.htm files) and Active Server Pages (.asp files).

Your primary tool is a Web page editor. You can work with Web pages using a number of editors, depending on your preference and what you are doing with the Web page. The default editor is the Visual InterDev HTML editor, which allows you to work in Design view to see your page as it will look in a browser. Alternatively, you can work in Source view, which allows you to work with the "raw" HTML text and objects on your pages.

Editing HTML

To display text in a Web page, you simply start a new HTML document (.htm file) or ASP page (.asp file) and enter text onto the page. If you want to format the text, or add features such as images or links to your page, you use HTML (Hypertext Markup Language).

If you aren't an HTML expert, you will probably want be familiar with how HTML works before getting started. Knowing HTML can help you get the most out of your Web-based application.

When you are creating HTML text, you most often use Design view in the editor, which displays text with all the formatting applied, the way you see documents in a word processor. Design view is the default editor for .htm files.

For more precise control over your document, you can edit in Source view, which displays your text and the HTML tags that are used to format it.

Working with HTML Tags

HTML is built around *tags*, which are formatting instructions embedded in the text. Tags are surrounded by angle brackets (< and >) to distinguish them from the surrounding text. Tags are also typically used in pairs around the text you want to format — an opening tag, the text, and a closing tag, which is marked with a slash (/). For example, the following line shows how you would format some text as bold (<B> tag) and some as italic (<I> tag):

```
<B>Welcome</B> to my <I>home page</I>.
```

When displayed in a browser, this line would look like this:

Welcome to my *home page.*

HTML includes tags for character formatting and paragraph formatting (for example, centering a paragraph). Tags are also used to specify features such as images, links, tables, and forms. The following table provides a brief outline of commonly used tags.

To insert	Use	Example
New paragraph	<P>	<P>This is a paragraph</P><P>This is a second paragraph</P>
Image	<IMG> plus the name of the image file to display	<IMG SRC="Monalisa.gif">
Link	<A> plus the address of the page to jump to, followed by link text	<A HREF="home.htm">Go to Home Page</A>

Table	<TABLE> to define the table
	<TR> for each row
	<TD> for each cell
		<TABLE>
		  <TR>
		    <TD>Row 1, Cell 1
		    </TD>
		  </TR>
		  <TR>
		    <TD>Row 2, Cell 1
		    </TD>
		  </TR>
		</TABLE>

Form	<FORM> to define the form <INPUT> to define controls such as text boxes and buttons	<FORM> Name: <INPUT Type="Text"> <INPUT Type="Submit"> </FORM>

Note You can easily create HTML elements in your documents using commands from the HTML menu.

For details, refer to the topics "Inserting Links and Bookmarks in Web Pages" and "Creating and Editing Tables in the HTML Editor" in the Visual InterDev online documentation.

To create forms, you can use Visual InterDev design-time controls. For details, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Creating HTML in the Editor

The Visual InterDev editor allows you create HTML in new pages or existing ones. For details, see "Creating Pages" in Chapter 2, "Web Basics."

When you are editing, you can work with HTML in different ways:

- In Design view, you can format text and paragraphs the way you would in a word processor and the editor will embed the appropriate HTML tags for you.

  For example, if you select text and click the bold button in the toolbar, the editor inserts <B> and </B> tags around the selected text. For more complex tags, such as an image or table tag, you can insert an entire element by filling in choices in a dialog box.

- In Source view, you can see HTML tags and edit them directly. You can also embed HTML tags using toolbar buttons or menu commands, as you can in Design view. For example, you can insert a link by choosing a menu command and specifying a target page.

  The editor colors HTML text in Source view to help you distinguish the different parts of HTML tags at a glance. For example, tag names are displayed in one color, tab attributes such as SIZE= in another, and so on. Unrecognized text is colored also, which helps you find errors, such as unmatched tag brackets, quickly.

  > **Tip** You can change the editor font style and colors in the Options window. From the Tools menu, choose Options, expand the Text Editor node, and choose Fonts and Colors.

- In Quick view, you can see what .htm files will look like in Microsoft Internet Explorer. (Quick view might not show you an accurate rendering of .asp files.)

At any point during your editing you can switch between views to see the effects of edits you are making.

To switch between views

- At the bottom of the editing window, choose the tab for the view you want.

  Note When you switch from **Source** to **Design view**, the editor might adjust white space (for example, spaces and tabs) and completes any incomplete HTML tags that exist in the Web page. For details, refer to the topic "Design View, HTML Editor" in the Visual InterDev online documentation.

When you initially create or open a Web page, it is displayed in either Design or Source view, depending on the default for the type of Web page you are editing.

To change the default view

1. From the **Tools** menu, choose **Options**.
2. In the left pane, expand **HTML**.
3. In the **Initial view** area, choose the default view for HTML (.htm or .html) pages and ASP (.asp) pages.

Adding Components to a Page

In addition to typing HTML text, you can add a variety of components to your page, including:

- Controls and objects
- References to other files, including images, links, and style sheets
- Database connections
- HTML text

In general, you can add components to your page by dragging them from either the Toolbox or the Project Explorer.

Adding Controls and Objects

When you are working in Microsoft Visual InterDev, the Toolbox lists controls that you can use on your page. These include:

- **Visual InterDev design-time controls** User interface controls such as text boxes and buttons that allow you to use standard object-oriented techniques for creating and scripting Web pages.
- **ActiveX controls** Controls that are registered on your computer.
- **HTML controls** Standard HTML controls such as text areas and buttons.
- **Server components** A list of the components and objects supported on Microsoft Internet Information Server (IIS) that you can use in server script. These include ActiveX Data Objects (ADO), Index Server objects, and more.

To use any of these components, drag them from the Toolbox onto your page at the location you want them to appear. Most controls and objects are displayed graphically in the HTML editor in both Design view and Source view. In Source view, you can specify that you want to see the text version of an object.

Note You can customize the Toolbox by adding new elements and tabs. For details, refer to the topic "Toolbox" in the Visual InterDev online documentation.

Adding References to Files

When you add images, links, documents, and style sheets to your page, you don't actually embed the element in your page. Instead, you add a reference to that element. For example, when you add an image, you are adding an HTML <IMG> tag that includes the name of the .gif file containing the image.

You can add a referenced element to your page by dragging it from the Project Explorer. You can do this with:

- **Images** If you drag an image from the Project Explorer to the page (for example, from the images node), the HTML editor creates an <IMG> tag.
- **Web pages** By dragging an .htm or .asp file onto the page, you automatically create a hyperlink to that document.
- **Documents** You can drag a text file or word processing document from the Project Explorer onto the page to create a link that allows users to download the document.
- **Style sheets** If you drag a style sheet (.css file) from the Project Explorer into the header of the document, you automatically create a link to that style sheet.

You can drag elements onto the page from any drag source, including Windows Explorer. When you do, the URL of the element is made relative to your project root.

> **Note** You might need to fix the URL of the link in Source view to be correct for your deployed project.

Adding Database Connectivity

If your Web pages interact with a database, you can easily create data commands in your Web project and data-bound controls on a page. Using drag and drop, you can:

- Add a *data connection* to your Web project. A data connection contains information required to connect to a specific database in a specific location.
- Add a *data command* to an existing data connection. A data command is a pointer to a database object — a table or view, stored procedure, database diagram, or query.
- Create data-bound controls on the page.

For more information about data connectivity, see Chapter 3, "Database Basics" and Chapter 18, "Database Concepts." For details about using data-bound controls, see "Getting Records" and "Displaying Data on Your Web Page" in Chapter 19, "Viewing Data."

Storing and Reusing HTML Text

If you want to reuse any element in your page — HTML text, a control, a script, a link, or any other element that you can select — you can store it on the Toolbox. Select the text, and then drag it to the Toolbox. When you drop it, the Toolbox creates a new element called HTML Fragment. You can then drag the element from the Toolbox to any other page.

If you want to store multiple elements this way, you can give them meaningful names and even create a special tab in the Toolbox for them.

Adding Scripts

You can create basic Web pages using nothing more than text and HTML tags. However, if you want to create sophisticated, data-driven applications, you can add *script* to your Web pages.

Scripts are programs that run when users display your Web page. They can be simple or complex, depending on your needs. You can include either client scripts or server scripts.

For example, you can use script to create these types of Web pages:

- A page that includes the current time and date along with the text.
- A page that displays the number of times that the Web site has been visited.
- A page that displays a form for users to fill in and then returns requested information or updates a database.
- A page that performs database operations that include transaction processing and other sophisticated data management operations.

You can write script in a variety of scripting languages. Two common scripting languages are Microsoft Visual Basic, Scripting Edition (VBScript) and JScript, Microsoft's implementation of the ECMAScript language.

You can choose whichever language you prefer, and you can even use different languages for different scripts on the same Web page.

The Microsoft Visual InterDev editor helps you create script with these features:

- Source view, where you can write script directly.
- Colored text that clearly shows you the different elements of your script statements.
- IntelliSense, which helps you create error-free script statements by presenting you with the names of methods and properties as soon as you've typed in the name of an object.
- The Script Outline Window, which displays the client and server objects in your page, and a list of the events that you can write script for.

Note You can use Visual InterDev design-time controls to generate script for common tasks such as querying a database, presenting an input form, or displaying a report. For details, see Chapter 19, "Viewing Data," Chapter 20, "Modifying Data," and Chapter 24, "Scripting with Design-Time Controls and Script Objects."

For more information about scripting, see "Editing and Scripting" and "Scripts in Web Applications" in Chapter 23, "Scripting Concepts."

After you've written scripts, you can use the built-in debugger commands to help you find errors in them. For more information, see "The Script Debugging Process" in Chapter 23, "Scripting Concepts" and Chapter 26, "Debugging Your Pages."

Choosing an Alternate Editor

Microsoft Visual InterDev allows you to choose from a variety of editors to work with files in your project. The default editors are:

- **The HTML editor**, which allows you to edit the layout of your Web page and create scripts. The HTML editor opens automatically when you create or open an HTML page or ASP page.
- **The style sheet editor**, which opens when you create or edit a cascading style sheet (.css file).
- **The text editor**, which opens when you create or edit a text file.

If you prefer, you can use an alternate editor. Choices include:

- Microsoft FrontPage, which you might use if you are already familiar with FrontPage, or if you want to take advantage of specific FrontPage features such as WebBots.
- An external editor of your choice.

You can choose a specific editor to open a page as needed, or you can make it the default editor for that type of page.

To open a Web page in a selected editor

1. In the Project Explorer, right-click the name of the file and then choose **Open With**.
2. Select an editor to use. If you want to make your choice the default editor, choose **Set as Default**.
3. Choose **OK**.

CHAPTER 5

Walkthroughs

The topics in this section provide mini-tutorials that walk you through some typical Web application development scenarios using Microsoft Visual InterDev. These walkthroughs are intended to provide:

- A guide to accomplish the primary tasks you'll need in your own work.
- Step-by-step instructions to be used as a learning exercise.
- A means to get acquainted with Visual InterDev.
- Pointers to more detailed information in other topics.

Creating a Home Page

Creating a home page for your Web site is a great way to become familiar with a new Web authoring and development product. If you already know how to create navigation bars, how to design page layouts, and how to code with JavaScript, this is an opportunity to see how to accomplish these tasks using Microsoft Visual InterDev.

In Visual InterDev, creating a home page consists of some or all of these steps:

- Creating a Visual InterDev Web Project
- Adding the Home Page
- Including Graphics
- Applying a Consistent Visual Design
- Adding Navigation to Your Pages
- Customizing Your Home Page
- Maintaining Your Home Page

Creating a Visual InterDev Web Project

Before you can begin drafting a home page, you must first create a Web project. A Visual InterDev Web project organizes the files used for a Web application, such as images, documents, and HTML pages.

When you create a Web project, you specify a Web server and project name for the Web application. You can also specify a page layout and a visual presentation style for all pages.

To create a Visual InterDev Web project

1. From the **File** menu, choose **New Project**.

   The **New Project** dialog box appears.

2. On the **New** tab, choose **Visual InterDev Projects** from the left pane and then select **New Web Project** from the right pane.

3. In the **Name** text box, enter a name for the Web project.

4. Click **Open**.

   The **Web Project Wizard** appears.

The Web Project wizard steps you through the process of selecting a master Web server, a Web application name, as well as a default theme and layout for the Web application.

For more information, see "Creating a Web Project" in Chapter 1, "Web Project Management," and Chapter 10, "Managing Web Projects."

Adding the Home Page

You can easily add a home page to your Web application using a site diagram. A site diagram is an electronic whiteboard that allows you to rapidly prototype a new Web application with skeleton HTML or ASP pages you can later customize.

To create a new home page

1. From the **Project** menu, choose **Add Item**.

   The **New Tab (Add Item Dialog Box)** appears.

2. From the **Web Project Files** folder, select **Site Diagram**.

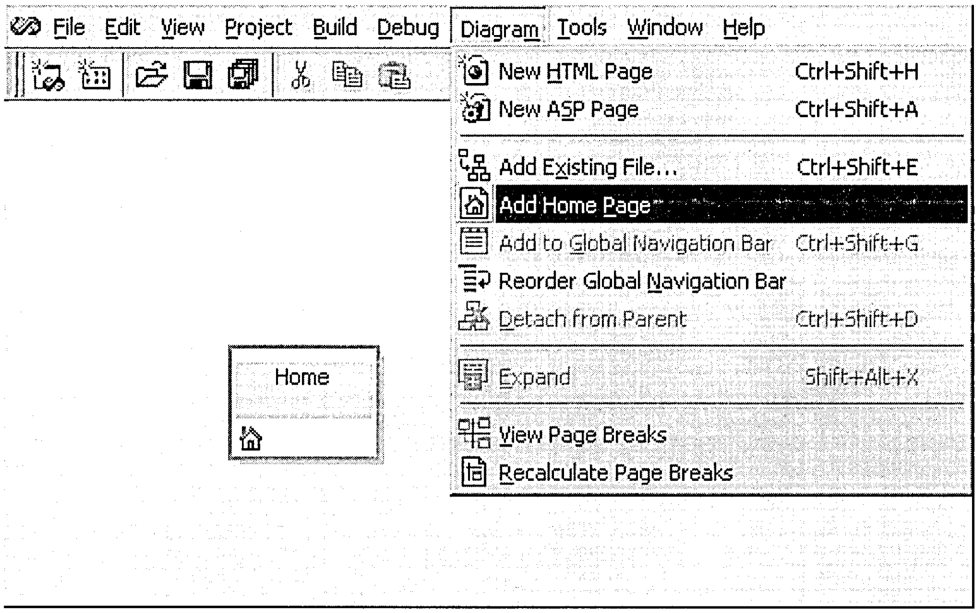
3. In the **Name** text box, type a name for the site diagram.

4. Click **Open**.

   A new site diagram opens.

5. From the **Diagram** menu, choose **Add Home Page**.

A page graphic appears in the site diagram with the name "Home."


File Edit View Project Build Debug Diagram Tools Window Help
New HTML Page Ctrl+Shift+H
New ASP Page Ctrl+Shift+A
Add Existing File... Ctrl+Shift+E
Add Home Page
Add to Global Navigation Bar Ctrl+Shift+G
Reorder Global Navigation Bar
Detach from Parent Ctrl+Shift+D
Expand Shift+Alt+X
View Page Breaks
Recalculate Page Breaks
Home

6. From the **File** menu, choose **Save**.

   The changes to the site diagram have been saved and Visual InterDev creates a new home page called `Default.htm or Default.asp` depending on the Web server you use.

   > **Note** Microsoft Internet Information Server specifies `default.asp` for a home page and Microsoft Personal Web Server specifies `default.htm` for a home page by default.

For more information, see Chapter 12, "Designing a Web Site," and Chapter 14, "Managing a Site Diagram."

Including Graphics

Adding multimedia, such as .gif, .wav, or .avi files, to a Web page helps liven the overall appearance and presentation of a page. You can easily add graphics and other HTML elements to your home page using the WYSIWYG approach of Design View in the HTML editor.

To include a graphic on your home page

1. In the **Project Explorer**, right-click your home page and choose **Open**.

   > **Note** Microsoft Internet Information Server specifies `default.asp` for a home page and Microsoft Personal Web Server specifies `default.htm` for a home page by default.

2. In the **HTML** editor, select the **Design** tab.

3. From the **HTML** menu, choose **Image**.

   The **Insert Image Dialog Box** appears.

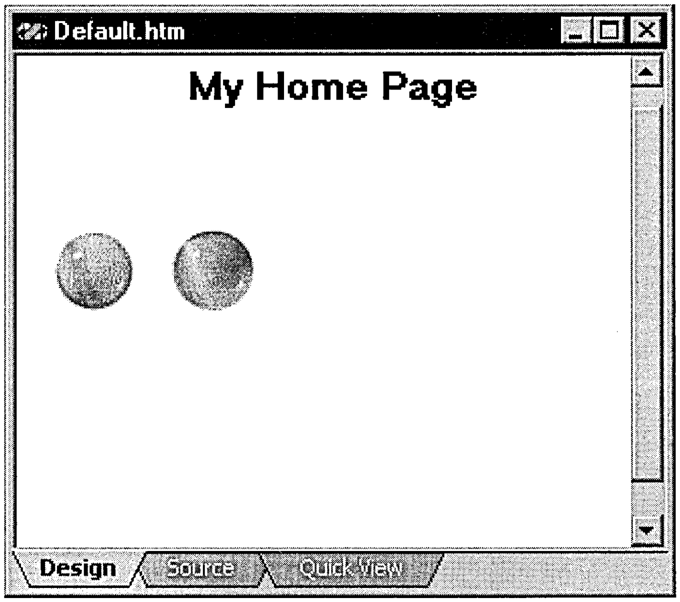
4. In the **Picture Source** text box, type the name of the .gif or .jpeg file.

   – or –

   Click **Browse** to search for the image file.

5. In the **Alternate Text** box, type the information you want users to see if the image doesn't display.

6. Click **OK**.



Applying a Consistent Visual Design

Establishing a consistent visual appearance for pages gives your Web site a professionally designed look. You can use a Microsoft theme to give the pages in your Web application visual consistency. Themes can be applied to an entire project, or to individual pages.

To create a consistent visual look using Microsoft themes

1. In the **Project Explorer**, select your home page.

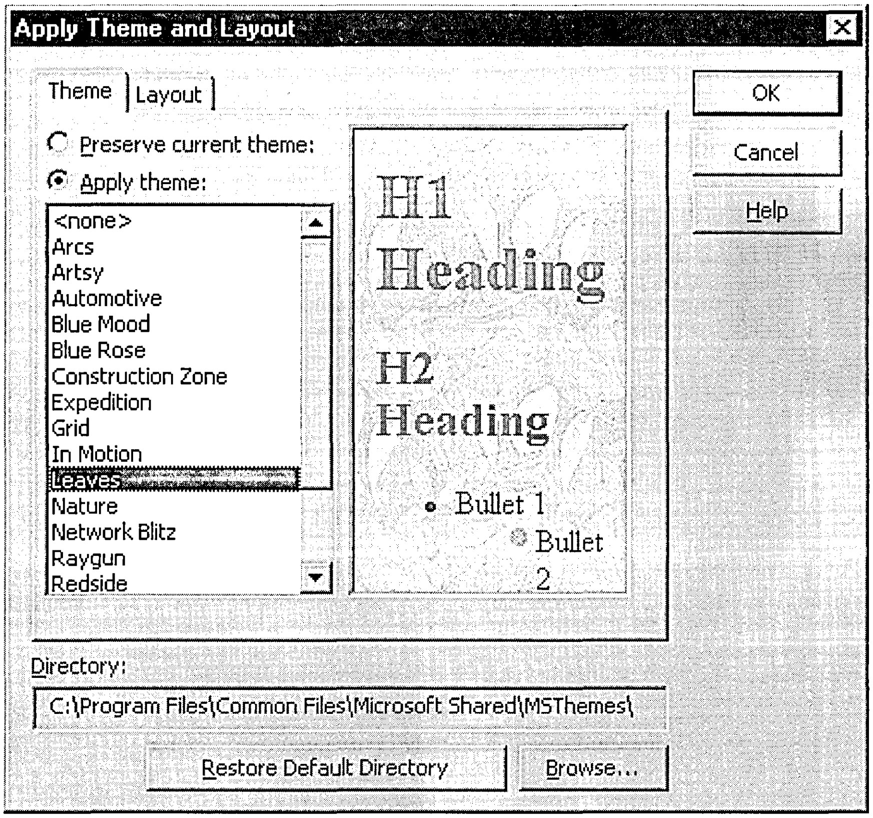
   Note Microsoft Internet Information Server specifies `default.asp` for a home page and Microsoft Personal Web Server specifies `default.htm` for a home page by default.

2. From the **Edit** menu, choose **Apply Theme and Layout.**

   The **Apply Theme and Layout** dialog box appears.

3. On the **Theme** tab, choose a theme name.

   A graphic of the theme appears in the preview pane.



Apply Theme and Layout

Theme | Layout

( ) Preserve current theme:
(•) Apply theme:

<none>
Arcs
Artsy
Automotive
Blue Mood
Blue Rose
Construction Zone
Expedition
Grid
In Motion
Leaves
Nature
Network Blitz
Raygun
Redside

Directory:
C:\Program Files\Common Files\Microsoft Shared\MSThemes\

4. Click **OK**.

For more information, see “Site Consistency” in Chapter 11, “Site Design,” and Chapter 17, “Customizing Page Appearance.”

Adding Navigation to Your Pages

Navigation bars contain the links that allow your users to move through your Web site. Making that navigation consistent will make it easier for your users to browse your site. You can automatically generate navigation bars using layouts or a design-time control and site diagrams.

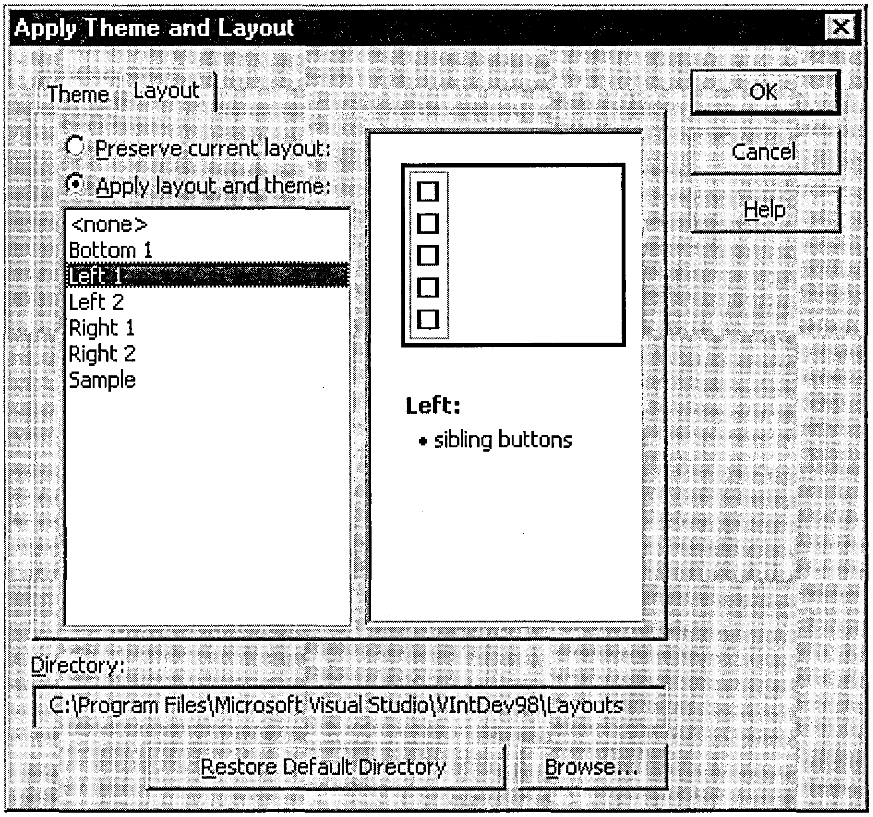
When you apply a layout or use the PageNavbar control, you can use site diagrams to design and maintain the navigation structure of a Web page. Visual InterDev then uses the navigation structure you designed to generate the links on the navigation bars for your pages.

To add a layout to a page

1. In the **Project Explorer**, select your home page.
2. From the **Edit** menu, choose **Apply Theme and Layout**.

   The **Apply Theme and Layout** dialog box appears.
3. On the **Layout** tab, choose a layout name.

   A graphic of the layout appears in the preview pane.


Apply Theme and Layout

Theme | Layout

( ) Preserve current layout:
(•) Apply layout and theme:

<none>
Bottom 1
Left 1
Left 2
Right 1
Right 2
Sample

Left:
• sibling buttons

Directory:
C:\Program Files\Microsoft Visual Studio\VIntDev98\Layouts

4. Click **OK**.

For more information about site navigation, see Chapter 13, "Designing Site Navigation."

Customizing Your Home Page

Once you have used the built-in features to create and design the basic structure of your home page, you can easily customize your pages. Use HTML text and tags to create the content for your page. You can also use script to perform tasks to enhance your Web pages. For example, you can use with client-side script to generate text that tells visitors when the home page was last updated.

To add an update message

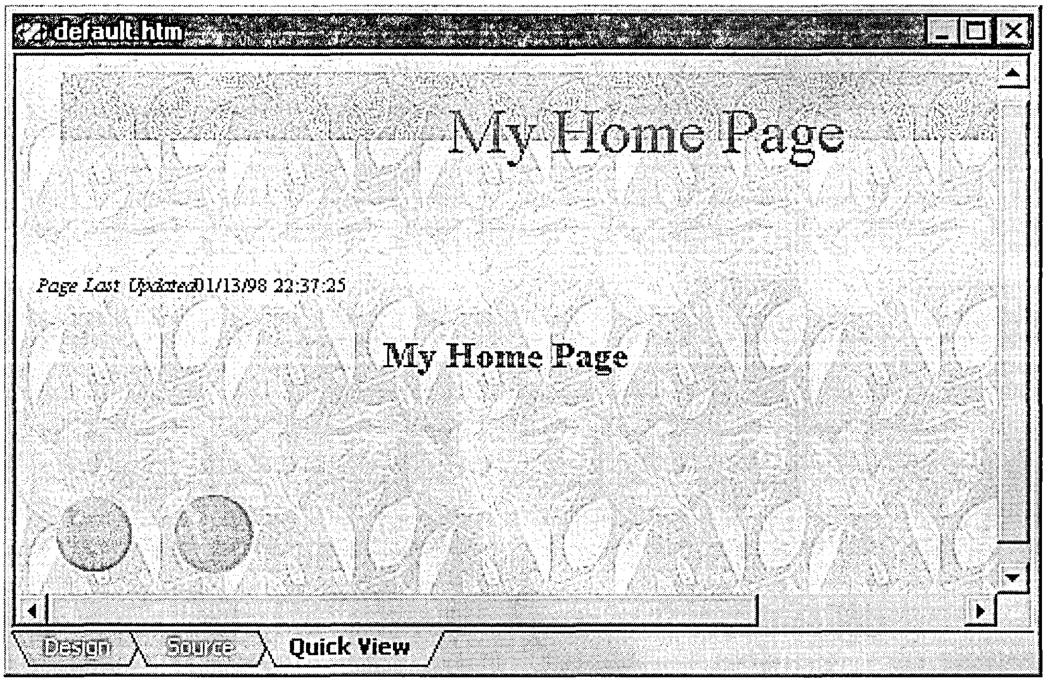
1. From the **Project Explorer**, right-click the home page and choose **Open**.

   The home page opens in the HTML editor.

2. On the **Source** tab, type the following code after the `<BODY>` tag:

```
<Script language="JavaScript">
<!--
 document.write("<I>Page Last Updated:</I> "+document.lastModified);
// -->
</Script>
```

3. On the **Standard** toolbar, click **Save**.
4. To view the last updated text, click the **Quick View** tab.



*Page Last Updated:*01/13/98 23:37:25

For more information, see “Adding Scripts” in Chapter 4, “Editing Basics,” and Chapter 25, “Scripting with HTML Elements.”

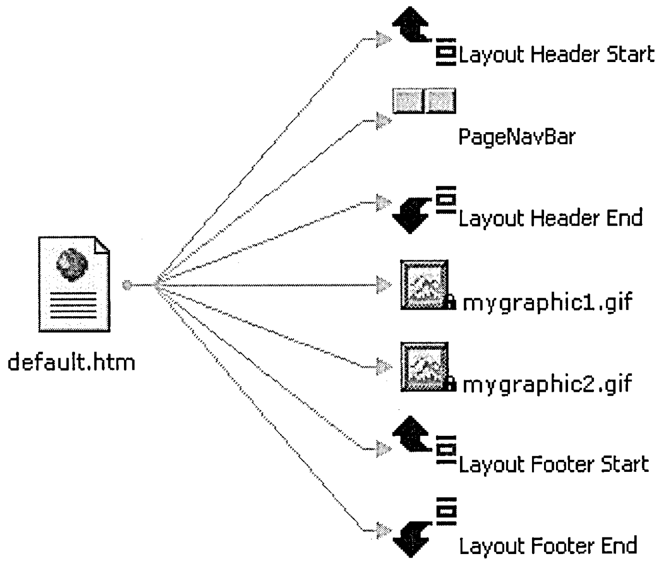
Maintaining Your Home Page

Once you have created your home page, the task switches to maintaining that page. It is important to your users that the links work correctly. You can maintain the links between items in your home page using Link View. You can quickly see if a link is broken and read the error for the link.

To view the links for your home page

- From the **Project Explorer**, right-click the home page and select **View Links**.

  A link diagram appears displaying the home page in the middle of the diagram. The link diagram can be filtered to show different types of links to make managing those links easier. The link diagram can be filtered to show different types of links to make managing those links easier.



If you link more items to your home page, you can use Link View to verify them. For more information, see "Link Verification" in Chapter 11, "Site Design," and "Repairing Links" in Chapter 16, "Maintaining Links."

Debugging Script

Debugging script allows you to find errors in your script, such as syntax errors, run-time errors, and logic errors, before you publish that script to the World Wide Web. You probably already know how to set breakpoints for debugging and how to step through lines of script. Now you can see how to accomplish these tasks using Microsoft Visual InterDev.

In Visual InterDev, debugging script consists of some or all of these steps, once you have added script to a page:

- Adding Client Script to an HTML Page
- Setting Breakpoints
- Stepping Through Lines of Script
- Changing the Value of a Variable

Adding Client Script to an HTML Page

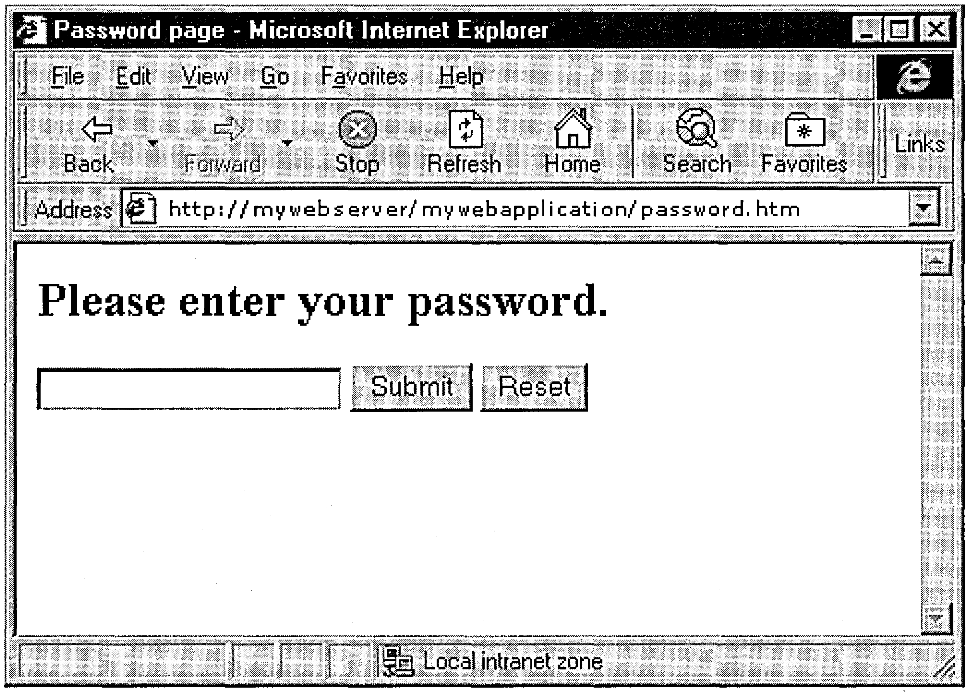
Debugging client script is an easy way to become familiar with debugging in Visual InterDev. Client-side script is processed on the browser, while server-side script is processed on the Web server. For more information about client and server script, see "Debugging Client Script" and "Debugging Server Script" in Chapter 26, "Debugging Your Pages."

To use this walkthrough, you will need to create a file with script to debug. The sample client script, located between the `<SCRIPT> </SCRIPT>` tags in the example below, validates the length of password entries made in a simple form. The remaining HTML tags define the form and do not participate in the script debugging process.

To use the script below, create a new .htm file and switch to Source view in the HTML editor. Copy the script and HTML then right-click and choose Paste As Text from the shortcut menu. For more information, see Chapter 25, "Scripting with HTML Elements."

```
<SCRIPT LANGUAGE="JavaScript">
function validatePassword()
{
   var password;
   password = document.frm1.txtPassword.value;
   // debugger;
   alert("You entered " + password);
   if (password.length < 4) {
      alert("You must enter 4 characters or more!");
      document.frm1.txtPassword.select();
      return false;
   }
   else {
      alert ("Your form is being submitted!");
      return true;
   }
}
</SCRIPT>
<P><H2>Please enter a password.</H2>
<P>
<FORM NAME="frm1" METHOD="Post" ACTION="Process.asp" OnSubmit="return validatePassword()">
   <INPUT NAME="txtPassword" TYPE="Password">
   <INPUT NAME="btnSubmit" TYPE="Submit" VALUE="Submit">
   <INPUT NAME="btnReset" TYPE="Reset" VALUE="Reset">
</FORM>
```

Note JavaScript is case sensitive, so the sample script above must appear in your .htm file exactly as it does here.


Password page - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites
Links
Address
Submit
Reset
Local intranet zone

http://mywebserver/mywebapplication/password.htm

Please enter your password.

When the user clicks the Submit button, the script checks that the password length is four characters or greater. The script generates a message box listing the password the user entered. If the password is four characters or greater in length, the user is then redirected to the page called by the Action element of the <FORM> tag.

If you haven't created Process.asp, the file that the form will call, the browser will display the "object not found" error. If the password is fewer than four characters in length, a message box appears indicating the character length requirement.

Setting Breakpoints

To specify a place in the script where you want to stop and examine the state of the process, you can use breakpoints. Breakpoints indicate the line of script you want to stop on in the application. You can then step through, step into, or step over lines of script individually to find errors. Breakpoints show up as red octagons to the left of a line of script in Source view.

> **Note** To debug client scripts in Microsoft Internet Explorer, you must be using Internet Explorer 4.0 and debugging must be enabled. For details, see the Internet Explorer documentation.

To set breakpoints

1. Open the .htm file and select **Source view**.
2. Place the cursor in the line of script you want to set a breakpoint in.
3. From the **Debug** menu, choose **Insert Breakpoint**.

   A red octagon appears to the left of the line of script.

   Note You can also set or remove a breakpoint by placing the cursor in a line of script and pressing F9.

To launch the debugger

1. In the **Project Explorer**, right-click the .htm file and select **Set As Start Page**.
2. Launch Internet Explorer 4.0 and load the .htm page into the browser.
3. From the **Debug** menu in Visual InterDev, choose **Processes**.

   The **Processes** dialog box appears.
4. In the **Process** area, select **Microsoft Internet Explorer,** and then select **Attach**.

   Microsoft Internet Explorer and your machine name appear in the **Debugged Processes** area.
5. In Internet Explorer, enter a password and choose **Submit**.

The script executes until Internet Explorer reaches the breakpoint. When Internet Explorer reaches the breakpoint, it stops and displays the source script in Source view.

A yellow arrow superimposed over the breakpoint icon indicates the line of script where the debugger stopped.

For more information, see "Debugging Client Script" in Chapter 26, "Debugging Your Pages."

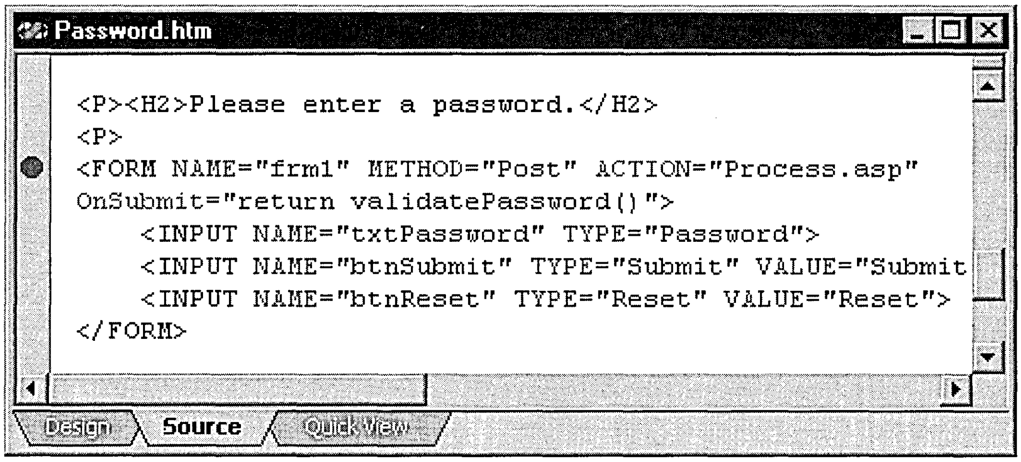
Stepping Through Lines of Script

To execute script one line at a time, you can use Step Into. After stepping through each line, you can view the effects of the statement by looking at the page in Internet Explorer.

To step into lines of script

1. Open the .htm file in **Source** view and set a breakpoint at the following line of script:

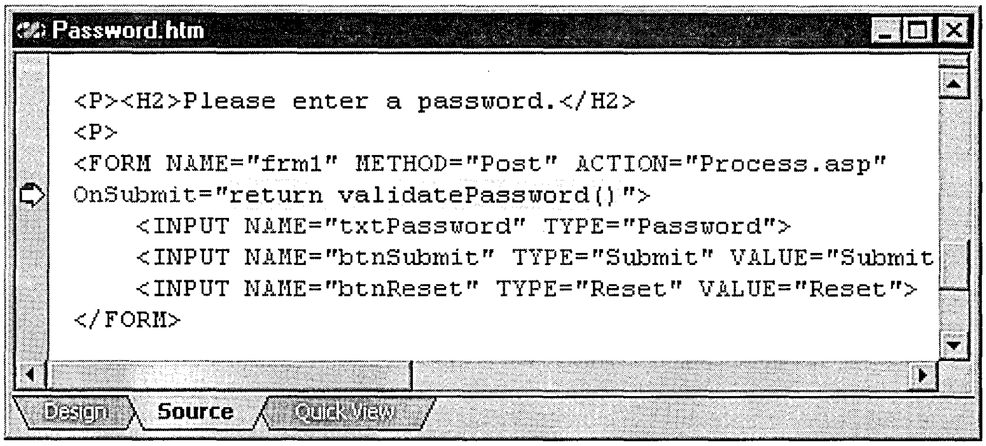
   ```
   <FORM NAME="frm1" METHOD="Post" ACTION="Process.asp" OnSubmit="return
   validatePassword()">
   ```


Password.htm

```
<P><H2>Please enter a password.</H2>
<P>
<FORM NAME="frm1" METHOD="Post" ACTION="Process.asp"
OnSubmit="return validatePassword()">
    <INPUT NAME="txtPassword" TYPE="Password">
    <INPUT NAME="btnSubmit" TYPE="Submit" VALUE="Submit
    <INPUT NAME="btnReset" TYPE="Reset" VALUE="Reset">
</FORM>
```

Note When you place a breakpoint in the line of HTML above, you are actually specifying that the script stop executing when you click the Submit button, `OnSubmit="return validatePassword()"`.

2. Launch the debugger.
3. Enter a two-character password and choose **Submit**.


Password.htm

```
<P><H2>Please enter a password.</H2>
<P>
<FORM NAME="frm1" METHOD="Post" ACTION="Process.asp"
OnSubmit="return validatePassword()">
    <INPUT NAME="txtPassword" TYPE="Password">
    <INPUT NAME="btnSubmit" TYPE="Submit" VALUE="Submit
    <INPUT NAME="btnReset" TYPE="Reset" VALUE="Reset">
</FORM>
```

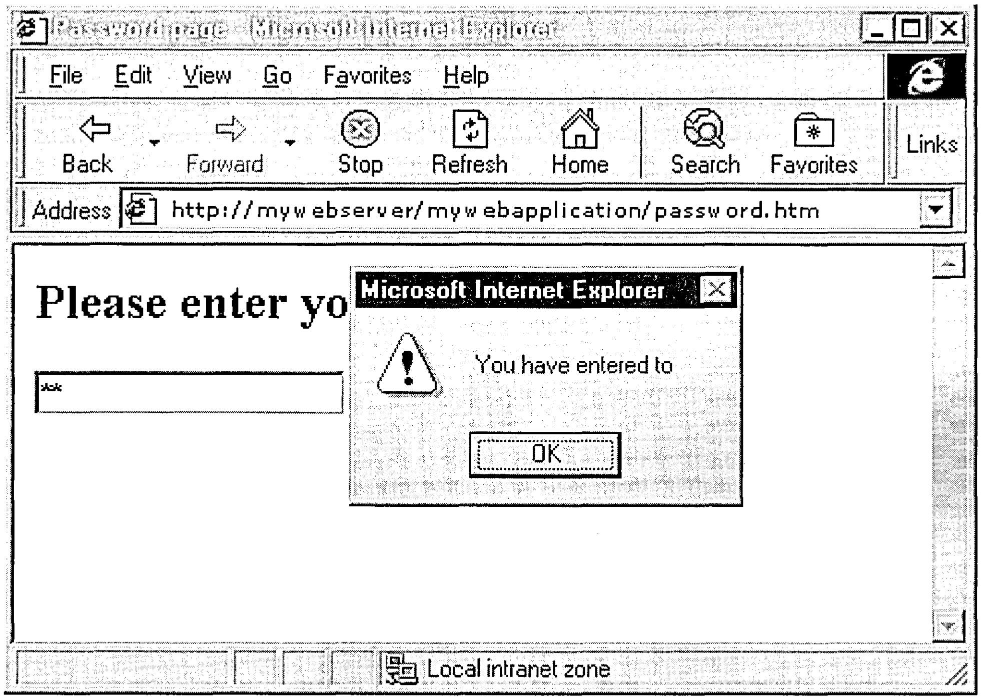
Internet Explorer reaches the breakpoint and switches to **Source** view. A yellow arrow appears superimposed over the breakpoint icon.

4. From the **Debug** menu, choose **Step Into,** and then choose **Step Into** a second time.

   The following line of script is highlighted:

   ```
   alert("You entered " + password);
   ```

5. From the **Debug** menu, choose **Step Into** again.


Password page - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
Links
Address
http://mywebserver/mywebapplication/password.htm
Please enter yo
**
Microsoft Internet Explorer
You have entered to
OK
Local intranet zone

Internet Explorer executes the alert script and displays the message box.

6. Choose **OK**.

   Internet Explorer reaches the next line of script and switches to **Source** view.

You can continue stepping into lines of script and view the results in Internet Explorer.

To execute procedures in script as a single unit, you can use Step Over. Step Over allows you to skip a procedure or function and step through the next procedure or function.

After stepping over a procedure, you can view the effects of the procedure by looking at the page in Internet Explorer.

To step over lines of script

1. Open the .htm file in **Source** view and set a breakpoint at the following line of script:

```
<FORM NAME="frm1" METHOD="Post" ACTION="Process.asp" OnSubmit="return
validatePassword()">
```

   Note When you place a breakpoint in the line of HTML above, you are actually specifying that the script stop executing when you click the Submit button, `OnSubmit="return validatePassword()"`.

2. Launch the debugger.

3. Enter a three-character password and choose **Submit**.

   Internet Explorer reaches the breakpoint and switches to **Source** view. A yellow arrow appears superimposed over the breakpoint icon.

4. From the **Debug** menu, choose **Step Over**.

5. Choose **OK** in each alert message.

   Internet Explorer executes the following script as a procedure unit:

   ```
   password = document.frml.txtPassword.value;
   //debugger;
   alert("You entered " + password);
   if (password.length < 4){
      alert("You must enter 4 characters or more!");
      document.frml.txtPassword.select();
      return false;
   ```

Internet Explorer reaches the line of script after the procedure unit and switches back to Source view. For more information, see Chapter 26, "Debugging Your Pages."

Changing the Value of a Variable

You can assign variables in your script as you debug. You change the value of a variable from the Immediate window. When Internet Explorer reaches a breakpoint and switches to Source view, you can then change the value of a variable and view the results in the Locals window.

To change the value of a variable

1. Open the .htm file in **Source** view and set a breakpoint at the following line of script:

   ```
   <FORM NAME="frml" METHOD="Post" ACTION="Process.asp" ONSUBMIT="return
   validatePassword()">
   ```

2. Launch the debugger.

3. Enter a three-character password and choose **Submit**.

   Internet Explorer reaches the breakpoint and switches to **Source** view. A yellow arrow appears superimposed over the breakpoint icon.

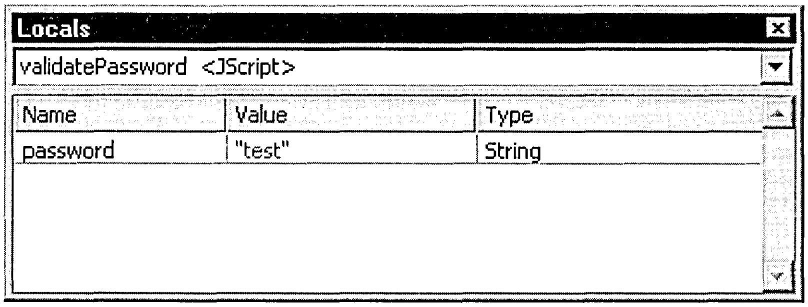
4. From the **Debug** menu in Visual InterDev, choose **Step Into**.

5. From the **View** menu, select **Debug Windows,** and then select **Immediate**.

6. In the **Immediate** window, type the following:

   ```
   password="test"
   ```

7. Press **Enter**.
8. From the **View** menu, select **Debug Windows,** and then select **Locals**.

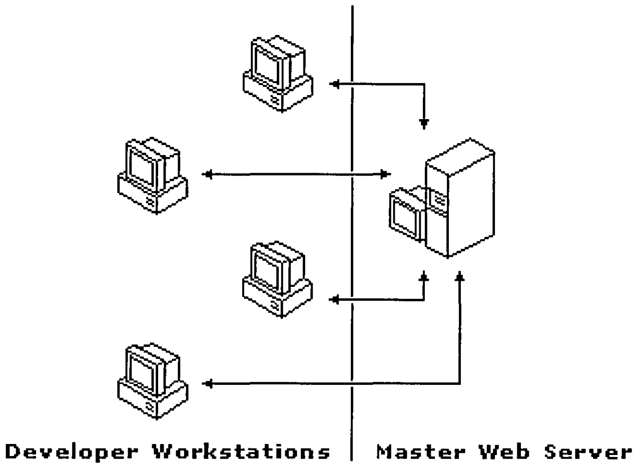


Name	Value	Type
password	"test"	String

The **Locals** window displays the new value of the variable you entered in the Immediate window.

Working with Multiple Developers

Sometimes it takes more than a single developer to create and maintain a Web application. When you work with a Web project, the master Web application is stored on a master Web server. When you work in a team taking advantage of Visual InterDev's developer isolation feature, individual team members work in local mode, with a local copy of the master application. This enables multiple developers to access the same master Web application from their local Web projects.



In addition to working in local mode, to facilitate team-based Web application development, you can use the built-in source control features of Visual InterDev.

In this walkthrough, the following scenarios are discussed:

- Using Source Control Features of Visual InterDev
  - Creating a Master Web Application
  - Sharing Web Application Files on the Master Web Server
- Controlling Sources with Microsoft Visual SourceSafe

Using Source Control Features of Visual InterDev

Visual InterDev's built-in source control features allow multiple developers to work on the same files at the same time. When there are conflicts between different versions of the file, Visual InterDev's source control helps you merge the differences.

Visual InterDev allows multiple developers to create their own unique local Web projects that access the same master Web application, which resides on a master Web server. Each developer is able to get working copies of files, edit them, and then update the master server.

If you have a copy of Microsoft Visual SourceSafe, you can use that to control your sources instead of Visual InterDev's built-in source control features. Visual SourceSafe is discussed later in this chapter under "Controlling Sources with Microsoft Visual SourceSafe."

Creating a Master Web Application

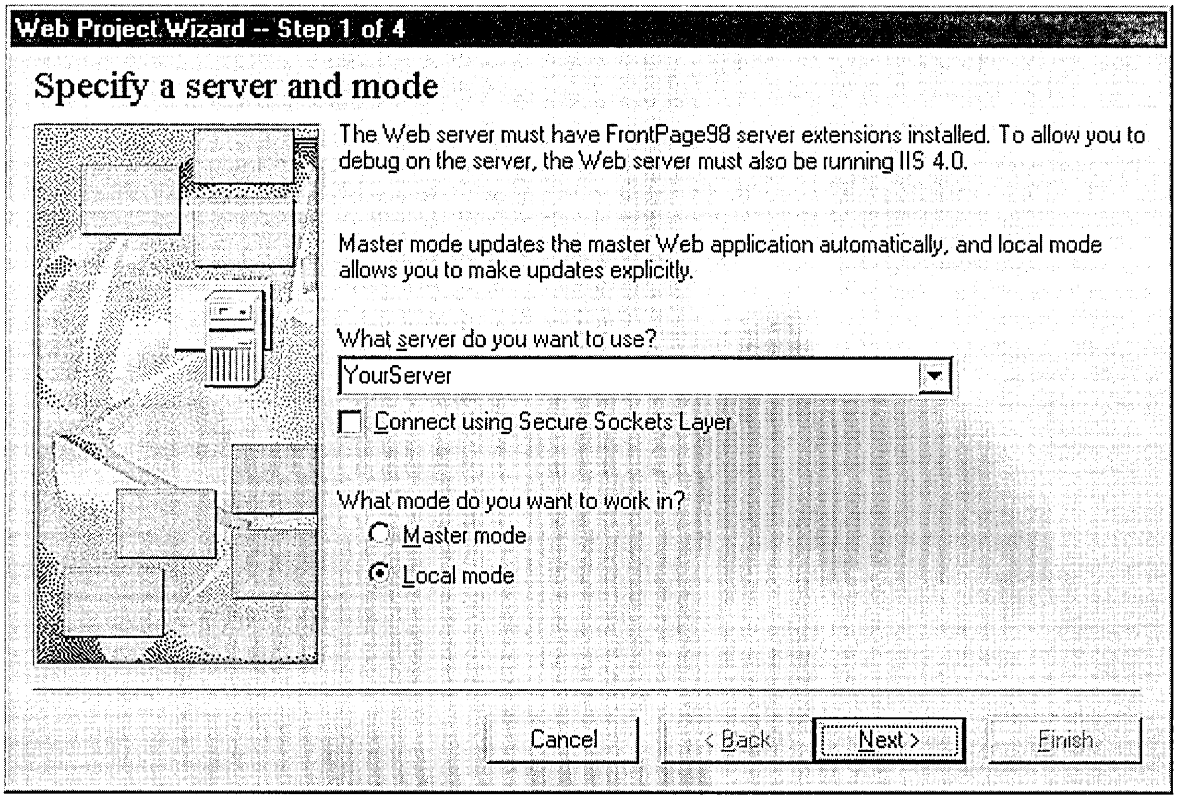
You can create a master Web application on the master Web server and then share it with other developers on your development team.

To create a master Web application

1. From the **File** menu, choose **New Project**. The **New Project** dialog box appears.
2. On the **New** tab, choose **Visual InterDev Projects** from the left pane and then select **New Web Project** from the right pane.
3. In the **Name** text box, enter a name for the Web project.
4. In the **Location** box, enter a location on your computer for the new local project.

   Note The name and location settings are for your local project only. These settings are not saved to the server and do not affect the name of the Web application or the Web site.
5. Click **Open**.

The **Web Project Wizard** appears.


Web Project Wizard -- Step 1 of 4

Specify a server and mode

The Web server must have FrontPage98 server extensions installed. To allow you to debug on the server, the Web server must also be running IIS 4.0.

Master mode updates the master Web application automatically, and local mode allows you to make updates explicitly.

What server do you want to use?

YourServer

☐ Connect using Secure Sockets Layer

What mode do you want to work in?

○ Master mode

● Local mode

Step 1: Specify a server and mode

Specify the name of your master Web server and select Local mode. Local mode is typically the preferred mode for each developer working in a multi-developer environment. This allows the developer to edit and test files independently from the master Web application.

Step 2: Specify your Web

Select **Create a new Web application** and type in a name.

Step 3: Apply a layout

It is not necessary to use a layout. If you want, you can select **<none>** now and apply a layout at a later time.

Step 4: Apply a theme

It is not necessary to use a theme. If you want, you can select **<none>** now and apply a theme at a later time.

6. Click **Finish.**

Visual InterDev simultaneously creates the new Web application on the master Web server and a new local project on your machine. After the new project is created, you see it displayed in the Project Explorer in your solution.

You are now able to use the commands on the Project menu to add HTML and ASP pages, get working copies, edit files, and update the server.

Now it's time to share files with one or more additional developers on your team. All new developers need to create their own local projects that access the existing master Web application. They follow steps that are similar to the ones above, except they do not create a new Web application.

To create local projects for an existing application

1. The new developer clicks **New Project** on the **File** menu to display the **New Project** dialog box.
2. On the **New** tab, the developer selects **Visual InterDev Web Projects** in the left pane and then **New Web Project** in the right pane.
3. The developer types his own settings in the **Name** and **Location** boxes for the new local project and then chooses **Open**.

   Note It is up to the developer's discretion whether or not to use the same name as the first developer. Using the same name can make it easier to identify a project that points to a shared Web application on the master Web server.

4. Click **Open**.

   The **Web Project Wizard** appears.

   Step 1: Specify a server and mode

   The new developer specifies the same server as the first developer and also selects **Local mode**.

   Step 2: Specify your Web

   The new developer selects the **Connect to an existing Web application on <server name>** option.

   The drop-down list contains a list of all available Web applications on the specified server. The new developer selects the appropriate Web application from the list.

5. Click **Finish**.

Visual InterDev accesses the master Web application on the master Web server and creates the new local project on the developer's machine. The developer then sees the new project in the **Project Explorer**. The developer now has access to all the files that are in the Web application on the master Web server.

For more information about working with local copies of a Web application, see Chapter 7, "Working Locally," and "Project Architecture" in Chapter 6, "Web Project Concepts."

Sharing Web Application Files on the Master Web Server

You and other developers can now access the same files in the master Web application. Each developer is able to get the latest version of each file. He can also get working copies, edit them locally, and then update the server with the changes.

Visual InterDev's source control features prevent you and the other developers from overwriting each other's work on the master Web server.

In the following scenarios Developer A and Developer B each have local projects on their machines that access the same master Web application.

To synchronize local projects with the master Web application

1. Developer A chooses **Add Item** on the **Project** menu to add multiple files to his project.

   The files are now in his local Web application, but are not in the master Web application. The (flag) icon indicates that a file exists in a local project but not in the master project.

2. Developer A adds the files to the master Web application by selecting the files and then clicking the **Project** menu, **Web Files**, and then **Add to Master Web**.

   The new files are copied to the master Web application. However, Developer B's project still reflects the former state of the master Web application.

3. Developer B synchronizes her project with the master Web application by clicking the **Project** menu, **Web** Project, and then **Synchronize Files**.

   Tip When working with other developers, it is a good idea to synchronize your local project frequently.

   Developer B's local project now reflects the changes made to the master Web server by Developer A.

To resolve conflicts between files

1. Developer A wants to work on a local copy of Filename.htm. He selects the file in the Project Explorer, chooses **Web Files** from the **Project** menu, and then chooses **Get Working Copy**. He begins to edit the file.

2. Developer B also wants to work on Filename.htm. She selects the file in the Project Explorer and clicks the **Project** menu, **Web Files**, and then **Get Working Copy**.

   Even though Developer A has already obtained a working copy of the file, Visual InterDev also allows Developer B to obtain a working copy.

3. Developer A has saved the file and wants to update the master Web application with his latest version. He clicks the **Project** menu, **Web Files**, and then **Release Working Copy** The master Web application is updated to reflect Developer A's changes.

4. Now Developer B has saved her changes and is finished with the file. She clicks the **Project** menu, **Web Files**, and then **Release Working Copy**.

   Visual InterDev compares Developer B's working copy against the master Web application's version and detects merge conflicts. The **Merge** dialog box is displayed to resolve conflicts.

5. Developer B can view all conflicts and choose how to resolve them. After resolving the conflicts, the master Web application is updated with the resolved version of the file.

6. Developer A's list of files in the Project Explorer is now out of synch with the master Web application. To get the latest version of the file in his local Web application, he chooses **Web Files** from the **Project** menu, and then chooses **Get Latest Version**.

For more information about working with others on Web application files, see Chapter 10, "Managing Web Projects" and Chapter 7, "Working Locally."

Controlling Sources with Microsoft Visual SourceSafe

If you have Microsoft Visual SourceSafe, you can extend the capabilities of Visual InterDev's built-in source control to include version control, rollback, and other source control features.

To use Visual SourceSafe with Visual InterDev

1. In order to use Visual SourceSafe with a master Web application, you need to first install it on the master Web server.

   Because Visual InterDev is compatible with Visual SourceSafe, it is not necessary to install Visual SourceSafe on the client (your local machine).

   For information about installing Visual SourceSafe, see "Setting Up Source Control on a Web Server" in Chapter 8, "Working with Multiple Developers."

2. After you have installed Visual SourceSafe on the server, you need to enable it on the master Web application. Once one developer has enabled it on the master Web application. Other developers must refresh their local project to take advantage of source control.

   To enable Visual SourceSafe as the source control, see "Adding Source Control to a Web Application" in Chapter 8, "Working with Multiple Developers."

Now, instead of "getting" and "releasing" files, you "check out" and "check in" files.

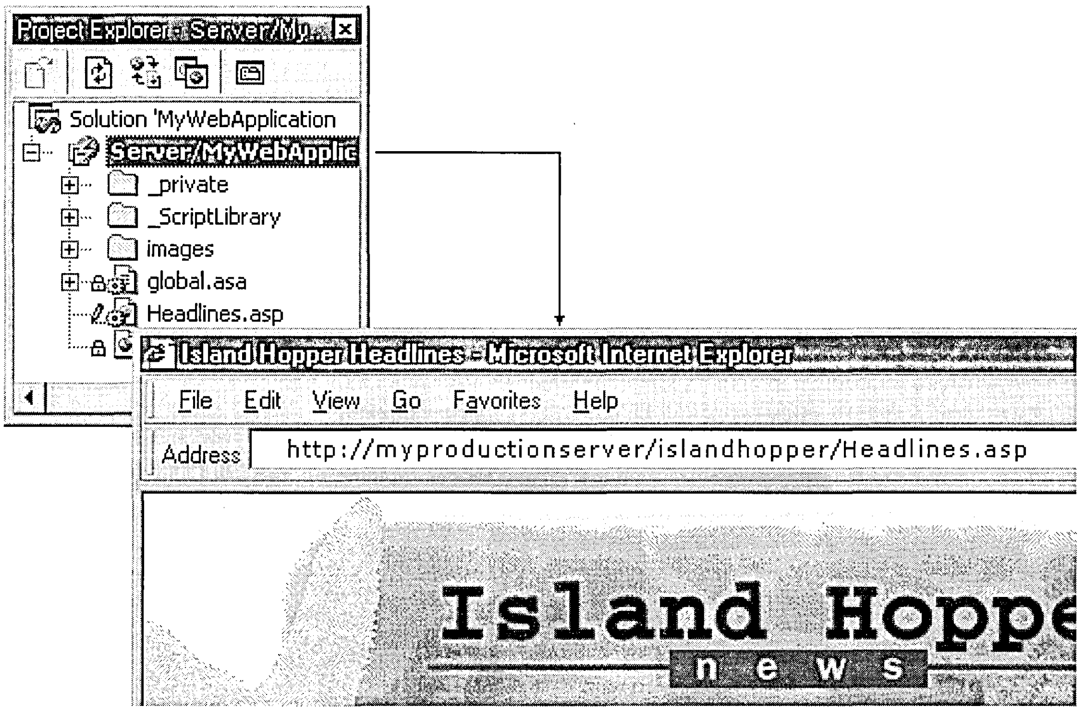
For more information about using Visual SourceSafe, see Chapter 8, "Working with Multiple Developers" and the Visual SourceSafe documentation.

Deploying a Web Application

Deploying a Web application creates a duplicate of your Web application in another location. For example, you might want to make the application available to end users on a production server or add a copy to an archive. The copy of the Web application created when you deploy it is separate from your master Web application.

Once you have finished developing your application, Microsoft Visual InterDev makes deployment easy. You simply specify the URL for the application and Visual InterDev handles the details of copying the application to another server.

Deployment takes your application from a project to a published Web application.


Project Explorer - Server/My...
Solution 'MyWebApplication
Server/MyWebApplic
_private
_ScriptLibrary
images
global.asa
Headlines.asp
Island Hopper Headlines - Microsoft Internet Explorer
File
Edit
View
Go
Favorites
Help
Address
http://myproductionserver/islandhopper/Headlines.asp
Island Hoppe
news

In the figure above, the URL for the deployed application shows that the production Web server is `islehop1`, the application name is `myproductionserver`, and the start page for the application is `Headlines.asp`. For more information about identifying start pages, see the documentation for your Web server, such as Microsoft Internet Information Server.

In Visual InterDev, deployment consists of some or all of these steps:

- Preparing for Deployment
- Deploying to the Web Server
- Verifying Production Server Content

Preparing for Deployment

You can make deployment easier by verifying a few things before you deploy. For more information about previewing and debugging, see "Previewing Pages" in Chapter 2, "Web Basics," and Chapter 26, "Debugging Your Pages."

Deployment Check List

✓ Do the links work?

Sometimes a link can be valid, but it jumps to the wrong page. You might want to verify that all of your links jump to the appropriate page. For more information about verifying links, see "Repairing Links" in Chapter 16, "Maintaining Links" and "Viewing Links for an Item in a Project" in Chapter 15, "Viewing Links for an Item."

✓ Are all files used by the application included in the Web project?

You can include any file that your application references. For example, if you provide documents to download, such as a user's guide or production information, you can include the document file in your Web project. For more information, see "Adding Files" in Chapter 1, "Web Project Management."

✓ Are all files updated on the master Web server?

If you have haven't released all of your working copies, the master Web may not have your latest versions. For more information about updating the master Web server, see "Synchronizing Master and Local Files" in Chapter 10, "Managing Web Projects."

✓ Does your data connection point to the production database?

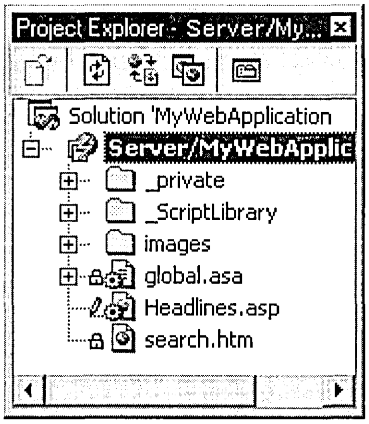
If your Web application will be using a different database than the database you used during development, you need to make sure the data connection in your project points to the appropriate database. For more information about preparing a data connection for deployment, see "Data Access Architecture" in Chapter 18, "Database Concepts," or Chapter 28, "Web Application Deployment"

Deploying to the Web Server

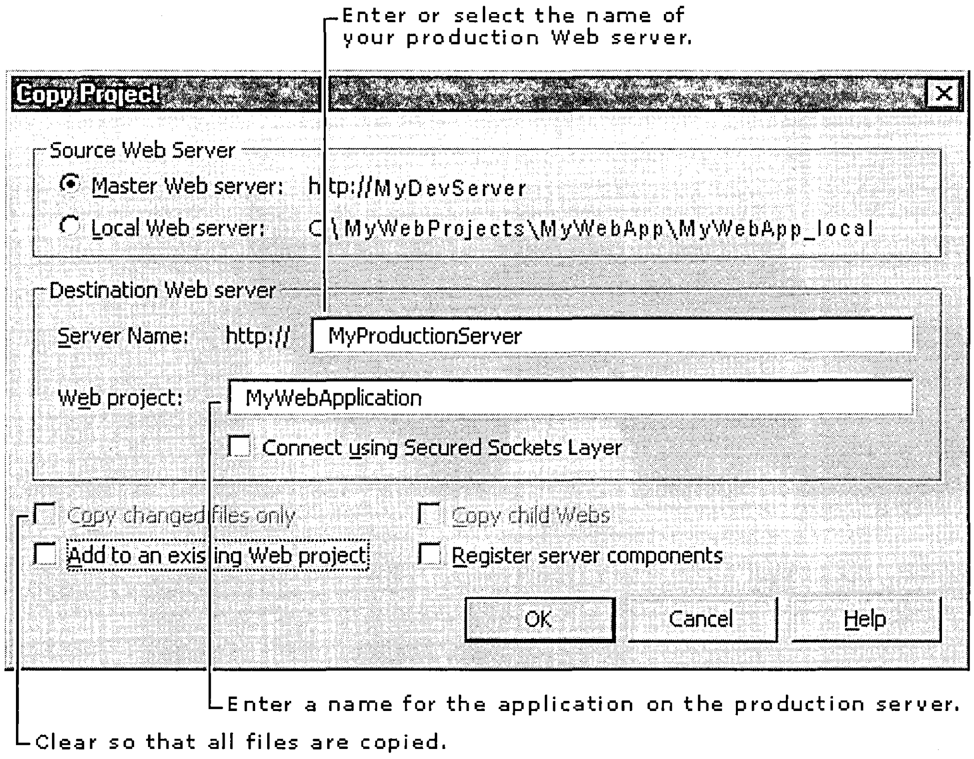
Once you have your project ready for deployment, you can quickly make your application available to your users. This procedure assumes that your production server has Microsoft FrontPage Server Extensions installed. For information about deploying to a server without FrontPage Server Extensions, see Chapter 28, "Web Application Deployment."

To deploy the Web application

1. In the **Project Explorer**, select the project for the Web application you want to deploy.



2. From the **Project** menu, choose **Web Project** and then **Copy Web Application** to display the **Copy Project** dialog box.


Enter or select the name of your production Web server.
Copy Project
Source Web Server
Master Web server: http://MyDevServer
Local Web server: C:\MyWebProjects\MyWebApp\MyWebApp_local
Destination Web server
Server Name: http:// MyProductionServer
Web project: MyWebApplication
Connect using Secured Sockets Layer
Copy changed files only
Copy child Webs
Add to an existing Web project
Register server components
OK
Cancel
Help
Enter a name for the application on the production server.
Clear so that all files are copied.

Visual InterDev creates the application root on the production server specified and copies your Web application. Your Web application is now ready to run in its new location.

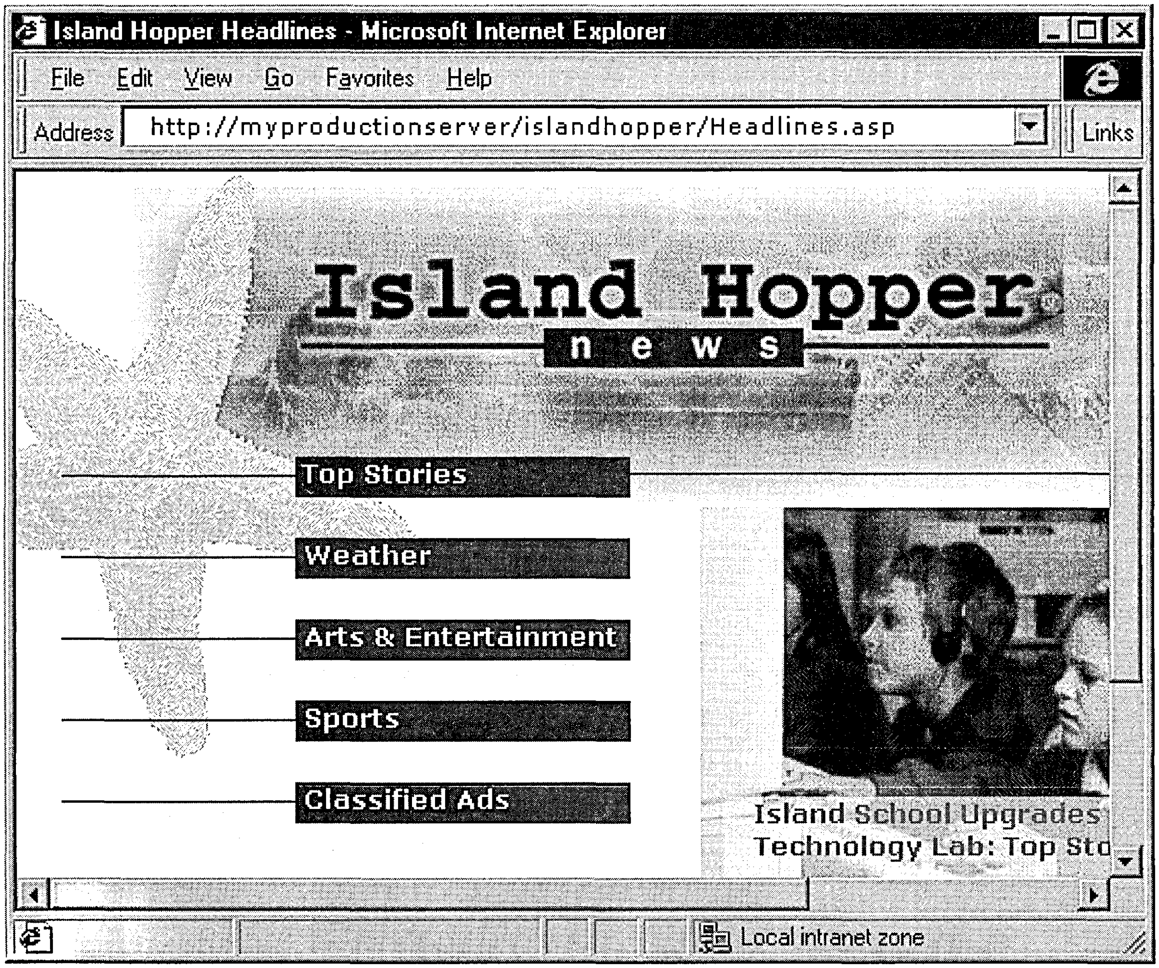
Verifying Production Server Content

You can verify the application by viewing it in your Web browser. You might also have someone else run the application to make sure it is running properly. For best results, test the application on a computer that does not contain a copy of the Web project files.

To verify your deployed application

- In the address box of a Web browser, enter the full URL to the application on the production server.

Your application is ready for your users to browse.


Island Hopper Headlines - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address
http://myproductionserver/islandhopper/Headlines.asp
Links
Island Hopper news
Top Stories
Weather
Arts & Entertainment
Sports
Classified Ads
Island School Upgrades Technology Lab: Top Sto
Local intranet zone

Simplifying Data Entry Pages

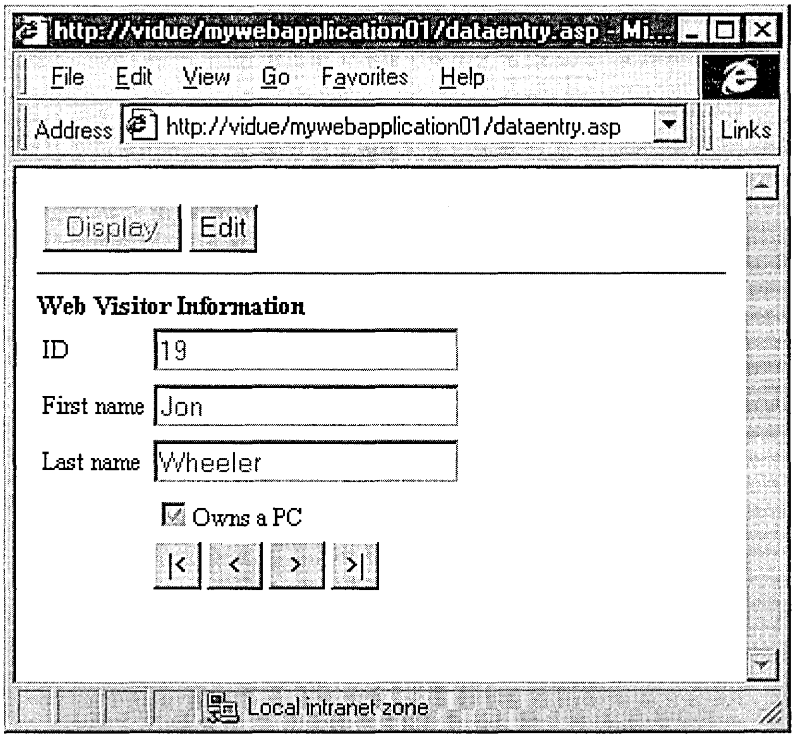
Would you like to maintain a single page that accommodates all aspects of data entry instead of several pages, each with similar sets of labels, text boxes, and buttons? Would you like to concentrate on the functionality and leave the details of scripting and multiple object models to Microsoft Visual InterDev?

Using the new FormManager design-time control to generate run-time script, you can create a versatile page that uses modes for displaying, editing, and adding records to your database.

Instead of scripting your own HTML forms, the modes you specify for the FormManager handle property settings for controls and events on the page. Using modes, you can simplify your Web application design by creating multipurpose pages.

The FormManager control makes creating a data entry page quick and simple, and later on, maintenance is easy. Just open the file and change the FormManager's properties. For more information about this control, see "Creating Event-Driven Forms" in Chapter 3, "Database Basics."

A Data-Bound Form on a Web Page

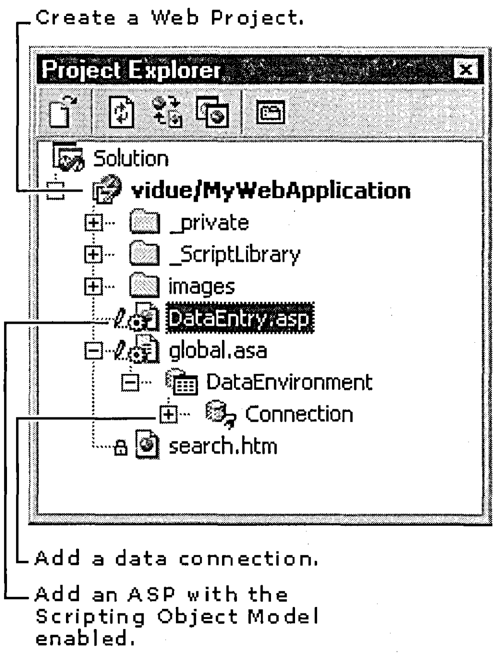

http://vidue/mywebapplication01/dataentry.asp - Mi...
File Edit View Go Favorites Help
Address
http://vidue/mywebapplication01/dataentry.asp
Links
Display
Edit
Web Visitor Information
ID
19
First name
Jon
Last name
Wheeler
Owns a PC
|<
<
>
>|
Local intranet zone

To accomplish the data entry tasks, this page has two modes.

- **Display mode** provides a read-only view of a record. This is also the mode used when the page is opened.
- **Edit mode** allows the user to enter and save changes to the record, add records, and delete records.

To simplify the example, this page displays a few fields for each record. In a real application, you can place as many fields as you need to accomplish your goal. The data-binding concepts are the same regardless of the number of fields.

Before creating a data entry page, be sure you have the following items ready.



You can also create HTML pages using these controls. For more information, see "Changing Target Platforms" and "Creating Forms With Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Once you are ready to begin, you can add data entry controls and modes to your ASP page with the following steps:

- Creating the Data Entry Form
- Simplifying Scripting with Form Modes
- Completing the Data Entry Page

Creating the Data Entry Form

You can create a form on a Web page using data-bound controls. You may have used HTML `<FORM>` and `<INPUT>` tags in the past, but data entry forms are most easily implemented and maintained using Recordset and data-bound controls because you get the same scripting object model in both ASP and Microsoft Internet Explorer 4.0 DHTML.

To add the Recordset control to your page

- From the toolbox, drag the Recordset control to your page. For details, see "Displaying Records" in Chapter 3, "Database Basics."

  Note The Scripting Object Model must be enabled to use the design-time controls.

For more information about the scripting object model, see Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Specifying an Updatable Recordset Control

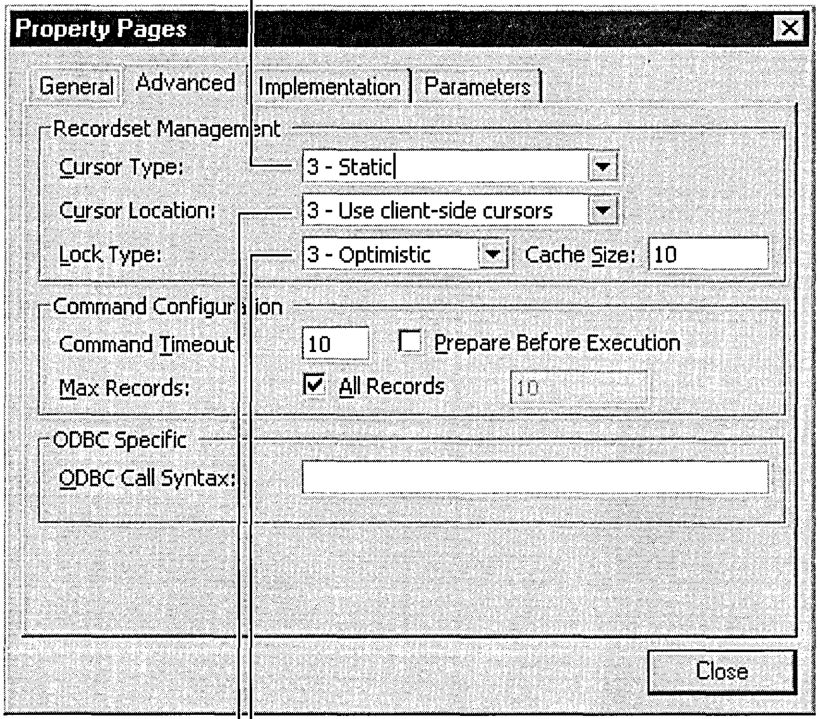
After you drag the Recordset control to the page, choose the source of data and set the lock type to anything other than read-only.

In the example, the Recordset control uses the DEntry table of the Gallery database provided with the Visual InterDev samples available when you install Visual InterDev. The type of lock used is "Optimistic" so that the form can update records.

Also, make sure the recordset is based on a database object that is updatable. For example, views without a primary key or a unique index are not updatable, regardless of the cursor type you select. For more information about updating records and specifying record sets, see Chapter 20, "Modifying Data." For more information about the Gallery, see the Visual InterDev online documentation.

Settings for an Updatable Recordset Control

Set the cursor type.



Property Pages

General | Advanced | Implementation | Parameters

Recordset Management

Cursor Type: 3 - Static

Cursor Location: 3 - Use client-side cursors

Lock Type: 3 - Optimistic   Cache Size: 10

Command Configuration

Command Timeout: 10   [ ] Prepare Before Execution

Max Records: [✓] All Records   10

ODBC Specific

ODBC Call Syntax:

Close

Choose an updateable lock type (any but "read-only").

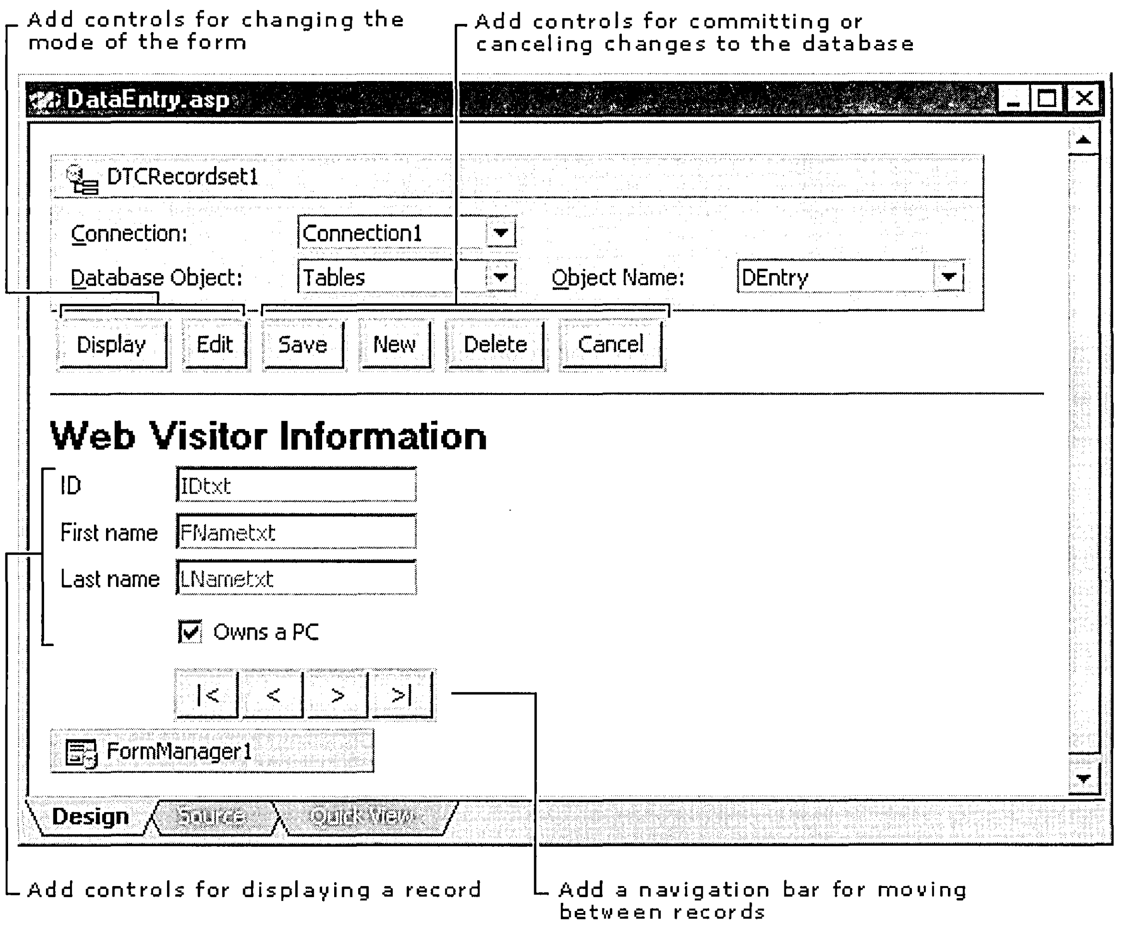
Choose where the cursor is processed.

Adding Controls for Record Display and Mode Transitions

Once you have a Recordset control on your page, you can create the form for displaying the fields you want to show the user and for making changes to the records.

Using the design-time controls from the toolbox, you can quickly specify which recordset fields to appear on the page. The field is a property of the control and makes the data display easy to add and, as long as the fields in the recordset do not change, simple to maintain. For more information about using data-bound controls to display records, see "Displaying Records" in Chapter 3, "Database Basics."

Forms Made with Data-Bound Controls are Easy to Create and Maintain.



To finish the visual design of the page, add and format text to provide a title for the form. Also, create the row of buttons by adding a table to the page and dragging Button controls into the cells of the table.

For the record display, drag Label, Textbox, and RecordSetNavBar controls to the page. In the example, the textbox controls and the Checkbox control are bound to the recordset using the property pages to specify the field that the control displays. For a list of all of the initial property settings for the controls on this page, see “Completing the Data Entry Page” later in this chapter.

Simplifying Scripting with Form Modes

Instead of creating and maintaining several pages with variations of the same control set, the FormManager consolidates it all in a single page that can handle multiple tasks and is easy to maintain.

Using the FormManger control, you can specify modes for your form without scripting the controls and changes. The modes handle changes to the controls and updates to the records.

In addition, you no longer need to scroll through page after page to find the method or event you want to change. The FormManager Properties Dialog Box summarizes all of the property changes and event handling for the page into two tabs.

Using the FormManager, you can specify modes for your form without scripting the controls and changes. Of course, if you have user-defined functions, you can reference them in the FormManager control as well. For example, you might have a validation function that you want to use in the form. You can use the FormManager to call the function and validate data whenever the recordset is updated.

Specifying modes covers three main concepts:

- **Identifying each mode and specifying the property settings and methods for the controls while the mode is active.** In the data entry form, the modes specify when the Save, Add, and Cancel buttons are hidden. In Edit mode, the Save and Cancel button are shown, but the Add button is hidden. In Insert mode, the Add button is shown and the Save button is hidden.
- **Specifying the transition events that move between the modes or trigger specific actions.** For example, clicking the Edit button transitions the form from Display mode into Edit mode.
- **Adding the actions that occur after the transition event is triggered but before the transition is complete.** For example, clicking the Save button not only switches from Edit mode to Display mode, but also updates the recordset with changes the user made in the form.

To add Display and Edit modes to the data entry form, complete the following steps.

To specify modes

1. Open or create an ASP page that has controls for displaying records and buttons for switching modes.

    Tip Before adding modes, make sure you have all of the controls you need on the form. For example, you may need to add buttons, such as an Edit button for moving to the Edit mode and a Save button for triggering transition events in that mode.

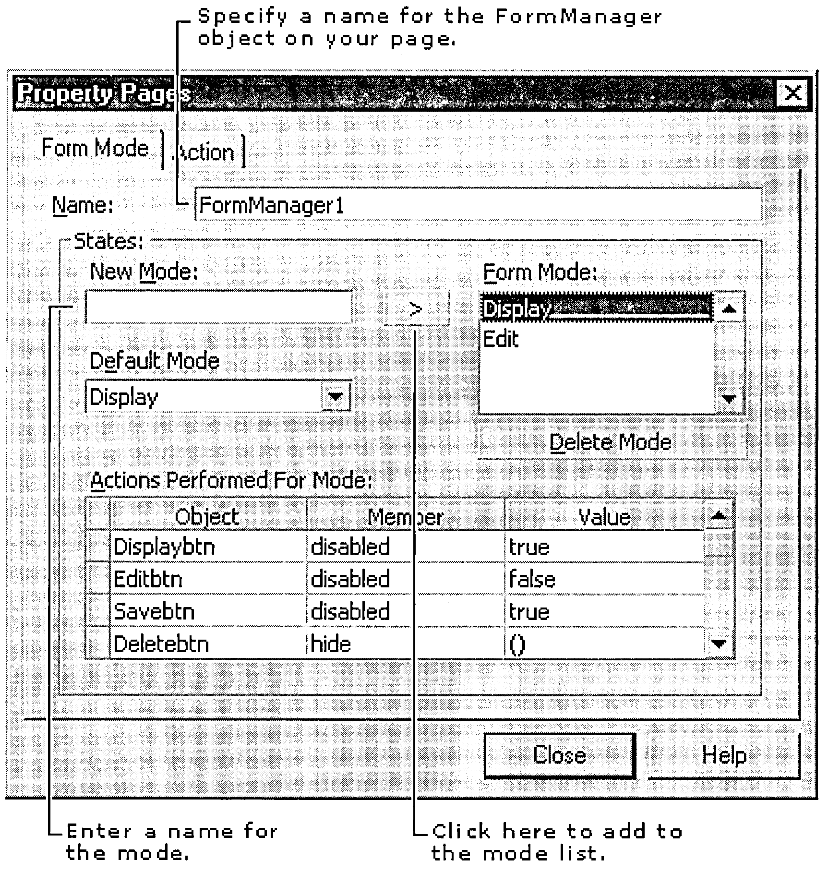
    If you decide to add more controls you can do that later, but adding controls before adding the modes makes them readily available to the FormManager control. This also applies to functions and methods you want to script in the page.

2. From the toolbox, add a **FormManager** control. Right-click the **FormManager** in the editor to display its property pages.
3. Identify the modes you want and the control settings for each.

A plan for two modes and their control settings

Display mode	Edit mode
Display controls are read-only	Display controls are write-enabled
Edit button is enabled	Display button is enabled
Save and Cancel buttons are hidden	Save and Cancel buttons are visible and enabled

4. On the **Form Mode** tab, add a new mode for each mode you have identified.



Specify a name for the FormManager object on your page.

Name: FormManager1

Default Mode: Display

Object	Member	Value
Displaybtn	disabled	true
Editbtn	disabled	false
Savebtn	disabled	true
Deletebtn	hide	()

Enter a name for the mode.

Click here to add to the mode list.

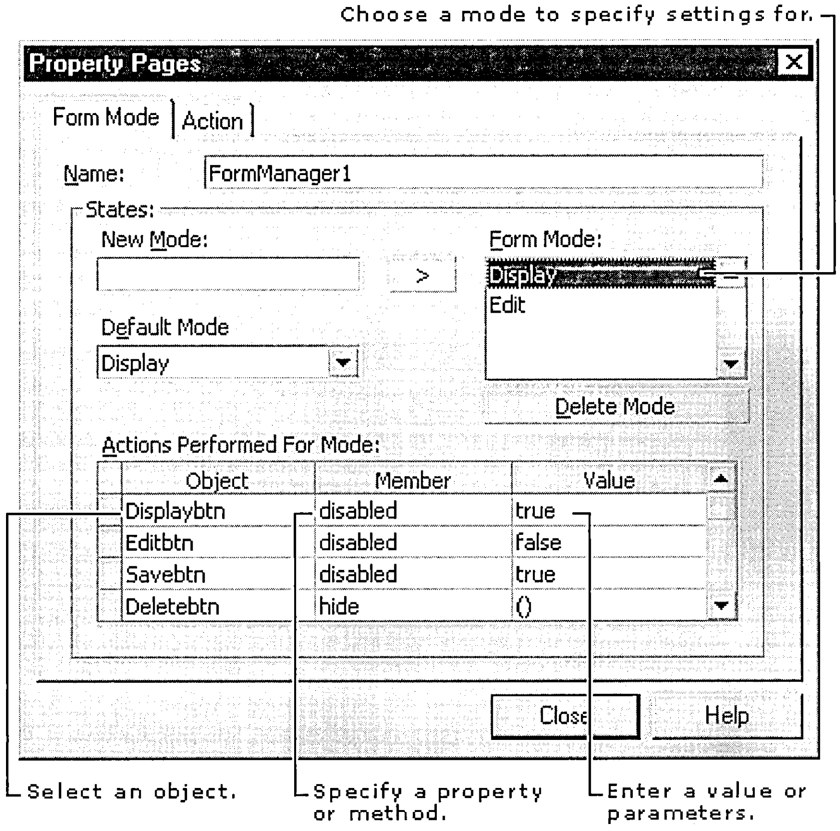
5. For each mode, specify the property settings and method calls for the display controls and the form mode buttons on the **Form Mode** tab.

   Note For each property you want to set for a control, add a line in the Property/Value pairs grid. For example, if you want the button, btnNew, to be both visible and disabled for the mode, in one line set the disabled property to true and in another line, specify the Show() method. If you want to call a function you created, you can leave the Object column blank and enter the function's name in the Member column and any parameters in the Value column. Be sure to include parenthesis around the parameters.

   Tip To ensure proper form behavior, set the properties and methods for each control in each mode. The Web environment is stateless so settings for controls may remain into the next mode unexpectedly. For example, specify the Hide() method for controls that should not be visible during that mode and Show() for controls that should be visible. If you don't specify the Hide() or Show() methods for each visual element in each mode, the buttons or textboxes may be visible or hidden depending on a setting specified for those controls in the previous mode.

   For a complete list of the mode settings for this form, see "Completing the Data Entry Page" later in this chapter.

Specify the Property Settings and Method Calls for the Mode.


Choose a mode to specify settings for.
Property Pages
Form Mode
Action
New Mode:
>
Form Mode:
Delete Mode
Close
Help
Select an object.
Specify a property or method.
Enter a value or parameters.

Name: FormManager1

States:

Default Mode: Display

Form Mode: Display, Edit

Actions Performed For Mode:

Object	Member	Value
Displaybtn	disabled	true
Editbtn	disabled	false
Savebtn	disabled	true
Deletebtn	hide	()

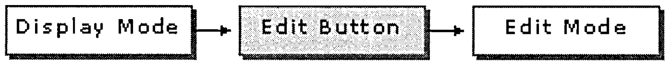
Note Be sure that the disabled property is properly set. If you want to enable the control, its disabled property needs to be "false." These values are case-sensitive. It is easy to inadvertently set it to the opposite.

Also, to make a control visible or hidden, use the Show() and Hide() methods. The server will translate them into the appropriate tags and attributes when it sends the HTML to the Web browser.

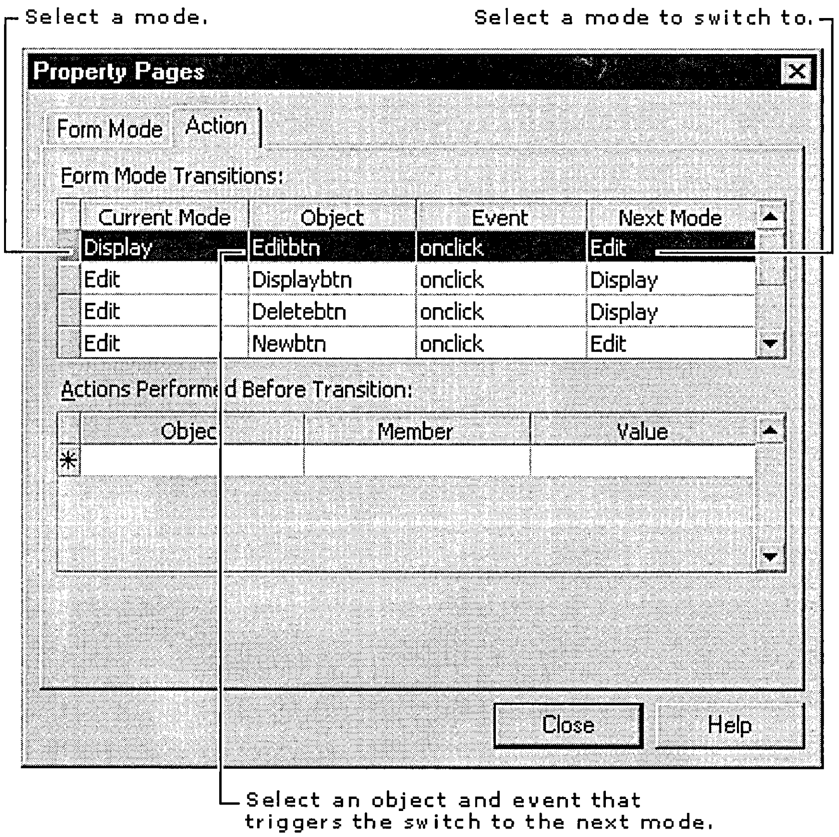
6. On the **Action** tab, specify transition events for each mode button on the form. For a complete list of the action settings for this form, see "Completing the Data Entry Page" later in this chapter.

   For example, when the Edit button is clicked, the form switches from Display mode to Edit mode. Without additional scripting, the mode determines the changes the controls' properties as specified in the Form Mode tab.

The Edit button triggers the switch to the next mode.



While in Display mode, the Edit button specifies only that a new mode is active. Additional actions are not required.


Select a mode.
Select a mode to switch to.
Property Pages
Form Mode
Action
Close
Help
Select an object and event that triggers the switch to the next mode.

Form Mode Transitions:

Current Mode	Object	Event	Next Mode
Display	Editbtn	onclick	Edit
Edit	Displaybtn	onclick	Display
Edit	Deletebtn	onclick	Display
Edit	Newbtn	onclick	Edit

Actions Performed Before Transition:

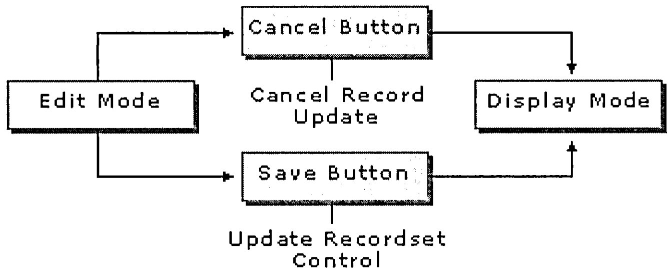
Object	Member	Value

Some transition events require actions in order to complete the entire task intended by the mode.

7. As needed, identify and specify actions to complete between the event and the next mode.

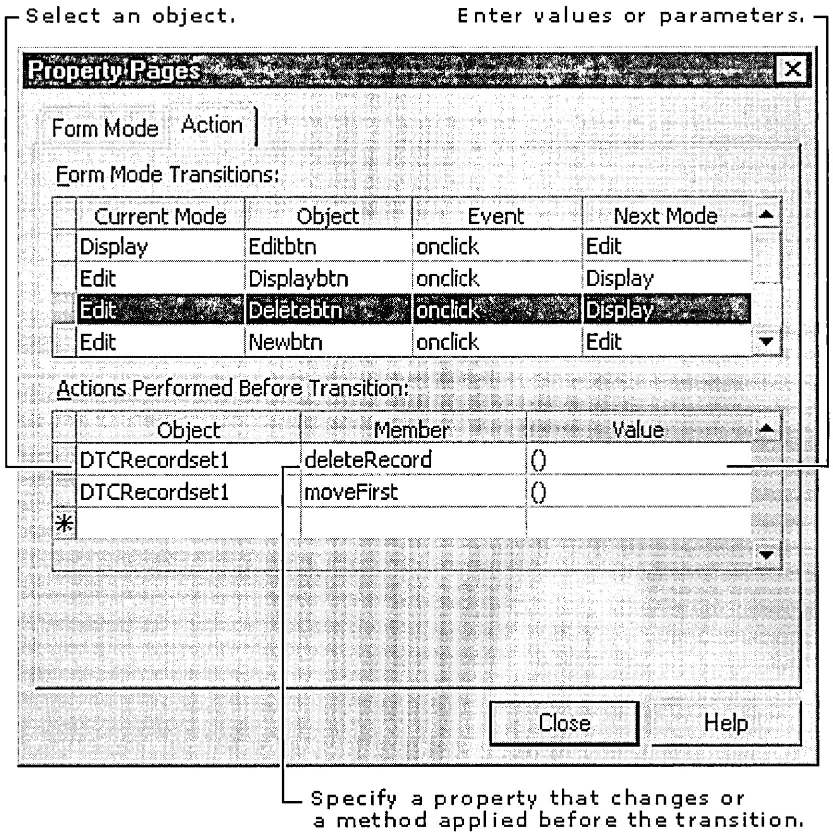
For example, in Edit mode when the Cancel button is clicked, the changes should be discarded before switching to Display mode. When the Save button is clicked, the recordset should be updated before switching to display mode.

A plan for the transitions and actions between modes



On the **Action** tab, these actions are specified as shown in the following figure.

The actions occur after the event happens and just before the mode switches.


Select an object.
Enter values or parameters.
Property Pages
Form Mode
Action
Close
Help
Specify a property that changes or a method applied before the transition.

Form Mode Transitions:

Current Mode	Object	Event	Next Mode
Display	Editbtn	onclick	Edit
Edit	Displaybtn	onclick	Display
Edit	Deletebtn	onclick	Display
Edit	Newbtn	onclick	Edit

Actions Performed Before Transition:

Object	Member	Value
DTCRecordset1	deleteRecord	()
DTCRecordset1	moveFirst	()

8. Choose **Close**.

Note If you add more controls and/or user-defined functions to the page, be sure to incorporate them into the modes using the FormManager control.

Completing the Data Entry Page

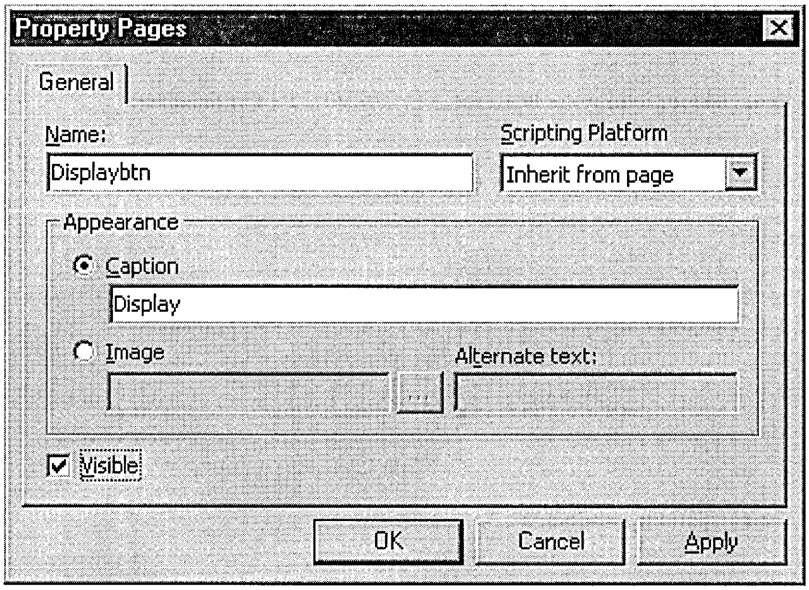
Now that you understand the basic use of the FormManager control, you can finish specifying the modes and actions for the data entry page. You can use the tables below for additional settings you might make to complete the data entry page.

- Initial Property Settings for the Controls on the Form
- Completing the Mode Settings for Controls
- Mode Transitions and Actions

Initial Property Settings for the Controls

You can set the initial properties for the controls on the form on the General tab of the property pages for the control.

Example of Property Setting for a Button



Name: Displaybtn

Scripting Platform: Inherit from page

Appearance

(●) Caption: Display

( ) Image

Alternate text:

[✓] Visible

The following table shows the property settings for the controls on the form before you add the modes.

Controls for the record	Initial settings
Recordset	Connection to the Gallery database used with the Visual InterDev samples. Database object = DEntry table Lock type = Optimistic or Batch optimistic
Label1 Inlbl	Visible
Text box1 Idtxt	Disabled (read-only), visible
Label2 Fnamelbl	Visible
Text box 2 Fnametxt	Disabled (read-only), visible
Label 3 Lnamelbl	Visible
Text box 3 Lnametxt	Disabled (read-only), visible
Label 4 PCchk	Visible
RecordSetNavBar	Visible, set to recordsetdtc1

Controls for form modes	Initial settings

Display Button	Name = Displaybtn
	Caption = Display
	Visible = true
	Disabled = true

Edit Button	Name = Editbtn
	Caption = Edit
	Visible = true
	Disabled = false

Save Button	Name = Savebtn
	Caption = Save Record
	Visible = false
	Disabled = false

New Button	Name = Newbtn
	Caption = New
	Visible = true
	Disabled = false

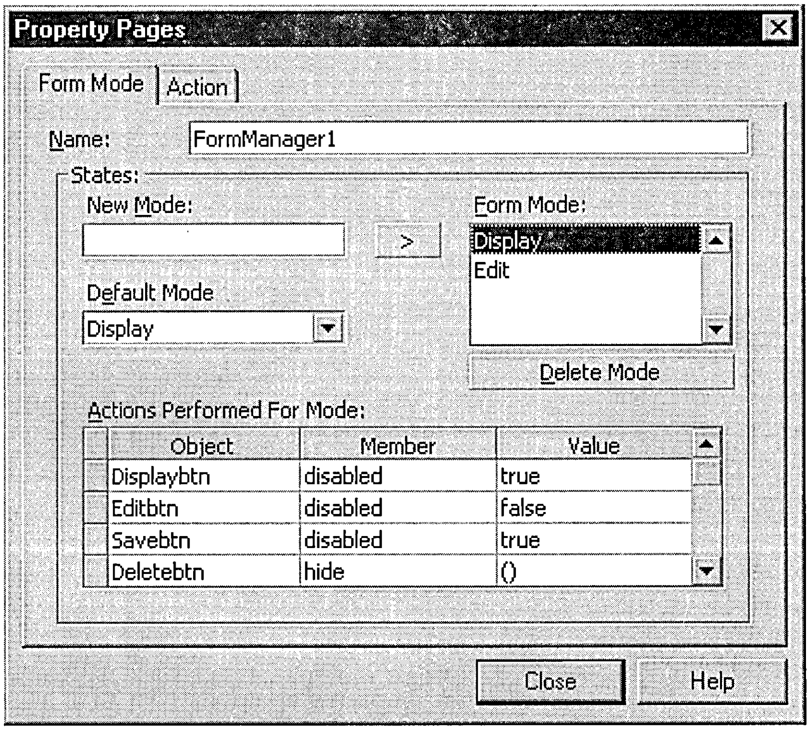
Delete Button	Name = Deletebtn
	Caption = Delete
	Visible = true
	Disabled = false

Cancel Button	Name = Cancelbtn
	Caption = Cancel
	Visible = false
	Disabled = false

Completing the Mode Settings for Controls

You can complete the mode settings for the data entry form by adding the Property/Value pairs into the Form Mode tab.

Use the Form Mode tab of the FormManager control to specify a mode.



Name: FormManager1

Form Mode: Display, Edit

Default Mode: Display

Actions Performed For Mode:

Object	Member	Value
Displaybtn	disabled	true
Editbtn	disabled	false
Savebtn	disabled	true
Deletebtn	hide	()

This tables shows the entire set of Property/Value pairs needed to complete the data entry page.

Record display controls	Display state	Edit state
Label1	Show( )	Show( )

Text box1	Show( )
	Disabled = true
		Show( )
		Disabled = false

Label2	Show( )	Show( )

Text box 2	Show( )
	Disabled = true
		Show( )
		Disabled = false

Label 3	Show( )	Show( )

(continued)

Record display controls	Display state	Edit state

Text box 3	Show( )
	Disabled = true
		Show( )
		Disabled = false

Label 4	Show( )	Show( )

Text box 4	Show( )
	Disabled = true
		Show( )
		Disabled = false

RecordSetNavBar	Show( )	Hide( )
Mode event buttons	**Display mode**	**Edit mode**

Display Button	Show( )
	Disabled = true
		Show( )
		Disabled = false

Edit Button	Show( )
	Disabled = false
		Show( )
		Disabled = true

Mode action buttons	**Display mode**	**Edit mode**

Save Button	Hide( )
	Disabled = true
		Show( )
		Disabled = false

Insert Button	Hide( )
	Disabled = true
		Show( )
		Disabled = false

Add Button	Hide( )
	Disabled = true
		Show( )
		Disabled = false

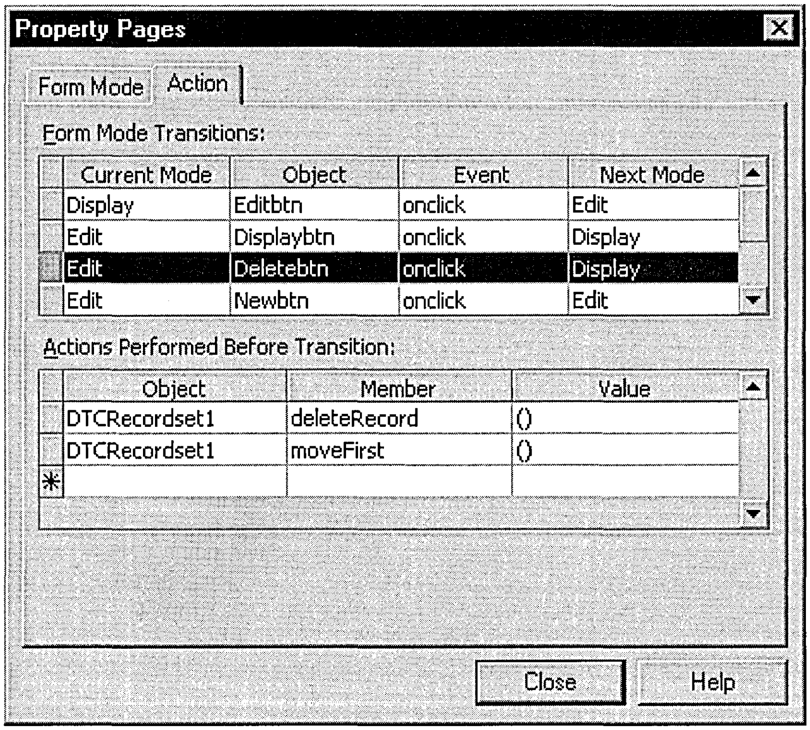
Cancel Button	Hide( )
	Disabled = true
		Show( )
		Disabled = false

Delete Button	Hide( )
	Disabled = true
		Show( )
		Disabled = false

Mode Transitions and Actions

You can complete the transition and action settings for the data entry form by identifying the events you want to use to switch between modes and the actions that occur before the switch is completed.

Use the Action tab of the FormManager control to specify transitions and actions.



Form Mode Transitions:

Current Mode	Object	Event	Next Mode
Display	Editbtn	onclick	Edit
Edit	Displaybtn	onclick	Display
Edit	Deletebtn	onclick	Display
Edit	Newbtn	onclick	Edit

Actions Performed Before Transition:

Object	Member	Value
DTCRecordset1	deleteRecord	()
DTCRecordset1	moveFirst	()
*		

This table shows the settings for both panels in the Action tab of the FormManager control. Remember, the mode itself specifies properties and methods on the display controls. The following settings specify events that trigger the mode and the actions that must be completed before entering the next mode.

Current mode	Object and event	Next mode	Button actions
Display	btnEdit	Edit	None
Edit	btnDisplay	Display	None
Edit	btnSave	Display	Recordset.Update( )

Edit	btnCancel	Display	Recordset.UpdateCancel( )
			Recordset.MoveNext( )

Recordset.AddRecord( )
Recordset.MoveLast( )
	btnNew	Display	

Recordset.DeleteRecord( )
Recordset.MoveFirst( )
	btnDelete	Display	

PART 2

Creating Web Projects

Part 2 provides a detailed look at Web applications and Web projects including information about working modes, source code control, security, and the development cycle.

Chapter 6 Web Project Concepts

Chapter 6 deals with some of the fundamental Web project concepts that underlie Visual InterDev Web applications, including "Project Architecture," "Web File Processing," "The Web Application Development Cycle," "Source Control," and "Security."

Chapter 7 Working Locally

In Visual InterDev, you have two options when working online. You can work directly with the master Web files in master mode, or you can develop your own working model of the Web application or its parts in local mode. This chapter covers how to work locally and how to integrate with the master Web application when you are working locally.

Chapter 8 Working with Multiple Developers

Whether you are working on a team or by yourself, you can track and save your changes to files easily with Microsoft Visual SourceSafe. This chapter explains using source control with your Web application.

Chapter 9 Adding Security

This chapter covers security-related issues such as "Adding Security Pages," "Setting Web Application Permissions," and "Connecting to a Proxy Server."

Chapter 10 Managing Web Projects

As you work on your Web application files, you might want to change the structure of your project or make copies to other servers. This chapter discusses "Synchronizing Master and Local Files," "Copying Web Projects," and "Reorganizing Project Structure."

CHAPTER 6

Web Project Concepts

Project Architecture

The project architecture in Microsoft Visual InterDev is designed to provide you with everything you need to create robust Web sites and Web applications. By working through a project you can concentrate on the details unique to your Web application and leave the issues of file and source management to Visual InterDev.

The following sections provide descriptions of projects, Web applications, and system components.

- The Web Project and Two Web Applications
- Local Mode and Master Mode
- The Project and Web Applications

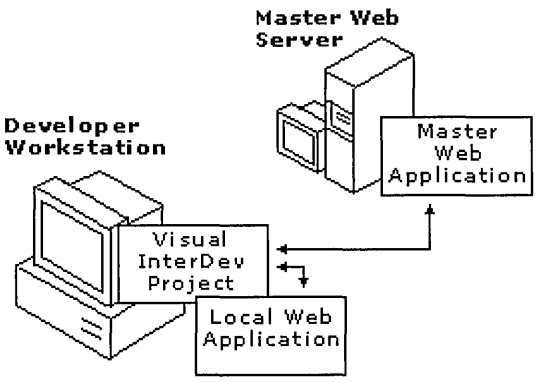
The Web Project and Two Web Applications

A Web application contains the files that hold your Web content and functionality — such as .htm, .asp, and image files — for all or a portion of your Web site. Through the Visual InterDev Web project, you can identify and manipulate your Web application files. The project is not part of the Web application file set. Instead, it is a file used by Visual InterDev to point to the files associated with your Web application.

Each developer creates a Web project in order to work on the Web application. The Web project provides a graphical representation of the Web application files in the Project Explorer. The Web project file remains on the developer's local computer and is not part of the Web application. When you deploy a Web application, the project file does not get copied with it, only the files containing your Web content and functionality.

When you create a Web project, you actually work with two separate Web applications. One is the master set of files on the master Web server. A second local set of files exists on your computer. When you make changes to the Web files in the project, those changes are made directly on the local set of files.

A Web project points to two Web applications



The project architecture provides you with several benefits. Visual InterDev uses the Web project with its pointers to local and master versions of the Web application to carry out your commands and to maintain the two file sets.

Project Explorer for managing Web files The Project Explorer gives you a single window for managing the Web files in each of the Web applications. You can use the Project Explorer to choose files to work with and to perform a variety of tasks.

The Project Explorer provides information about the files, including the status of each of the files. The icons next to the file names communicate information about the specific files.

Isolated development This project architecture allows you to work independently from the master set of files so that others using the master are not affected by the changes you make to the files until you are finished working on the files. The project does this by managing two versions of the Web application simultaneously. One version is the master version of the Web files and can be available to other Web developers or to your end-users. The second version resides locally for isolated development, testing, and debugging on your local machine.

For example, in the figure above, you can see that one directory containing the Web files is located on a server. This is the master Web application. The second set of the files is located on your drive C. This is the local Web application. Through the Project Explorer, a Web project facilitates the management of these files. You can work on the local set of files, and when you are ready, update the master set of files.

Since the project is a file that is independent from the Web application, you can delete the project from your local computer without affecting the master Web application files. You can also use Visual InterDev to delete both the project and the Web application if you want.

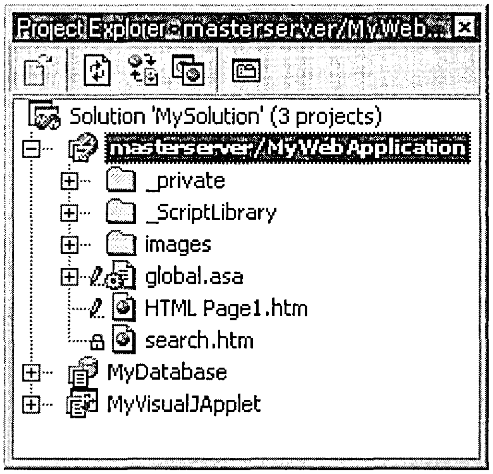
Easy access for multiple developers Using projects to manage the Web application files allows many developers to work on the Web application simultaneously. Each developer creates a project that points to the master application. Using their own projects, they can work independently on their local machines.

Projects in a Solution

Each Visual InterDev Web project you create in the Project Explorer is also part of a solution. A solution can contain several projects of the same type or of different types. For example, you might have a solution that contains one project consisting of a Web application that you make available to the general public and a separate project containing Web pages designed for a restricted group of users. Although both projects are in a single solution, each points to a different Web application that may reside on different Web servers.

Your solution can also include other types of projects, such as a database project or a project created in Microsoft Visual J++. The .sln file stores the information pointing to the projects that are in the solution. When you open a solution file, the projects within the solution are also opened in the Project Explorer. Like the project file, this solution file is not considered a Web application and does not exist in the master Web application.

One solution can contain several projects


Project Explorer - masterserver/MyWeb...
Solution 'MySolution' (3 projects)
masterserver/MyWebApplication
_private
_ScriptLibrary
images
global.asa
HTML Page1.htm
search.htm
MyDatabase
MyVisualJApplet

If you create a Web project and do not have a solution open, one is created for you.

Local Mode and Master Mode

A project always maintains two copies of the Web application files, the master files and the local files. When you make changes to files, those changes are made on the local copy of the Web application files. The project's working mode determines when changes you make to your local files are sent to the master Web application.

A project can work in one of two modes: local and master. Typically, you pick a mode when you create a project and stay with that mode throughout your development cycle.

Local mode Changes you make are saved only to your local version of the file set. The master Web application is only updated with your changes when you specifically choose to do so.

Master mode Changes you make to your local version of a file are saved to the local version and the master version at the same time. Two sets of files are still maintained, but they are updated simultaneously.

> **Note** In addition to the two project modes, you can also work offline if you do not want to connect to a Web server. For more information, see "Working Offline," later in this chapter.

For more information about using project modes, see "Web File Processing," and "The Web Application Development Cycle," later in this chapter; or Chapter 7, "Working Locally."

The Project and Web Applications

You can find the project folder and its local Web application folder on the developer workstation. The developer workstation generally contains the "local" copy of the Web project files. The master Web server folder is located on the master Web server containing the files available to other developers or, possibly, the end user.

Local Folders and Files

When you create a new Web project, you can reference either an existing Web application on the master Web server or you can create a new Web application. When you create a new project, Visual InterDev creates the following on your workstation:

- **A new folder** contains a local copy of the Web application. This folder includes a subfolder for the local Web application that is used by the Web server on the workstation when you preview and debug the pages.
- **Web project definition files** store information Visual InterDev uses to maintain and manage the project. For example, you have the solution definition file (.sln file), the project definition file (.vip file), and the Visual InterDev Cache file (.vic file).
- **A newly created project** appears in the Project Explorer. If you choose to create a new Web application, the files in the project reflect the files you specified in the wizard, such as a search page (search.htm). If you choose to reference an existing Web application, your new project mirrors the structure and contents of that Web application.
- **A new virtual root** is specified on the local Web server if a Web server is present.

Master Web Server Files

If you choose to create a new Web application at the same time that you create a new project, Visual InterDev creates the following on the master Web server:

- **A master Web application folder** contains the master copies of the Web application. Typically, this folder is a direct subfolder of the Web server's root directory and has the same name as the Web application. This folder also includes several hidden folders that store the metadata about your files. If you are running Microsoft Internet Information Server, you also get a virtual root. For more information about virtual directories, see the Internet Information Server documentation.
- **A new application root** points to the Web application folder and has permissions set to allow pages to be read and script to be executed.
- **An empty startup page (Global.asa file) and a search page (Search.htm file)** might appear in the root of the Web application folder, depending on the options you selected. The Global.asa file allows you to control the Web application in a similar manner as the main function or program in a traditional application. It is executed when a visitor opens a page of the Web application and controls a variety of events for the Web application.

Examples of File Names and Locations

To provide an example of where Visual InterDev places Web project files after you create a new project, the following table contains sample paths for a typical set of files.

File type	Project folder[1]	Local Web application subfolder[1]	Web server folder[2]
Solution definition (.sln)	\personal\projectname	No	No
Web project definition (.vip)	\personal\projectname	No	No
Web application file (Global.asa)		\personal\projectname\projectname_local\global.asa	\Default Web Site\Web application name
Web application pages (.htm, .asp)		\personal\projectname\projectname_local\Web application pages	\Default Web Site\application name

[1] For Windows NT, the default directory for Web projects is \winnt\Profiles\<username>\Personal.
[2] For Internet Information Server, the default root is Default Web Site. Your Web server may be set up differently.

Web File Processing

How and when Web files are processed depends on a variety of factors, including the stage of the development cycle you are in, the project mode you are working in, and the needs of your Web application.

The following sections describe system components that affect your files and what happens to your files at run time, design time, and test time.

- System Components Affecting Your Web Application
- Web Server Connections
- Run-Time Interaction of Components
- Local Mode Interaction
- Master Mode Interaction
- Offline Projects

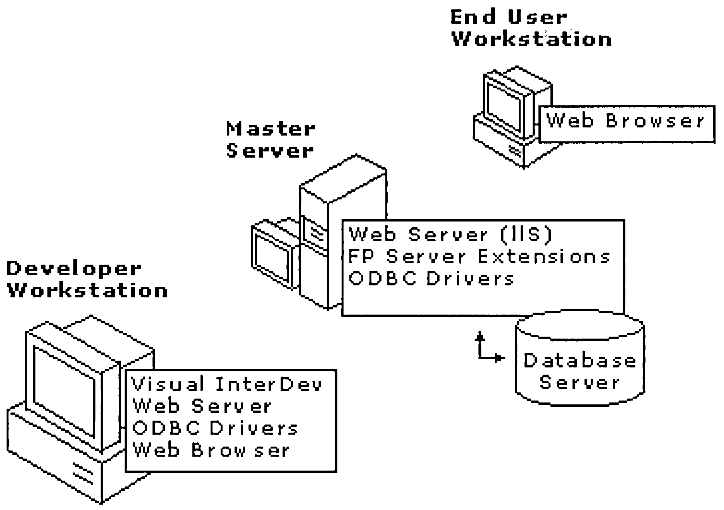
System Components Affecting Your Web Application

Each Web application requires a combination of hardware and software pieces that are needed for developing a Web application that incorporates a database. The following table provides a brief description of each.

Component	Function
Developer workstation	Creating a Web project, storing the project information, and connecting to other components
Database server	Storing database definitions and data used by the Web application, if required
Web server with FrontPage Extensions	Storing the master Web application, sending your Web application pages to the client, and processing server script and data requests including connections to the database server
Web browser	Testing and previewing the pages and processing client script

These components may reside on a single computer, or they can exist on two or more computers in a network.

Typical set of Web system components



Web Server Connections

The detailed interaction of the components depends on the mode of your project. The following section shows the difference between the interaction of system components for designing and testing in local and master modes.

Typically the system components interact using HTTP, except for the database components which are likely to use a Local Area Network (LAN) connection or Wide Area Network (WAN).

Visual InterDev communicates with the master Web server via HTTP. Communicating via HTTP allows you to develop Web applications in a distributed environment where the master Web server and local development machine might only be connected via the Internet. An HTTP connection also allows you to connect to the master Web server through firewalls and proxy servers that protect you from unauthorized access to your local development site. For information on configuring Visual InterDev for a proxy server, see “Connecting to a Proxy Server” in Chapter 9, “Adding Security.”

If your Web application uses a database server, you need to add an ODBC database connection to your Web project to take advantage of the Microsoft Visual Database Tools. For more information about creating a data connection and using databases with your Web application, see Chapter 18, “Database Concepts.”

After you have established a database connection, you can use the Visual Database Tools to manipulate the database.

Run-Time Interaction of Components

The simplest scenario for understanding how the components work together occurs at run time. At run time, the Web server is the central component in the Web application. It receives requests for Web pages from the user's Web browser, sends any requests for data or database commands to the database server, receives data from the database server, processes server script, and sends pages and data back to the user's browser.

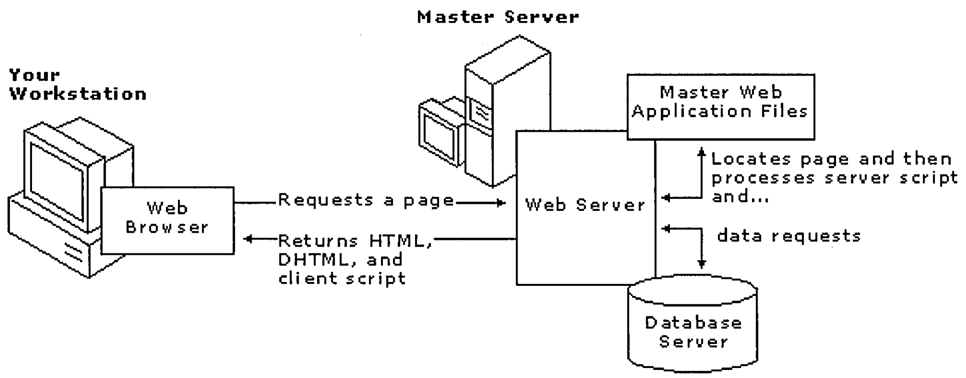
When your Web application is ready for others to see, you can make the Web pages available using the master Web server or you can deploy the Web application to a production Web server.

A production server that is separate from your master Web server is recommended because it provides an added layer of protection. For example, if you have multiple developers updating the master server with their local files, it is possible that the changes made by one developer can conflict with changes made by another, and "break" parts of the Web application. With a production server, you can test the master server and only update the application shown to the public when the "final" working version of the Web application is available.

If you plan to use a production Web server, Visual InterDev makes it easy to copy the Web application from the master Web server to a production Web server where end users can browse the Web application. For more information, see Chapter 28, "Web Application Deployment."

The following diagram shows the interaction between the system components at run time. Notice that the project is not used at run time. The project is used only at design time and during testing and debugging.

Interaction of system components at run time


Master Server
Your Workstation
Master Web Application Files
Web Browser
Requests a page
Web Server
Locates page and then processes server script and...
data requests
Returns HTML, DHTML, and client script
Database Server

Local Mode Interaction

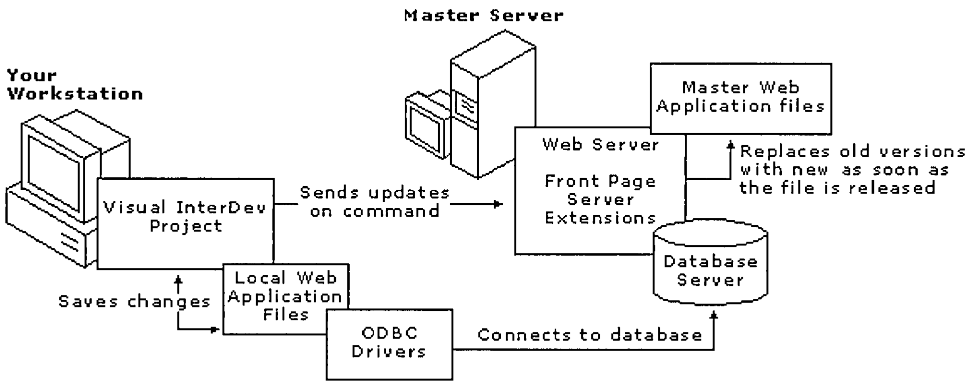
Local mode is designed to provide you with an isolated version of the Web application files for development. The following sections show how the various components work to isolate your work yet quickly send your changes to the master version at your command.

Design-Time Interaction of Components

The development of Web application files takes place on local copies, which are updated on the master server only when you explicitly update the master server with the local copies.

If someone has made changes to the file while you were working locally, you can review the differences and merge the files when you update the master server. The Visual InterDev project plays a key role in the development of the Web application files. The project gives you a view of both the local and master Web application files.

Design-time interaction of components in local mode



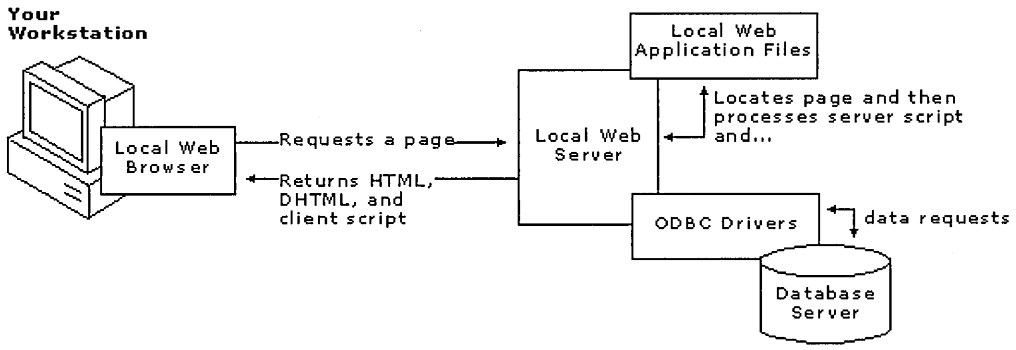
Test-Time Interaction of Components

In local mode, the Web files you test are the local copies stored in the project folder on your computer. You can test the files by using the Quick View tab of the editor or by previewing the Web application in your Web browser.

The Quick View tab does not use your local Web server, if your workstation has one, and will not process the server script in your Web application.

Previewing in your Web browser, you can test your client script. If your workstation has a Web server, you can also test your server script and ASPs in your Web browser. The following figure shows the interaction between components while you're testing your Web application in local mode.

Test-time interaction of components in local mode


Your Workstation
Local Web Browser
Requests a page
Returns HTML, DHTML, and client script
Local Web Server
Local Web Application Files
Locates page and then processes server script and...
ODBC Drivers
data requests
Database Server

For more information about testing your Web application, see "The Web Application Development Cycle," later in this chapter; "Link Verification" in Chapter 11, "Site Design"; and "The Script Debugging Process" in Chapter 23, "Scripting Concepts."

Master Mode Interaction

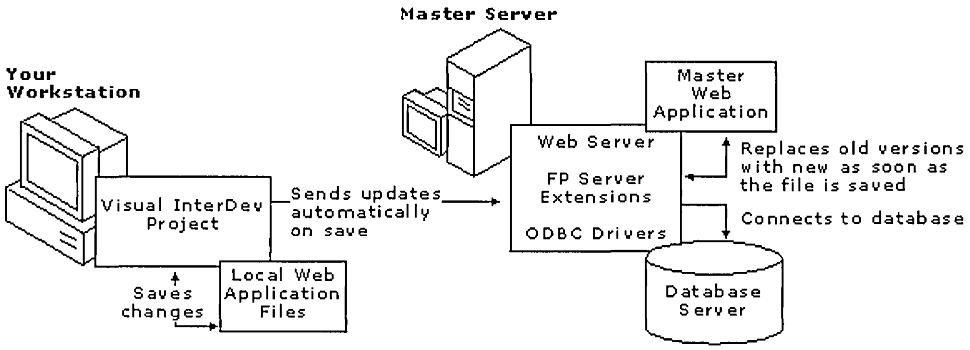
Master mode propagates the changes you save to files automatically to the master Web application. This can affect the results of other developers working on the Web application. Master mode is useful if you are the only person who modifies the Web files, but is not recommended in multi-developer situations because intermediate changes to files may affect others working on the application.

Design-Time Interaction of Components

The design-time interaction between the system components depends on the tasks you are performing. In master mode, development of files takes place on local copies but changes are updated on the master server immediately when you save changes to the files.

If someone has made changes to the file, you are prompted to review the differences and merge the files when you release your working copy. The Visual InterDev project plays a key role in the development of the Web application files.

Design-time interaction of components in master mode

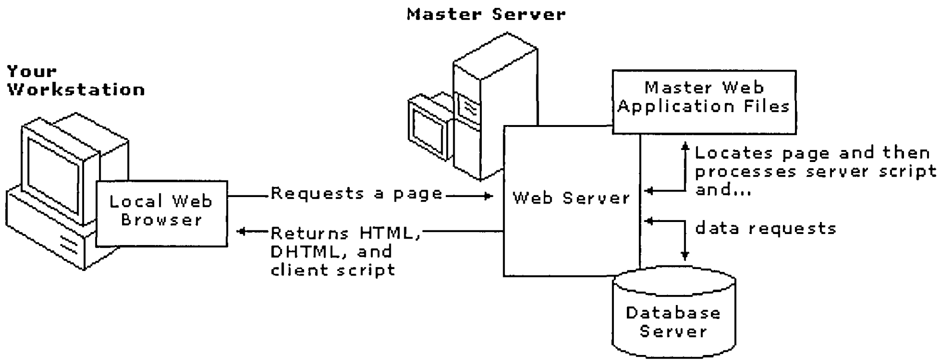


Test-Time Interaction of Components

You can test in master mode by previewing the Web application in your Web browser, by using the preview mode of the editor, or by verifying links in Link View.

The copies you are testing are those on the master Web server. If you use the editor's Quick View tab, you are viewing the local file and the server script is not processed even if you have a local Web server.

Test-time interaction of components in master mode


Master Server
Your Workstation
Local Web Browser
Requests a page
Returns HTML, DHTML, and client script
Web Server
Master Web Application Files
Locates page and then processes server script and...
data requests
Database Server

The local Web application and local Web server are not used during testing in Master mode.

For more information about testing your Web application, see the next section, "The Web Application Development Cycle"; "Link Verification" in Chapter 11, "Site Design"; and "The Script Debugging Process" in Chapter 23, "Scripting Concepts."

Offline Projects

You can take your project offline and continue to work on your Web application files. For example, you might need to work on your files while completely disconnected from the servers if you use a laptop.

Working offline is similar to local mode, except that you cannot update the Web server until you are back on line.

For more information about the project modes or working offline, see Chapter 7, "Working Locally" and "Working Offline," later in this chapter.

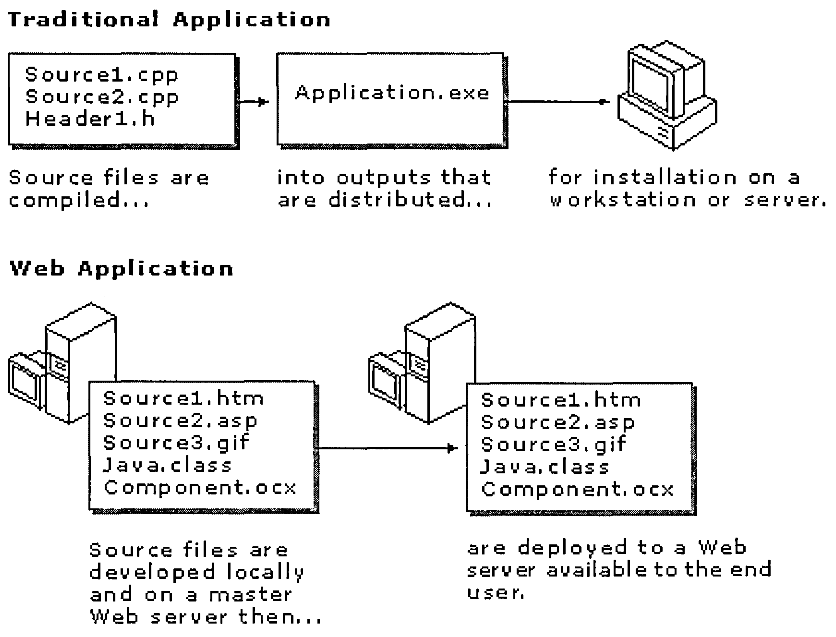
The Web Application Development Cycle

The development of Web applications requires the same iterative phases as other applications, however, the Web application itself is not the same as traditional applications.

A traditional application requires a special set of files during development, but distributes different outputs. For example, the class files, image files, and source code files used to develop a Java applet can reside in a single Microsoft Visual J++ project. For distribution, however, those files are compiled into a single executable that is independent of the files kept in the developer's Visual J++ project. The resulting executable does not require the presence of the source code files that were used to develop it.

Web applications, on the other hand, are composed of the same set of files used during development and after deployment. There is no compiled executable file produced that becomes the Web application. For example, the .htm, .asp, and .exe files in your Web project are the same files you propagate to your production Web server. The source code, or script, in these Web files is executed on the client or server only when a browser requests the Web page.

Source files and outputs for traditional and Web applications


Traditional Application
Source1.cpp
Source2.cpp
Header1.h
Application.exe
Source files are compiled...
into outputs that are distributed...
for installation on a workstation or server.
Web Application
Source1.htm
Source2.asp
Source3.gif
Java.class
Component.ocx
Source1.htm
Source2.asp
Source3.gif
Java.class
Component.ocx
Source files are developed locally and on a master Web server then...
are deployed to a Web server available to the end user.

You typically consider the following process in developing your Microsoft Visual InterDev Web application:

- Planning a Web Application
- Building a Solution
- Creating and Testing Web Items
- Deploying and Maintaining a Web Application

Planning a Web Application

You can use the Visual InterDev tools to get a fast start on your Web application development. You can save even more time with careful planning. If you address a few issues in the early stages of development, you are better able to implement features and functionality efficiently. For example, the design decisions you make at the start can impact how you create items in the Web application.

When you plan your Web application, consider the following activities before creating your Web solution.

- Identifying the Audience and the Browser
- Specifying the Web Application's Purpose
- Determining Content
- Analyzing Development Resources

Identifying the Audience and the Browser

A primary consideration in planning is identifying who will use the Web application — the audience. You can define your Web application's audience by identifying those who have access to the Web application and the type of Web browser they use. Access can come through an intranet, the Internet, or an extranet. An extranet is an area on a Web site available only to a set of registered visitors.

In many respects, access to the Web application helps you to determine the type of browser your Web application needs to accommodate. For example, if the Web application is designed for an intranet, typically you know which browser everyone is using and can program your Web application to take advantage of its features.

The capabilities of the Web browser also shape your plans for using client and server script in your Web application. Using server script and Active Server Pages (ASP), you can generate browser-independent pages easily. The server processes the server script and then sends HTML to the browser for processing. In contrast, if you know what type of browser the user will have, you can use client script to generate the page and minimize the load on the server. Depending on the needs of your Web application, you can incorporate both to take advantage of your system and the capabilities of your target audience.
The audience also determines what type of run-time security your Web application needs. Do you want everyone to be able to read the Web application's pages? Are some pages for everyone and other pages only for those meeting certain criteria? For example, you might want your download page available to those who completed a particular form. For more information about run-time security, see "Security," later in this chapter, and visit The Security Advisor Site at http://www.microsoft.com/Security/.

Specifying the Web Application's Purpose

Another key factor in development is the purpose of the Web application. For example, your Web application may be a set of announcement pages that inform visitors of current events. Maybe your Web application is an order-entry component on a large commercial Web site. Your Web application might be a game designed to attract Web visitors and introduce them to products sold on the site.

After you clearly specify the Web application's purpose, you can determine the features and functions that define the visitor's experience.

Determining Content

The content includes all of the Web items and design elements used to specify the features of your Web application and fulfill the Web application's purpose. For example, content for announcement pages can be implemented using simple HTML pages or using a database connection to populate an Active Server Page. An order-entry Web application can incorporate a variety of forms and database transactions using server script processing and client script processing. A game Web application can be created using Java applets or dynamic HTML and page objects to take advantage of quick client-side processing.

Before you plan the details, you might want to review the variety of content you need to consider and explore which Visual InterDev tools can implement your design quickly.

Some tools, such as the wizards and templates, quickly implement content without your having to plan the details. Other tools, such as the design-time controls, let you worry about the details of a design while the tool takes care of the code and script for implementing the design.

You can consider this list to help plan content to include in your Web application.

- Visual design and site navigation using themes and layouts
- Web application introduction and general information
- Database integration for displaying, updating, adding, or deleting records
- Web application control
- Forms and interactive pages
- Script for conditional processing
- Design-time and run-time security

Details about tools to use and implementation ideas are covered in "Building a Solution," and "Creating and Testing Web Items," later in this chapter.

Analyzing Development Resources

Your development resources affect how you plan and design your Web application. Here are some questions to help you analyze your development resource combination.

Who is on your development team?

Do you have a team of Web developers? Does your team include content authors and graphic designers? Do you want to use source control? Should source control be set to allow multiple checkouts of a single file or should it impose exclusive checkouts? What security levels do each of the team members need? What tools do developers need on their workstations?

The answers to these questions help you decide whether you need to use source control and what type of design-time security you need to consider. For more information, see "Security," later in this chapter.

If you are on a team, you probably want everyone to work in Web projects using local mode. Local mode allows the developer to work in isolation from changes made by other developers. For more information about project modes, see Chapter 7, "Working Locally."

By working with a team of developers, you can blend the skills of different developers to create Web applications that would be difficult or impossible for a single developer to create. When working on a team, the following concepts become important.

Work Simultaneously with Web Projects Visual InterDev allows multiple developers to work with files in the same Web application at the same time. For multiple developers to work on the same Web application, it must first exist on the master server. Developers can then create local project files that point to this Web application. This allows developers to maintain their own option settings and their own local copies of the master files.

If your team includes authors using Microsoft FrontPage, they can also refer to your Web application in one of their projects. For more information on creating Web projects, see "Project Architecture,"earlier in this chapter. For more information about using FrontPage with Web applications, see "Using FrontPage and Visual InterDev to Create Web Sites" in Chapter 29, "Integration Tasks."

Source Control Because all files reside on the master server, more than one person can work with the Web application files at a time. Multiple users can get files from the server, work with them, and then update the server copy with their changes.

To prevent more than one user from changing the same file at the same time, you can install Microsoft Visual SourceSafe 5.0 or later and enable source control for your project. For more information about using Visual SourceSafe with Visual InterDev, see "Source Control," later in this chapter.

Multiple Checkouts and Merge Resolution With multiple developers, you can also have more than one person working on the same file at a time. For example, you might have an author working on the content and a developer changing some script in the same file. When the file is updated on the master server, the second person saving the changes has the opportunity to merge the differences. For more information, see Chapter 8, "Working with Multiple Developers."

What is your development system?

Do you have a single developer workstation with both the server and client components needed to develop and test a Web application? Are your Web server components, database development components, and developer tools on separate machines? What level of design-time security does your Web application require?

The location of the components affects how you test your Web items and the Web application. For example, Web applications tested locally without a local Web server cannot process Active Server Pages. If you don't have a copy of a Web server on your developer workstation, you might want to consider using a Web project in master mode so that you do not have to explicitly update the master and test on the master server.

What is your testing system?

Do you have a testing or staging server that the team uses simultaneously for development? Do your team members each develop their files locally, then move them up to the staging server? Where does your test database reside?

The locations for testing help determine which project mode you want to use. Preferably testing and development of individual pages is isolated from the testing performed on the master version by setting your project to use local mode.

What is your deployment system?

Are you deploying your Web application on the same Web server that you are using for testing? Is your master server also your production server? Where does your production database reside? What level of run-time security does your Web application require of the deployment server?

The relationship of your testing server and production server determines how you perform some server management and file management tasks. Preferably, Web applications under development and the production versions are maintained on separate machines. For more information, see "Project Architecture," earlier in this chapter.

Developing Enterprise Applications

As most people use the term, an enterprise application is scalable, distributed, component-based, and mission-critical. Enterprise applications tend to be data-centric, and must meet stringent requirements for security, administration, and maintenance.

Developing extensive enterprise-level applications adds a level or two of complexity to the Web application development process. But Visual InterDev, in conjunction with other Microsoft Visual Studio tools, allows you to create these complex applications.

All the points in this topic that apply to Web applications apply as well to enterprise applications, but here are a couple of additional points you'll want to keep in mind when developing enterprise applications:

- Plan extensively. Think about scalability and performance in the design phase. If you anticipate a lot of network traffic, plan to take advantage of Microsoft Transaction Server technology and ASP connection pooling. Separate business logic, databases, and user interfaces as much as possible.
- Create components when possible. Encapsulating functionality into ActiveX controls and COM objects allows you to share and maintain code more efficiently. Microsoft Visual Basic, Visual FoxPro and Visual C++ make it easy to encapsulate your code into components that can be leveraged from Visual InterDev applications.

For a detailed discussion of enterprise application development, see *Developing for the Enterprise.* "What is an Enterprise Application" presents a good overview of the enterprise application model and it is a good place to start when planning your enterprise application.

Building a Solution

A solution allows you to work on several projects at one time efficiently. For example, a solution can have both a Web and a Microsoft Visual J++ project in it. The Web project manages all of the files that make up the pages of the Web application. For example, you might have several pages that make up the front-end to your database. Or you can create and deploy some pages to inform your users that the Web application is in development. For this set of pages, you can create one Web project for publishing some basic Web pages that announce the Web application and keep visitors updated on when the full Web application will be available.

You can add a second Web project to the solution for developing a prototype of the final Web application. Also, if you plan to create a database, you can add a database project. Your Web project can also include Web components created in Visual Basic or Visual C++.

In building a solution, you will probably want to perform the following general activities:

- Make a Web project to create and manage the files in your Web application.
- Add a database connection or create a new database.
- Develop a prototype by adding Web items to your Web project.

Making a Web Project

To minimize the time and effort you spend in creating a Web application, Visual InterDev sets up the basic components you need and gives you the option to standardize the look and feel from the start.

To set up a Web project, you need to provide the following information.

- Name of the master Web server
- Name for the local project and physical directories
- Name for the Web application and virtual root
- Mode for updating the master server: master or local
- Theme (optional)
- Layout (optional)

After you have created a Web project, you can add items that implement the features you identified during the planning phase.

Working Independently

Your Web project actually supports two sets of Web application files. One resides on the master Web server and one locally in your project directory. The local version allows you to develop and test independently from the master version depending on the mode of the project. For more information, see Chapter 7, "Working Locally."

Your project can operate in one of two modes: local mode or master mode. In local mode, you can save and test the changes you make to files in the local Web application without changing the master copy. You work in isolation from changes made by others to the master Web application. Your changes are not saved to the master Web server until you release the local copy. For more information about the two modes, see "Specifying a Project Mode" in Chapter 7, "Working Locally."

When you work in local mode, the changes you make are applied to a local disk-based Web application called the local Web application. The local Web application mirrors the structure of the master Web application as it was when you last updated the master Web application, refreshed the project, or synchronized the file sets. For more information, see "Synchronizing Master and Local Files" in Chapter 10, "Managing Web Projects."

At times your local Web application can contain a different set of files than the master Web application. For example, if you add a file while in local mode, the file is immediately added to your local Web application but not to the master. If team members update the master from their local Web application, your Web project will not show those changes. To resolve these issues, update the master Web application, refresh your project, or synchronize your local Web application with the master Web application. For more information, see "Updating the Master Web Application" and "Updating the Local Web Application" in Chapter 7, "Working Locally."

Working Offline

If you do not need to access the server or if the server is down, you can work offline. When working offline, commands that require the server are not available. For example, you can make changes to files in your local directory, but you can't release the changes and update the master Web application. For more information, see "Working Offline" in Chapter 7, "Working Locally."

In addition, the functionality available for testing your project offline depends on the system components on your developer workstation. If you have a Web server on your developer workstation, the project sets up a Web application root that allows you to test against a Web server and server extensions on your workstation. If you do not have a Web server and the appropriate extensions, the file is loaded directly from the local Web directory. For more information about the components you need for testing various Web items, see "Project Architecture, earlier in this chapter.

Working Outside of a Project

If you add, create, or delete Web application files outside of Visual InterDev and the Project Explorer, you need to refresh the project when you return to the Project Explorer. For example, if you add or delete a file using the Windows Explorer, the project would not show that change to the local directory and the local files until you refresh the project. For more information about refreshing the Web project, see "Synchronizing Master and Local Files" in Chapter 10, "Managing Web Projects."

Adding a Database

If your Web application plans include database access, you can integrate existing databases or modify a database specifically for your Web application. In the Web project, just add a connection to the database and you can use information from the database in your Web application files. This connection also displays a Data View window you can use to view the objects in the database and adds a data environment to your project.

You can also modify the database objects from your Web project using the Microsoft Visual Database Tools. After you have a database connection, you can continue working from within the Web project to develop a prototype. For more information on the Visual Database Tools, see Chapter 18.

Developing a Prototype

After you have created your Web project and data connection, you can quickly develop a prototype of your Web application. Some developers wait to add the look and feel elements of their Web application until after they have finished implementing the core features and functionality. You may have already added those elements to your Web application when you created the project and chose a theme and layout. For more information about themes and layouts, see "Site Consistency" in Chapter 11, "Site Design."

Create and Organize a Set of Pages

Using your content plans, you can outline the HTML and ASP pages for your Web application by creating a site diagram. A site diagram provides a graphical view of the pages in a Web project. The pages automatically incorporate the theme and layout defaults you selected when you set up your Web project.

After you have setup the initial prototype pages and Web application organization, you can modify them to meet your Web application's requirements.

By establishing hierarchical relationships between the pages in a site diagram, you can easily design the navigation links for your Web pages. If you want to modify the look and feel, you can do that quickly by changing the theme or layout. For more information about site diagrams, see Chapter 12, "Designing a Web Site."

Author Text to Introduce the Web Application

Web application pages typically have some information that rarely changes. HTML is used to implement this text.

Your Web application should probably include at least one page with information that introduces the purpose of the Web application and informs the Web visitor how to use the Web application or a specific page. Typically this information is provided on the start page for the Web application and provides a common entry point to the Web application. It can include other design elements such as forms, data, or interactive objects.

If you are designing for the Internet, this page should be browser-independent and inform the user of any special software or browser requirements your Web application needs to run properly.

Implement Data and Script

After you have created your initial set of pages, you can open the pages in the editor and add the text, images, and script that fulfill your Web application's purpose. One of the main considerations in determining how you implement content is whether you want the processing performed on the client or the server. This answer depends on the audience, their browser capabilities, and the resources on your deployment system. For a summary of Web items to consider using, see the following section, "Creating and Testing Web Items."

Whether you decide to use client or server scripting, Visual InterDev offers a Web page editor and a data environment that allows you to implement your functionality quickly. For example, you can use the data environment to create a recordset object for use on multiple pages. The editor also offers several data-bound design-time controls you can easily add to your pages that display those records. For more information, see "Scripts in Web Applications" and "Document Elements" in Chapter 23, "Scripting Concepts."

Control the Web Application

To control your pages as a Web application, you can use server processing with Active Server Pages and its special file, the Global.asa file. In the Global.asa file, you can script session- and Web application-level events and variables that are available to all of the pages in the Web application that use server script.

Creating and Testing Web Items

Visual InterDev provides a variety of visual designers and ActiveX controls you can use to implement your feature set.

Identifying Items to Use

When planning your Web application's functionality, you might want to consider using a variety of items to specify the features in your Web application. Since a Web application actually consists of a set of files that reside on a Web server, you can identify each item by its file type. For example, you can incorporate any of the following items into your Web application.

To add	Consider
Static pages or client script	.htm files
Server script	.asp files
Data publishing	Recordsets
Design tools	Design-time controls
Visual design	Templates, themes, layouts, images and multimedia files
Interactive pages	ActiveX controls
Integrated Web solution	Output files from other projects such as applets
Downloadable documents	Document files, spreadsheets

Creating Simple Information Pages

You might want to create a set of straight HTML pages that announce that your new Web application is in development, when to expect it to be deployed, and what your user can expect to gain from it. A few static information pages can hold the place of your Web application and create interest in using it. You can deploy this set of pages to establish a Web presence that can be as useful on an intranet as it is on the Internet.

After deploying your introductory version of these pages, they can easily evolve into the start pages for your dynamic Web application. After your Web application is complete, all you need to do is deploy the full set of files and replace the previous versions with your improved pages and the additional pages that make up the Web application. Links to the original introduction pages will continue to work to bring the user directly into your Web application.

You can use a site diagram if you want to create several pages or simply create a single HTML page that contains the announcement. You might want to add images or other effects to make the page more visually appealing.

Publishing Data Dynamically

A Web application can serve as the front-end to any ODBC data source. You can design pages that display data or allow users to update, add, or delete records. For displaying data, you can use data-bound design-time controls. You can choose whether the data is processed on the server and sent as text to the client or you can implement some dynamic data capabilities on the client. For more information on data connections, see Chapter 18, "Database Concepts," and "Connecting to a Database" in Chapter 3, "Database Basics."

Scripting Interactive Pages

Typically a Web application includes pages with forms or other items that require or respond to user actions. Forms are made up of intrinsic elements of HTML. You can use design-time controls to quickly tailor elements on a form to fulfill your Web application's requirements. With dynamic HTML, you can create dynamic effects on the pages and script them to respond to events or the passage of time. For more information, see Part 5, "Editing and Scripting."

Although many of the Visual InterDev wizards and tools generate the basic code and script for many elements of your Web application, you probably want to use certain criteria to control the results on your Web page. For example, you might want certain pages that appear for a user who completed a form and a prompt to finish the form for users who did not fill out the form. For more information about using the editor and scripting pages, see "The Scripting Object Model," or "Scripts in Web Applications," in Chapter 23, "Scripting Concepts."

Adding Run-Time Security

You can protect your resources from unauthorized use by implementing security. In addition to the security you set through your server, you can also add password pages and user verification script through your Web application. For example, you can add pages that integrate an HTML form that collects user identification and password with server script that compares the information to a database. For more information about security guidelines and adding security to a Web application, see "Security," later in this chapter.

Designing Visual Impact and Site Navigation

The visual design and site navigation determines your Web visitor's initial impression of your Web application. The way you apply colors, images, and sounds can make your Web application intuitive and easy to use. Carefully designing your site navigation makes it easy for the user to move from page to page through your Web application. Using a site diagram, you can graphically lay out your pages and specify the navigation at the same time.

Before creating your Web application, plan the general look and feel of your site. Or, choose a Visual InterDev theme and layout to give your Web application consistent visual impact while you create the pages in your Web application. When you create a project, you can specify one of the themes provided by Visual InterDev.

If you have a theme design in mind, you can create your own themes, templates, and layouts to quickly add consistency to the Web application.

Themes and Layout Files

Visual InterDev offers you several themes and layout files you can use to give your Web application a consistent look and feel. For more information, see Chapter 17, "Customizing Page Appearance."

Site Navigation

Site Designer provides a graphical method of designing your site navigation. Create site diagrams to define the hierarchical relationships between pages. When you specify the relationships, the site diagram automatically specifies the links between the pages and adds them to your Web pages. For more information, see Chapter 13, "Designing Site Navigation."

Dynamic HTML and ActiveX Controls

You can use Dynamic HTML to lay out text, images, and multimedia items on HTML pages. Dynamic HTML provides you with a rich set of controls for scripting multimedia pages. These controls give you a set of objects and parameters you can change dynamically to change visual and sound effects on your page. For more information about using controls, see Chapter 24, "Scripting with Design-Time Control and Script Objects."

Testing Links and Debugging Script

You can test the entire Web application as you progress. The two main areas to test are links and script.

Find Broken Links and Pages Without Links

In creating a Web application, one of the primary concerns is the verification and testing of links between the Web pages. Visual InterDev provides tools for you to verify and test the links within your Web application.

Check links on a single page Using Link View, you can verify that each link on a page points to a page that exists. Through the link diagram, you can see a graphic representation of the links to and from your Web page. Link View only verifies that a link points to a page that is an existing Web page. It does not determine whether the link points to the correct page. For more information about Link View, see "Link Verification" in Chapter 11, "Site Design."

Check links in the entire Web application If you want to find out all of the broken links and files without links in the entire Web application, you can generate a list using the Broken Link Report. For more information, see "Repairing Links" in Chapter 16, "Maintaining Links."

Verify link destinations To test that each link goes to the correct page, you need to open the page in a browser and click each link. To test links while your project is in local mode or offline, you need to make sure that all the files you want to view are in the local directory. If the target file is not in the local directory, the browser can't resolve the link.

Automatically repair links Visual InterDev also offers a link repair feature that automatically fixes links when you make changes to the files. For example, if you rename a file, all links to that file must be updated with the new file name. Even simply renaming or moving a single file may force many other files in the project to be changed.

Link repair modifies the necessary files on your behalf, updating the links to point to the correct file. The link changes are first applied to the master Web application and then to the local Web application and all affected files are updated. If an affected file is open in an external editor or if another developer has the file open, the links in those files must be fixed manually. For more information, see "Repairing Links" in Chapter 16, "Maintaining Links."

Debugging Script

You can debug script much as you would in a traditional Web application. When you preview your pages, you can find syntax errors, run-time errors, and logic errors.

Using the editor and its debugger, you can control the execution of your script and monitor values of variables and properties to identify the cause for errors you encounter. For more information about debugging script, see "The Script Debugging Process" in Chapter 23, "Scripting Concepts," or Chapter 26, "Debugging Your Pages."

Deploying and Maintaining a Web Application

After you have tested the Web application and are satisfied with its performance, you can deploy it to the Web server you make available to your users. Since a Web application is actually a set of files, you need to copy the virtual root and its file set to the production Web server. If your pages use the virtual root as the basis for their links, all links should still work. If your project includes a dependent project, you need to make sure the server component outputs are properly registered on the server. For more information about registering components, see Part 6, "Building Integrated Solutions."

The advantage of a Web application is easy maintenance. To upgrade a Web application you don't need to recompile and redistribute an entirely new executable file. All you need to do is add new files and replace previous versions.

During an upgrade, your users are not interrupted while they are working because their browser is using the original copy of the file it got from the Web server. When the file is replaced with a newer version, the user sees the new version only after refreshing the page not while viewing it. The upgrade to a new file set is seamless. For more information, see Chapter 30, "Deploying and Maintaining Web Applications."

Source Control

Your Web application can easily grow into a large set of files. To keep your Web application fresh and current, you may need a team of people to create and maintain those files. Whether you are working on a team or by yourself, you can track and save your changes to files easily using Microsoft Visual SourceSafe. To manage changes to your Web files, you can use Visual SourceSafe in unison with Microsoft Visual InterDev.

Using source control, you and other Web team members can share files, modify them independently, and later merge the changes. Visual SourceSafe also saves past versions of the files in a database, tracks the date and time of changes, and provides an option to keep a comment log.

The most commonly used Visual SourceSafe commands are available directly from within Visual InterDev. The rest of the Visual SourceSafe command set is always available from the Visual SourceSafe Explorer.

To use source control with Visual InterDev, you need to understand:

- The Components for Source Control
- Your Interaction with Visual SourceSafe
- The Interaction Between Visual InterDev and Visual SourceSafe

The Components for Source Control

To use source control with your Web files, you need the following items.

- **Visual SourceSafe server** installed on your master Web server with the integration option selected. For more information, see "Installing Visual SourceSafe on a Web Server" in Chapter 8, "Working with Multiple Developers."

  Note Installing the Visual SourceSafe client on your developer workstation is optional. The client is not necessary for checking your Web application files in and out as long as Visual SourceSafe is installed on the master Web server.

- **FrontPage Extensions** installed on the master Web server. Visual InterDev uses the FrontPage Extensions to send commands to Visual SourceSafe.
- **Visual InterDev** installed on your developer workstation.
- **Source control enabled** on the Web application and set to use Visual SourceSafe on the master Web server.

  Note For best results with source control, use Visual SourceSafe on a server running Microsoft Windows NT with an NTFS file system.

Your Interaction with Visual SourceSafe

You can interact with Visual SourceSafe on several levels depending on the task you want to complete. Some tasks need to be performed only once, while others may take place on a regular basis.

The following table shows a list of typical tasks you might perform, how often you will need to perform the tasks, and the interface you use to complete the task. For more information about each task, use the link provided in the table.

Task	Frequency	Software used
Installing and setting up Visual SourceSafe	Once for each master Web server	Windows and Visual SourceSafe Setup
Setting options for multi-user checkout	Occasionally as needed	Visual SourceSafe Administrator
Adding source control to a Web application	Once for each Web application	Visual InterDev
Checking files in and out	Often as needed	Visual InterDev

(continued)

Task	Frequency	Software used
Resolving merge conflicts	Occasionally as needed	Visual InterDev
Setting checkout code options	Occasionally as needed	Visual InterDev
Reviewing file history	Occasionally as needed	Visual SourceSafe Explorer*
Restoring past versions	Occasionally as needed	Visual SourceSafe Explorer*

* If you're using a proxy server for remote access, you won't be able to use the Visual SourceSafe Explorer to view your Visual SourceSafe database because the Explorer requires a LAN connection. For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts."

The following sections describe file tracking, simultaneous checkout and merging, history details, and other Visual SourceSafe options.

File Tracking

After you have Visual SourceSafe installed and enabled for your Web application, you can typically work directly in Visual InterDev. Visual SourceSafe and Visual InterDev work together seamlessly to track your file changes as you add, save, and remove files in your Web project. Any changes you make to your Web application files are propagated to the Visual SourceSafe project without requiring you to open Visual SourceSafe.

When you check out a file in your Web project, you get a write-enabled copy of the file in your local Web. When you check the file in, the new version is copied to the master Web application on the master server and is sent to the Visual SourceSafe repository. Your local version becomes read-only. If you have been working offline, when you go online and check in a file, the new version is copied to both the master Web application and the Visual SourceSafe repository. For more information, see "Working under Source Control" in Chapter 8, "Working with Multiple Developers."

Simultaneous Checkout and Merging

If Visual SourceSafe is set to allow simultaneous checkouts, you are warned if someone else has made changes to the file while you had it checked out. When this happens, you can review the differences between the two files and choose to merge them, accept specific changes, or disregard changes in either version.

For example, you might be working on the script and someone else might be changing content on a Web page. When you check in the file, you can merge the two versions into one file containing both changes.

The option for simultaneous checkout is set using the Visual SourceSafe Administrator. For more information about multiple checkouts, see "Resolving Merge Conflicts" in Chapter 8, "Working with Multiple Developers."

File History Details

Visual SourceSafe offers the option to keep notes about each check in by prompting you for a comment when you check in a file. If you do not plan to keep a comment history, you can use the Source Control options in the Visual SourceSafe Administrator to prevent the comment dialog box from appearing.

Visual SourceSafe Options

You can also set additional options for how Visual SourceSafe provides alerts and handles changes to files. These settings affects the Visual SourceSafe options for the Web application and, therefore, everyone who works on the Web application. For more information about setting options, see "Options Command" in the Visual SourceSafe topics.

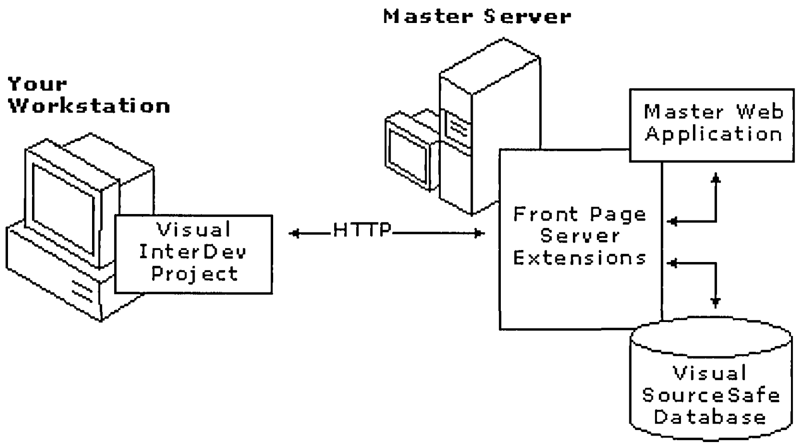
You can also set check out options that determine whether you always get a write-enabled copy, a read-only copy, or a prompt to choose. For more information about setting default options, see "Checking a File Out" in Chapter 8, "Working with Multiple Developers."

The Interaction Between Visual InterDev and Visual SourceSafe

Visual InterDev and Visual SourceSafe interact through the FrontPage server extensions. Visual SourceSafe and the FrontPage extensions must reside on the same machine working against the same master Web server. If you have Microsoft Internet Information Server 4.0, you can have multiple virtual master Web servers, but a single Visual SourceSafe project can reference only one.

The FrontPage server extensions provide HTTP connection services between Visual InterDev and other services on the Web server. When you check in a file, the file moves from Visual InterDev on your developer workstation to the FrontPage extensions, which update the master Web application and give a copy to Visual SourceSafe. Visual SourceSafe adds the copy to its repository along with a record of the date, time, and user who checked the file in. If you choose to add comments, this text is included as well.

Visual InterDev and the FrontPage server extensions



This table shows what happens when you perform common file commands.

Task	Action
Check out and open a file	**Local and master mode** Checks if the file is already checked out exclusively. If not, checks out the file and places a write-enabled copy of the file in your local Web application. **If you are offline**: You cannot check out files while you are disconnected from the server. You need to check them out before going offline.
Save a file	**Local mode** Saves the file to the local Web application The file is not saved to the master Web application and is not checked in to Visual SourceSafe. **Master mode** Saves the file to the local Web application and updates the copy on the master Web application. The file is not checked in to Visual SourceSafe. **If you are offline**: The file is saved to the local Web application, but not to the master Web application. The file is not checked in to Visual SourceSafe.
Save and check in a file	**Local mode and master mode** Checks in the file and updates the copy in Visual SourceSafe and in the master Web application. Depending on your check-in options, may also remove the write-enabled copy from the local Web application. **If you are offline**: You cannot check in files while disconnected from the server. You need to check them in after you go back online.

Security

The World Wide Web and HTTP provide the largest imaginable audience for Web applications and a proportionately increased need for security. In addition, Web development teams can now span the globe. To control just how that audience and team work with Web application files and gain access to the system that supports them requires security.

Security for Web applications is a complicated subject because it can be set at several levels in several different ways. The choices depend on the system and servers used and the needs of the Web application.

Some of the considerations are:

- Locations for Setting Security
- User Authentication
- Permitting Access to Web Applications, Folders, and Files
- Guidelines for Design-Time Security
- Guidelines for Run-Time Security
- Securing HTTP Transmissions (SSL)
- Administration and Maintenance Considerations

For more information about security, see "Authentication and Security for Internet Developers" in the MSDN Technical Articles section or see the Microsoft Security Advisor Web site at http://www.microsoft.com/security/.

Locations for Setting Security

You can set security options in several locations depending on your system and the assumptions you can make about your visitors. For example, basic security for an intranet can be handled and maintained in parallel with the security for the network itself. For an Internet Web application, you can add security through the pages provided to the Web browser using the Web application.

The following table shows a summary of many options you have for security locations. As you can see, you may need to use several interfaces to get the results you want. This table assumes you are using Microsoft Windows NT with Microsoft Internet Information Server for your Web services and Microsoft SQL Server for your database server.

Location	Example interface
Operating system	User accounts specified in Windows NT* User Manager
Folder	Sharing properties set in the Windows NT Explorer
Web server	User accounts and server properties set in Internet Information Server Administrator

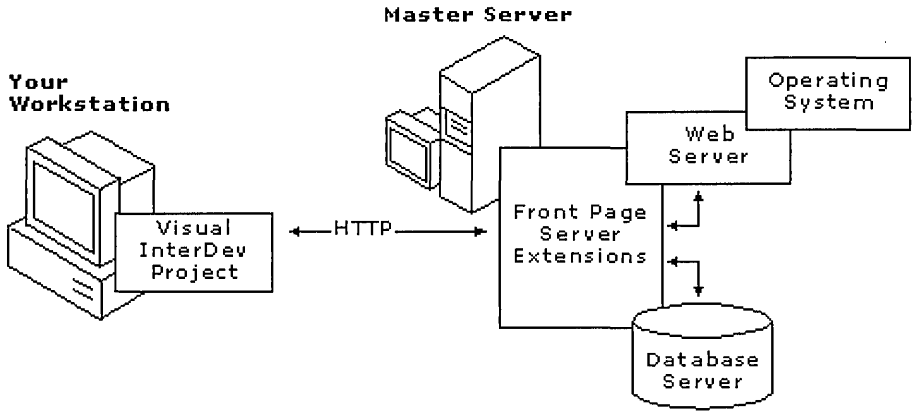
(continued)

Location	Example interface
Virtual root	Internet Information Server Administrator
Database	Permissions specified in SQL Server, Enterprise Manager
Web application	Web application properties set in Visual InterDev Web Permissions and stored in a session
Page	Server script written in Visual InterDev, Source Code Editor
Source control	User accounts and permissions specified in Visual SourceSafe Administrator

* Applies only to Web servers on Microsoft Windows NT with FS file system. NTFS allows you to specify per-file Access Control Lists (ACLs). Use of the FAT file system, whether on a Microsoft Windows 95 or Windows NT server, greatly reduces the security features available.

For all interactions with your Web server, Visual InterDev uses the Microsoft FrontPage server extensions that may use the existing security features of your operating system and Web server. For example, the extensions are integrated with Windows NT and Internet Information Server to manage Web application security. You interact with the FrontPage extensions through the Visual InterDev user interface.

Visual InterDev and security components



When planning which security options to use for your Web application and where you want to set those options, it is helpful to consider four types of access to your site. You may want to:

- Allow any Web visitor to execute ASP pages and read HTML pages at run time.
- Restrict access to registered Web visitors at run time.
- Allow Web developers and authors to write to your files at design time.
- Restrict Web administration to certain authorized users.

For a complete understanding of security options affecting your Web application, read about the security features of the FrontPage server extensions, Internet Information Server, and Windows NT. If you use a database in your Web application, you might want to become familiar with the security models of ODBC databases, such as Microsoft Access or SQL Server. For more information about setting security options for Windows NT and Internet Information Server, see the documentation for those products.

User Authentication

In Web security, your central concerns are identifying a user and controlling the user's access to your Web application and its resources. You accomplish this by implementing measures to authenticate users and specify access permissions.

In choosing your security options, you need to specify how the user provides an identification to your Web application, how that identity is verified, and what level of access or permissions that user is allowed. The following table provides guidelines to help you choose the appropriate options for your Web application.

Location	Example interface
Operating system	Domain and user account specified in Windows NT
Web server	User accounts and server properties set in Internet Information Server Administrator
Database	Permissions specified in SQL Server, Enterprise Manager
Web application	Web application properties set in Visual InterDev Web Permissions dialog box, login pages, and variables stored in a session

Each of these locations provides a different feature set for implementing user identification, verification, and permission levels. Depending on the systems and software you are using, the feature sets may work together to set options at the operating system level or they may change related options at other levels. For example, if you set permissions on a file or folder in the Visual InterDev Web Permissions dialog box, you are also setting the permissions in Windows NT.

Operating System Level

Security choices you make for the operating system depend on the features and options your operating system offers. For example, if you are using Windows NT, you can implement authentication for a large number of people without explicitly specifying user accounts by using a special user account, referred to as the "guest user." This account is set up by default. If you want to distinguish between individual users and track them, you need to implement security options at a different level than your operating system.

In addition to authentication, the file system used by your operating system also affects the permissions you are able to grant to the user. For example, Windows NT may use either Window NT File System (NTFS) or File Allocation Table (FAT). NTFS allows you to specify an Access Control List (ACL) for files and folders so you can control access more granularly. FAT, on the other hand, does not provide the ability to specify lists and offers less control.

Web Server Level

Once the operating system has verified the identity of the user, the Web server can also evaluate the user's identity. The Web server primarily controls access at run time. For example, if you are using the default configuration for Internet Information Server, the Web server processes anonymous requests as the anonymous Internet Information Server user, IUSR_<*machinename*>.

The anonymous user set up by Internet Information Server is similar to an Internet guest account defined in Windows NT User Manager. However, password changes are not handled automatically. Changes to passwords for one requires an explicit change to the other.

If you have set the anonymous account to allow access to the requested file, Internet Information Server allows access to the file and satisfies the request. Otherwise, Internet Information Server rejects the request, returns an error to the client and informs the client of the authentication methods that Internet Information Server supports.

For more information, see the security topics in the documentation for the Web server you are using.

Database Level

Visual InterDev allows you to connect to a wide variety of data sources. The security measures you set for your database depend on the database management system you are using. The following paragraphs provide a few tips for setting security on databases used in Web applications. For the most recent information about setting security for your database, see the security topics in your database management system's documentation.

For file-based systems such as .mdb files in Microsoft Access, you can control security through the sharing permissions available on the folders and files for the operating system. If you are using SQL Server, you can use the features for granting and revoking privileges offered within the database management system. For example, Microsoft SQL Server has a Security Manager that allows you to specify the privileges available to a single user or a group of users.

Although SQL Server for Windows NT offers three types of login security, the standard security choice is recommended. Standard login security requires a login ID and a password to access the server. Your Web application provides this information through the data connection and your users do not need to provide any additional identification.

While not recommended for all systems, some Web applications might use integrated security. If you choose to use integrated security, keep in mind that SQL Server will use Windows NT security to authenticate users so the Web application and database must reside on the same server. Windows NT does not delegate security to another server.

Web Application Level

In your Web application, you can take advantage of the security features provided through FrontPage server extensions and through the Global.asa file processing available with Active Server Pages.

FrontPage Server Extensions

Visual InterDev works through the FrontPage server extensions to provide the ability to manage design-time Web permissions using the underlying security model of the host operating system on the master Web server.

If your operating system is Windows NT with the NTFS file system, the FrontPage extensions manage access for administrators and authors using file ACLs for the DLLs in the following table. These directories are hidden to the Web server but available to the file system:

Function	DLL	Location
Administrative (i.e., setting Web permissions)	Admin.dll	<Webdir>/_vti_bin/_vti_adm
Authoring (i.e., opening a file)	Author.dll	<Webdir>/_vti_bin/_vti_aut
Browsing (i.e., viewing links)	Dvwssr.dll	<Webdir>/_vti_bin/_vti_aut

When you perform a function, such as changing permissions on a Web application, your request is sent over HTTP at the server and routed to one of these ISAPI DLLs. For example, a request to perform an administrative function is handled by that Web application's Admin.dll. In the request, the client provides credentials that identify the user who is logged in to the client workstation. This user must have read permission (equivalent to read and execute individual permissions) for the DLL handling the request; otherwise, the request is denied.

Thus FrontPage restricts who may perform a given request by controlling read permission on the DLLs in _vti_bin. Whenever a change is made to a Web application's permissions via the Web Permissions dialog box, the FrontPage extensions on the server modify the ACLs on the DLL's _vti_adm and _vti_aut directories in that Web application's _vti_bin directory accordingly.

Note FrontPage does not change ACLs on content files to manage design-time security; it only changes ACLs on the directories that contain the gatekeeper files admin.dll, author.dll, and dvwssr.dll. FrontPage manipulates content file ACLs to manage run-time security, which is the topic of the next section.

Global.asa File Processing

Visual InterDev supports Web applications that make full use of Active Server Pages. One of the features of an Active Server Page is the automatic processing of a Web application's Global.asa file, if present, at the start of a new session with that Web application. You can use the Web application and session to control access and processing of your Web application. The user never actually sees the Global.asa file and the security provisions you add to it.

Permitting Access to Web Applications, Folders, and Files

After specifying a way of identifying users, you can control access to your system resources through setting permissions. Typically, you set permissions based on files and directories.

At the operating system level, you can set permissions for individual user to read, write, execute, delete, change permissions on, or take ownership of a file.

In addition, Windows NT defines some standard permissions that combine one or more individual permissions. It is these standard permissions that are usually assigned to files or directories. For instance, having change permission is equivalent to having read, write, execute, and delete individual permissions. Having full control is equivalent to having all the individual permissions. Having the standard read permission is equivalent to having individual read and execute permissions.

In this document, standard permissions are used unless otherwise noted.

Controlling User Access to Resources with ACLs

Identity and permissions together determine a user's access to resources. The entire set of permissions assigned to a resource, for example, a file or directory, is called an Access Control List (ACL). When a user attempts to access a resource, the operating system checks the ACL of the resource to verify that the user has the proper permissions for the type of access being attempted. For instance, if a user who has read-only permission on a file attempts to delete that file, that user is denied access and cannot delete the file.

Here is a simple example: A Web site manager wants to use security to protect a single file on the site, say Sample.htm, from being read by users browsing the site anonymously. By simply changing the permissions on Sample.htm so that the anonymous user account does not have read permissions for the file, anonymous users will not be able to view Sample.htm.

Note The permissions you set for the Windows NT group, "Everyone," also includes the anonymous user.

Visual InterDev Web Applications

Visual InterDev exposes the security model defined by FrontPage for Webs hosted on a server running the FrontPage server extensions. This model is summarized here, and described in more detail in "Using the FrontPage 97 Server Extensions with the Microsoft Internet Information Server" in the FrontPage documentation. It extends to Web applications the permissions model discussed above for files and directories.

Levels of Web Access

FrontPage server extensions define three levels of access for a Web application. An individual user, or an entire group of users, is either assigned one of these three levels or has no access. Access is defined on a per-Web application basis. The operation of defining access on a Web is referred to as "setting Web permissions." The three levels of access are administer, author, and browse. These are described below.

Access level	Permission to
Administer	Author and browse files
	If on a root Web, create or delete Web applications on the root Web server
	If on a Web application, change permissions on that Web application
Author	Browse, open, and modify files in that Web application, including adding or deleting content
Browse	View the content of that Web through a browser By default, when a Web application is created, browse access is granted to all users. You can restrict browse access for a Web application to specific users who have been given browse access.

Guidelines for Design-Time Security

Design-time security is managed by controlling which users and groups of users are granted administer and author access to Web applications.

To give a user full design-time access to a Web application, the user should be granted administer access to that Web application. If you want a user to be able to create new Web applications on a server, the user needs administer privileges on the root Web of that server. To give a user enough access to fully modify an existing Web application, but not to delete it, the user should be granted author access to the Web application.

During design time, you might need to provide various levels of access to members of your Web development team. Not all members need administrative privileges. Some members may need only write access to certain folders or directories. By default, Web applications inherit the same rights that the root had at the time the Web application was first created. For more information about specifying privileges for design time, see "Setting Web Application Permissions" in Chapter 9, "Adding Security."

Controlling Developer Access to your Development Web Application

If you want to control the privileges that members of your Web team have for changing your Web application files, you can use Visual InterDev along with settings at other locations. You can use the following tables to help you choose security settings that are appropriate for your Web applications.

Location for setting	Typical Intranet setting
Operating system	Windows NT Challenge/Response (NTLM)
Windows Explorer folder (OS)	Add user group or individuals to permissions list
Web server	Server-specific
Database	In the Enterprise Manager, set security type to Standard
Visual InterDev Web project	Add user to Visual InterDev Web Permissions list
Visual SourceSafe	Add users to Visual SourceSafe Administrator

The following table shows typical settings for security on the Internet.

Location for setting	Typical Internet setting
Operating system	Basic clear text
Windows Explorer folder (OS)	Add user group or individuals to permissions list
Web server	Same as operating system
Database	In the Enterprise Manager, set security type to Standard
Visual InterDev Web project	Add user to Visual InterDev Web Permissions list
Visual SourceSafe	Add users to Visual SourceSafe Administrator

Specifying Web Administrators

When you first install your Web server, you are prompted to supply a Web administrator password. You can specify that other users have the same administrator privileges for your Web applications. For example, you might want someone else to add new users or change Web permissions.

You can use the following table to help you choose what security options are appropriate for your Web applications.

Location for setting	**Typical Intranet setting**
Operating system (Windows NT)	Windows NT Challenge/Response (NTLM)
Windows Explorer folder (OS)	Add user group or individuals to permissions list
Web server	User groups and password settings must match the operating system
Database	In Enterprise Manager, provide appropriate database permissions
Web application	Global.asa file
Visual InterDev Web project	Add user to Visual InterDev Web Permissions list

This table shows typical settings for the Internet.

Location for setting	**Typical Internet setting**
Operating system (Windows NT)	Basic clear text
Windows Explorer folder (OS)	Add user group or individuals to permissions list
Web server	Allow Anonymous
Database	Provide TCP/IP access and special .dll In Enterprise Manager, provide appropriate database permissions
Web application	Global.asa file
Visual InterDev Web project	Add user to Visual InterDev Web Permissions list

Using Source Control

Applying source control to your Web application during development is another way you can add security during design time. Visual SourceSafe added to your Web server helps control who can check out and modify files in the Web application.

Guidelines for Run-Time Security

After a default installation of Microsoft Internet Information Server and FrontPage server extensions, Web applications on a FrontPage server may be read anonymously by any browser that can make an HTTP request to the server. To secure their Web applications from such unrestricted access, Visual InterDev authors use run-time security.

As part of the development process, authors routinely test their Web applications by using a Web browser of their choice to read their Web pages. For this reason authors must be aware of run-time security considerations even while authoring.

Note If you use the same Web server for production as you do for development and testing, then the run-time security becomes the same as production security. However, if a Web application is put into production on a different server, the security settings on the production server should be reviewed and adjusted as necessary.

For example, if the production server has FrontPage extensions and the Web application was transferred to the production server using the Copy Web command, security settings need to be reestablished through Visual InterDev. If the production server is an Internet Information Server without FrontPage extensions, then the security should be established using Windows Explorer to manage file and directory ACLs on the Web applications content.

Run-time security is managed, first, by controlling whether all users or only registered users are permitted to browse a Web application, and, second, by controlling which users are granted browse access to those Web applications that only permit registered browsers.

Browsing Permissions

The FrontPage security model facilitates the use of two access levels for end users: unrestricted browsing and restricted browsing.

Unrestricted Browsing

Unrestricted browsing is the default setting for new Web applications. All users may read (browse) the content of a Web application that permits unrestricted browsing. End-user requests to read Web pages are satisfied anonymously; that is, end users are not required to identify themselves before fetching pages.

Note Because end-users are never required to identify themselves, it makes no difference whether they have browse access to the Web application. For an explanation of how the unrestricted browsing state is implemented using the Allow Anonymous setting of Internet Information Server, see "Run-Time Security Considerations," later in this chapter.

Restricted Browsing

Using the Web Permissions dialog box, any root Web or sub-Web application may be set to restrict browsing to only registered users. The FrontPage extensions remove read access for the anonymous user from all the content files of the Web application. An anonymous end-user's request to read a page of that Web application is then rejected by the Internet Information Server. End-users must authenticate their identity in the Windows NT domain, and the authenticated user account must have browse access to the Web application before the request will be satisfied. When a user is given browse access to a Web application, the FrontPage extensions grant the user account read access to all the content files of that Web application. (See "Run-Time Security Considerations," later in this chapter.)

For complete control over who can browse a Web application, restrict the Web application so that only registered users can browse it, and then give specific users and/or groups browse access to the Web application.

Run-Time Security Implementation Details

The FrontPage extensions manage run-time Web permissions using the underlying security model of the WWW server software. Here we consider only the case where this is Internet Information Server 3.0 running on Windows NT with the NTFS file system. Security on non-Internet Information Server servers is beyond the scope of this topic. Security features on non-NTFS file systems are severely limited.

FrontPage Run-Time Security

FrontPage manages run-time security differently than design-time security. This is because run-time HTTP requests to read pages are not directed to ISAPI DLLs, as are design-time requests. Internet Information Server goes directly to the file being requested and attempts to read it, as described below. For this reason FrontPage directly alters the ACLs on content files in a Web application in order to control run-time security.

Run-Time Security Considerations

At run time, you might want to allow everyone to read the Web applications pages or restrict access to certain authorized users.

Allowing Access to Any Web Visitor

You can set up your system to allow access to any Web visitor. The following table shows some typical settings you can use to control security for your Web application. Other settings and combinations may produce run-time errors for certain Web visitors.

Location for setting	Intranet setting	Internet setting
Operating system (Windows NT) / Web server	Allow anonymous	Allow anonymous
Folder	Permissions are handled through the Web server	Add Anonymous ID to appropriate User Groups, also permissions are handled through the Web server
Database	SQL Server — Standard	SQL Server — Standard
Web application	Nothing special	Nothing special
Page	Nothing special	Nothing special

Unrestricted Browsing

When a Web application is set to allow unrestricted browsing, FrontPage grants the anonymous user account standard read access to all the content files in that Web application.

Restricting Web Visitor Access

You can restrict access to your Web application by setting security options in several areas. Some developers set up a site that has one Web application with unlimited access for introductory and general content that links to another Web application that is available only to authorized visitors. Typically this second Web application requires the visitor to login so that an ID and password can be compared against a database of registered users.

You can use the following table to help you choose what security options are appropriate for your Web application.

Location for setting	Intranet setting	Internet setting
Operating system (Windows NT)/ Web server (IIS)	Windows NT Challenge/Response (NTLM) Add permissions for User Groups and/or individuals; user groups and password settings must match the operating system	Basic clear text Add anonymous user ID to appropriate user groups
Windows Explorer folder (OS)	Add user group or individuals to permissions list	Add user group or individuals to permissions list
Database	Standard	Standard
Web application	Global.asa file	Global.asa file — identify authorized users and store session state for a user
Page	Use .asp files and server script	Use .asp files and server script

Registered Users

When a Web application is set to allow only registered users to browse it, FrontPage removes all anonymous user account permissions from the Web application's content files. Since Internet Information Server impersonates the anonymous user during all requests that do not contain user credentials (when Allow Anonymous is checked), this causes such requests to fail initially and begin the authentication process described above.

Securing HTTP Transmissions (SSL)

SSL (Secure Sockets Layer) is a protocol that is used to protect HTTP transmissions from unwanted eavesdropping. Visual InterDev supports the optional use of SSL over the client/server link at design time and when performing a Web Copy operation. You can also use SSL during run time. You choose to use an SSL connection when you create a new Web project or initiate a Web Copy operation.

When an SSL connection is in use, all HTTP requests and replies are encrypted before transmission and decrypted after receipt. This process does require additional time and slows performance.

Except for this encryption/decryption step, all Visual InterDev operations take place exactly as they would over a non-SSL connection. You must obtain and install an SSL certificate on your Web server before you can initiate an SSL connection. For more information on obtaining and installing an SSL certificate on an Internet Information Server, see "Securing Your Site Against Intruders" in the Internet Information Server documentation.

Administration and Maintenance Considerations

In choosing which settings are appropriate for your Web application and system, you might also consider the administration requirements to maintain your security plan.

For example, if you choose to use Windows NT Challenge Response, the Web permissions are based on the individuals and user groups defined for that server. This setting works well for intranets because user profiles are likely to be setup and maintained on a regular basis and the quantity of user is known. Web permissions can be provided at the same time as network permissions.

If, however, your Web application is running on the Internet and you use the Windows NT Challenge Response, the number of potential requests for profiles could make user administration a full-time job for the system administrator. Also, operating systems typically have a limit on the number of user profiles you can specify.

Simplifying Run-Time Security

As was the case with design-time security, you can simplify the task of managing run-time security by creating a local group, say Web_Readers, on the Web server, and giving the group browse access to the root Web application. Be sure to restrict the root Web application so that only registered users can browse it. When new sub-Web applications are created, keep their default setting of inheriting permissions from the root Web application.

Now the task of permitting a new user to browse the Web applications on this server is reduced to simply adding the new user's name to the Web_Readers group using the Windows NT User Manager. For more information about adding groups, see "Setting Web Application Permissions" in Chapter 9, "Adding Security."

Managing Security for Multiple Web Applications

Permission settings are stored and applied on a Web-by-Web basis. You can save time by creating sub-Web applications that inherit the permissions from the root Web above it.

Any changes in root Web permissions apply to all sub-Web applications that inherit from the root Web. An administrator may not change permissions directly on a sub-Web application until it is set to have its own, unique permissions.

Managing Permissions Efficiently

For convenience, Windows NT permits the definition of named collections of users, called "groups." Once a group has been defined and users have been added to it, each user in the group can be granted or denied access to a resource by changing the group's permissions on that resource.

CHAPTER 7

Working Locally

In Microsoft Visual InterDev, you have two options when working online. You can work directly with the master Web files in master mode or you can develop your own working model of the Web application or its parts in local mode.

In local mode, you make changes to a write-enabled copy of the Web application without affecting the master Web application. You can use the local mode functionality to prototype new versions of your Web application without changing your master Web files or interfering with the work done by other developers.

In addition to local and master mode, if for some reason you need to work without a connection to the master Web server, you can choose to work offline. For example, you can place a local copy of your Web application on a laptop and work on the files offline. For more information, see "Working Offline," later in this chapter.

Developing Applications Independently

You might want to work on your Web application in isolation without changing your master Web application or being affected by changes to the master Web files made by other Web developers.

In Microsoft Visual InterDev, you can use local mode to make changes to a Web page and then test how your page works in the context of several other pages without impacting the master file or other developers. If your master Web server is your production server, any changes you make to the local Web files will not affect your Web visitors. You can open multiple files, make changes, test them as a local Web application, and then release your local copies to update the master Web application. Other developers or your visitors see the changes only after the files have been released.

You can use the following major steps to work locally and to identify other tasks you might want to do. Detailed procedures for each major step are provided in related topics.

To work locally

1. Open or create a project for the Web application. For more information, see "Creating a Web Project" in Chapter 1, "Web Project Management."
2. Make sure your project is set for local mode. For more information about selecting a mode, see the next section, "Specifying a Project Mode."
3. Make sure you have a local copy of the files you want to work on. For more information about getting the latest version or a local copy of a file, see "Getting Master Copies Locally," later in this chapter.
4. Open the files, edit the files, and save your changes. Since your project is in local mode, the changes are saved to your local Web application only; the master Web application does not change. For more information, see the procedures for editing and saving your changes in Chapter 2, "Web Basics."
5. If you have moved files or manually added links, you can do a quick visual check of the links using Link View. For more information about using Link View, see "Link Verification" in Chapter 11, "Site Design."
6. Preview the working version of the Web application in the Web browser. If you don't have a Web server on your developer workstation, the file is loaded with a file URL and server-script won't work. If you do have a local Web server, your .asp files use the local Web server to process the server script. A virtual root is automatically created and the page is loaded into the Web browser. For more information about previewing, see "Previewing Pages" in Chapter 2, "Web Basics."
7. Update the master Web application. When you are satisfied with your working versions, you can release your local copy to update the master Web application. For more information, see "Updating the Master Web Application," later in this chapter.

Specifying a Project Mode

You can switch between local and master modes for your project. Local mode is recommended because it allows you to work in "isolation" from the master server. Local mode ensures that your local copies of the files are not affected by updates made to the master Web application by other developers. This mode also keeps your changes from affecting others using the master Web application.

To specify local mode for your project

1. In the Project Explorer, select the project you want in local mode.
2. From the **Project** menu, choose **Web Project** and then choose **Working Mode**.
3. From the submenu, choose **Local**.

If you work alone and do not use the master server as your production server, you can work directly in master mode without impacting others using the application.

To switch to master mode

1. In the Project Explorer, select the project you want to switch to master mode.
2. From the **Project** menu, choose **Web Project** and then choose **Working Mode**.
3. From the submenu, choose **Master**.

Getting Master Copies Locally

To work and test locally, you need copies of your Web files in your local Web application. You can get copies of the master versions from the master Web server.

You have the option of getting a write-enabled or a read-only copy of the files. If you want to edit the files, get write-enabled copies of the files. For testing your Web application, you also need to get copies of related files you might want to use during testing. These related files can be either write-enabled or read-only.

To get a write-enabled copy of a file

1. In the Project Explorer, select the file you want to edit.
2. From the **Project** menu, point to **Web Files**, and then choose **Get Working Copy**.

   Note If you are using source control, you may need to choose **Checkout** from the **Source Control** submenu on the **Project** menu.

A write-enabled working copy of the file is placed in your local Web application. With write-enabled copies of your files in your local Web application, you are ready to begin editing, and saving files in your application. For more information about these operations, see Chapter 1, "Web Project Management"; Chapter 2, "Web Basics"; and Chapter 4, "Editing Basics."

To test pages with links specified with relative paths, you need to copy the related files from the master Web application to your local directory. For example, if you want to test a link in the file MyFile.htm, which points to the file TargetFile.htm, you need to have a copy of TargetFile.htm in your local Web application. Relative paths for links between files within the same application are recommended as they are portable if the Web application is copied to another Web server.

To get related files

1. In the Project Explorer, select the related files you want.
2. From the **Project** menu, point to **Web Files**, and then choose **Get Latest Version**.

   Note If you are using source control, you may need to choose **Get Latest Version** from the **Source Control** submenu on the **Project** menu.

The latest master version is copied to your local Web application. If the local file was read-only, it will be read-only in your local Web application. If you already had a write-enabled copy and the master version is newer, you will be prompted to merge the two versions, replace the local with the master, or skip updating and keep the local version.

You might want to place a copy of the entire Web application on your local computer. Depending on the size of your application, this operation may take quite a bit of time and space on your computer.

To copy the entire Web application to your local directory

1. In the Project Explorer, select the project you want to get.
2. From the **Project** menu, point to **Web Project**, and then choose **Refresh**.
3. From the **Project** menu, point to **Web Project**, and then choose **Get Latest Version**.

   Note If you are using source control, you need to select the files individually and choose **Get Latest Version** from the **Source Control** submenu on the **Project** menu.

If you have any write-enabled copies and the master version is newer, you will be prompted to select merge, replace, or skip.

Previewing a Web Application

When you are ready to test your local Web application, you can view its files in the Web browser. Previewing your local Web application allows you to test its functionality before releasing the copies to the master server. You can also test your links visually using Link View before running the application in a Web browser. For more information about testing links, see "Link Verification" in Chapter 11, "Site Design," and Chapter 16, "Maintaining Links."

The testing functionality available in your local Web application prototype depends on the system components on your local computer. You can always test your .htm files and your client script using the local Web application.

To fully test .asp files with server script, however, you need to have a local version of a Web server that supports .asp files. For more information about getting local versions, see "Getting Master Copies Locally," earlier in this chapter. For more information about what you can test, see "Test-Time Interaction of Components" in Chapter 6, "Web Project Concepts."

As long as you have a connection to the master Web server, you can still verify your links to other files without getting a local copy of those files. Even in local mode, Visual InterDev contacts the master Web server to verify links.

To test links while you are working offline, however, you need to get copies of the files to use in your local Web application because the connection to the master Web server is not available.

To preview the local Web application

- From the **View** menu, choose **View in Browser**.

If a Web server resides on the same computer as the local Web application, you can test your server script as well as your client script and HTML. If you do not have a Web server on the same computer, you can test only your client script and HTML. Server script requires a Web server for processing.

You might want to review how the current version of master Web application performs.

To preview the master Web application

- If your project is in local mode, you need to open your Web browser and specify the URL for the Web application on the master Web server.

  – or –

  If your project is in master mode, choose **View in Browser** from the **View** menu.

Updating the Master Web Application

After you have saved your changes to your local Web application and you are ready to propagate your changes to the master Web application, you can release the working copies.

To update the master Web application

1. In the Project Explorer, select the files you want to save to the master Web application.
2. From **Project** menu, point to **Web Files**, and then choose **Release Working Copy**.

Your local versions of the files are copied to the master Web server and become the current versions on the master Web application. If someone else has made changes to the master version since you got the copy you changed, the Merge dialog box appears and allows you to review differences and accept or reject the changes. For more information, see "Resolving Merge Conflicts" in Chapter 8, "Working with Multiple Developers."

You might not want your changes propagated to the master Web server. For example, you may have decided you like the way the current version handled a scripting challenge better than the changes you just made to the local file. You can discard your current write-enabled copy, leaving the master version in tact.

To discard changes to a local copy

1. In the Project Explorer, select the files you do not want to release to the master Web application.
2. From **Project** menu, point to **Web Files**, and then choose **Discard Changes**.

Your local copy of each selected file is replaced by the master version of the file.

Updating the Local Web Application

While working in local mode, your local copies of the Web application files are protected from the changes made by other team members to the master Web application. You might want to explore the changes made by other Web developers. For example, others may have made changes to existing files or added new files to the master Web application. If you want to get the latest file list from the master, you can refresh your project. If you want to get the latest versions of files that you already have locally, you can synchronize your local Web application with the master Web application. Also, if someone has enabled source control on the Web project since you last opened the project, you need to refresh your project to take advantage of the source control features.

To get changes to the project structure

1. In the Project Explorer, select the project you want to refresh.
2. From **Project** menu, point to **Web Project**, and then choose **Refresh Project View**.

   The file list in the Project Explorer and local read-only files are updated to reflect the current state of the application on the master server. Although the Project Explorer view changes, write-enabled copies of local files are not changed even if the master version of that file has been changed.

   For example, if you have a local copy of a write-enabled file that was deleted from the master application, your local copy remains as it was before the refresh. However, read-only copies of local files are updated to reflect any changes made on the master server. For example, such changes as newer versions, deleted files, and renamed files are made locally.

   Note Refreshing the project view does not place new local copies of write-enabled files on your machine. For information about getting the latest versions of files, see "Synchronizing Master and Local Files" in Chapter 10, "Managing Web Projects."

You might want to get updated files from the master server, but not the entire Web application. You can synchronize your local version of specific files with the master version.

To get changed files

1. In the Project Explorer, select the project with the files you want to get.
2. From **Project** menu, point to **Web Project**, and then choose **Synchronize Files**.

Working Offline

You might want to take your laptop to a meeting to demonstrate your current version of the application. You can incorporate suggestions and changes right at the meeting without a connection to your master Web server. With your project offline, you can open a project, edit local (working) copies of files, preview changes, etc., without a connection to the Web server.

To work offline

1. Open or create a project for the Web application. For more information, see "Creating a Web Project" in Chapter 1, "Web Project Management."
2. Make sure you have a local copy of the files you want to work on. For more information about getting the latest version or a local copy of a file, see "Getting Master Copies Locally" or "Updating the Local Web Application," earlier in this chapter.
3. Set your project to work offline. For more information, see the procedure below.

   When your project is offline, you work in the project just as you would for local mode, except that commands that update, restructure, or need information from the master Web application are not available. For example, you can't get the latest versions, release working copies, or move files in the project.
4. Open files, edit your files, and save your changes.

   Since you are working offline, the changes are saved to your local Web application only. For more information, see the procedures for editing and saving your changes in Chapter 2, "Web Basics."
5. Preview the local version of the Web application in the Web browser.

   If you don't have a Web server on your computer, the file is loaded with a file URL and server-script won't work. If you do have a Web server, your .asp files use the local Web server to process the server script. A virtual root is automatically created and the page is loaded into the Web browser. For more information about previewing, see "Previewing Pages" in Chapter 2, "Web Basics."

   If you have a database server, you can also perform database-related functions.
6. Set your project to work online.

   If your project is in local mode when you switch to offline, you might want to update the master Web application after you return online. If your project is in master mode, the master Web is automatically updated when you return online. For more information, see "Updating the Master Web Application," earlier in this chapter.

To set your project to offline

1. In the Project Explorer, select the Web project you want to take offline.
2. Make sure you have a local copy of the files you want to work on and related files you need for testing in your local Web application. For more information about getting the latest version or a write-enabled copy of a file, see the procedures in "Getting Master Copies Locally," earlier in this chapter.
3. From the **Project** menu, choose **Web Project**, choose **Working Mode**, and then choose **Work Offline**.

   Note If you want to make changes to the site structure, rename files, or move files, you need to be online.

CHAPTER 8

Working with Multiple Developers

Whether you are working on a team or by yourself, you can track and save your changes to files easily with Microsoft Visual SourceSafe. When using source control on your Web applications, you might perform some tasks every day, some occasionally, and others only once.

Setting Up Source Control on a Web Server

To install Microsoft Visual SourceSafe and set it up to work with Microsoft Visual InterDev, you need to complete several procedures using a couple of different interfaces.

The first step involves installing Visual SourceSafe on the master Web server using Microsoft Windows and Visual SourceSafe Setup.

To set up source control on your Web server

1. Install Visual SourceSafe on the master Web server. For more information, see "Installing Visual SourceSafe on a Web Server," in the next section.
2. In Visual SourceSafe, grant Visual SourceSafe permission to users. For more information, see "To grant permissions to a user," in the next section.
3. For Web servers on Microsoft Windows NT, add permissions to the anonymous user account. For more information, see "To add permissions for the Anonymous user account," in the next section.
4. Share source control between Web projects.

Installing Visual SourceSafe on a Web Server

When installing Visual SourceSafe, be sure to install the Visual SourceSafe server on the same computer as the master Web server. Make sure the option for integration with other visual products is selected. Without that option set, Visual SourceSafe runs independently and does not recognize the commands used by Visual InterDev. If you plan to check files in and out from within Visual InterDev, you don't need to install the Visual SourceSafe client on your local computer.

The following procedure assumes you have Microsoft Internet Information Server and FrontPage Server Extensions already installed on your master Web server.

To install Visual SourceSafe on the master Web server

1. Run the Visual SourceSafe setup program.
2. Choose **Custom** and make sure that you are installing at least the following components:
   - Create SourceSafe Database
   - Administrative Programs
   - Enable SourceSafe Integration
3. If you are running Windows 95 and Microsoft Personal Web Server, download and install the Distributed Component Object Model (DCOM).

If you need to download the Distributed Component Object Model (DCOM), go to the Microsoft corporate Web site at http://www.microsoft.com/search/default.asp and search for "DCOM."

After you've installed Visual SourceSafe on a Web server, you must grant read/write permissions to all users whom you want to be able to author files using Visual InterDev or Microsoft FrontPage.

To grant permissions to a user

1. Run the **Visual SourceSafe Administrator** application. This application should appear in the **Programs** section of the **Start** menu. If it doesn't, check that you installed all of the components listed above.
2. Select the **Users** menu and then choose **Add User**.
3. Enter the user's name and, if desired, leave the **Password** box blank. Make sure the **Read-Only** option is not selected.

   Note If you assign a password, the user will have an additional login step to complete when using the project.

In addition to granting read/write permissions to specific users, if you have installed Visual SourceSafe on a Windows NT server, you must also add permissions for the Anonymous user account.

To add permissions for the Anonymous user account

1. On the server, open **Internet Information Manager**.
2. In the Microsoft Management Console, choose **Internet Information Server**, and then select the computer that is running the **Visual Source Safe Administrator**.
3. Select **Default Web Site** and open its properties dialog box.
4. On the **Directory Security** tab, choose **Edit** to change the **Authentication Methods**.

5. In the **Anonymous User Account** dialog box, copy the username.

   Note By default, this property setting is IUSR_*machinename*, where *machinename* is the name of the server where you have installed the Visual SourceSafe Administrator.

6. In the **Visual SourceSafe Administrator** application, create a user account for the **Anonymous** user by choosing **Add User** from the **Users** menu and pasting in the value of the **UserName** property. Leave the **Password** box blank and make sure that the **Read-Only** option is not selected.

When Visual SourceSafe tracks changes, it uses the operating system to identify and record who made the changes.

Certain operating systems can only recognize and record the anonymous user name for changes. If your Web server is running Windows 95, or Windows NT using the File Allocation Tables (FAT) file system, then all files checked out through Visual SourceSafe will always be checked out to the same user account. This user account might not represent the user performing the operation.

On a Windows NT FAT system, the anonymous user account performs all source control operations on the server. On a Windows 95 system, the user account specified when Windows 95 was started performs all source control operations on the server.

Adding Source Control to a Web Application

Once you've installed and set up Microsoft Visual SourceSafe, you can enable source control for a Web application using any Web project that references that application. Only one developer needs to enable source control for the application.

Visual SourceSafe provides two ways to control source files.

- **Exclusive Checkout.** The default option which allows only one user to check out a file at a time.
- **Multiple Checkouts.** An option that allows more than one person to check out the same file at a time. For information about resolving conflicts resulting from multiple checkouts, see "Resolving Merge Conflicts," later in this chapter.

After a developer has enabled source control for a Web application, other developers with open projects that reference the Web application must either refresh or reopen their Web projects for source control to take effect on those projects.

To enable source control for a Web application

1. In Visual InterDev, open or create a Web project that references the Web application you want to place under source control.
2. In the Project Explorer, select the project you want to use with source control.

3. From the Project menu, select Source Control, and choose Add to Source Control.
4. In the Enable Source Control dialog box, verify that the project name is the one you want for the Source Control Project, and then click OK.

   You can enter a different name for the Visual SourceSafe project if you want. If you use a different name, it must be preceded by the dollar sign ($) and forward slash (/). For example: $/MyWebApplication.

If you want to enable multiple checkouts, you need administrator privileges to the Visual SourceSafe Administrator on your master Web server.

To enable multiple checkouts for a Visual SourceSafe project

1. In Visual SourceSafe Administrator, from the **Tools** menu, choose **Options**.
2. On the **General** tab, choose **Allow multiple checkouts**.
3. Click **OK**.

If you do not know whether a Web application has source control enabled, you can check the property sheet for your project that references that Web application.

To determine whether source control has been enabled for a Web application

- In the Project Explorer, check for a lock icon next to the project name.

  If the project is under source control, the lock icon will appear next to the project name. If the project is not under source control, there will be no lock icon.

You can also disable source control for a Web application from any Web project that references that Web application.

To disable source control for a Web application

1. Open or create a new Web project that references the Web application you want to remove from source control.
2. In the Project Explorer, select the project you want to remove from source control.
3. From the **Project** menu, select **Source Control**, and then choose **Disconnect Web Project**.

When you disable source control for a Web application, the Visual SourceSafe project created when you first enabled source control for that Web application remains on the Visual SourceSafe server. This means you can re-enable source control and choose that Visual SourceSafe project for your Web application.

Re-enabling source control after disabling it may cause some unexpected results. For example, if you disable source control, then delete files from the Web application, then re-enable source control, the files that you deleted will show up in your Web project again. This happens because the files were deleted outside of source control and so the original Visual SourceSafe project still shows them as part of the project. You can remove the Visual SourceSafe project by deleting it from within Visual SourceSafe Explorer.

Working Under Source Control

Once a Web application has source control enabled, you check your files in and out instead of getting and releasing working copies as you did before.

By checking files in and out through the Web projects in Microsoft Visual InterDev, team members have more control over when the master Web application and Visual SourceSafe repository are updated with newer versions from your local Web applications.

Note If you use the Visual SourceSafe client instead of the Visual InterDev features to perform source control tasks, be sure to refresh your Web project in Visual InterDev before continuing with your work.

Adding Files to a Source Control-Enabled Web Application

Once you've enabled source control for a Web application, any files that you create or add to that Web application using Visual InterDev are added automatically to the Visual SourceSafe project. This includes files that are created in any Visual InterDev editor or files that you create in other applications and add to a Web project.

Note If you add files directly to the project directory on the server, through the file system, those files will not be automatically added to the Visual SourceSafe project. To add those files to the Visual SourceSafe project, you need to refresh the project view by synchronizing the local Web application with the master Web application.

To synchronize the local and master Web applications

1. In the Project Explorer, select the Web project.
2. From the **Project** menu, select **Web Project**, and then choose **Synchronize Files**.

   Your view of the Project Explorer is refreshed and all files contained in the project directory appear in the Project Explorer.

   If you see an icon with a red "X," the associated file is not in the Visual SourceSafe project.

If you have a file in your Web project that for some reason is not under source control, you can add it manually.

To add a file manually to source control

1. In Visual InterDev, select the file in the Project Explorer that you want to add to source control.
2. From the **Project** menu, select **Source Control**, and then choose **Add to Source Control**.

To remove files from source control

1. In Visual InterDev, select the file in the Project Explorer that you want to add to source control.
2. From the **Project** menu, select **Source Control**, and then choose **Remove from Source Control**.

For more information about using Visual SourceSafe, see "Troubleshooting Source Control," later in this chapter.

Checking a File Out

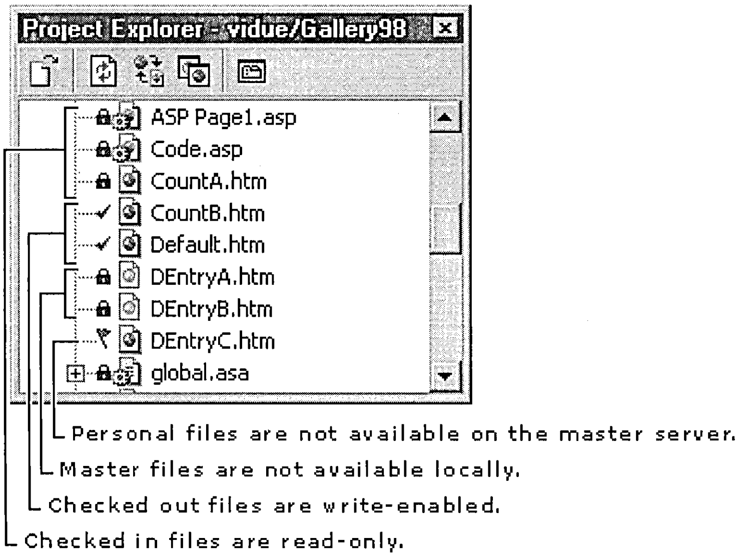
You can check out a file directly in the Visual InterDev Project Explorer without running the Visual SourceSafe client. By default, when you check out a file, no one else can get a write-enable copy of the file. If you request a local copy of a file that another user has already checked out, Visual SourceSafe displays a message indicating that another user has the file checked out, and only allows you to get a read-only version of the file.

To check out a file

1. In the Project Explorer, select the file you want to check out.
2. From the **Project** menu, select **Source Control**, and then choose **Check Out**.

In the Project Explorer, files that are write-enabled in the local directory are shown in color, while read-only copies and files without any local copy appear in shades of gray.

File Status in the Project Explorer


Project Explorer - vidue/Gallery98
ASP Page1.asp
Code.asp
CountA.htm
CountB.htm
Default.htm
DEntryA.htm
DEntryB.htm
DEntryC.htm
global.asa
Personal files are not available on the master server.
Master files are not available locally.
Checked out files are write-enabled.
Checked in files are read-only.

When a file is checked out, you get a write-enabled copy of the file and Visual SourceSafe marks the file as checked out in the Project Explorer. When you check in the local copy, the file is marked as checked in and all changes to the files are saved to the master Web server.

By default, you are prompted to choose between a write-enabled or a read-only copy of the file. If you know you always want one or the other, you can set Visual SourceSafe to always provide one or the other.

To specify how files are checked out

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Project** and then **Web Projects**.
3. From the **When opening a file** box, choose the source control option you want.
4. Click **OK**.

Allowing Multiple Checkouts

If your team members want to be able to work on the same file simultaneously, your Visual SourceSafe administrator can allow more than one person to check out and edit a file by allowing multiple checkouts. This option allows more than one person to work on a file at the same time.

When multiple checkouts are allowed, each user who checks out the file gets a write-enabled copy of the file. If you are the second person to check the file in, you can review the differences and choose to merge the two versions.

To allow multiple checkouts

1. On your master Web server, open the Visual SourceSafe Administrator.
2. From the **Tools** menu, choose **Options**.
3. On the **General** tab, choose **Allow Multiple Checkouts**.
4. Click **OK**.

Getting the Master Version of a File

If you want to review the latest version of a file without making changes to the file, you can get a copy of the master version of a file without checking the file out. When you get the latest version of a file, a read-only copy of the master file is placed on your local Web server. For example, you might want to get the latest version of files that link to the files you are developing.

To get a copy of the master version of a file

- From the **Project** menu, select **Source Control**, and then choose **Get Latest Version**.

  If you already have a local copy of the file, Visual InterDev displays a dialog box in which you choose to keep the existing local file or to replace it with the master copy. A copy of the requested file is placed on your local Web application.

  Note If the local directory is not synchronized with the master Web directory, this command may fail. If this happens, use the **Refresh** command on the **View** menu to synchronize the project with the Web directory, and then try the command again.

Checking Files In

When you check in a file, you replace the current version of the file in Visual SourceSafe with a revised version of the file from your local Web directory. Visual InterDev automatically interacts with Visual SourceSafe to check the file in; you don't have to perform any additional steps.

To check in a file

- From the **Project** menu, select **Source Control**, and then choose **Check In**.

  Note The Check In command is different from the Save command. The Save command updates the file in the local or master Web applications, but does not update the source control version in Visual SourceSafe.

After the file is checked in, the version you originally checked out is placed in the Visual SourceSafe history for that file.

If you made changes to the file, but do not want to save them as the latest version in Visual SourceSafe, you can use the Discard Changes command to abandon all changes made since getting the local copy.

To discard changes made to a file

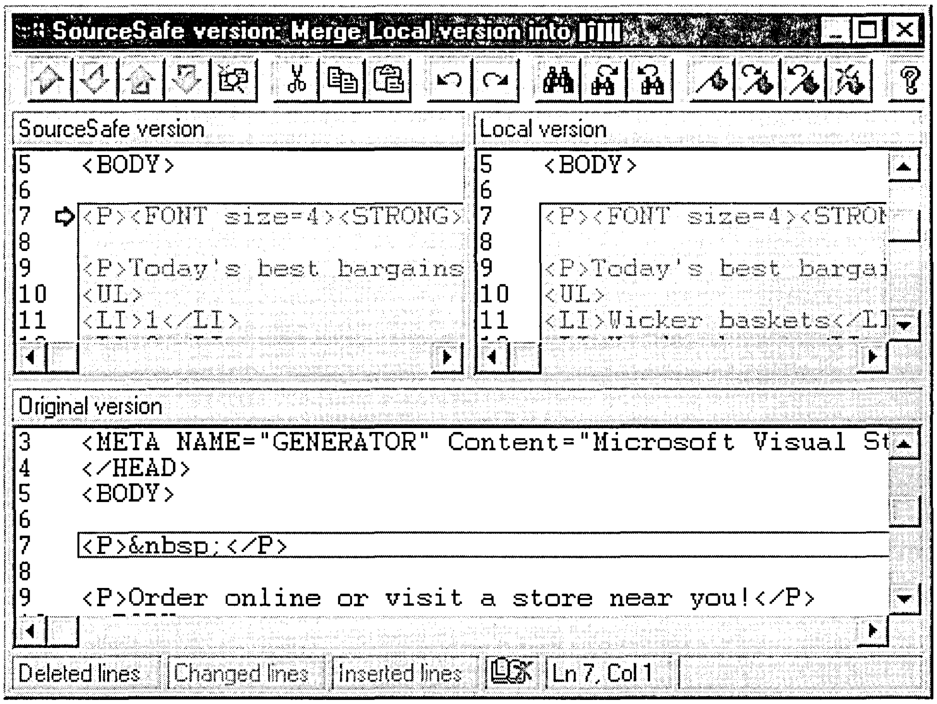
- From the **Project** menu, select **Source Control**, and then choose **Undo Check Out**.

Resolving Merge Conflicts

When multiple checkouts are allowed, you might need to resolve merge conflicts if other team members have made changes to the file you are checking in.

When you check in a file in that has been altered by another user, Visual InterDev detects the merge conflicts and displays the Merge dialog box for you to review the two different versions. For example, you check out a file to change some client script and a Web author checks out the same file to rewrite some of the content. If the Web author checks in the modified file before you, the Web author's version of the file becomes the new master version. When you check in your copy of the file, the Merge dialog box appears allowing you to merge the two versions and save the merged file as the latest version.

If someone made changes to the master version after you got the copy you changed, the Merge dialog box appears when you release the your local copy. In the Merge dialog box, you can review differences between the files and accept or reject the changes.


SourceSafe version: Merge Local version into [illegible]

SourceSafe version

```
<BODY>

<P><FONT size=4><STRONG>

<P>Today's best bargains
<UL>
<LI>1</LI>
```

Local version

```
<BODY>

<P><FONT size=4><STRON

<P>Today's best bargai
<UL>
<LI>Wicker baskets</LI
```

Original version

```
<META NAME="GENERATOR" Content="Microsoft Visual St
</HEAD>
<BODY>

<P>&nbsp;</P>

<P>Order online or visit a store near you!</P>
```

Deleted lines | Changed lines | Inserted lines | Ln 7, Col 1

For example, in the Visual SourceSafe version is the master version that someone saved to the server. The local version reflects the latest local copy. The Original version pane allows you to edit the file and shows the merge decisions you have made.

To merge the files

1. In the bottom pane, locate and select the conflict area you want to resolve.
2. Right-click and from the menu, choose the changes you want to merge.
3. Close the dialog box and choose Save.

   Both the master and the local copies reflect the changes you saved.

You can also view differences by comparing the version in Visual SourceSafe and the local version.

To manually view differences

- From the **Project** menu, select **Source Control**, and then choose **Compare Versions**.

For more information about comparing files, see the **File Difference Options** dialog box.

Troubleshooting Source Control

If you experience difficulty in using Microsoft Visual InterDev with Microsoft Visual SourceSafe, you may need to make changes outside of Visual InterDev in the applications that support Visual InterDev. You can check the following items to resolve issues with enabling and using Visual InterDev with Visual SourceSafe.

Appropriate Login and Permissions

- On your local computer, make sure that you are logged in as yourself, at your client machine. If someone else has used your computer and doesn't have the same permissions as you, you may be denied permissions. To solve the problem, log in as yourself to get appropriate access to files.
- In the Visual SourceSafe Administrator, make sure that you are entered as a valid user. Also, make sure your server's anonymous user is in the user list. By default, the anonymous user name is IUSR_<*machinename*>. If you don't know your computer's anonymous user name, you can find it in the Internet Information Server configuration in the properties for the WWW service.
- In the Microsoft Internet Information Server Manager, make sure that either Basic authentication or NT Challenge/Response is enabled. If the No Authentication option is selected, access is not available to modify projects.

Proper Software Installation

- On your Web server, check that you have installed the FrontPage Server Extensions, Active Server Pages, and Visual SourceSafe. If Visual SourceSafe was installed using the Typical or Client installations, you need to reinstall and use the "Custom" or "Server" installation, and select the option to "Enable Source Safe Integration."

Visual SourceSafe Connection

If you have Visual SourceSafe installed on your local computer in addition to the Web server, make sure the LAN connection between your computer and the server is working and that the appropriate permissions are set.

You do not need Visual SourceSafe on your local computer; you only need Visual InterDev. If you want to use the Visual SourceSafe Explorer to work with your Web files directly in the VSS database, you can install the Visual SourceSafe client on your local computer.

CHAPTER 9

Adding Security

Security can be added to your Web application at many levels and depends on your system and the users you want to accommodate.

For a complete understanding of security options affecting your Web application, read about the security features of Microsoft FrontPage, your Web server such as Microsoft Internet Information Server, and your operating system such as Microsoft Windows NT.

If you use a database in your Web application, you might want to become familiar with the security models of ODBC databases, such as Microsoft Access or SQL Server.

Adding Security Pages

You can combine HTML forms with client and server script to secure your site. Since ASP pages generate what your visitor sees dynamically before sending it to the browser, you can also control what code and text is visible as source code in your visitor's browser. Typical security pages include login or registration pages. For example, you can add login pages that control entrance to the rest of the application.

The actual pages and script you add to establish security depend on what your design goals are. The following procedure provides one way to implement security pages on your site.

To add security pages

1. Add an HTML form to your main site page that gathers user identification and password information. For details, see "Gathering Information with HTML Forms" in Chapter 25, "Scripting with HTML Elements."
2. In the main site page, add client script that verifies that values were entered in the form. For details, see "Adding Scripts" in Chapter 4, "Editing Basics."
3. In a database, add a table that stores the user identification and password information.
4. In the project, add a connection to the database and table. For details, see "Connecting to a Database" in Chapter 3, "Database Basics."
5. Create an ASP page that retrieves a record based on the user input and compares the input with the values in the database.
6. Add server script that handles failed logins and successful logins that link to the site.

Setting Web Application Permissions

Every Web application on the Web server has permission settings that identify authorized users and specify their privileges. In Microsoft Visual InterDev, you can set the design-time permissions for your Web application. By default, a new Web application inherits the same permissions as the root directory. You can customize these permissions, and then control the permissions for individual users and for groups.

Setting Web application permissions involves the following:

- Customizing Permissions for a Web Application
- Setting Permissions for Individuals
- Setting Permissions for Groups
- Setting Permissions for Computers
- Simplifying Design-Time Security
- Revoking Permissions

Customizing Permissions for a Web Application

You can set unique permissions for the Web application of the currently active Web project. Unique permissions must be set for a Web application before you can modify its users and groups.

To set unique permissions for a Web application

1. In the Project Explorer, select the project you want to set permissions for.
2. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
3. On the **Settings** tab, select **Use unique permissions for this Web application.** This specifies that the current Web application does not inherit its permissions setting from the root Web application.
4. Choose **Apply.**

   Note You need to apply the change to unique settings in this tab before the other tabs are available for changes.

Setting Permissions for Individuals

After you have set custom permissions for your Web application, you can add individual users and control their permissions.

To add users to a Web application

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Settings** tab, select **Use unique permissions for this Web application** and then **Apply**.
3. On the **Users** tab, choose **Add**.
4. In the **Add Names** box, type the domain and user name for the new user in this format: *domain\username*.

   – or –

   In the **Obtain list from** box, select the domain and add users from the **Names** field.
5. In the **New users can** box, select the permission level of the new users.

   Note that the selected permission level applies to all new users in the **Add Names** field.
6. Choose **Apply**.

You can edit the permissions for each user.

To change permissions for a user

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Users** tab, select the user to edit and choose **Edit**.
3. In the **Edit Users** box, select the new level of permission for the user.
4. Choose **Apply**.

Setting Permissions for Groups

You can control the read and write permissions granted to groups of users.

Note If your master Web server is on an operating system using the File Allocation Table file system (FAT), such as Windows 95, instead of NT File System (NTFS), you cannot change the permissions for groups.

To add groups of users to a Web application

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Settings** tab, select **Use unique permissions for this Web application** and then **Apply**.
3. On the **Groups** tab, choose **Add**.

4. In the **Add Names** box, type the domain and group name for the new groups in this format: *domain\group name*.

   – or –

   In the **Obtain list from** box, select the domain and add groups from the **Names** field.
5. In the **New users can** box, select the permission level of the new groups. Note that the selected permission level applies to all new groups in the **Add Names** field.
6. Choose **Apply**.

Once a group has been given permissions, you can change the permissions of the group.

To change permissions for a group

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Groups** tab select the group to edit and choose **Edit**.
3. In the **Edit Groups** dialog box, select the new level of permission for the group.
4. Choose **Apply**.

Setting Permissions for Computers

You can control the read and write permissions granted to computers based on an IP address.

> **Note** If your master Web server is on an operating system using the File Allocation Table file system (FAT), such as Windows 95, instead of NT File System (NTFS), you cannot change the permissions for computers.

To add a computer to a Web application

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Settings** tab, select **Use unique permissions for this Web application** and then **Apply**.
3. On the **Computers** tab, choose **Add**.
4. In the **IP Address** boxes, type the address for the computer.

   > **Note** You can specify a group of computers that have the same initial address by using an asterisk (*).
5. In the **Computer permissions** box, select the permission level for the computer.
6. Choose **Apply**.

Once a computer has been given permissions, you can change the permissions of the computers.

To change permissions for a computer

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. On the **Computers** tab select the computer or computer mask to edit and choose **Edit**.
3. In the **Edit Computer** dialog box, select the new level of permission for the group.
4. Choose **Apply**.

Simplifying Design-Time Security

You can simplify design-time security easily by modifying Web permissions once and then using the Microsoft Windows NT User Manager utility for subsequent security changes.

> **Note** If your master Web server is on an operating system using the File Allocation Table file system (FAT), such as Windows 95, you cannot change the permissions for groups.

To simplify design-time security

1. In the Windows NT User Manager on your Web server, create two new local groups of users.

   For example, you might create one for administrators called Web_Admins and one for developers called Web_Devs.
2. In Visual InterDev, add the new groups to the Web permission.

When you create new Web applications on this server, you can choose to have the new Web application inherit the permissions of the root directory.

You can grant new users administer or author access to the Web applications on this server by simply adding their names to the appropriate group using the Windows NT User Manager, without reinvoking the Web Permissions dialog box.

However, if you want to maintain a single sub-Web application with different permissions than the other applications on the server, you must set it up to have unique permissions and then manage its users and groups separately.

Revoking Permissions

You can revoke the permissions that an individual or group has by deleting them from the user or group lists.

To delete user or group permissions on a Web application

1. From the **Projects** menu, choose **Web Project** and then **Web Permissions**.
2. Select the user to remove from the **Users** tab or the group to remove from the **Group** tab.

   Note If your master Web server is on an operating system using the File Allocation Table file system (FAT), such as Windows 95, instead of NT File System (NTFS), you cannot change the permissions for groups.
3. Choose **Remove**.
4. Choose **Apply**.

Connecting to a Proxy Server

If your system allows you to connect to the proxy server, you can connect to it using Microsoft Visual InterDev. You might want to work on Internet Web applications on a Web server outside of your intranet. For example, you might need to change some Web files located on a Web server beyond your corporate firewall.

With Visual InterDev, you can connect to the proxy server in order to access the Internet. The specific connection and capabilities available to you depend on the system you are working on and the one you want to connect to.

You can either specify that the project use the same proxy settings as the Internet settings for the operating system or you can specify a different proxy server.

To specify the same proxy server as operating system

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Projects** and then **Web Proxy Settings**.
3. On the **Web Project Proxy** tab, select **Use system settings**.

The settings for the system appear on the tab, but are read-only. To change the actual setting, you need to change the Internet settings for the operating system. For example, if you were using Windows NT Workstation, then you would go to the Control Panel and change the proxy server settings in the Internet Properties dialog box.

If you do not want to use the same proxy server as the operating system, you can specify a different one for the Web project to use.

To specify a proxy server for only the Web project

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Projects** and then **Web Proxy Settings**.
3. On the **Web Project Proxy** tab, clear **Use system settings.**
4. Enter the name of the proxy server you want to use in the **HTTP Proxy** box.
5. In the **List of hosts without proxy** box, enter any IP addresses for servers that you want to access without going through the proxy server.
6. Select the **Do not use proxy server for local (intranet) address** box, if you want local addresses to be processed without using the proxy server.

   Note When you create or copy a Web application, you can specify the target Web server using either the server name or by explicitly entering its IP address. If using the IP address instead of the name, and it is a local or Intranet address, you need to enter the IP address in the List of hosts without proxy box. Although an IP address may identify a local address, IP addresses are sometimes processed through the proxy server anyway and may generate an authentication error.

Visual InterDev uses the Web project proxy server that you specify only for operations on files in the Web project. You can separately specify a proxy server to be used by a Web browser. If you want to specify a proxy server for Microsoft Internet Explorer, you can do so on the Connection tab of the Internet Explorer Internet Options dialog box.

CHAPTER 10

Managing Web Projects

As you work on your Web application files, you might want to change the structure of your project or make copies to other servers.

Synchronizing Master and Local Files

Since Microsoft Visual InterDev 6.0 keeps two copies of the application while you are working, you may find that some files are on the local but not the master server and vice versa. For example, if you are working with multiple developers in local mode or offline, you might to need make sure that your local file list matches the master application.

Tasks you may want to perform:

- **Compare a master copy to a local version** to view the differences between the two files.
- **Update the master application** to reflect changes you have made to the local application. If you are in master mode, this happens automatically.
- **Synchronize the local file set with the master**. If you are working with multiple developers and want to refresh your local version of files others have worked on, you can get the latest version of read-only files currently in your local application.
- **Refresh the Project Explorer** to reflect the current list of files in the master and local applications. Although you may not need the local application to include all of the files in the master, you might want to see the entire file set that makes up both applications.

  Files on the master server that are not in your local directory are called master files. Local files that are not yet copied to the master directory are called personal files.

Comparing Master and Local Copies

You can look at the differences in source code between the master and local copies of a file.

To compare the local and master copies of a file

1. In the Project Explorer, select the file you want to compare.

   Note If you are working in master mode, don't save the file before comparing. If you do save, the changes will be automatically saved to the master server and the files will be identical.

2. From **Project** menu, select **Web Files**, and then choose **Compare to Master Web**.

   The File Difference dialog box appears with the differences highlighted in gray. If the files aren't different, a message states that they are identical.

For more information about comparing files, see the File Difference dialog box. If you are using source control with your project, see "Resolving Merge Conflicts" in Chapter 8, "Working with Multiple Developers."

Updating the Master Application

After you have saved your changes to your local Web application and you are ready to add your changes to the master Web application, you can release the local copies.

To update the master Web application

1. In the Project Explorer, select the files you want to save to the master Web application.

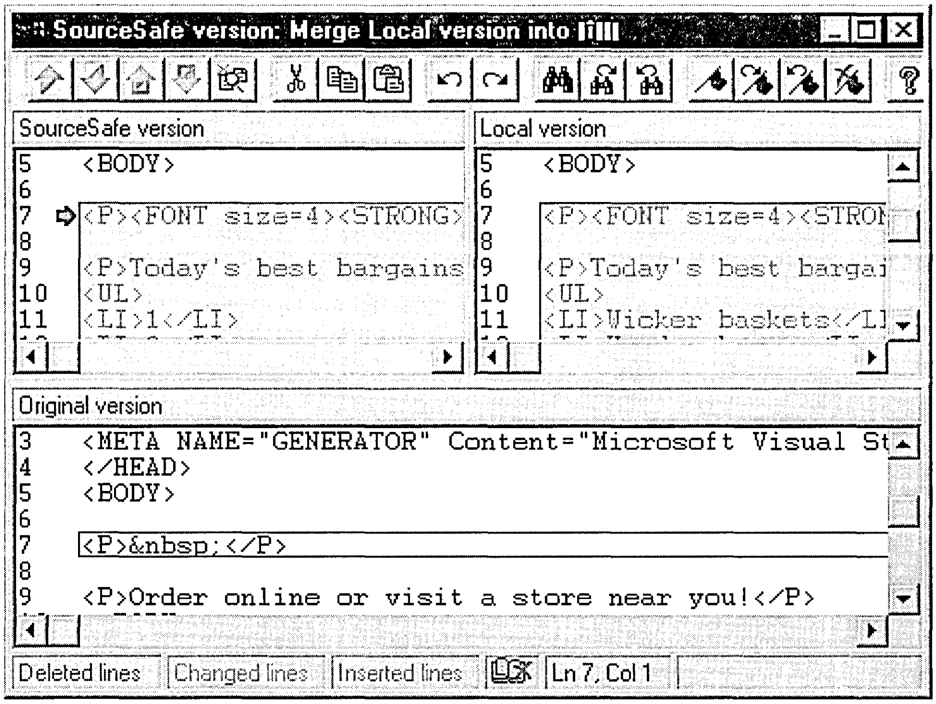
2. From **Project** menu, select **Web Files**, and then choose **Release Working Copy**.

   Note If your project is under source control, choose **Check In** from the **Source Control** submenu on the **Project** menu.

Your local versions of the files are copied to the master Web server and become the current versions on the master Web application.

If someone made changes to the master version after you got the copy you changed, the Merge dialog box appears when you release the your local copy. In the Merge dialog box, you can review differences between the files and accept or reject the changes.

For example, in the following picture, the SourceSafe version is the master version that someone saved to the server. The local version reflects the latest local copy. The Original version pane allows you to edit the file and shows the merge decisions you have made.



```
<BODY>

<P><FONT size=4><STRONG>

<P>Today's best bargains
<UL>
<LI>1</LI>
```

```
<BODY>

<P><FONT size=4><STRO

<P>Today's best bargai
<UL>
<LI>Wicker baskets</LI
```

```
<META NAME="GENERATOR" Content="Microsoft Visual St
</HEAD>
<BODY>

<P>&nbsp;</P>

<P>Order online or visit a store near you!</P>
```

To merge the files

1. In the bottom pane, locate and select the conflict area you want to resolve.
2. Right-click and from the menu, choose the changes you want to merge.
3. Close the dialog box and choose Save.

   Both the master and the local copies reflect the changes you saved.

You might not want your changes propagated to the master Web server. For example, you may have decided you like the script in the current version better than the changes you just made to the file. You can discard your current local copy, leaving the master version intact.

To discard changes to a local copy

1. In the Project Explorer, select the files you do not want to save to the master Web application.
2. From **Project** menu, select **Web Files**, and then choose **Discard changes**.

   Note If your project is under source control, choose **Undo Checkout** from the **Source Control** submenu on the **Project** menu.

Your local copy of each selected file is replaced by the master version of the file.

Synchronizing the Local Web Application

If you want to get the latest file list from the master Web application, you can refresh your project. If you want to get the new versions of read-only files that you already have locally, you can synchronize your local Web application with the master Web application. To replace your write-enabled files with master copies, you can get the latest versions.

To get changed files from the master Web application

1. In the Project Explorer, select the project with the files you want to get.
2. From **Project** menu, select **Web Project**, and then choose **Synchronize Files**.

Refreshing the Project Explorer

The Project Explorer shows the list of files for the Web application whether they are in the master or local Web application. If someone has added, deleted, moved, or renamed files from the master server, your any changes made to the master application.

Also, if someone has enabled or disabled source control on the Web project, you need to refresh your project to see those changes.

To view the current list of files in both the master and local applications

1. In the Project Explorer, select the project you want to refresh.
2. From **Project** menu, select **Web Project**, and then choose **Refresh Project View**.

Copying Web Projects

You can copy an entire Web application to another server or duplicate the application on the same server under a new Web application name. To copy to another directory or another server, you'll need administrator-level privileges on that server in order to create a new directory. If the destination server does not have Microsoft FrontPage extensions, you need to use Posting Acceptor 2.0. Posting Acceptor 2.0 is available in the Enterprise Edition of Visual Studio.

To copy a Web application

1. In the **Project Explorer**, select the project that points to the Web application you want to copy.
2. From the **Project** menu, choose **Web Project** and then **Copy Web Application**.
3. In the **Copy Project** dialog box, choose the copy of the application you want to copy. Typically, you would choose the master version.
4. In the **Server name** box, enter the name of the destination Web server.
5. In the **Web project** box, enter the name you want in the URL.

6. Clear the **Copy changed files only** checkbox.

   Note If the destination server has a Secured Socket Layer (SSL) certificate and you want to use secure HTTP transmissions during the copy operations, select the **Connect using Secured Socket Layer** box. For more information about Secured Socket Layers, see "Securing HTTP Transmissions (SSL)" in Chapter 6, "Web Project Concepts."

7. Check the options that you want to enable and then choose **OK**.

   - To copy only files updated or added since the last time the Web was copied, choose **Copy changed files only**.
   - To copy the Web application as a subfolder to an existing destination Web application, choose **Add to an existing Web project**. This option merges the files and folders of the Web application you are copying into the destination Web application. The destination Web application must already exist; if it does not, you get an error.
   - If the Web application is the root application and you want to copy all of the Web applications on the source server to the destination server, choose **Copy child Webs**.
   - If you have components that were marked for server registration and you want to register them on the destination server when they are copied, choose **Register server components**. See "Deploying an Integrated Web Solution" in Chapter 30, "Deploying and Maintaining Web Applications."

     Note If you have components marked for server registration in their property pages and you clear this option, those components will be unregistered on the destination server after the operation is complete.

A new application root is created on the Web server and the files in the Web application are copied to that new folder. The name you specified in the Copy Project dialog box becomes part of the application's URL.

You can now perform a final verification of the application. The entire Web application, except its security settings, is duplicated on the destination server under the new name. The newly copied Web application inherits the destination's root project security settings. For more information about security settings, see Chapter 9, "Adding Security."

After the copy is complete, the source project remains open and unchanged in the current solution. To access the newly copied Web application, you need to create and open a project that references the copy on the destination server.

Reorganizing Project Structure

You can add, delete, rename, or move files or folders in the Project Explorer. When you rename or move items, the changes are applied immediately to both the master application and the local application whether your project is in master or local mode. Other developers who have the project open can see the changes applied to the master application after they refresh their Project Explorer.

Caution If the local application is out of synch with the master version, operations to reorganize the project may fail. You might want to refresh or synchronize your project before reorganizing the project.

Reorganizing a project also affects the links in the files. Files that you rename, delete, or move are called target files. Files that link to the target files are called referring files.

You can choose whether referring files are updated to reflect the changes you make to target files. By default, you are prompted to update the referring files. For more information about setting link repair options, see "Repairing Links" in Chapter 16, "Maintaining Links."

To delete files or folders

1. In the Project Explorer, choose files or folders you want to delete.
2. From the **Edit** menu, choose **Delete**.

You might need to change the names of files or folders.

To rename files or folders

1. In the Project Explorer, choose the file you want to rename.
2. On the **File** menu, choose **Rename**, and then type the new name.

   Note If you have a working copy of a file, you need to release it before you are allowed to rename it.

You might want to change the location of files or folders in your Web application.

To move files or folders

1. Make sure working copies of the files are released.
2. In the Project Explorer, select the files or folder and then drag them to the destination within the Project Explorer.

You might want to copy files or folders in your Web application.

To copy files or folders

1. In the Project Explorer, select the files or folder you want to copy.
2. Press CTRL and then drag them to the destination within the Project Explorer.

PART 3

Designing Sites

Part 3 provides information about designing the visual appearance and navigational structure for a Web application, as well as viewing and maintaining links.

Chapter 11 Site Design

This chapter introduces the Visual InterDev tools and techniques for Site Design, Link Verification, and Site Consistency.

Chapter 12 Designing a Web Site

You can rapidly design and create a new Web application or redesign an existing Web site with Site Designer. Chapter 12 explains how to use Site Designer to create a site diagram, add pages to the diagram, apply themes and layouts, and print the site diagram.

Chapter 13 Designing Site Navigation

Site diagrams give you a visual way to design how users navigate through your site and provide a fast and efficient way to manage site navigation. This chapter covers "Arranging Pages in a Site Diagram," "Adding a Page to the Global Navigation Bars," "Removing a Page from a Site Diagram," and "Deleting Pages in a Site Diagram."

Chapter 14 Managing a Site Diagram

Chapter 14 delves into the ways you can take advantage of the features of a Site Diagram to manage your Web pages directly from the Site Diagram.

Chapter 15 Viewing Links for an Item

You can use Link View to create a graphical representation of the links to and from items in a Web application. Chapter 15 covers using Link View to see and manage the links between files in your Web application.

Chapter 16 Maintaining Links

Visual InterDev provides tools to help keep your Web page links up to date. Chapter 16 describes "Filtering Links," "Viewing Link Diagrams," "Changing a Link Diagram Layout," and "Repairing Links."

Chapter 17 Customizing Page Appearance

With Visual InterDev tools, you can quickly design consistent Web pages. Templates create standard pages that can contain common HTML and script. Style sheets instruct the browser to override browser default fonts, sizes, and colors. Themes employ a common set of graphics. Layouts define how the navigation and content are laid out on regions of a page.

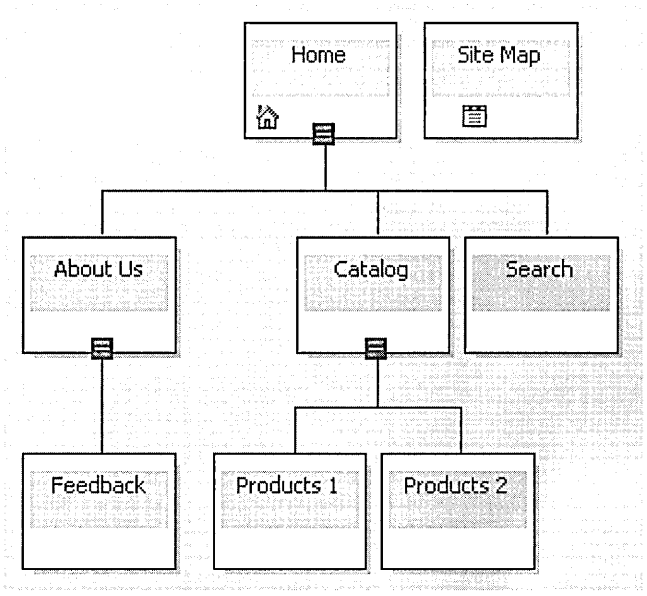
CHAPTER 11

Site Design

Designing a good Web application takes research, careful planning, and thorough testing of your Web application ideas. When creating your site, you need to consider how to organize your information and how your users will navigate through your Web application.

Site Designer makes it easy to visually prototype a new Web application or redesign an existing site with site diagrams. You can easily add new pages to your Web application and create navigational links between pages in the site diagram.

When you save the diagram, Site Designer adds the pages you created to your project and updates the navigational links in your files. Site diagrams can make designing a Web application as simple as dragging and dropping files.



Even a well designed site requires maintenance to keep the content fresh and accessible. Site Designer helps you with this by providing a graphical way for you to design and maintain the navigation structure for a Web application.

You can easily arrange and reorganize the navigation links between the pages to define how users navigate through your Web application. Within minutes, you can create new pages and define the hierarchical relationship between those pages by dragging graphical representations of the pages into position on the site diagram. Saving the changes to the site diagram updates the navigation bars in your site.

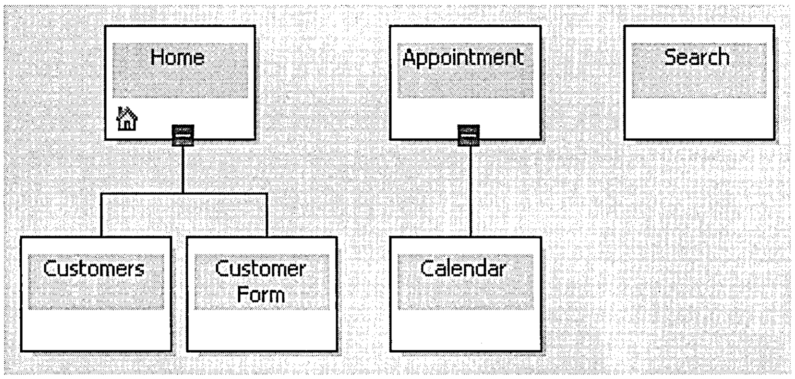
For more information about the functionality of site diagrams, see:

- Site Diagrams
- Prototyping a Web Site
- Designing Site Navigation
- Organizing a Site Diagram

Site Diagrams

Site diagrams are graphical representations of the pages in your Web application that provide information about the navigation structure between pages. Site diagrams give you a way to interact with the Web application's site structure file, which stores information about the navigation structure between pages for your Web application.

Within a site diagram, you can create groups of pages with hierarchical relationships. These groups of pages are called trees. These hierarchical relationships are used to define the navigation bar links in a site. You can easily move pages around the diagram to change the hierarchical relationships. Each Web application can have multiple site diagrams and each site diagram can have multiple trees.



Square nodes represent pages in a site diagram. Pages can have several adorning icons indicating the type of page it is: an external page, global navigation bar page, modified page, or home page.

To create relationships between pages, you drag the pages beside or underneath one another. The link lines that appear when you drag a page underneath or beside another page identify the type of relationship created between pages.

Prototyping a Web Site

Site diagrams provide a way for you to rapidly design and populate a Web application. You can create new pages for your project directly in a site diagram. When you save the site diagram, the site structure file is updated and all new pages are created and added to the project. The new pages are named according to the label names you typed in the site diagram.

If you have applied themes or defined a layout for the Web application, the new pages are created using these defaults. The navigation structure for the site is also automatically incorporated and updated for each page.

> **Note** Each project has a single site structure file that all the site diagrams for a project interact with. Changes you make in one site diagram can impact another site diagram for the Web application if both site diagrams share pages.

For more information, see Chapter 12, "Designing a Web Site."

Designing Site Navigation

Site diagrams provide a visual way for you to design and modify the navigation structure for a new Web application or to maintain the navigation for an existing application. When you save new pages in a site diagram, the navigation links are added to the pages based on project defaults and the relationships defined in the site diagram.

You can create several site diagrams for portions of a Web application or a single site diagram for the entire application. In the site diagram, you design the hierarchical relationships that define the navigation structure for your site. In addition, you can specify which pages will appear on the global navigation bar for the application or you can add a home page to the site diagram.

For more information, see Chapter 13, "Designing Site Navigation."

Organizing a Site Diagram

Site Designer allows you to manipulate the layout and organization of a site diagram to make it easier to work with the navigational structure of your site.

You can change the magnification of a site diagram or automatically size the site diagram to fit within the current window. You can also expand and collapse child pages in a tree. Use the Rotate command to switch between horizontal and vertical tree layouts. You can also reposition any tree within the site diagram by simply dragging the tree to another location.

For more information, see Chapter 14, "Managing a Site Diagram."

Link Verification

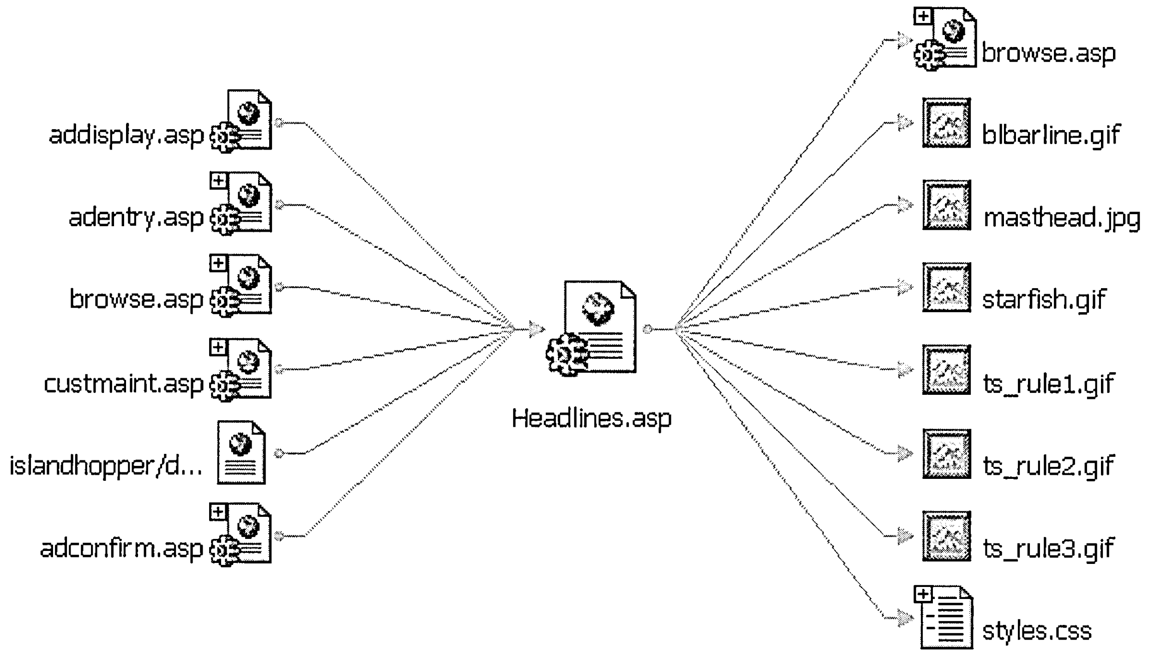
Web applications contain multiple links to HTML pages, graphics, and other files. The larger the Web application, the more complicated verifying and maintaining these links can be.

You can use Link View to create link diagrams to identify the links between files in a Web application and to find broken links between items. Link diagrams provide a graphical view of the links between files to help you easily maintain these links in your Web application.

In Link View, an item is a resource, such as an .htm file, a design-time control, or an image file, that is part of a Web application. Items in a Web application are represented by graphical icons and the relationships between these items are represented by arrows in the link diagram.

Items in a link diagram are joined by lines that represent links between the items. Arrows communicate the direction of links, either in or out. You can understand the link relationships between items by analyzing the graphical representation of those relationships in a link diagram.

With Link View, you can view the links for a page or item in your Web application or any page on the World Wide Web. You can also use a link diagram as a launching point for previewing items in a Web browser, or for editing items in their default editors. For more information about previewing and editing items, see "Viewing Link Diagrams" in Chapter 16, "Maintaining Links."



To learn more about Link View and the functionality of link diagrams, see the following sections:

- Link Diagram Layouts
- Verifying Items
- Identifying Broken Links
- Viewing Links for an Item
- Showing and Hiding Items with Filters
- Expanding and Collapsing Links to Other Items

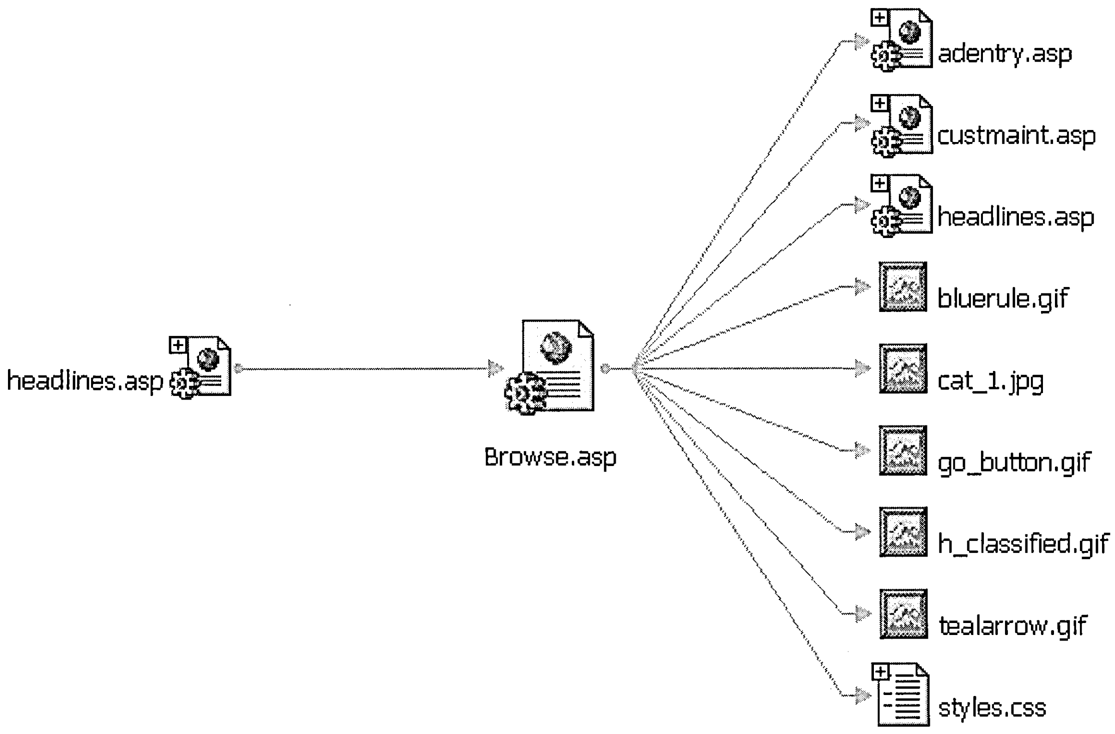
Link Diagram Layouts

Link View uses two layouts for link diagrams: the horizontal layout and the radial layout. Each layout has a slightly different focus and together they can help you track links between items in your Web application. For details about diagram layouts, see "Changing a Link Diagram Layout" in Chapter 16, "Maintaining Links."

Horizontal Layout

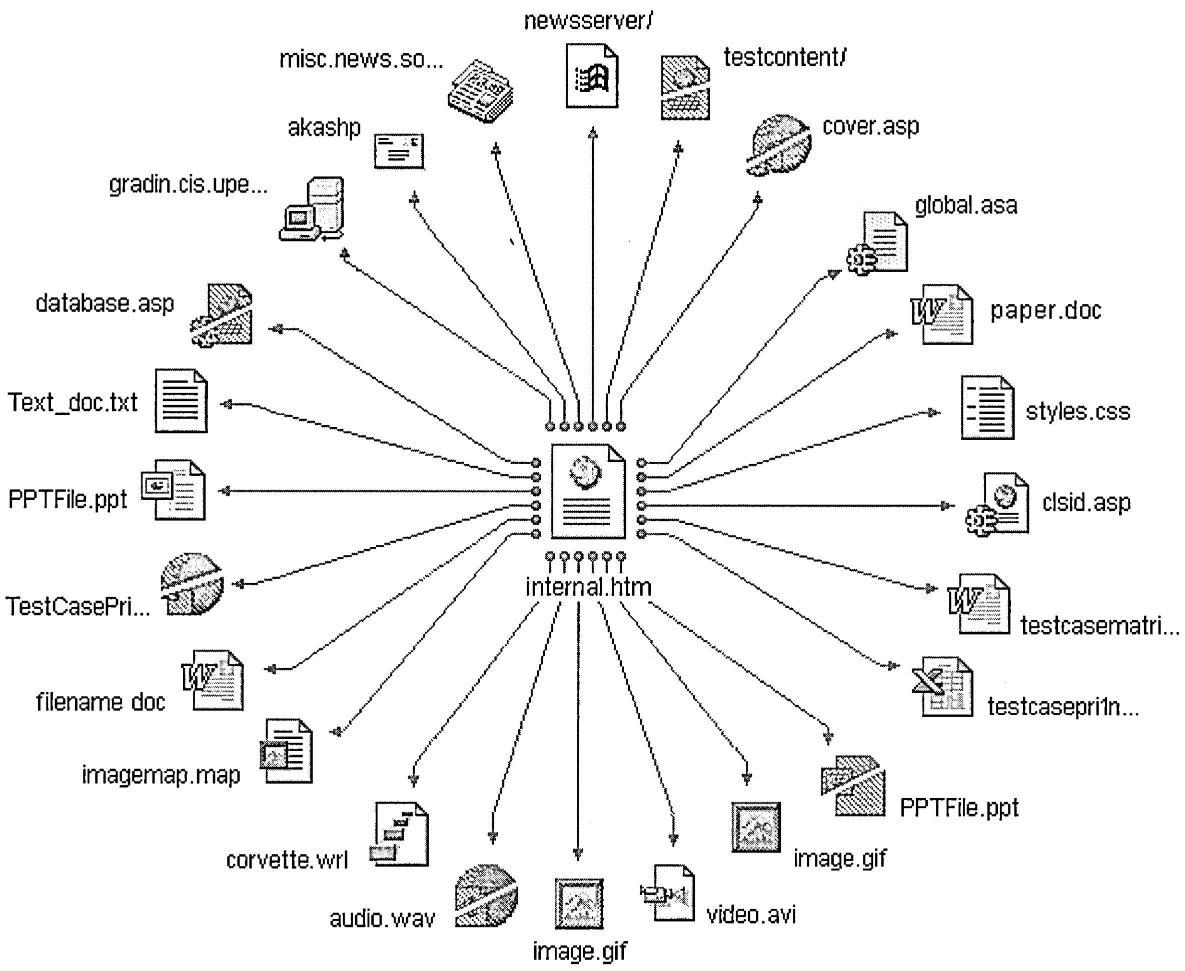
By default, the horizontal layout appears when you open a link diagram for an item. This layout shows both the in links and the out links for an item.

In the horizontal layout, in links appear to the left of the expanded item and out links appear to the right of the expanded item.



Radial Layout

The radial layout shows either in links or out links for an item, but not both at the same time. In radial layout, the link diagram shows all the items linked to the expanded item arranged radially around it.



Verifying Items

When you create or refresh a link diagram for an item, Link View retrieves information about the links either from information stored on the master Web server (in the link index file and the site structure file) or by searching for links on the World Wide Web. Whether the item is part of the Web application determines how Link View will retrieve the link information.

- **For items in a Web application** Link View uses the information stored on the master Web server to quickly determine whether an item is valid or broken.

  If the information on the Web server has changed since you created the link diagram, you must use the Refresh command to retrieve the most current link information for the Web application.

- **For items outside of a Web application** Link View dynamically retrieves link information by searching for the destination of the link on the World Wide Web. External items are displayed in the link diagram as valid, broken, unknown, or pending. Link diagrams for external items build slower than link diagrams for internal items due to varying Internet access speeds.

  Note Initially, all external items in a link diagram are displayed as unknown. Use the Verify or Expand commands to have Link View determine if the link is valid or broken.

Link View uses the following icons to indicate the state of the link.

Link State	Icon	Meaning
Valid	📄	The item the link points to exists.
Broken	❌	The item the link points to does not exist.
Unknown	❓	The item the link points to may or may not exist. Use the Verify command to have Link View determine if the link is valid or broken.
Pending	⌛	Link View is searching for the item the link points to on the World Wide Web.

Note When a link to an item is determined to be broken, you can position the pointer over the item's icon to display its ToolTip. The ToolTip contains information about why the item could not be found. For example, if Link View tries to verify a link to an HTML page that no longer exists, the ToolTip for the broken link will contain an http error message.

Identifying Broken Links

Broken links can appear for a number of reasons. For example, the destination file may have been deleted or the link may contain a typing error.

In Link View, broken links are indicated graphically in the link diagram by the broken link icon. The ToolTip for the item containing the broken links displays the reason for the broken link.

In addition to using the link diagram, you can quickly create a list of all the broken links in your Web application by creating a Broken Links Report. A broken links report contains a list of all broken links in a Web application, as well as a list of files that have no in links. Once you have identified the broken links in your Web application, you can then repair the links.

For more information about identifying and fixing broken links, see "Repairing Links" in Chapter 16, "Maintaining Links."

Viewing Links for an Item

When you open a link diagram on an item, that item is represented by a large icon in the middle of the link diagram. The links shown for the item depend on the diagram layout and any filters applied.

You can create a link diagram on any of the following:

- An item in your Web project. For more information, see "Viewing Links for an Item in a Project" in Chapter 15, "Viewing Links for an Item."
- An item in an open link diagram. For more information, see "Viewing Links for an Item in a Project" in Chapter 15, "Viewing Links for an Item."
- Any valid URL. For more information, see "Viewing Links for an Item on the WWW" in Chapter 15, "Viewing Links for an Item."

Showing and Hiding Items with Filters

You can filter a link diagram to show only the types of items you want to see. Filtering is useful if you are viewing a particularly large Web application or if you want to view a specific type of resource. For example, you may want to use a filter to exclude multimedia files — such as audio and video files — that are used in a Web application or to show only graphic files included on a page. For more information about displaying links, see "Filtering Links" in Chapter 16, "Maintaining Links."

Filtering a link diagram to show only out links or only in links can be helpful. For example, showing in links enables you to determine the impact of renaming or deleting an item because you can see other pages that currently link to that item. For more information, see "Changing a Link Diagram Layout" in Chapter 16, "Maintaining Links."

Expanding and Collapsing Links to Other Items

If you want to see more detail in the current link diagram, you can expand links for a selected item to show the in or out links for that item. The links that appear depend on the filters currently set for the link diagram. For example, if you set the filters to show only HTML pages in your link diagram, expanding links on a page will display only links to other HTML pages. You can also collapse links for an item in a link diagram. Collapsing a link hides the links for that item and reduces the size of a large link diagram for a site.

For more information about how to display linked items, see "Viewing Link Diagrams" in Chapter 16, "Maintaining Links."

Site Consistency

One of the key differences between an amateur and a professional Web site is consistency. Professional sites employ well-designed pages that are consistent across the site in their look and functionality.

When your site is consistent, you have a site that:

Looks professional A professional site is well designed and consistent in its use of the design. Users are more comfortable with a site that's easy to navigate and easy to read.

Saves development time When you take steps to include consistency in your development cycle, the number of tasks is reduced and simplified.

Saves authoring time Using templates, layouts, and themes can save time when authoring new files, letting the author focus on writing the actual content.

To give your Web site a crisp, professional look and feel, you can take advantage of the many tools available with Microsoft Visual InterDev. The larger your team is, the more effective these tools become.

The Site Consistency Process

When you design consistency into your site, you typically perform some or all of the following tasks.

1. Create style sheets to define how your pages appear.
2. Design a common layout that meets your requirements for navigation and page appearance.
3. Create custom templates for authors to create new pages similar to existing ones.
4. Employ a theme that shares common graphics across your site.

Each of these tasks work together to provide a consistent look that assures users you've done a professional job.

Site Consistency in Visual InterDev

For each site consistency task, Visual InterDev offers you:

- Pre-packaged solutions.
- Tools to help you create your own from scratch.

Depending on the size of your site and your team, you might use solutions right out of the box or you may customize them a great deal. The Visual InterDev solutions for consistency include:

- Cascading Style Sheets
- Templates
- Layouts
- Themes

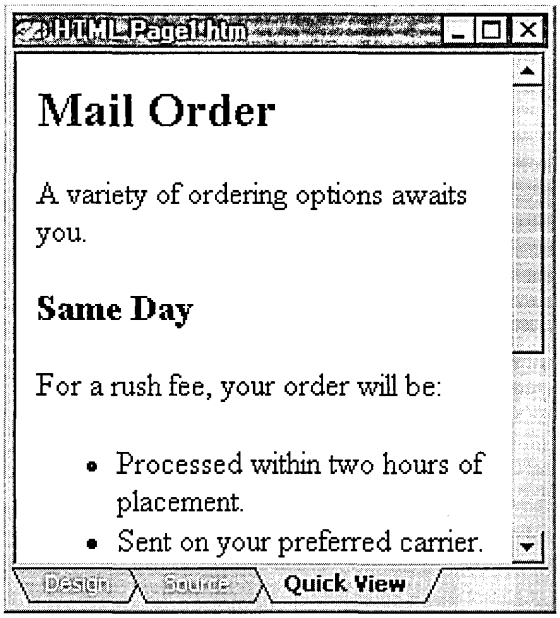
Cascading Style Sheets

Cascading style sheets (CSS) give you the ability to define a set of styles that override the browser's standard methods for rendering HTML. This lets you give your pages a unique and consistent design. Style sheets allow you to define the attributes of any tag. For example, you can adjust the font, line spacing, justification, and border properties.

You apply a style sheet to a page by adding a <LINK> tag in the HTML document heading - between <HEAD> and </HEAD>. A <LINK> tag referencing a CSS file would look like this:

```
<LINK REL="stylesheet" TYPE="text/css" HREF="YourStyles/COLOR0.CSS">
```

Therefore, you can have as many pages as you want referencing the same style sheet. Because all the style information is stored in a single file, it is easier to maintain and saves space on the Web server.



Mail Order

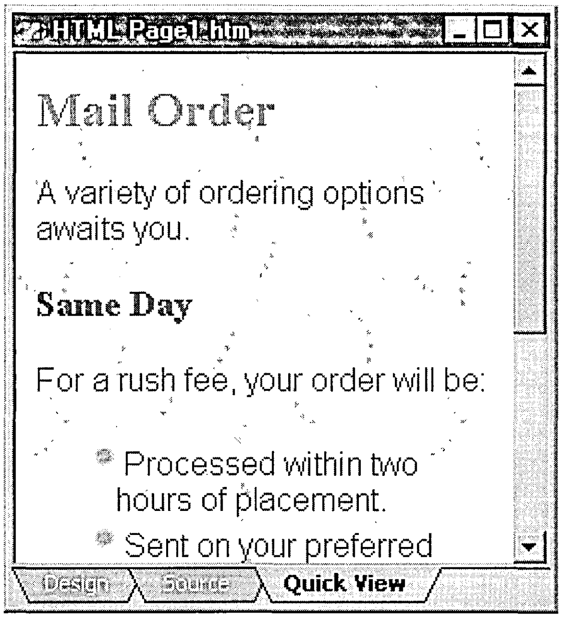
A variety of ordering options awaits you.

Same Day

For a rush fee, your order will be:

- Processed within two hours of placement.
- Sent on your preferred carrier.

Page without style sheet



Mail Order

A variety of ordering options awaits you.

Same Day

For a rush fee, your order will be:

- Processed within two hours of placement.
- Sent on your preferred

Page linked to a style sheet

Visual InterDev includes a CSS editor that has an easy-to-use interface for setting style properties. You can use the editor's graphical interface so that you don't have to edit raw CSS text. For information on using the CSS editor, see "Editing Style Sheets" in Chapter 17, "Customizing Page Appearance."

Not all browsers support the use of cascading style sheets. However, when cascading styles are used correctly, browsers that don't support them will ignore them and still render the page readable. You should keep this in mind when designing with styles to make sure that the appearance of your page degrades gracefully on older platforms. CSS 1 is supported by Microsoft Internet Explorer 4.0 and Netscape Navigator 4.0. Internet Explorer 3.0 supports a subset of CSS 1.

The use of style sheets is endless — varying from the simple to the extremely complex. For a more detailed description of CSS, see CSS Attributes reference and also see the World Wide Web Consortium's CSS specification on the Internet at http://www.w3.org/pub/WWW/TR/REC-CSS1.

Style Sheets in Visual InterDev

Visual InterDev comes with a variety of style sheets:

- A standard template is used in the Add Item dialog box when you create a new style sheet, from which you can create and modify styles.
- Approximately 60 themes come with Visual InterDev. Each theme contains one or more style sheets.

You can use any of these style sheets as a model for creating your own style sheet, or you can simply make a copy and customize it to suit your needs.

Previewing a Style Sheet

When you are editing a style sheet in the CSS editor, you can use the Preview tab to see how it would look if it were applied to any page, local or on the World Wide Web.

> **Note** If the page you are previewing already has a style sheet attached, Visual InterDev recognizes that style sheet as the primary style sheet and correctly applies the one that you are editing in a secondary position.

Templates

Templates can be an invaluable resource. Even though a template is just a master file from which new copies are made, you can use templates for HTML documents, ASP pages, CSS files, or any other text-based file. Because it takes time to create a well-designed page, the time you invest in templates pays off many times over.

By default, all templates appear in the Add Item dialog box. To add a template to the available list, add it to the following directory:

```
C:\Program Files\Microsoft Visual Studio\VintDev98\Templates\Web Project Items
```

> **Note** After adding a template to the "Web Project Items" directory, you must restart Visual InterDev in order to display the template in the Add Item dialog box.

Take a look at the directory to see the actual templates that ship with Visual InterDev.

Interactive Templates

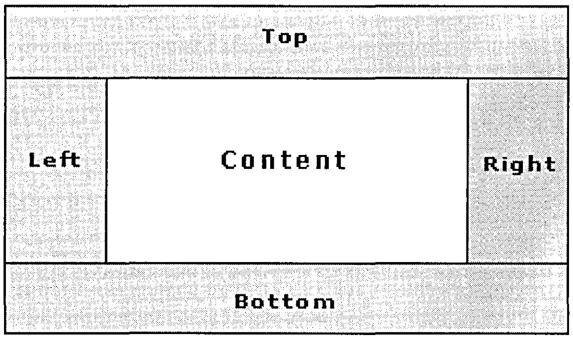
You can use parameters in your templates to prompt a template user for additional information, like a theme name or a company name. This allows your templates to be more flexible. For more information, see "Creating Custom Templates" in Chapter 17, "Customizing Page Appearance."

Layouts

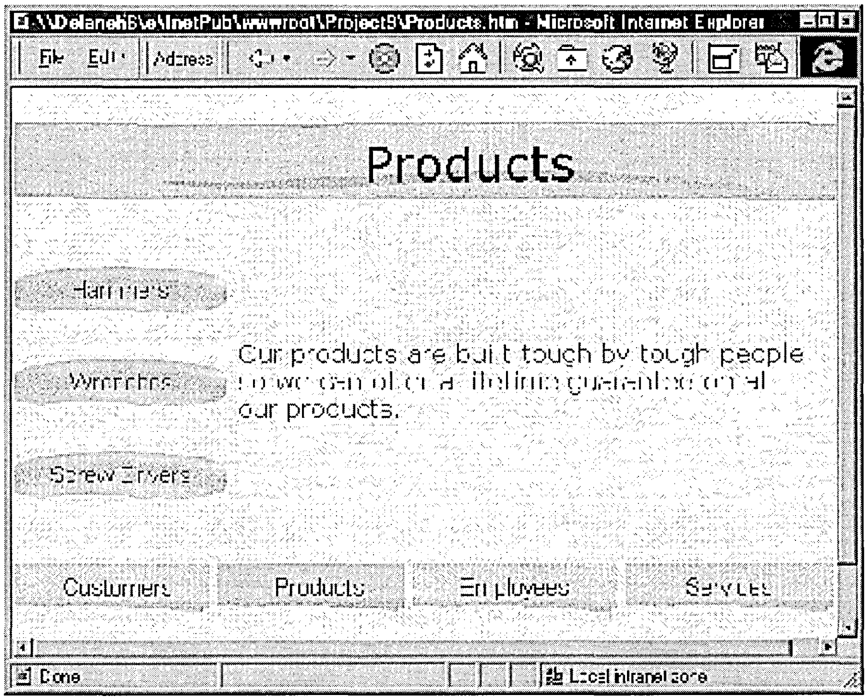
Layouts define how the content of a page is visually laid out and how users can navigate among parent, children, and sibling pages by using a PageNavbar design-time control.

How a Layout Works

Layouts control how the regions of a page are laid out. Layouts are comprised from different possible regions of a page. For example, most layouts utilize two or three of the following five possible regions:



Visual InterDev creates the regions by using HTML tables. The result looks something like the picture below. In the following picture, the Top, Left, Bottom, and Content regions were utilized.



Our products are built tough by tough people [illegible] our products.

The HTML table information is stored in a Layout Header section and Layout Footer section of the Web page. The header and footer Layout design-time controls bracket the content of your page.

Each layout is based on a unique HTML template. The template stores the layout information. When the layout is applied to a page, Visual InterDev adds the layout design-time controls to the page.

> **Important** Do not manually alter the run-time text of the Layout design-time controls. If this text is altered, Visual InterDev might not be able to parse the information correctly. Also, when you apply a different layout, Visual InterDev deletes the current Layout design-time controls and replaces them with new ones.

Because the layout information is maintained within a template, it is easy to change which layout is applied to the page without affecting content. For step-by-step instructions on applying or customizing a layout, see "Laying Out Pages" in Chapter 17, "Customizing Page Appearance."

Layouts Installed on Your Workstation

The list of layouts that are installed on your machine was determined when you chose the Typical or Custom option while setting up Visual InterDev on your computer.

Typical Install Option The typical install option copies all 18 layouts to your machine.

Custom Install Option The Custom install option lets you choose either of the following:

- 0 layouts (no layouts)
- Typical (18 layouts)

Directory Location of Layouts

Layouts reside in the "Layouts" directory, which is found at:

```
C:\Program Files\Microsoft Visual Studio\VintDev98\Layouts
```

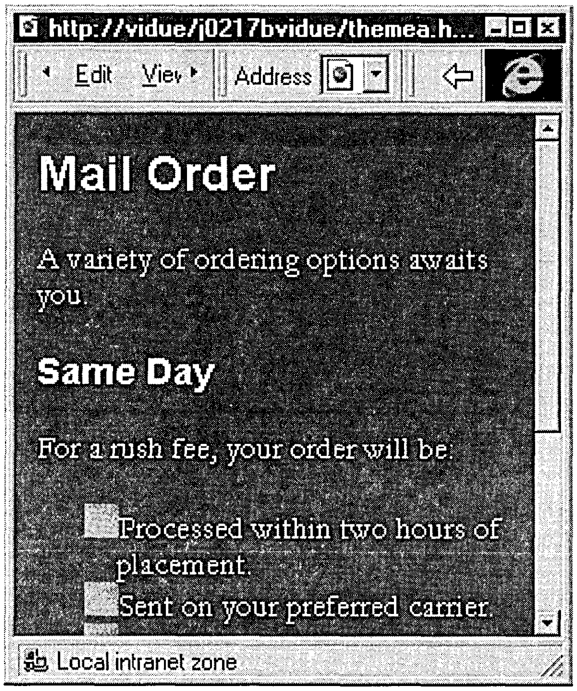
The files that make up a layout are added to the master server automatically when you apply a layout so that the layout can be shared with other developers who might be working on the same project and need access to the files as well. Layouts are stored on the master server in a directory named "Layouts" in the project's top-level directory. These files are needed at design time, but not at run time.

Themes

Themes are comprised of a set of graphics and one or more cascading style sheets that control styles, font, and graphics. You apply themes to pages so that there is visual consistency shared across your pages.

You can set a default theme for an entire project so that each page that you create in that project will have the same theme applied. Even if a default theme is set for a project, you can choose to override the default on particular files where you might want to apply a different theme or no theme at all.

When you set a default theme for your project, you automatically benefit from the theme's pre-packaged design. Another advantage of themes is the ability to easily switch between different themes, without changing content, to give your site a whole different look.



Mail Order

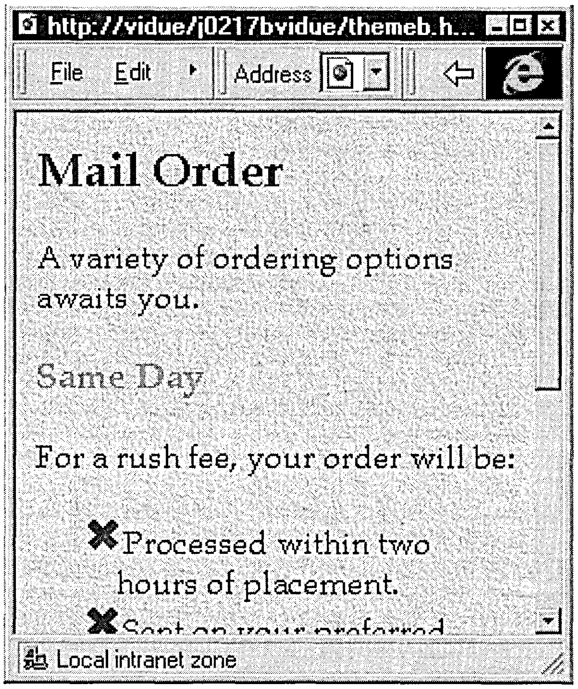
A variety of ordering options awaits you.

Same Day

For a rush fee, your order will be:

- Processed within two hours of placement.
- Sent on your preferred carrier.

Artsy theme applied to page



Mail Order

A variety of ordering options awaits you.

Same Day

For a rush fee, your order will be:

- Processed within two hours of placement.

Expedition theme applied to page

You can use one of the many themes that come with Visual InterDev, modify one of the existing themes, or create your own custom theme. For step-by-step information about applying and creating themes, see "Applying Themes" in Chapter 17, "Customizing Page Appearance."

How Themes are Referenced in a Page

When you apply a theme to a page, Visual InterDev inserts <LINK> tags within the <HEAD> block. For example, the following text is inserted when the Artsy theme is applied:

```
<LINK REL="stylesheet" TYPE="text/css" HREF="Themes/artsy/COLOR0.CSS" VI98THEME="artsy">
<LINK REL="stylesheet" TYPE="text/css" HREF="Themes/artsy/GRAPH0.CSS" VI98THEME="artsy">
<LINK REL="stylesheet" TYPE="text/css" HREF="Themes/artsy/THEME.CSS" VI98THEME="artsy">
<LINK REL="stylesheet" TYPE="text/css" HREF="Themes/artsy/CUSTOM.CSS" VI98THEME="artsy">
```

In the example above, four <LINK> tags have been inserted because the Artsy theme uses four cascading style sheets.

Themes Installed on Your Local Machine

The list of themes that are installed on your machine was determined when you chose Typical or Custom while setting up Visual InterDev on your computer.

Typical Install Option The Typical install option copies 17 themes to your machine. These 17 themes include:

- New themes that were created for Visual InterDev 6.0.
- Upgraded versions of themes that shipped with Visual InterDev 1.0.
- Upgraded versions of Microsoft FrontPage 98 themes.

Custom Install Option The Custom install option lets you choose one of the following:

- 0 themes (no themes)
- Typical (17 themes)
- Complete (all themes)

Location of Themes on Your Workstation

Each set of files that makes up a theme is maintained in a single directory. The directory name is identical to the theme name.

The theme directories are stored locally in the "Themes" directory, which is typically found at:

```
C:\Program Files\Common Files\Microsoft Shared\Themes
```

> **Note** In Visual InterDev 1.0, the theme files were spread across different directories. In Visual InterDev 6.0, all the files that make up a theme reside in the same directory.

Location of Themes on the Server

Themes are stored on the master server in a directory named "Themes" in the project's top-level directory. The files that make up a theme are added to the master server automatically when you apply a theme.

When you deploy your Web application, include the Themes directory and sub-directories when you copy the files to the production server.

Sharing Themes Between Visual InterDev and FrontPage

You can access a master application on the server from either a Visual InterDev Web project or a FrontPage project. This allows team members to use their preferred tool of choice. For example, a team might have developers who use Visual InterDev's enhanced scripting capabilities to access the master application and designer and writers who prefer to use the design features of FrontPage to access the same master application.

For information about the differences between Visual InterDev and FrontPage, see "Using FrontPage and Visual InterDev to Create Web Sites" in Chapter 29, "Integration Tasks."

Additional Graphics in Themes

Each theme comes with an assortment of additional graphics to complement the basic ones that are used by default. You can use these additional graphics to enhance the design of your pages. Look inside a theme's directory to see all the .GIF images that are available in a particular theme.

> **Tip** When you drag an image from the Project Explorer onto a page, Visual InterDev automatically inserts an <IMG> tag with the correct URL.

When you use images in the Themes directory, be sure to include the correct URL in the <IMG> tag.

> **Note** If you apply a new theme to the page, you will need to manually update the URLs of all images that you added by hand.

Putting it All Together

After you've decided how the tools discussed above interest you, you can use them to simplify the process of creating and maintaining a visually appealing and consistent Web site.You can:

- Use the Web Project wizard to create a new project. The wizard prompts you for a default theme and layout.
- Apply a theme and a layout to an existing Web project or individual file. To select a project or file name in the Project Explorer, right-click, and select **Apply Theme and Layout**.

When a theme or a layout is applied, it is automatically copied to the master server so that it can be shared with all team members.

If you want to share a template with team members, make the master copy available to them. Each team member must then make a copy and place it in the Templates directory on the workstation.

CHAPTER 12

Designing a Web Site

You can rapidly design and create a new Web application or redesign an existing Web site with Site Designer. In a site diagram, you visually create, add, and organize pages in a Web application and design the navigation structure for the site. For more information about designing the navigation structure of a site, see chapter 13, "Designing Site Navigation."

Site Designer allows you to create new pages directly in a site diagram or add existing pages from the current Web application or other applications. After you have created a site diagram, you can apply a theme to the pages or print the diagram.

Creating a Site Diagram

You can use a site diagram to easily design and build the structure of a Web application. In addition, site diagrams help you to define and modify the navigation links between pages. For more information, see Chapter 13, "Designing Site Navigation."

Site diagrams can be used to create new Web applications or to redesign the structure of existing Web applications. A Web application can have multiple site diagrams. For example, you can create different site diagrams to define the navigation structure between separate areas of your Web application.

To create a site diagram

1. Open or create a Web project.
2. From the **Project** menu, choose **Add Item.**

   The **Add Item** dialog box appears.
3. On the **New** tab, select the **Web Project Files** folder and then choose **Site Diagram** from the pane on the right.
4. In the **Name** text box, type a name for the site diagram. Note that site diagram files have a .wdm extension.
5. Click **Open.**

   A new site diagram appears.

Note When you create the first site diagram for a Web application, Site Designer automatically adds a home page for the site to the diagram. If no home page exists, Site Designer creates one and adds it to the project when you save the site diagram.

Site Designer determines existing home pages based on information on the Web server. In general, Site Designer considers files such as `default.htm`, `default.asp`, or `index.htm` to be home pages. Microsoft Internet Information Server specifies `default.asp` for a home page and Microsoft Personal Web Server specifies `default.htm` for a home page.

You can add pages to a site diagram from the current project or from other applications. You can also use Site Designer to add new pages to the site diagram and the Web project. For more information, see the next section. "Adding Pages to a Site Diagram."

Adding Pages to a Site Diagram

You can add pages to your site diagram in a number of ways. You can drag files from the Project Explorer into the site diagram or you can create new files directly in the site diagram. You can also drag files from other applications, such as Microsoft Internet Explorer or Microsoft FrontPage.
After you have added pages to the site diagram, you can then arrange the pages to create hierarchical relationships. Navigation bars use these hierarchical relationships to determine which navigation links to include on the navigation bars for each page in the site diagram. For more information, see "Arranging Pages in a Site Diagram," in Chapter 13, "Designing Site Navigation."

Note Each .htm file can only appear once in a site diagram. Other files, such as .asp files, .gif files, and external pages, can appear multiple times in a site diagram.

To create new pages in the site diagram

1. Open or create a site diagram.
2. From the **Diagram** menu, choose **New HTML Page** or **New ASP Page**.

   – or –

   Right-click the diagram, and then choose **New HTML Page** or **New ASP Page** from the shortcut menu.

A new page is added to the site diagram. You can then drag the page to position it in the diagram. The new page is given a default name. For more information about renaming a page in a site diagram, see "Renaming Page Labels," in Chapter 14, "Managing a Site Diagram."

Each .htm file can only appear once in a site diagram, but .asp files can appear multiple times in site diagrams using different parameters. You can add files to site diagrams by dragging and dropping the files from the Project Explorer into the site diagram or with the Add Existing File command.

Use the Add Existing File command when you want to create a new instance of a file that already exists in your site diagrams. Use the drag and drop method when you do not want to create a new instance of a file but want to include a file in the current site diagram that already exists in another site diagram. If you attempt to drag in a file that already exists in the current site diagram, Site Designer highlights the existing file in the diagram but does not add the file again.

To add pages to a site diagram from a project

1. Open or create a site diagram.
2. From the **Diagram** menu, choose **Add Existing File.**

   The Choose URL dialog box appears.
3. In the **Projects** list, select the project name or folder for the file.
4. In the **Contents** list, select the file you want to add to the diagram.
5. In the **Parameters** drop-down list, enter a parameter value or bookmark value.
6. Choose **OK.**

   If the file you specify already exists in a site diagram, a message box appears. To add the existing instance, choose Yes. To create a new instance, choose No. If the file does not exist in any site diagram, Site Designer adds the file to the current diagram.

   You can then drag the page to position it in the diagram.

To add pages from another application

1. Open or create a site diagram.
2. Open the application containing the page you want to add to the site diagram.
3. Drag the link from the source application into Visual InterDev.
4. Drop the file in the site diagram.

If the file already exists in another site diagram, Site Designer will add the file and its navigation structure to the current diagram. If the file does not exist in any site diagram, Site Designer adds the file to the current diagram.

Note Each page you add to a site diagram from another application or project displays the External page icon. An external page cannot have child pages in a site diagram.

Adding a Home Page to a Site Diagram

You can add an existing home page to a site diagram or create a new home page for the Web application. When you create the first site diagram for a Web application, Site Designer automatically adds the site's home page to the diagram. If no home page exists, Site Designer creates one and adds it to the project when you save the site diagram.

If a home page already exists in the Web application, Site Designer identifies the home page based on information on the Web server. In general, Site Designer considers files such as `default.asp`, `default.htm`, or `index.htm` to be home pages. Microsoft Internet Information Server specifies `default.asp` for a home page and Microsoft Personal Web Server specifies `default.htm` for a home page.

To add a home page to a diagram

1. Open or create a site diagram.
2. From the **Diagram** menu, choose **Add Home Page**.

   – or –

   Right-click the diagram, and then choose **Add Home Page** from the shortcut menu.

Site Designer automatically adds the current home page for the Web application to the site diagram. When you save the site diagram, the new home page file is added to the current project.

Mapping Multiple Pages to an .asp File

With Site Designer, you can quickly prototype a Web application without worrying about how you will use the pages in your site. For example, if you have added three .htm pages to your site diagram, you can later map these pages to an .asp file and have the navigation bars for the pages display correctly.

Site Designer allows you to map pages in a site diagram to a new or existing Active Server Page in the site structure file. You can also map several pages to an ASP page using different parameters.

Note You cannot map an external page to a new or existing Active Server Page.

To map multiple pages to an ASP page

1. In the site diagram, select the page you want to map to an .asp file.
2. From the **View** menu, choose **Property Pages**.

   The Page Tab (Page Properties Dialog Box) appears.
3. Select the **Browse** button.

   The Choose URL dialog box appears.
4. In the **Contents** text box, select the .asp file to use.
5. In the **Parameters** drop-down list, enter a parameter value.
6. Click **OK**.

The navigation bar links will reflect that the selected page is a parameter of an .asp file and will link to it correctly.

Using the Choose URL dialog box does not add script to the .asp file. You must still add the appropriate script to the .asp file to get the .asp file to display the pages correctly.

For example, if you have mapped `Books.htm` and `Clothes.htm` to `Products.asp` as Type=Books and Type=Clothes, you would need to include the following script in `Products.asp` so that `Books.htm` and `Clothes.htm` would display properly.

```
<% Select Case Request .QueryString ("Type") %>
   <% Case "Books" %>
      Books
   <% Case "Clothes" %>
      Clothes
<% End Select %>
```

Converting Active Server Pages to HTM Pages

You can create files in Site Designer without regard to the file type and then you can convert individual ASP pages to HTML pages as needed.

To convert ASP pages to HTML pages

1. Select the .asp file in the Project Explorer that you want to convert.
2. Right-click the page, and choose **Rename** from the shortcut menu.

   Note The Rename command is available for local files only.
3. Change the .asp extension to .htm.
4. Press **Enter**.
5. In the message box, click **OK.**
6. In the Link Repair dialog box, click **Yes**.

   The .asp file icon is replaced with the appropriate .htm file icon in the **Project Explorer**.

Applying a Theme and Layout to Pages

Themes and layouts provide a consistent visual design for information on pages. You can apply themes and layouts to new pages created using Site Designer or to existing pages in a site diagram.

When you use layouts or the PageNavbar design-time control, you can take advantage of the navigation structure design capabilities of Site Designer. For more information, see Chapter 13, "Designing Site Navigation."

To apply themes and layouts

1. In a site diagram, select the pages you want to have themes and layouts.
2. From the **Edit** menu, click **Apply Theme and Layout**.

   The **Apply Theme and Layout** dialog box appears.
3. On the **Theme** tab, select the name of the theme you want to apply.
4. On the **Layout** tab, select the name of the layout you want to apply.
5. Click **OK**.

Printing a Site Diagram

You can print a site diagram for use in presentations or other occasions. Before printing, you can preview the page breaks for the site diagram.

Site Designer sets the page breaks for a site diagram the first time you print the diagram. If you later add pages to the diagram, Site Designer does not automatically reset the page breaks. You can, however, reset the page breaks manually.

You can change the size of your site diagram to make it fit on a page. For more information, see "Managing the Size of a Site Diagram," in Chapter 14, "Managing a Site Diagram."

To print a site diagram

- From the **File** menu, choose **Print**.

To view the page breaks for a site diagram

- From the **Diagram** menu, choose **View Page Breaks**.

  Dashed lines appear on the site diagram to indicate page breaks.

To reset page breaks

- From the **Diagram** menu, choose **Recalculate Page Breaks**.

  The page breaks are now reset to begin at the top left corner of the diagram.

CHAPTER 13

Designing Site Navigation

Site diagrams give you a visual way to design how users navigate through your site and provide a fast and efficient way to manage site navigation. Simply by dragging page icons into place in your diagram, you can create relationships between your Web pages. These relationships are reflected in the Web application's site structure file, which contains the navigation-related information for pages in the Web application.

When you use layouts or the PageNavbar design-time control, Site Designer interprets the information in the site structure file to determine the types of links to include in the navigation bars for each Web page in the site diagram.

Once the navigation bars are in place, you can modify the links included in the navigation bars for a site by adding, moving, and deleting pages from a site diagram. If you have created a new Web application with a layout or if you have applied a layout or PageNavbar design-time control to an existing Web application, you can automatically update the navigation bar links in your pages with Site Designer.

Arranging Pages in a Site Diagram

Site diagrams provide an easy way to design and update the navigation structure of a Web application. In a site diagram, you create hierarchical relationships between pages by grouping pages into trees. Trees contain one or more parent pages and one or more child pages. Each Web application can have multiple site diagrams, and each site diagram can have multiple trees.

> **Tip** Each .htm file can only appear once in a site diagram. Other files, such as .asp files, .gif files, and external pages, can appear multiple times in a site diagram.

You drag and drop pages beside or beneath one another in a site diagram to create parent, child, and sibling hierarchical relationships. Use the link lines to aid you in creating hierarchical relationships.

Layouts and the PageNavbar design-time control use trees and independent pages to determine the types of navigation bar links to include in a page.

To create separate groups of pages

1. Select a page in the site diagram.
2. From the **Diagram** menu, choose **Detach from Parent**.

   The selected page and all its children are now separated from the group of pages above them.

You can easily create child-parent relationships between pages.

To create a parent-child relationship between pages

1. Select the top page of the source tree you want to use as the child pages.
2. Drag the child pages below the parent page until the link line shows the pages connected.
3. Drop the child pages.

You can also create sibling relationships between pages.

To create a sibling relationship between pages

1. Select a source page in the site diagram.
2. Drag and drop the source page beside the target sibling page.

   Note Use the link lines to determine the placement of pages either to the right or the left of the target sibling page.

Adding a Page to the Global Navigation Bars

You can create hierarchical relationships between pages using site diagrams. These relationships are used to determine the types of links included on the navigation bars for a page.

You can also indicate that a page appears as a default link on the global navigation bar for a site. For example, you may want your help page and search page to be available as a navigation bar link from any page in your site. Site Designer allows you to do this by specifying that a page is a global navigation bar page.

Note You must use a layout or the PageNavbar design-time control to take full advantage of this option.

To add a page to all site navigation bars

1. Select a page in the site diagram.
2. From the **Diagram** menu, choose **Add to Global Navigation Bar**.

   A message box appears if the page is currently a child page.

The page is now available as a navigation bar link from all the navigation bars in the site. In the site diagram, the page displays the global navigation bar icon.

To remove a page from all site navigation bars

1. Select the global navigation bar page in the site diagram.
2. From the **Diagram** menu, choose **Remove from Global Navigation Bar**.

   – or –

   Drag and drop the global navigation bar page over another page to create a parent-child relationship.

You can also specify the order in which global navigation bar page links appear on the navigation bars for a site.

To order global navigation bar links

1. From the **Diagram** menu, choose **Reorder Global Navigation Bar**.

   The **Reorder Global Navigation bar** dialog box appears.
2. In the **Order of pages in global navigation bar** list box, select a page.
3. Select **Move Up** to move the page above another page.

   – or –

   Select **Move Down** to move the page below another page.
4. Select **OK**.

   Note You must save the current site diagram in order for the changes to the global navigation bar page link ordering to appear.

Removing a Page from a Site Diagram

You can easily remove pages from a site diagram without deleting the associated file from your project. Removing a page from the site diagram does not automatically remove the page from the project.

You can also delete pages from the navigation bars or the Web project.

Note To remove a parent page without removing the child pages, you must detach all the child pages from the parent page before using the Remove command.

To remove a page from a site diagram

1. Select the page or pages in the site diagram.
2. From the **Edit** menu, choose **Remove**.

Deleting Pages in a Site Diagram

You can delete pages from the site navigation bars or you can delete pages from the Web project from within a site diagram. When deleting pages from the site diagram, the files associated with the page icons are deleted from the Web application.

You can also remove pages from a site diagram without deleting the page from the site navigation bars or from the Web project.

Note To remove or delete a parent page without removing or deleting the child pages, you must detach all the child pages from the parent page before using the Delete or Remove commands.

To delete pages from navigation bars

1. Select the pages you wish to remove from the navigation bars.
2. From the **Edit** menu, choose **Delete**.

   The **Delete Pages** dialog box appears.
3. Select the **Remove these pages from all navigation bars** option, and then click **OK**.

The selected pages are removed from the current site diagram.

To delete pages from a project

1. Select the pages you wish to remove from the Web application.
2. From the **Edit** menu, choose **Delete**.

   The **Delete Pages** dialog box appears.
3. Select the **Delete these pages from the Web project** option, and then click **OK**.

The selected pages are removed from the current site diagram and the files are deleted from the current project.

CHAPTER 14

Managing a Site Diagram

You can open, rename, save, and delete site diagrams from a Web application. You can also change the magnification of a site diagram, rotate trees in a site diagram, and expand and collapse pages in a site diagram to make viewing the structure of the Web application easier. Use site diagrams as a launching point for editing the source for a page or for previewing the page in a browser.

Managing the Size of a Site Diagram

You can manipulate the site diagram to show more or less information, as needed. You can easily manage the size of a site diagram by expanding or collapsing pages, changing the magnification of a diagram, and rotating trees to fit your screen.

To expand or collapse a group of pages

- Click the plus sign (+) on the page to expand the page group.
- Click the minus sign (–) on the page to collapse the page group.

You can zoom in and zoom out on a site diagram to fit the diagram on your screen.

To change the magnification of the diagram

1. From the **View** menu, choose **Zoom**.
2. Select a zoom level or type a custom level.
3. Click **OK**.

– or –

1. Right-click the diagram.
2. From the shortcut menu, choose **Zoom**.
3. From the **Zoom** submenu, choose **Fit**.

You can change the layout of the site diagram by rotating the trees in the diagram horizontally or vertically.

To rotate the trees in the diagram

1. Right-click the diagram.
2. From the shortcut menu, choose **Rotate.**

   The page group rotates either up 90 degrees to show all children to the right of the parent pages or down 90 degrees to show all children below the parent pages.

Viewing the Full Page Path

You can view the file name and path for a page in a site diagram by viewing its ToolTip. For files external to the current project, the ToolTip displays the page label and the full URL for the file. For files in the current project, the ToolTip displays the page label and the project path for the file. You can also view this information in the property page for the page.

To view the path for a page

- Position the pointer over the page until the ToolTip appears.

  – or –

1. Select a page in the site diagram.
2. Right-click the page, and then choose **Properties** from the shortcut menu.

   The property page for the selected page appears.

Previewing a Page in a Browser

You can launch a browser, such as Microsoft Internet Explorer, to preview Web pages and see how they will appear to the user.

To preview pages in a browser

1. In the site diagram, right-click a page.
2. From the shortcut menu, choose **View in Browser**.

If you do not want to preview the page in the default browser, you can open the page using a different browser.

To preview pages in a specific browser

1. In the site diagram, right-click a page.
2. From the shortcut menu, choose **Browse with.**

   The **Browse With** dialog box appears.
3. Select a browser from the **Browser list** and click **Browse.**

Editing the Source for a Page

You can open an editor to modify the source code of a page from within a site diagram.

To edit the source for a page

1. In the site diagram, right-click a page.
2. From the shortcut menu, choose **Open**.

If you do not want to edit the page in the default editor, you can specify a different editor.

To edit the source for a page in a specific editor

1. In the site diagram, right-click a page.
2. From the shortcut menu, choose **Open with.**

   The **Open With** dialog box appears.
3. Select a program from the list and click **Open**.

Renaming Page Labels

Each page in a site diagram has a label that you can change. This label specifies the text that appears on the navigation bar for that page's link.

Global navigation bar page labels exceeding 15 characters may display multiple button graphics for a single link on the global navigation bar. To remedy this, use labels that are under 15 characters or select a smaller font size for the global navbar class in the theme's .css file.

> **Note** When you create new pages using the New HTML Page or New ASP Page command, the text you type for the label of a page becomes the name of the file that is created when you save the site diagram.

To rename a page label

1. Select a page in the site diagram.
2. Type a new name or phrase and press ENTER.

The new label text appears on the page in the site diagram. When you save the modified site diagram, the change to the navigation bar link is made in the site structure file and the navigation bars for the site are updated.

Opening a Site Diagram

You can open a site diagram from the Project Explorer. Site diagrams have a .wdm file extension in the current project.

Note If another user makes changes to a site diagram since you last updated your Web project files, those changes will not appear in your site diagram. To view the latest changes, close the site diagram and synchronize the Web project.

To open a site diagram

1. In the Project Explorer, right-click the name of the diagram that you want to open.
2. From the shortcut menu, choose **Open**.

   – or –

   Double-click the name of the diagram you want to open.

   The site diagram appears.

You can now move the pages, add new pages, or remove pages from the site diagram.

Note If you open a site diagram after changing your system colors, the site diagram colors may not display properly. To correct this, save and close the diagram before opening the diagram again.

Renaming a Site Diagram

You can change the name of an existing site diagram from the Project Explorer. Renaming a site diagram file does not affect the site structure file.

Note You cannot rename a site diagram while you are working offline.

To rename a site diagram

1. In the Project Explorer, right-click the name of the diagram that you want to rename.
2. From the shortcut menu, choose **Rename**.

   Note The Rename command is available for local files only.

3. Type a new name for the site diagram, and then press **ENTER**.

Saving a Site Diagram

When you save a site diagram, Site Designer updates the site structure file for the Web application with your changes. All pages in the project that have navigation bars will have their navigation bar links updated as well. Any pages you add to the site diagram with the New HTML Page or New ASP Page command are automatically created and added to the current project.

Note You can save changes to site diagrams in master and local modes only. You cannot edit or save a site diagram in offline mode.

To save a site diagram

- From the **Standard** toolbar, choose **Save**.

The site diagram is saved with a .wdm file extension in the current Web application. The modified page icon is removed from the pages you edited in the diagram.

Caution If another user has made changes to the site that conflict with the site diagram you are trying to save, your changes may not be saved. Instead, an error message will appear, detailing the site structure file conflicts.

Deleting a Site Diagram

You can delete a site diagram from a Web application in the Project Explorer. When you delete a site diagram, you delete the .wdm file from the project; you do not delete any of the pages represented in the site diagram.

To delete a site diagram

1. In the Project Explorer, right-click the name of the diagram that you want to delete.
2. From the shortcut menu, choose **Delete**.

   – or –

   Press the DELETE key.

CHAPTER 15

Viewing Links for an Item

You can use Link View to create a graphical representation of the links to and from items in a Web application. This graphical link representation, or link diagram, allows you to visualize the links for an item. A link diagram also helps you to identify broken links and to understand the link interactions between items in a Web application.

Use Link View to view the links for items in the current Web application or for items on the World Wide Web. You can also preview the items in a Web browser or edit the items by launching the default editor from the link diagram.

Viewing Links for an Item in a Project

You can view the links for any item located in a Web application, an item in an open link diagram, or any valid URL.

When you view the links for an item in your current Web application, you see a design-time view of the links. For example, design-time controls and other design-time-only links appear in a link diagram. Internal items appear as valid or broken in a link diagram.

To view links for an item in a project

- Right-click the item in the **Project Explorer**, and then choose **View Links**.

A link diagram appears with an icon representing the selected item in the center of the diagram. By default, new link diagrams open in horizontal layout, showing both the in links and out links for an item.

To view links for an item in a link diagram

- Right-click the item's icon in the link diagram, and then choose **View Links**.

A link diagram appears with an icon representing the selected item in the center of the diagram.

To view links for any URL, see the next section, "Viewing Links for an Item on the WWW."

Viewing Links for an Item on the WWW

You can view links for pages outside of your Web application by opening a link diagram on any valid URL. The URL can point to an internal item or to an item on the World Wide Web. When viewing external items, keep the following points in mind:

- Link View represents all external items with external icons.
- External items appear as valid, broken, unknown, or pending in a link diagram.
- Link View displays the run-time view of the links for an external URL. Links to items such as design-time controls do not appear in a link diagram for external items.
- You cannot open an external item in a default editor from a link diagram.

To view links for pages outside of a project

1. From the **Tools** menu, choose **View Links on WWW**.
2. In the **Address** edit box, type the full URL of the page you want to view, and then choose **OK**.

A new link diagram appears. The new link diagram does not replace the current link diagram, if one is open. You can switch between multiple link diagrams from the Window menu.

Refreshing a Link Diagram with Current Project Information

When working in a multiuser environment, changes may have been made to the Web application after you created your link diagram. You can refresh the link diagram at any time to ensure you are viewing the latest link information.

Link diagrams created using the View Links command use the link information stored on the master Web server in the link index file and site structure file. These files are not automatically updated when other users make changes to the site. You must manually request the latest link information with the Refresh command. Link View rebuilds the link diagram using the latest link information from the Web server.

Link diagrams created with the View Links on WWW command always use the most current link information to build a link diagram.

To refresh a link diagram with current project information

- On the **View** menu, choose **Refresh**.

Link View creates an updated link diagram for the selected item.

Previewing Items in a Browser

You can launch a browser to preview link diagram items and see how they will appear to the user. For example, if a link to an expanded item in your diagram leads to an HTML page, you can preview that page in your browser to see how it currently appears.

To preview an item in a browser

1. In the link diagram, right-click an item.
2. From the shortcut menu, choose **View in Browser**.

The item will appear inside your default browser.

If you do not want to preview the item in the default browser, you can specify a different browser.

To preview items in a specific browser

1. In the link diagram, right-click an item.
2. From the shortcut menu, choose **Browse with**.

   The **Browse With** dialog box appears.
3. Select a browser from the list, and then click **Browse**.

Launching Editors on Items in a Link Diagram

You can select an item in the link diagram and open it using the default editor associated with that item. For example, if a link diagram shows that an item contains a broken link, you can quickly open the item with the broken link in its editor and search for the problem.

The default editor for an item is determined by the file-type association set from either My Computer or the Windows Explorer. For example, if you open an item with a .doc extension, the document is opened in Microsoft Word. For more information on file-type associations, see Windows Help online.

To edit an item from Link View using the default editor

- In the link diagram, right-click the item and choose **Open** from the shortcut menu.

  – or –

  In the link diagram, double-click the item.

If you do not want to edit the page in the default editor, you can specify a different editor.

To edit the source for a page in a specific editor

1. In the link diagram, right-click an item.
2. From the shortcut menu, choose **Open with.**

   The **Open With** dialog box appears.
3. Select a program from the list, and then click **Open**.

Printing a Link Diagram

You can print any link diagram in Link View. You can also increase or decrease the size of the link diagrams you print to fit your page size requirements.

To scale a link diagram for printing

1. Open a link diagram.
2. From the **File** menu, choose **Print Setup**.

   The **Print Setup** dialog box appears.
3. Enter a percentage in the **Print scale** option to increase or decrease the size of the diagram for printing purposes.

You can easily print any link diagram.

To print a link diagram

- From the **File** menu, choose **Print**.

CHAPTER 16

Maintaining Links

Maintaining the links in your Web application is important to your users. With Link View, it is easy to visualize links and to fix broken links in your Web application.

You can quickly organize the links in a link diagram by filtering a link diagram to show only the types of files or only the types of links you want to view. For example, you could filter out multimedia files so you can focus on finding broken links in .htm or .asp files.

You can also manage the size and complexity of a link diagram by centering the diagram, changing the magnification of the diagram, or by expanding and collapsing links. By managing the size of the link diagram, you can fit the entire diagram in the current window or zoom in to view the file names for items. You can get a different view of your link diagram by switching between a horizontal layout and radial layout.

Filtering Links

You can filter a link diagram to show only the types of items you want to see. You can apply more than one filter category to a link diagram. In Link View, you can set filters based on both the types of items and types of links between items.

- Showing and Hiding Items
- Showing and Hiding Links

Showing and Hiding Items

Filters can be used to show and hide specific types of items in the link diagram. Available filters appear on the Diagram menu and are represented by buttons on the Link View toolbar.

The following filters apply to all types of items.

Filter category	Toolbar button	Displays
All items	🗐	All types of items
External files	🌐	All types of items outside of the project **Note** The external files filter category can be useful to hide all of the files that are not part of the current Web application.

To select one or more filters

- From the **Diagram** menu, choose the filters you want to turn on or off in the current link diagram:

Filter category	Toolbar button	Displays
Documents	🗎	Files associated with applications, such as Microsoft Word or Microsoft PowerPoint
Executable files	🗎	Files representing programs, applications, batch files, scripts, and DLLs
HTML pages	🗎	HTML pages, including special types of HTML layouts such as Active Server Pages
Multimedia files	🖼	Multimedia files, such as image, audio, video, and virtual reality files
Other protocols	🖳	Links using protocols other than http, such as mail, news, and telnet

Note When you set a filter, it applies to the current link diagram only. If multiple link diagrams are open, you must set filters for each diagram separately.

Showing and Hiding Links

You can manage the complexity of the link diagram by limiting the types of links that are shown in the diagram. For example, you can choose to hide repeated links and links within a page, or to show only inbound links.

Repeated Links

A repeated link occurs if one page has multiple links to another page in the link diagram. For example, if you have a Web page that contains multiple links to your glossary page, you may want to reduce the complexity of the link diagram by hiding all but the first link to the glossary page.

To show and hide repeated links

- Click **Show Repeated Links** on the Link View toolbar.

  When the Show Repeated Links button is pressed in, a separate icon for each repeated link appears in the link diagram. When the Show Repeated Links button is raised, multiple links from one item to another item in the link diagram are consolidated into a single link.

Bookmark Links

Bookmark links are links that refer to a different part of the same page. You can hide all the bookmark links in a link diagram to reduce the size and complexity of a link diagram.

To show links inside pages

- Click **Show Links Inside Pages** on the Link View toolbar.

  When the Show Links Inside Pages button is pressed in, a separate link arrow for each link within a page appears in the link diagram. When the Show Links Inside Pages button is raised, links inside pages do not appear in the link diagram.

In/Out Links

You can also filter the link diagram to show only out links or only in links by changing the diagram layout options. For example, showing in links enables you to determine the impact of renaming or deleting an item because you can see other pages that currently link to that item.

Filter category	Toolbar button	Displays
Show in links	→□	Links that point into an expanded item, indicating that the item at the other end of the line links to the expanded item.
Show out links	□→	Links that point out of an expanded item, indicating that the expanded item links to the items it points to.
Show in and out links	→□→	Both links that point out of an expanded item and links that point into an expanded item.

For more information, see "Changing a Link Diagram Layout," later in this chapter.

Viewing Link Diagrams

You can easily place a link diagram in the center of the current window, change the magnification of a link diagram, expand and collapse links, select items in a link diagram, and determine the URL of an item.

- Centering Link Diagrams
- Zooming Link Diagrams
- Expanding and Collapsing Links
- Selecting Items in a Link Diagram
- Identifying the Full URL of an Item

Centering Link Diagrams

By default, the entire link diagram is centered inside the current window when you first create the link diagram. When you expand the links for items in a link diagram, Link View horizontally scrolls to keep the expanded items in view.

You can recenter a link diagram around any selected item in the link diagram. You may want to do this when you have increased the size of the link diagram beyond the boundaries of your screen.

To center an item in the current window

1. Select the item in the link diagram you want to center in the current window.
2. From the **View** menu, choose **Center View**.

The selected item is centered inside the current window.

You can also recenter the original expanded item in the current window.

To recenter the original expanded item

1. Click the background of the link diagram to unselect any selected items.
2. From the **View** menu, choose **Center View**.

   The original expanded item appears in the center of the current window.

Zooming Link Diagrams

You can change the magnification of a link diagram to view your Web application from different aspects. When you zoom in, items appear larger and you can read their labels more easily. When you zoom out, you can see more of the whole site at a glance. Link diagrams are arranged around the initial expanded item.

To zoom a link diagram

- Select a zoom value from the **Zoom** control on the Link View toolbar.

  The link diagram resizes according to the zoom level you selected.

You can also enter any zoom level you want to resize the link diagram.

To set a custom zoom level

- Type any value directly into the **Zoom** control on the Link View toolbar. Link View accepts values from 10% to 400%.

  The link diagram resizes according to the zoom level you entered.

You can also automatically resize the link diagram to fit within the boundaries of the current window.

To fit the entire link diagram into the current window

- Select **To Fit** from the **Zoom** control on the Link View toolbar. When zooming to fit, Link View will not zoom smaller than 10% or larger than 100%.

Expanding and Collapsing Links

If you want to see more detail in the current link diagram, you can expand links for the selected item to show the links for that item. The links that appear depend on the filters currently set for the link diagram. For example, if you have set filters to show only HTML pages in your link diagram, expanding links on a page will display only links to other HTML pages.

You can also collapse links for an item in a link diagram. Collapsing a link hides the links for that item and reduces the size of a large link diagram for a site.

To expand links on an item

1. In the link diagram, right-click the item that you want to expand.
2. Choose **Expand Links**. The icon beside the command appears pressed in, and all links for the current item are displayed.

   Note In horizontal layout, Link View expands either in or out links for an item, but not both. If you expand the items to the left of the center item, Link View displays only the in links for those items. If you expand items to the right of the centered item, Link View displays only the out links for those items.

In some cases, an item does not have links to or from other items. Link View, therefore, does not enable the Expand Links command in these cases. You would also see no links if the page does not link to any items that meet the currently active filter criteria.

To collapse links on an item

1. In the link diagram, right-click the expanded page that you want to collapse.
2. Choose **Collapse Links**. All links for the current page are collapsed.

You can collapse links at any level. For example, if item A is expanded to show a link to item B and item B is then expanded, you can collapse the links to item A without first collapsing the links to item B.

Selecting Items in a Link Diagram

You can select any single item in a link diagram, and then execute commands on that item. For example, you can select an item and choose the Open command to view the item in its default editor.

To select an item in a link diagram

- Click the item.

You can select multiple items in a link diagram to execute commands on the set of selected items. For example, you can select multiple items and choose the **Expand** command to see all the links for all the selected items.

To select multiple items in a link diagram

- Individually select items by holding down the **CTRL** key while you click the items.

  – or –

  From the **Edit** menu, choose **Select All.**

  – or –

  Click anywhere in the link diagram and drag the pointer. A thin dotted line forms a rectangle as you drag. When you release the mouse button, all items within the rectangle are selected.

When you choose the Expand Links command on a group of selected items, Link View expands each item in the order that it was selected. If your selection includes an item that cannot be expanded, that item is skipped.

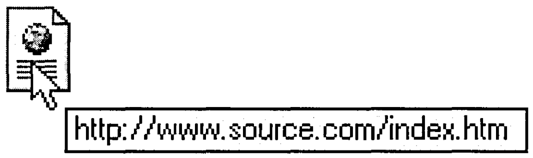
Identifying the Full URL of an Item

When an item appears in a link diagram, it is identified by a label that shows its file name. If you want to see the complete URL of an item, you can view the item's ToolTip.

Note For broken links, the ToolTip also displays the WinInet error for the item indicating why it is broken.

To view the full URL of an item

- Position the pointer over the item until the ToolTip appears.



http://www.source.com/index.htm

Changing a Link Diagram Layout

Link diagrams can have two different layouts: the horizontal layout and the radial layout. When you view the links for an item, the horizontal layout appears by default.

The horizontal layout shows both the in links and the out links for an item.

The radial layout appears when you choose to see either the in links or the out links for an item.

You can switch between the two link diagram layouts to get a different view of the links for an item.

To change the layout for a link diagram

- From the **Link View** toolbar, click **Change Diagram Layout**.

The link diagram changes to either the horizontal layout or the radial layout.

Repairing Links

In large Web sites, it is essential that you can easily maintain the links in your Web applications. To facilitate this task, Microsoft Visual InterDev offers the following assistance:

- Preventing Broken Links
- Updating Link Information
- Finding Broken Links and Unreferenced Files

Preventing Broken Links

When you reorganize your Web application by deleting or moving files and folders, you might break links within the application. Visual InterDev provides a way to automatically repair links that would be broken by deleting or moving files and folders.

With link repair enabled for a project, all relative links in target files are repaired. Target files are the files that are being renamed, moved, or deleted. Referring files are files that contain links to target files. Links in referring files are repaired depending on the setting of the Link Repair option in the Web Project Options (Projects — Options Dialog Box) in the Options dialog box. Link repair for a project is enabled or disabled from the General Tab (Project Properties Dialog Box).

To control automatic link repair

1. On the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, select **Projects,** and then select **Web Projects** in the navigation pane.

3. In the **Link repair** area, choose a link repair option.

To	Select
Always update links automatically on both the master Web server and the local Web server	Repair links in referring files
Never update links automatically	Don't repair links in referring files
Get a prompt and decide each time	Ask me each time

Warning When you work with files under source control, you should check all of your Web application files in before renaming or moving files. If a file is checked out exclusively by another user, the links within the file may not be updated properly.

You should also periodically update the link information on your local Web server and the master Web server to provide the most current information for link diagrams and the Broken Links Report.

Updating Link Information

If you work in a multiple developer environment, chances are high that the changes you or others make to the master Web server will change the link information for the Web application. To keep your link information current, you should periodically refresh the link information on both your local Web server and the master Web server.

You can refresh your link information by using the Recalculate Links command. In master mode and local mode, the Recalculate Links command first updates the link information on the master Web server and then updates the link information on the local Web server. When working offline, Recalculate Links updates only the link information on the local Web server.

Note Recalculating links can take several minutes, depending on the size of the Web application. While Visual InterDev recalculates links, you will be unable to work on other files in Visual InterDev.

To recalculate links explicitly

1. In the **Project Explorer**, select the root of the project.
2. From the **Project** menu, choose **Web Project**, and then choose **Recalculate Links**.

   Visual InterDev examines all of the files in the Web application and updates the link information. If your project includes a Search.htm file like the one generated by the Web Project Wizard, this file is also updated.

Recalculating links for a Web project provides the most current link information for link diagrams and the Broken Links Report.

Finding Broken Links and Unreferenced Files

In a large Web site, you need an easy way to find all of the broken links in your files. Visual InterDev provides two methods for identifying broken links: link diagrams and the Broken Links Report. A link diagram allows you to visually verify links for individual files or sets of files. For more information about visualizing broken links, see Chapter 15, "Viewing Links for an Item."

The Broken Links Report allows you to identify broken links and unreferenced files for an entire Web project. Unreferenced files are files in your Web project that do not have any in links. These files may be necessary for creating and maintaining your Web project but do not need to be deployed to the production server. For example, site diagram files (*.wdm) contain essential navigation structure information but are not needed on the production Web server.

> **Note** You can specify which file types are not deployed in the Web Project Options (Projects — Options Dialog Box) in the Enterprise Edition of Microsoft Visual Studio and Visual InterDev.

Unreferenced files may also be files that contain links that the Broken Links Report does not recognize or files that are no longer used and should be deleted.

> **Note** Creating a Broken Link Report can take several minutes, depending on the size of the Web application. While Visual InterDev verifies links, you will be unable to work on other files in Visual InterDev.

To identify broken links and unreferenced files

1. In the **Project Explorer**, choose the root of the project.
2. From the **View** menu, choose **Broken Links Report**.

   The Links Report message appears, indicating that the links for the Web application are being checked.

When the Broken Links Report is complete, the files with broken links and unreferenced files are listed in the Task List Window and the Output Window.

From the Task List window, you can double-click any file listed to open the file in the default editor.

CHAPTER 17

Customizing Page Appearance

Professional-looking Web sites stand apart because of their quality and consistency of design. Microsoft Visual InterDev helps you achieve a high standard of design quality with templates, style sheets, themes, and layouts.

Using any or all of these features enforces standardization across your Web site as well as shortens the time you spend on development and production:

- Templates create standard pages that can contain common HTML and script.
- Style sheets instruct the browser to override browser default fonts, sizes, and colors.
- Themes employ a common set of graphics.
- Layouts define how the navigation and content are laid out on regions of a page.

Creating Files with Templates

Any existing HTML or ASP page can be used as a template for creating new identical pages. Using templates can save you a lot of time as well as give your Web site a consistent look and feel. When creating a page, you can base it on a standard Microsoft Visual InterDev template or use one of your own custom templates.

When you create files for your Web project, Visual InterDev looks in the Templates directory for .htm, .asp, and other files and then displays a list of those templates in the right pane of the New tab of the Add Item dialog box.

To create a page based on a template

1. From the **File** menu, choose **New File**.

   – or –

   From the **Project** menu, choose **Add Item**.
2. In the **New** tab, select **Web Project Files** in the left pane.
3. Select the desired template in the right pane.

For information about creating your own templates, see "Creating Custom Templates" later in this chapter.

Editing Style Sheets

Cascading style sheets (CSS) give you the ability to define a set of styles that overrides the browser's standard methods for rendering HTML. This lets you give your pages a unique and consistent design. For an in-depth discussion on using style sheets with Microsoft Visual InterDev, see "Site Consistency" in Chapter 11, "Site Design."

You can create or edit cascading style sheets in the CSS editor. The CSS editor has an easy-to-use interface for setting style properties. You can use the editor's graphical interface so that you don't have to edit raw CSS text.

To create a style sheet in your Web project

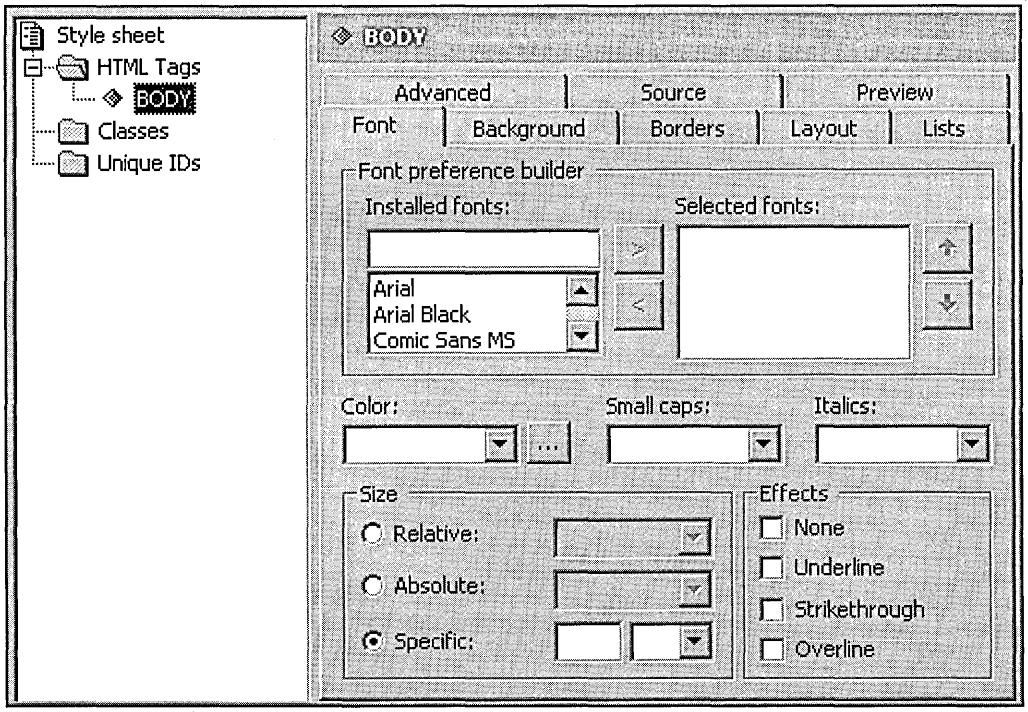
1. From the **Project** menu, select **Add Web Item** and then **Style Sheet**.
2. Type a name in the **Name** box and choose **Open**.

You can also create a style sheet independent from a Web project.

To create a style sheet outside of a Web project

1. From the **File** menu, choose **New File**.
2. On the left, choose **Visual InterDev**.
3. On the right, **Style Sheet**.
4. Type a name in the **Name** box and choose **OK**.

The CSS editor will be displayed with your new style sheet.



Using the CSS Editor

In the CSS editor you select an HTML tag, class, or unique ID in the left pane and then set its properties in the right pane. For information about the editor's interface, see "CSS Editor Window" in Visual InterDev's online documentation.

Adding HTML Tags for Editing

Depending on the style sheet that you are editing, you may have many or just a few tags. If you need to modify the style of a tag that is not listed, you can add it to the list in the left pane.

To add an HTML tag

- From the **StyleSheet** menu, choose **Insert HTML Tag**.

Editing HTML Tags

You can override the standard settings of an HTML tag.

To set style properties for an HTML tag

1. Choose the + icon to expand the **HTML Tags** node.
2. Select a tag.
3. In the right pane, select tabs and options to set the properties of the style.

Editing Classes

You create a class to contain certain properties that you apply to a specific tag or make available to all tags.

To create a class

1. From the **StyleSheet** menu, choose **Insert Class**.
2. In the **Insert New Class** dialog box, type a name in the **Class name** box.
3. If you want to apply the class to a specific tag, select **Applies only to the following tag** and select a tag from the list box.

After the class has been created and added to your style sheet, you can edit its properties in the right pane.

Editing Unique IDs

Unique IDs are used like classes except they can be used only once per page.

To create a unique ID

1. From the **StyleSheet** menu, choose **Insert Unique ID**.
2. In the **Insert Unique ID** dialog box, type a name in the **Unique ID** box.
3. If you want to apply the ID to a specific tag, select **Applies only to the following tag** and select a tag from the list box.

After the ID has been created and added to your style sheet, you can edit its properties in the right pane.

Previewing a Style Sheet

You can preview how any page, local or on the World Wide Web, would look if the style sheet were applied.

To preview a style sheet applied to a page

4. In the CSS editor, choose the **Preview** tab.
5. In the address box, type the URL of any page.

   Note If the page you are previewing already has a style sheet attached, Visual InterDev recognizes that style sheet as the primary style sheet and correctly applies the one that you are editing as secondary.

Applying Themes

Give your Web design a professional polish by applying themes. A theme is comprised of a set of graphics and a cascading style sheet (CSS) that controls the styles, font, and other elements. For information on site consistency including themes, see "Site Consistency" in Chapter 11, "Site Design."

You can set a default theme for an entire Web project, or you can apply a specific theme to an individual file.

Applying a Theme to a Project

When you create a Web project, you can choose a default theme for the project.

To set a default theme when creating a Web project

- Run the Web Project wizard and select a theme when prompted.

Now, by default, all pages that you create in the Web project will use the default theme.

You can still apply a project default theme even if you did not select one when creating the project.

To apply or change the default theme of an existing Web project

1. In the **Project Explorer**, select the project name.
2. From the **View** menu, choose **Property Pages.**
3. In the dialog box, select the **Appearance** tab.
4. Under **Default Theme and Layout**, click the **Change** button to display the **Apply Theme and Layout** dialog box.
5. Click the **Theme** tab.
6. Select **Apply theme** to activate the scrolling list of themes.
7. Select a theme from the list.

If you later decide that you don't want to use a theme, you can remove it from the project without affecting the content.

To remove a theme from a Web project

1. In the **Project Explorer**, select the project name.
2. From the **View** menu, choose **Property Pages.**
3. In the dialog box, select the **Appearance** tab.
4. Under **Default Theme and Layout**, click the **Change** button to display the **Apply Theme and Layout** dialog box..
5. Click the **Theme** tab.
6. Select **Apply** to activate the scrolling list of themes.
7. At the top of the list, select **<none>.**

At this point, the theme is no longer applied to the project, but the theme files still exist in the project. If you want to actually remove the files from the project, select the unwanted theme in the Themes folder and delete it.

Applying a Theme to a Single Page

You can apply a theme to a single page. If the project that contains the page already has a default theme, the page-level theme overrides the project-level one.

To apply or change the theme of a single page

1. In the **Project Explorer**, select an HTML or ASP page.
2. From the **Edit** menu, choose **Apply Theme and Layout.**
3. In the dialog box, select the **Theme** tab.
4. Select **Apply** and specify a theme.

If you decide that you do not want a particular page associated with a theme, you can opt to apply no theme or return to the project-level theme.

To remove a theme from a single page

1. In the **Project Explorer**, select an HTML or ASP page.
2. From the **Edit** menu, choose **Apply Theme and Layout**.
3. In the dialog box, select the **Theme** tab.
4. Select **Preserve Current Theme**.

   – or –

   Select **Apply** and then **<none>** from the list.

   Preserve Current Theme instructs Microsoft Visual InterDev to use the project's default theme, while **<none>** explicitly specifies that there is no theme applied to the page.

   Note The **Apply Theme and Layout** command is disabled when you are working offline.

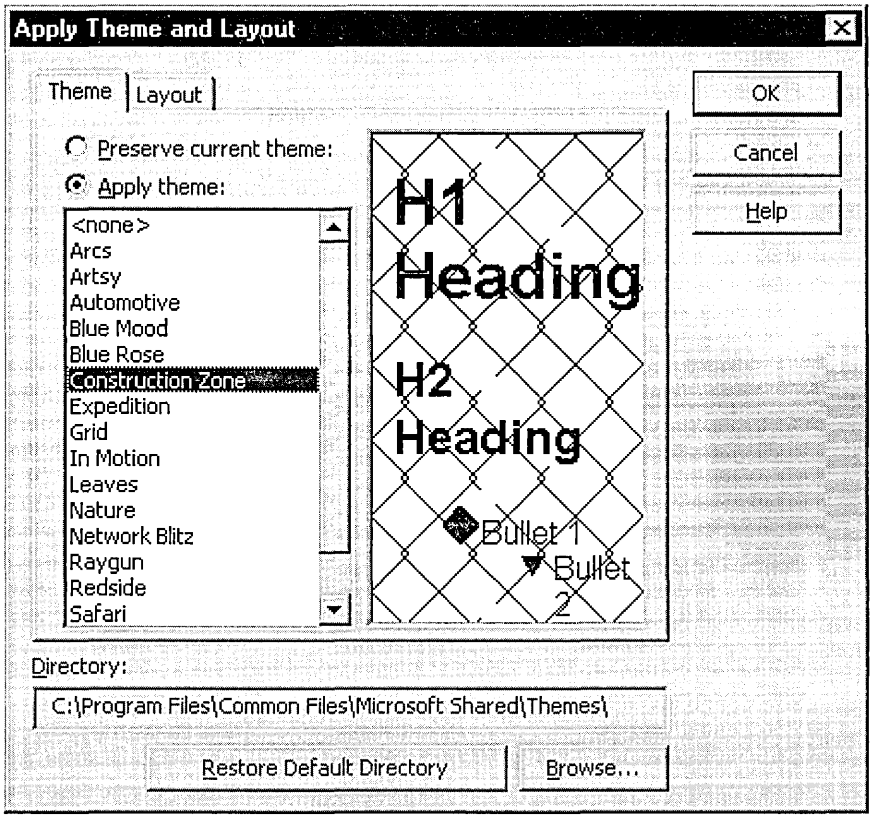
Previewing a Theme

You can browse through all the available themes in the Apply Theme and Layout dialog box. This preview allows you to test how a theme looks before you apply it to a page or project.

To preview a theme

1. In the **Project Explorer**, select an HTML or ASP page.
2. From the **Edit** menu, choose **Apply Theme and Layout**.
3. In the dialog box, select the **Theme** tab.
4. Select **Apply theme** to activate the list of themes.

5. Click on any theme on the left to display a preview on the right.



Preserve current theme:

Apply theme: Construction Zone

<none>
Arcs
Artsy
Automotive
Blue Mood
Blue Rose
Construction Zone
Expedition
Grid
In Motion
Leaves
Nature
Network Blitz
Raygun
Redside
Safari

Directory: C:\Program Files\Common Files\Microsoft Shared\Themes\

6. Cancel the dialog box if you do not want to apply the theme.

Visual InterDev fills the list with themes from both your local installation as well as the themes residing on the project's master server. For information concerning theme locations, see "Themes Installed on Your Local Machine" in Chapter 11, "Site Design."

Creating a Custom Theme

Creating a custom theme is as simple as supplying files in the correct locations.

To create a custom theme

1. Create a directory named after your theme.
2. Add your cascading style sheets to the theme directory.
3. Add your images to the theme directory.

Creating the Theme Directory

All the elements of a theme are maintained in a unique directory.

To create a theme directory

- On your local machine, create a directory in the _Themes directory.

  Note If you haven't yet applied a theme to a project or to one of the project files, then the Themes folder won't yet exist in your project.

The name that you give the directory determines the name of the theme.

Later, when you apply the theme using the **Apply Theme and Layout** dialog box, Visual InterDev will look in the _Themes directory and include your custom theme in the list of available themes. You will also be able to preview your theme in the dialog box along with the other themes.

The theme is automatically copied to the server when you apply the theme to a page or project. Visual InterDev copies the directory and all its files to the server.

Creating a Cascading Style Sheet for a Visual InterDev Theme

Cascading style sheets allow you to format the style of your page without affecting the content. When used in conjunction with themes, you typically:

- Define the font, name, size, and other attributes of tags.
- Specify image files to use with bulleted lists and horizontal rules.

You can create a cascading style sheet with the CSS editor. You can also use the editor to modify an existing style sheet. For information on the CSS editor, see "Editing Style Sheets," in this chapter, or "CSS Editor Window" in Visual InterDev's online documentation.

When you have finished editing your style sheet, save it and move it to your theme directory.

Adding Images to the Theme Directory

By default, Visual InterDev references images in the Themes directory. Copy all images that you want to include in your theme into your custom theme directory.

Sharing Customized Themes with FrontPage

Note Be careful if you are sharing a project between Visual InterDev users and Microsoft FrontPage users.

If you customize one of the themes that is supplied by Visual InterDev, and then apply the theme to a file using FrontPage, FrontPage might overwrite your customizations.

To avoid this, rename themes when you customize them and use new unique names. Also, in the theme's .inf file, set the `title=` line to reflect the new theme name.

Laying Out Pages

When you lay out a page you design it so that the user can easily scan the page for content, navigation, and orientation. Microsoft Visual InterDev includes a new feature, layouts, which makes it easy to organize your pages into logical regions.

When a layout is applied to a page, the layout:

- Defines how the content of the page is visually laid out in regions.
- Implements navigation among parent, children, and sibling pages using the PageNavBar design-time control.
- Includes a consistent use of a title region, if desired.

Applying a Layout to a Project

When you create a Web project, you can choose a default layout for the project.

To set a default layout when creating a Web project

- Run the Web Project wizard and select a layout when prompted.

Now, by default, all pages that you create in the Web project will use the default layout.

You can still use layouts if you choose not to use one when creating the project.

To apply or change the default layout of an existing Web project

1. In the **Project Explorer**, select the project name.
2. From the **View** menu, choose **Property Pages**.
3. In the dialog box, select the **Appearance** tab.
4. Under **Default Theme and Layout**, choose **Change** to specify a layout.

If you later decide that you don't want to use a layout, you can remove it from the project without affecting the content.

To remove a layout from a Web project

1. In the **Project Explorer**, select the project name.
2. From the **View** menu, choose **Property Pages**.
3. In the dialog box, select the **Appearance** tab.
4. Under **Default Theme and Layout**, choose **Change**.
5. Select **Apply** to activate the scrolling list of layouts.
6. At the top of the list, select **<none>**.

At this point, the layout is no longer applied to the project, but the layout files still exist in the project. If you want to actually remove the files from the project, select the unwanted layout in your project's Layouts and delete it.

Applying a Layout to a Single Page

You can apply a layout to a single page. If the project that contains the page already has a default layout, the page-level layout overrides the project default.

To apply or change the layout of a single page

1. In the **Project Explorer**, select an .htm or .asp file.
2. From the **Edit** menu, choose **Apply Theme and Layout**.
3. In the dialog box, select the **Layout** tab.
4. Select **Apply** and specify a layout.

If you later decide that you do not want a layout applied to a particular page, you can choose to apply no layout or return to the project default.

To remove a layout from a single page

1. In the **Project Explorer**, select an .htm or .asp file.
2. From the **Edit** menu, choose **Apply Theme and Layout**.
3. In the dialog box, select the **Layout** tab.
4. Select **Preserve Current Layout**.

   – or –

   Select the **Apply** option and then **<none>** from the list.

   Preserve Current Layout instructs Visual InterDev to use the project's default layout, while **<none>** explicitly specifies that there is no layout applied to the page.

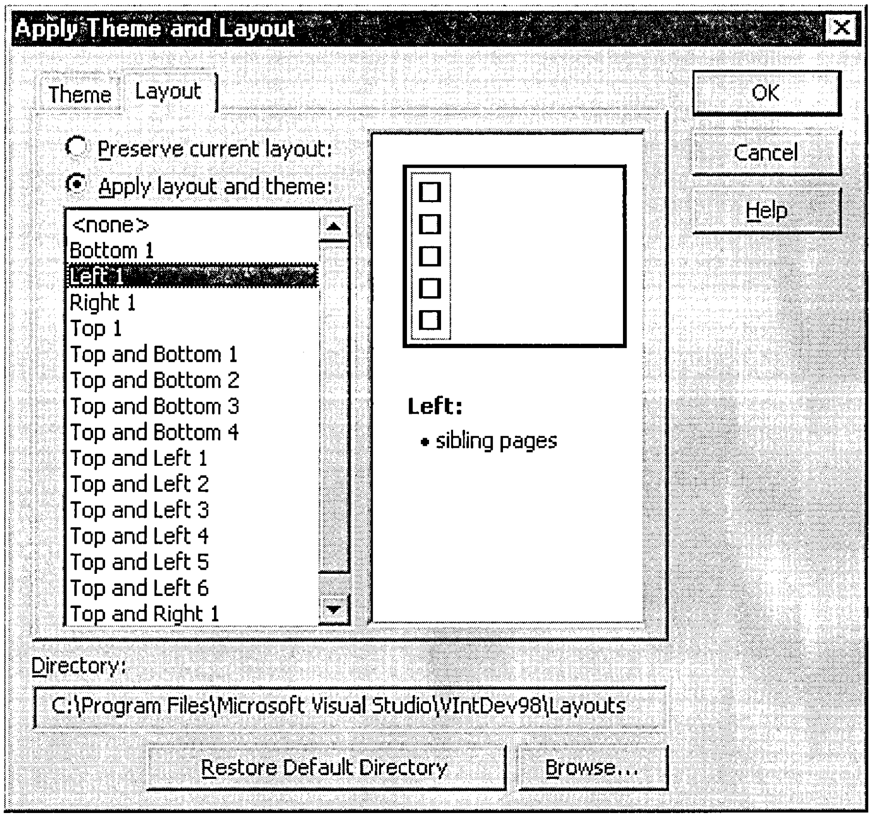
   Note The **Apply Theme and Layout** command is disabled when you are working offline.

Previewing a Layout

You can browse through the layouts in the **Apply Theme and Layout** dialog box. Images appear in the dialog box that are representative of the layouts themselves. This preview allows you to see how a layout looks before you apply it to a page or project.

To preview a layout

1. In the **Project Explorer**, select an .htm or .asp page.
2. From the **Edit** menu, choose **Apply Theme and Layout**.
3. In the dialog box, select the **Layout** tab.
4. Click on any layout on the left to display a preview on the right.



Apply layout and theme:

<none>
Bottom 1
Left 1
Right 1
Top 1
Top and Bottom 1
Top and Bottom 2
Top and Bottom 3
Top and Bottom 4
Top and Left 1
Top and Left 2
Top and Left 3
Top and Left 4
Top and Left 5
Top and Left 6
Top and Right 1

Left:
- sibling pages

Directory:
C:\Program Files\Microsoft Visual Studio\VIntDev98\Layouts

Creating Custom Layouts

You can create your own custom layout to meet your own design needs. Layouts are located in the Layouts directory of a project.

> **Note** If you haven't yet applied a layout to a project or to one of the project files, then the _Layouts folder won't yet exist in your project.

Each Visual InterDev layout is comprised of three files. You can open the files in the editor in order to become more acquainted with their contents. The files serve the following purposes:

File name	Purpose
name.inf	*name* is the name of the layout. *name* matches the name of the parent folder. This file contains information about the layout.
Layout.htm	This template contains the HTML that is inserted into a page when a layout is applied to the page.
Preview.htm	This file is called when the layout is selected in the **Apply Theme and Layout** dialog box.

The easiest way to create a custom layout is to copy an existing one and then modify it.

To create a custom layout template

1. In the project's _Layout folder, create a new folder and name it.
2. Copy the files from one of the existing layout folders and paste them into you new folder.
3. In your new folder, select the **.inf** file and rename it so that it matches the folder name.
4. Double-click the .inf file to open it.
5. On the `title=` line, specify a new title. This is the string that appears in the list of layouts in the **Apply Theme and Layout** dialog box.
6. Save and close the .inf file.
7. Double-click **Layout.htm** to open it in the editor.
8. Click the **Source** tab of the editor. Now you can see the Layout design-time controls and the HTML tags that define the layout's table. Modify the HTML tags and text to suit your needs.

   > **Warning** Do not modify the run-time text of the Layout design-time controls.

9. Save and close Layout.htm.
10. Double-click **Preview.htm** to display it in the editor. This file is called when the layout is selected in the **Apply Theme and Layout** dialog box. Edit the file to suit your needs.
11. Save and close Preview.htm.

Now Visual InterDev will automatically include your custom layout in the list of available layouts when you display the **Apply Theme and Layout** dialog box.

For detailed information on Layouts, see "Layouts" in Chapter 11, "Site Design."

Creating Custom Templates

A custom Microsoft Visual InterDev template is any .htm, .asp, or other text file that you create and place in the Templates directory. You can further customize the template by including parameters that prompt the user for input that is added to the new file based on the template.

A template can be:

File type	Result
A single .htm file	An HTML page
A single .asp file	An ASP page
Other text file	A text file

To create a custom template

- Move your .htm .asp, or other text file to the Templates directory. The installation default is "C:\Program Files\Microsoft Visual Studio\VintDev98\Templates."

You can further customize a template so that it prompts the user for input when creating a new file.

To prompt the user for input

- Use delimiters (<%# and #%>) in the template to create a parameter. Parameters can be up to 100 characters long and are case-insensitive.

  For example, to prompt your user with a question, place text between the delimiters, as follows:

  ```
  <B><%#What is the name of your department?#%> Expense Report</B>
  ```

  If the user responds with the input "Marketing," then the new file would contain this line:

  ```
  <B>Marketing Expense Report</B>
  ```

The following parameters are reserved by Visual InterDev for unique functions.

- <%#DataConnection#%>

  When Visual InterDev encounters this parameter, it prompts the template user to select one of the existing project data connections.

- <%#FilenameWithExtension#%>

  This parameter automatically inserts the template's file name and extension into the new .htm or .asp file.

- <%#FilenameWithoutExtension#%>

  This parameter automatically inserts the template's own file name without its extension into the new .htm or .asp file.

- <%#ThemeName#%>

  When Visual InterDev encounters this parameter, it prompts the template user to select one of the available themes.

For more detailed information on Visual InterDev templates, read the technical paper "Creating Your Own Visual InterDev Templates" on our Web site at http://www.microsoft.com/vinterdev/techmat/whitepapers/vitemplt.htm/.

PART 4

Integrating Databases

Part 4 provides information about data connections, database management, the data environment object model, server and client access to databases, as well as other data access information.

Chapter 18 Database Concepts

This chapter covers the concepts behind the Visual InterDev "Data Access Architecture," the "Data Environment," and "Data Binding."

Chapter 19 Viewing Data

In Visual InterDev, you can display data on your Web pages using data-bound design-time controls. This chapter describes "Getting Records" and "Displaying Data on your Web Page."

Chapter 20 Modifying Data

In addition to displaying data, you can create Web pages that allow users to modify data — update existing records, add records, and delete records.

Chapter 21 Accessing Databases Directly

This chapter covers debugging stored procedures and working with database commands through the Data Environment.

Chapter 22 Managing Database Projects

Database projects allow you extensive control over your SQL Server or Oracle database.

CHAPTER 18

Database Concepts

Data Access Architecture

Microsoft Visual InterDev allows you great flexibility in designing Web applications with a database component. You can use any database supported by ActiveX Data Objects (ADO) for which you have drivers, including Microsoft SQL Server, Microsoft Access, Oracle, and others.

You can interact directly with the database or use views, stored procedures, and other database entities to manage the database. The database can be physically located on the same computer as your Web server or on a different computer. You can do all your database access using the Web server, or you can access the database directly from a client computer.

To prepare for the data access features of Visual InterDev, consider these concepts first:

- Database Integration in Visual InterDev
- Web Servers and Databases
- Data Connections
- Database Management

Database Integration in Visual InterDev

Visual InterDev integrates a wide array of features to help you create Web applications with a database component:

- **Database projects** A type of project you can add to your Visual InterDev solution that includes tools required to build and manage your databases as a separate component from Web pages.
- **Data View window** A window that provides a live view of the data to which your database or Web project is currently connected. From the Data View window, you can launch tools to manage your database, whether you are in a database project or Web project.

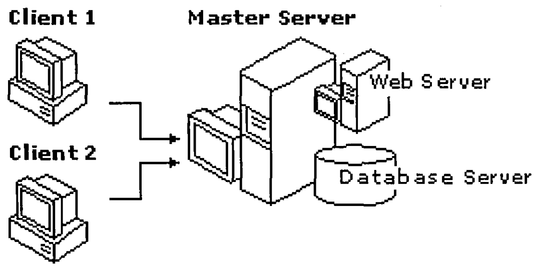
- **Microsoft Visual Database Tools** A set of tools for managing and querying your database graphically. The Database Designer allows you to create and modify table definitions, column definitions, and relationships between tables in Microsoft SQL Server and Oracle databases. Using the Query Designer, you can visually create and run SQL statements. A View Designer enables you to create views.
- **Data Environment** A repository in your Web project for information required to connect to and access data in databases. The data environment stores reusable connection strings that allow you to access databases from your Web pages. In addition, the data environment stores data commands that represent recordsets, based on database objects or SQL commands.
- **Data-bound controls** Controls such as text boxes, buttons, and so on, that you can put on a Web page and that are automatically bound to specific fields in a database record. Data-bound controls already include script required to make data connections, extract data, and update the database so you can build database access into your Web pages with little or no scripting.
- **Source control for database objects** A link between databases and your source control system so you can put SQL scripts and compiled stored procedures under source control. This makes development simpler and more secure in companies with more than one database programmer.

Web Servers and Databases

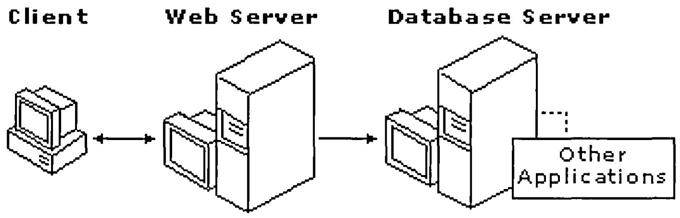
In Web applications that access databases, two server-like functions are occurring: the Web server handles requests for pages and a database server or equivalent software handles database access. Although these two server functions are part of the same application, each functions separately.

You can configure Web servers and database servers in various ways, depending on how you want people to use the database server, what the target audience is for your Web application, and how the application relates to other applications in your business. The following configuration options are possible:

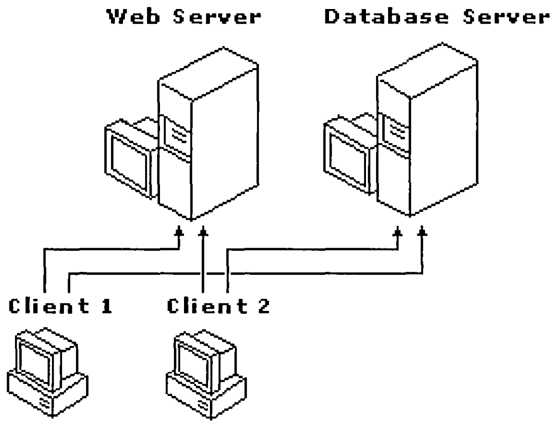
- The database server can run on the same computer as the Web server.



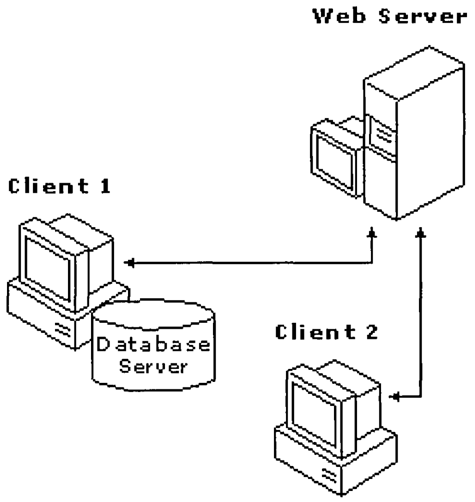
- The database server can be on a separate computer from the Web server. You might do this if you wanted to optimize each computer for its respective task, if you want to share a database server among several Web servers, or if the database server is used for applications other than the Web application.



- The database server and Web server can be entirely separate processes that do not communicate. This model is practical if client computers can access the database directly, which can increase performance.



- The database can be on a local (client) computer. You might do this in special cases, such as testing.



Using the Web server as a gateway to the database is the most common strategy, because it gives Web applications the widest reach — the server does the database work, so it doesn't matter what type of browser the user has. Server-side database access is therefore a good choice for public Internet sites, where users might access the site with any browser.

Client-side access, on the other hand, can provide a richer user experience that emphasizes performance, because the browser can manage data sets independently from the Web server. However, client-side access requires Internet Explorer DHTML as a browser, and also that the database be accessible via specific database drivers. Client-side database access is therefore most practical in intranet sites, such as a corporate Web site, where users access the site using a standard browser with predictable features.

For information about Web server and database server security, see "Security" in Chapter 6, "Web Project Concepts."

Data Connections

To use a database, you add a data connection to your Visual InterDev project, which tells it how to access the database. Typically, a data connection includes information such as:

- The type of database you are accessing (for example, Microsoft SQL Server) and the server name (if appropriate).
- The name of the database (for example, pubs).
- A user name.
- A password.

You can add as many data connections to your project as you need. For example, if your application requires access to two different databases, you would add two data connections. For details about how to add a data connection, see "Connecting to a Database" in Chapter 3, "Database Basics."

Creating Connections

When you are creating a data connection, Visual InterDev reads connection information from a DSN (data source name) on your computer. Your computer can provide the DSN information in either a file DSN (stored in a .dsn file) or a machine or system DSN (stored in the Windows registry of your computer).

File Data Source Name

In the case of a file DSN, Visual InterDev reads the connection information from the DSN and extracts the connection string. It then adds a connection node to your project's data environment.

The connection string extracted from the DSN is stored in your Web project in a binary file called DataEnvironment.asa. This is referred to as a "DSN-less" connection, because the Web project no longer needs the DSN to establish the connection. From then on, Visual InterDev can simply read the connection information from the binary file as needed.

Machine Data Source Name

In the case of using a machine or system DSN, a connection string is not used. Instead, you will have to recreate the DSN on each development machine as well as the Web server.

ODBC Drivers

At run time, the server must have the appropriate ODBC driver to make the connection to the data source. By default, ODBC drivers are installed on both your development computer and your server when you install Visual InterDev. However, if you deploy your applications to other servers, you must make sure that those servers also have the correct ODBC drivers.

Security with Data Connections

Generally, when you are creating an application, you want to have the widest possible range of privileges so you can manipulate the database and data as needed. However, when users access the data at run time, you want to limit them to the minimum privileges required to run the application.

Your first task is to create user profiles on the database. For example, you might create an administrator-level user profile for yourself to use during development. You can also create a user profile with privileges appropriate for users of your application. If the database server is on a different computer than the Web server, you must also make sure that the correct user profiles have been defined at the operating system level. For more information, see "Security" in Chapter 6, "Web Project Concepts."

Later, when you create a data connection, you can specify both design-time and run-time authorization. Design-time authorization is the user name and password you use when developing the application. Run-time authorization is the user name and password that Visual InterDev will use when connecting to the database while the application is running.

When specifying design-time authorization, you choose how secure your authorization is. For maximum security, you can choose to be prompted for a password each time you connect to the database. If you are not as concerned about security, you can choose not to be prompted. In that case, Visual InterDev encrypts your password and stores it in the project.

When you specify run-time authorization, you do not have this choice: you cannot prompt users for a password, because the prompting would necessarily have to occur on the Web server. Therefore, you must include the password with the user name. The password is encrypted and stored in the project so it can be passed to the database each time a user connects to the database when the application is running.

Database Management

In addition to helping you create Web pages that are linked to databases, Visual InterDev allows you to manage databases while directly connected to them at design time. Depending on the features of your database and your access privileges, you can use Visual InterDev tools to add, remove, or modify:

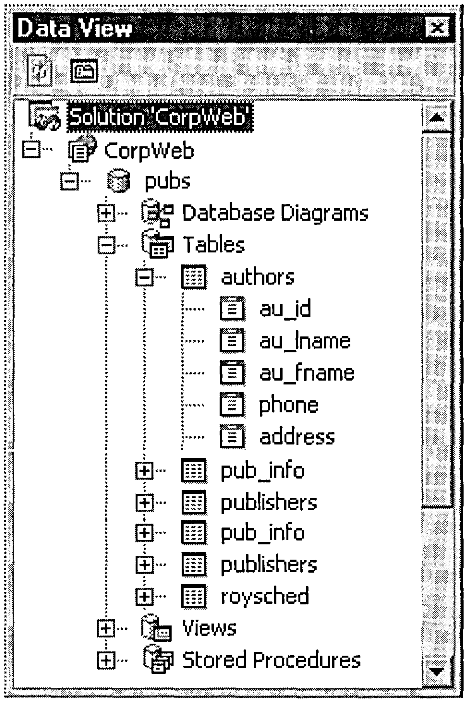
- Databases
- Tables or columns
- Views and synonyms
- Relationships between tables
- Indexes
- Constraints and triggers
- Stored procedures, functions, and packages
- Queries that return sets of data, or queries that modify a database by adding, updating, copying, or deleting records

  Note Features that allow you to make structural edits to databases are available only in the Enterprise Edition of Microsoft Visual Studio.

Managing databases is a separate task from adding database functionality to a Web application. Therefore, to manage databases in Visual InterDev, you create a database project. To help you perform various database management tasks, a database project in Visual InterDev provides the following tools:

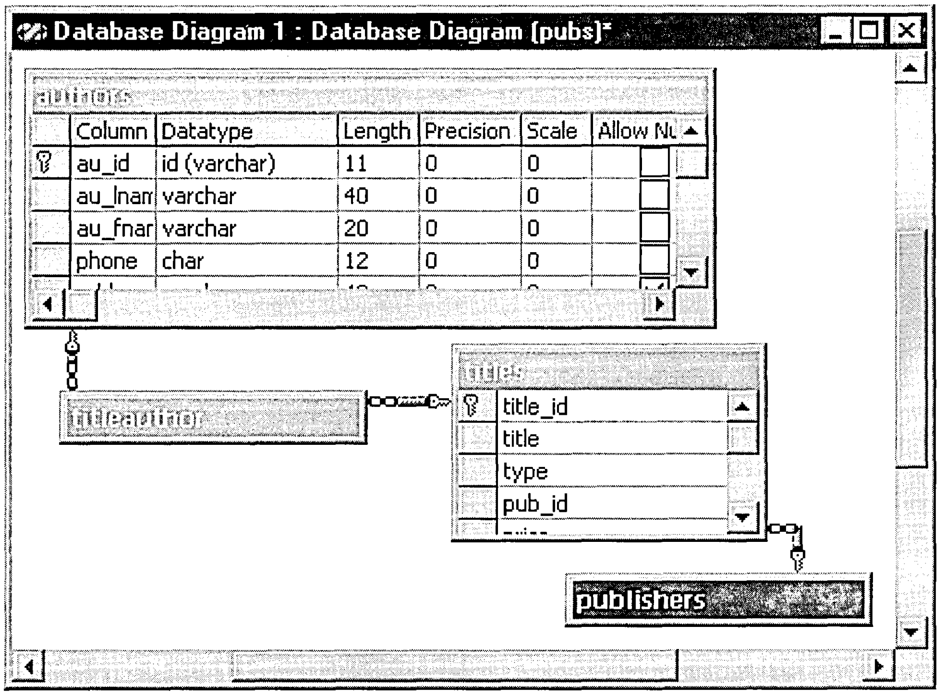
- **Data View window** A window that displays all the database objects that you can currently work with. From the Data View window, you can edit objects such as tables, views, stored procedures, and triggers.

Data View Window Showing Available Database Objects



- **Database Designer** A tool that displays your Microsoft SQL Server or Oracle database as a database diagram, which you can edit to add or modify table and column definitions, define relationships, create indexes, and add constraints.

Database Diagram Representing a Database Visually

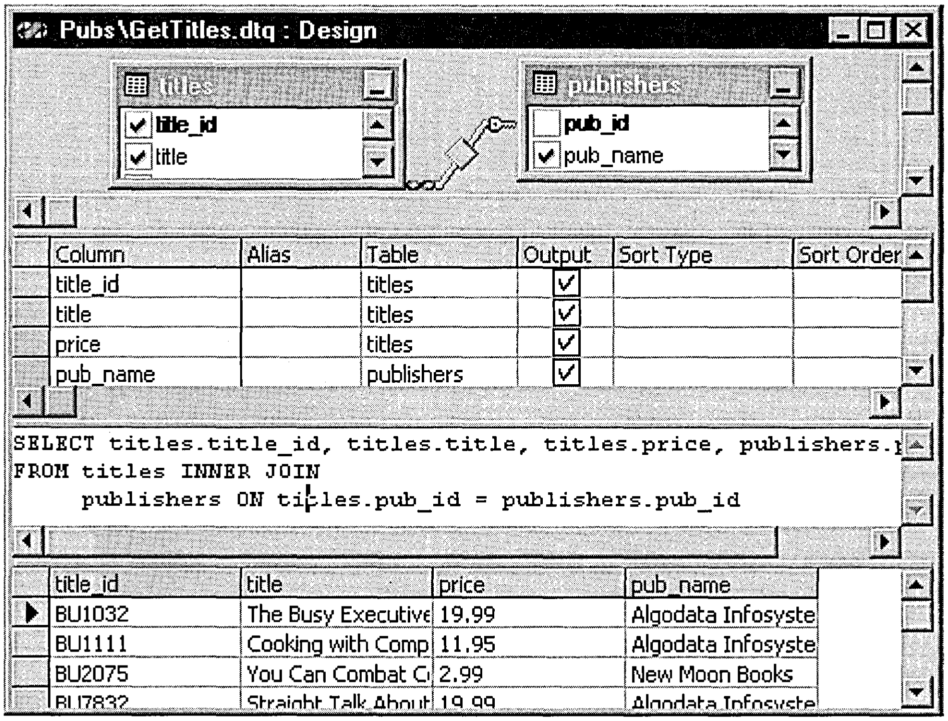


Column	Datatype	Length	Precision	Scale	Allow Nu
au_id	id (varchar)	11	0	0	☐
au_lnan	varchar	40	0	0	☐
au_fnar	varchar	20	0	0	☐
phone	char	12	0	0	☐

titles
title_id
title
type
pub_id

- **Query Designer** A designer that allows you to visually create an SQL statement to query or modify a database.

Query Designer Being Used to Create an SQL Statement



Column	Alias	Table	Output	Sort Type	Sort Order
title_id		titles	✓		
title		titles	✓		
price		titles	✓		
pub_name		publishers	✓		

```
SELECT titles.title_id, titles.title, titles.price, publishers.p
FROM titles INNER JOIN
    publishers ON titles.pub_id = publishers.pub_id
```

title_id	title	price	pub_name
BU1032	The Busy Executive	19.99	Algodata Infosyste
BU1111	Cooking with Comp	11.95	Algodata Infosyste
BU2075	You Can Combat C	2.99	New Moon Books
BU7832	Straight Talk About	19.99	Algodata Infosyste

- **View Designer** A version of the Query Designer that allows you to visually create the SQL Statement that defines a view.
- **Stored Procedure editor** A window for creating stored procedures that includes a link to the Query Designer to construct SQL statements.
- **Trigger editor** A window for creating triggers.
- **Script editor** A window for creating SQL scripts, which are SQL statements that are independent of any particular database. You can put SQL scripts under source control as well.

The Data Environment

The data environment is a repository in your Microsoft Visual InterDev Web project for the information required in server script to connect and manipulate data in databases. It provides a standard interface for creating reusable data-related objects and for placing them on Web pages.

> **Note** The data environment is available on the server. If you are designing a Web application that uses client access to data (using Microsoft Internet Explorer 4.0 DHTML), the data environment is available at design time, but not used at run time.

The data environment also provides an object you can reference in script, allowing you to access and manage database objects such as tables, views, stored procedures, and SQL commands programmatically. It provides an easy-to-use wrapper around ActiveX Data Objects (ADO), making these objects more accessible and easier to work with in Visual InterDev.

To understand the data environment, you must understand these concepts:

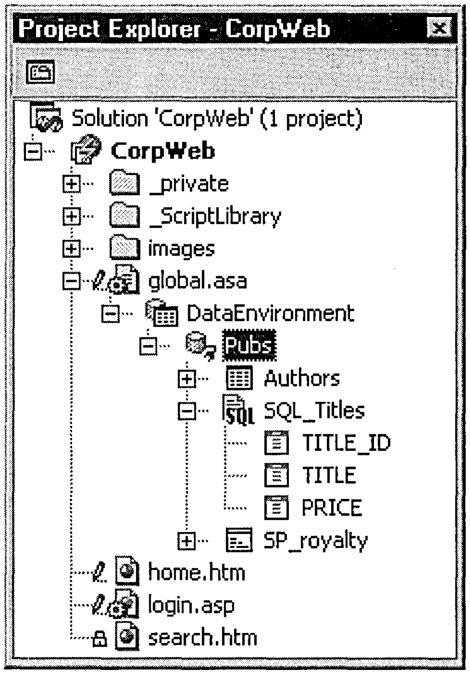
- Data Environment Contents
- Drag and Drop Scenarios in the Data Environment
- The Data Environment Object Model

Data Environment Contents

The primary component of the data environment is a data connection, which includes the information required to connect to one database with a specific user name. For example, your data environment might include a connection that links your application to the Pubs database on a Microsoft SQL Server under the user names Admin (at design time) and Guest (at run time). If your application requires access to multiple databases, you can add multiple data connections to your data environment.

Within each data connection, you can add one or more data commands (command objects), which define a set of data to work with. Command objects can reference a database object such as a table, query, view, synonym, stored procedure, or SQL statement. For example, you might create a command object that references the Authors table so you can display the contents of that table on a Web page. You could also define additional command objects to reference queries and stored procedures you can call to display and update data in other tables.

A Web Project Showing the Data Environment and Data Commands



Command objects are accessible to any page in your application. They therefore become reusable objects. If the underlying database changes, you can make a single change to the command object, and all Web pages that reference the command object will continue to work properly.

Each command object is a node that contains additional information relating to that command object. For example, a command object that references a table contains a list of columns in that table. A command object that references a stored procedure can contain a list of the columns returned by the procedure, or it can contain a list of the procedure's parameters. For information on adding Command objects, see "Getting Records" in Chapter 19, "Viewing Data."

Visual InterDev creates a data environment for your project the first time you define a data connection. As soon as you add the connection, Visual InterDev creates the DataEnvironment folder and adds it as a node under the Global.asa file. The data connection you added is displayed in the DataEnvironment node.

As you create additional data connections, they are added to the DataEnvironment node.

> **Note** You can only have one DataEnvironment node (and one data environment) in a Visual InterDev project. For more information on adding data connections, see "Connecting to a Database" in Chapter 3, "Database Basics."

Drag and Drop Scenarios in the Data Environment

An important feature of the data environment is that you can drag objects to and from it to simplify the process of adding database access to your application. To create new commands, you can drag database objects from the Data View window to the data environment. In addition, you can create data-bound controls on a page by dragging commands and database fields from the data environment to your page.

The following table summarizes how you can use drag and drop with the data environment.

Drag	From	Drop on	To
Database object	Data View	Connection in data environment	Create a command object for the database object you dragged. For example, dragging a table creates a command object whose Database Object type is table.
Command object or Field object	Data environment	Web page	Create a data-bound control.
Database object	Data View	Web page	Not allowed. Drag objects from the data environment instead.

For details about using data-bound controls, see Chapter 19, "Viewing Data."

The Data Environment Object Model

The data environment supports its own object model, which you can use when writing script to manipulate the data you want to display on your Web page. The data environment object model is based on the ActiveX Data Objects (ADO) object model, but is simpler to use.

In ADO, the main objects in the data environment object model are the Connection object, Command object, Recordset object, Field objects, and Parameter objects. Each of these ADO objects has its own properties and methods.

The data environment abstracts this object model to make it simpler to use. The data environment itself is an object that can be used in script and that contains these ADO objects. Within the data environment object, command objects are exposed as methods in script. You can call a command method to execute it and return the recordset referenced by the command or to execute its SQL command or stored procedure.

Each of the data environment objects also has properties you can set in the property page for that object or directly in script. For more information on the data environment object model and the properties for each object, see "Executing Database Commands Using the Data Environment" in Chapter 21, "Accessing Databases Directly."

Data Binding

Because of the interaction between browser, Web server, and database, accessing a database with Web pages is different than working with a database in a traditional application. Moreover, the process is different for server-based and client-based database access.

As explained later, if you use Visual InterDev tools to design database access for your application, you, as an application developer, will not need to worry about the underlying differences. Nonetheless, it is helpful to understand the following concepts:

- Server Access to Databases
- Client Access to Databases
- Data-Bound Controls and the Script Object Model
- Database Access Design in Visual InterDev Applications

Server Access to Databases

In server-based database access, the interaction between database and user shares some features with ordinary client-server database applications. However, the Web server sits between the two and introduces a layer of interaction with its own features.

The following characteristics dictate how the browser, Web server, and database server interact:

- The client (the browser, which presents data entry forms and reports) has no direct connection to the database. The client can only transmit requests to the Web server, which then passes them to and from the database.
- The database fulfills requests by sending recordsets to the Web server. The database server itself does not maintain recordsets, and by extension, does not track the current record in the recordset.
- Interaction with the database is handled by server scripts, not by client scripts. Server scripts are processed before a page is sent to the browser, so that when a user interacts with the page (by clicking a button, for example), all database access for that page has already been accomplished.
- The browser and the server are not in continual contact. The server does not maintain information about previous browser requests. For example, a server does not keep information about what record in a database the browser last requested.

As a result, interactions between users and databases in Web applications are handled differently than in traditional applications. For example, a common scenario is that the user sees a form containing database information and wants to page back and forth between records.

A typical sequence of events required to accomplish this task is something like this:

1. The user requests a page that contains the form.
2. When the server processes the page, it executes script that connects to the database, and then executes a query or stored procedure that returns one or more records.
3. The server script moves to one particular record out of the recordset.
4. The server script extracts data from the current record and writes the data into the Web page as HTML text. Information about the location of the current record, such as a bookmark or primary key, is stored in a global server variable or packaged up to be sent to the browser along with the page. Typically, the server then discards the recordset.
5. The server sends the page to the browser, where the user sees the database record in an HTML form.
6. The user clicks a "Next" button on the page. The button's script submits the HTML form to the server. Because the server has no connection to the browser, the button script must pass an indication to the server that the user is navigating to the next record. If the location of the current record was sent to the browser earlier, the browser sends that information back to the server.
7. At the server, server script parses the user's submission. In this case, the server detects that the user wants to navigate. The server then reissues the earlier query, which returns a new recordset. The server uses the bookmark or primary key saved earlier to navigate to the correct place in the recordset, and then moves one record forward and repeats steps 4 and 5.

The scenario seems complicated because in effect the Web server forgets about both the recordset and the user's state as soon as it sends the page to the browser. As an analogy, the process is like having a telephone conversation with a friend in which you hang up after each sentence. Each time you dial and reconnect, your friend has forgotten everything you've said.

It is possible to cache recordsets so that the query does not have to be repeated each time. However, when browsing large amounts of data, caching is not recommended. If you do cache, even a small number of users could easily overwhelm the server's resources. However, if the recordset contains only a single row and you are going to update it using an optimistic lock, it can be efficient to cache the recordset on the server.

With server access to the database, the client environment has no direct control over database access. When the server extracts information from the recordset and writes it into the page, the information becomes indistinguishable to client scripts from ordinary HTML text. Client scripts cannot directly execute a command to move in the database. Instead, the client scripts must send sufficient information to a server script so that the server script can pick up where it left off in the data.

> **Note** You can use design-time controls and the Visual InterDev scripting object model to create applications to make the process of handling user requests on the server transparent. For details, see Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Client Access to Databases

If your deployment environment makes it practical, you can create direct database access from the client to the database. You can then manage database access entirely from client scripts, which often results in faster database access. In addition, the development environment allows you to create a richer user experience by taking advantage of browser features.

To access a database from the client, you use features of Dynamic HTML specific to Microsoft Internet Explorer 4.0. All your users must have Internet Explorer 4.0 as their browser. In addition, you must keep your database on a server that supports the correct data access software, or at least use a properly configured server as a gateway to your database.

When you use client access to databases, the underlying interaction between application and database is simpler than that in server access. (If you use Visual InterDev tools to manage database access, the differences are invisible and the way you script for database access is the same for both types.)

To accomplish the navigation scenario described earlier under Server Access to Databases, the application follows these steps:

1. The user requests a page that contains the form.
2. The Web server passes the page to the browser. (If necessary, the server processes scripts on the page first, but server script is not required for database access.)
3. When the browser receives the page, an RDS (Remote Data Service) control on the page automatically opens the recordset and binds the data to the controls via the DHTML 4.0 data binding specifications.
4. The user clicks a "Next" button on the page. The button's script navigates to the next record in the recordset.

An important difference is that after the Web server has sent the page to the browser, the application does not require further requests to the server to manage database access, reducing substantially the number of browser-Web server round trips. In addition, because the client can cache the recordset, database actions such as navigation do not require that the recordset be regenerated. Finally, updates can be sent in batches for greater efficiency.

In all cases, the result is faster database access. You can create client-based database access using Microsoft Remote Data Service (RDS). For information about RDS, see the Microsoft RDS Web site http://www.microsoft.com/data/rds/.

Data-Bound Controls and the Script Object Model

Although you are not required to use Visual InterDev data-bound controls for database access in your Web application, they greatly simplify application development. Data-bound controls provide a complete set of user interface elements, and in addition include all the logic necessary to connect to, navigate in, and update recordsets.

Data-bound controls include:

- **Recordset.** Acts as the master control by binding to a database object. Other controls on the page in turn are bound to a recordset control.
- **Individual controls.** Display the contents of a single field in a database record, including text boxes, list boxes, check boxes, option buttons, and labels.
- **Grid.** Displays a set of records.
- **RecordsetNavbar.** Includes buttons that allow navigation to the next, previous, first, and last record in a result set.

In addition to making it easy to add user interface elements for displaying data, the data-bound design-time controls can take advantage of the Visual InterDev scripting object model. This object model accomplishes two tasks. First, it provides a consistent interface for scripting database access, regardless of what type of database you are working with or whether you are scripting server or client access to databases. Second, it creates a high-level model for database access and manipulation, hiding from you most of the complexity involved in Web-based database access.

When working with data-bound controls, you work primarily with the Recordset control, which uses the scripting object model to expose properties and methods that help you manage the records in a result set. For example, to navigate between records in a result set, you can call a moveNext or movePrevious method of the Recordset object. The individual data-bound controls likewise expose properties and methods that bind them to the Recordset control and that determine the controls' appearance and behavior.

For more information about working with controls and the scripting object model, see Chapter 19, "Viewing Data," and "The Scripting Object Model" in Chapter 23, "Scripting Concepts."

Database Access Design in Visual InterDev Applications

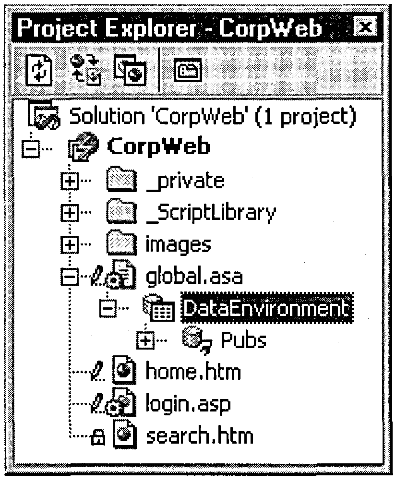
Regardless of what database you want to access, Visual InterDev includes tools that greatly simplify the process. This section provides an overview of how you create a Web application with a database component, highlighting where you can take advantage along the way of Visual InterDev features.

Although the underlying process is different for server-based and client-based database access, the Visual InterDev database tools make the difference transparent. With only a few differences, you will be able to create both types of access using the same procedures.

In outline form, the steps required to set up a Web page with database access are as follows:

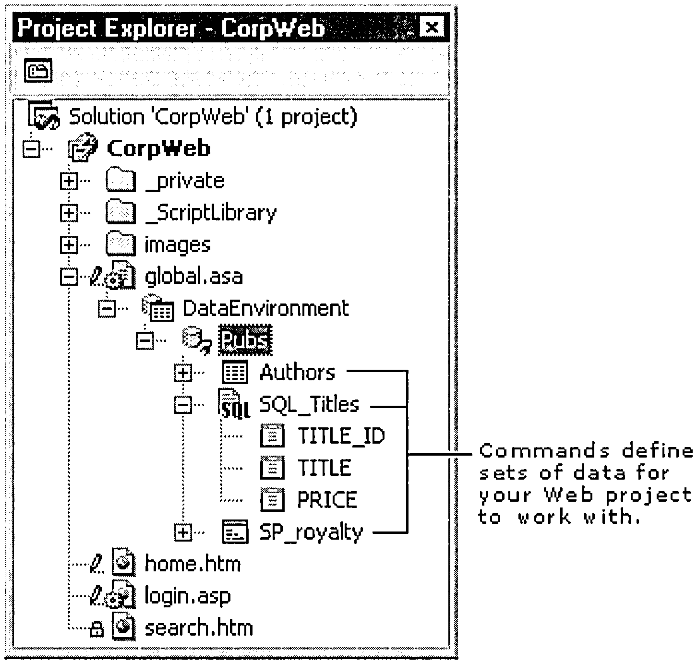
1. **Establish a connection to a database.** You specify a database to connect to by selecting Add Data Connection from the Project menu.

   When you do, Visual InterDev creates a data environment, which acts as a repository for all data connection information in your project. Because different Web pages can share a data connection, scripts in the Global.asa file of your Web project maintain the data environment.

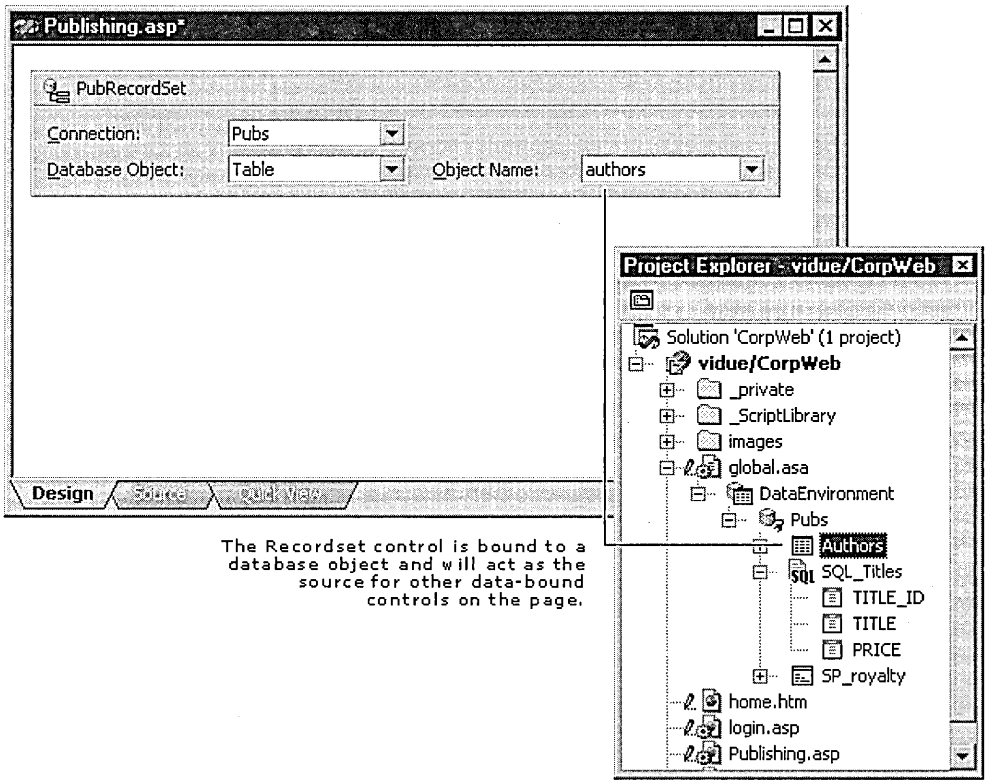
   

   If your application requires access to more than one database (including databases on different servers), you can establish multiple connections in the data environment.

2. **Define the data to use.** For each connection in the data environment, you create one or more *commands*, which are objects that encapsulate references to SQL statements, stored procedures, tables, views, or synonyms. Each command produces a different result set (which is also called a *cursor*). For example, if your application reads data from a table, a query, and a stored procedure, you can define three different commands.

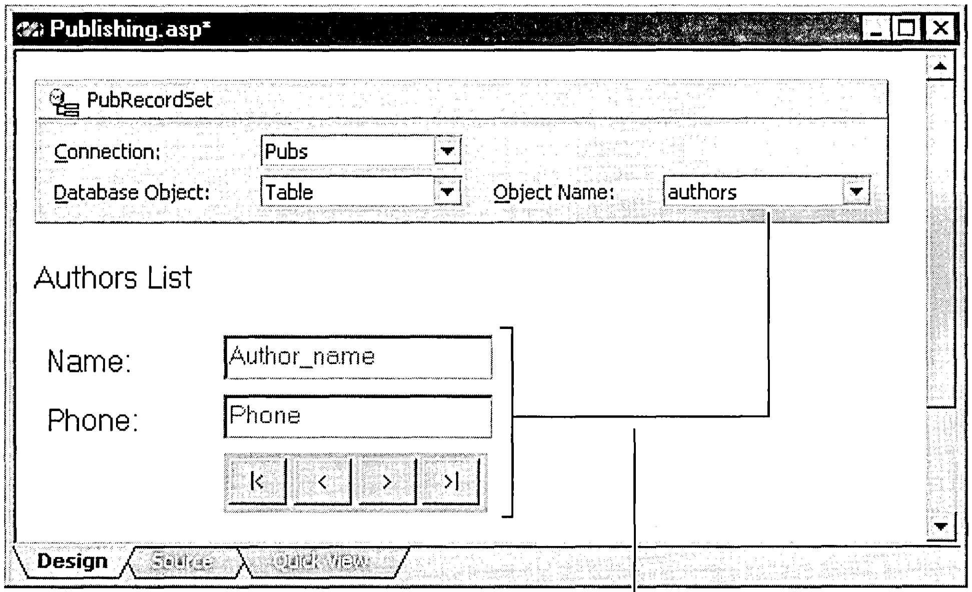


3. **Add a Recordset control to your Web page.** On the Web page where you want to use database data, you add a Recordset control. This control acts as the local controller when the page is running, providing an object for other data-bound controls to interact with. It is bound to database objects either directly or by using command objects in the data environment.



The Recordset control is bound to a database object and will act as the source for other data-bound controls on the page.

4. **Add data-bound controls to your Web page.** On the Web page where you have put the Recordset control, you add data-bound controls such as text boxes, check boxes, and buttons, as well as a navigation bar. You use these data-bound controls to define data entry forms, reports, or other user interface elements for the data in the database. Each data-bound control is linked to a field in a database via the Recordset control.



Connection: Pubs
Database Object: Table
Object Name: authors

Name: Author_name
Phone: Phone

Individual controls are bound to the database through the Recordset control.

5. **Add your own script.** If the data-bound controls do not provide you with the full database interaction that you want in your application, you can write additional script. For example, you can write script that validates the user's entry into a data-bound text control. You can also write script to perform functions such as updating, adding, or deleting records by calling methods in the Recordset object. Finally, you can write script that executes database commands directly by referencing the commands stored in the data environment.

The scenario listed in the steps above is simple, but even complex scenarios do not require substantially greater effort. For example, a Web page for a catalog application might include a drop-down combo box that customers can select a product from. The drop-down list would be produced by a second recordset (in addition to the recordset being updated by the catalog page itself). The data access tools in Visual InterDev allow you to easily maintain both of these recordsets and likewise make it easy to bind specific controls to each set.

If you want, you can also bypass the data-bound controls and create your own database access in script. In that case, you would still use the data environment to establish connections and commands. The data environment presents an object model that you can interact with using script in order to query the database and modify it. For details, see "Executing Database Commands Using the Data Environment" in Chapter 21, "Accessing Databases Directly."

CHAPTER 19

Viewing Data

In Microsoft Visual InterDev, you can display data on your Web pages using data-bound design-time controls. The new data environment makes this process easy by giving you the ability to create and manage all your data-bound controls from one location.

First, you make a data connection to an existing database. The data connection appears in the data environment (DataEnvironment) folder in your Web project. You can then easily add controls bound to this connection to an ASP or HTML page by creating Command objects in the data environment and dragging them to the page. Visual InterDev creates data-bound controls that display the data from the database.

You can also drag data-bound controls from the Toolbox to your ASP page to display data. For example, you can create a data connection and add a Recordset design-time control based on this connection to an ASP or HTML page. A number of other controls allow you to display data from the recordset defined by a Recordset control in different ways: text boxes, labels, list boxes, check boxes, option groups, and so on.

For even more flexibility in presenting your data, you can take advantage of the Grid design-time control. You can use this control to display data from multiple records in a grid format.

Getting Records

Getting and displaying records from a database is one of the key features of Microsoft Visual InterDev. You manipulate data using data-bound design-time controls, primarily the Recordset control.

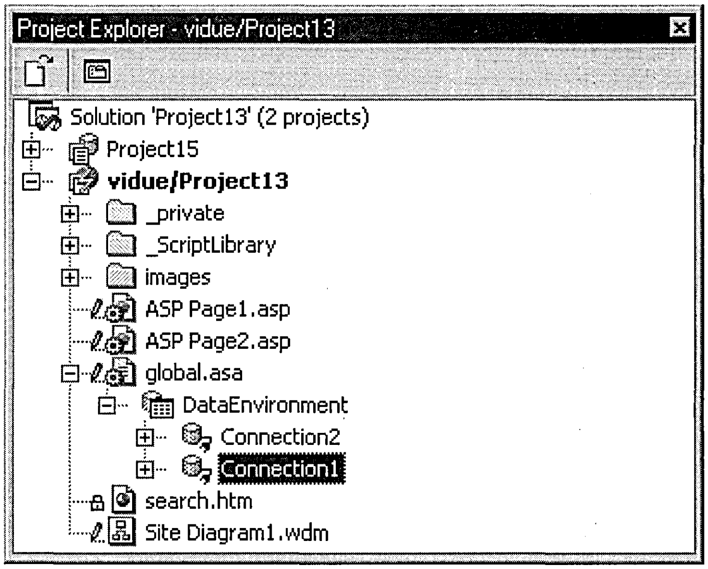
A Recordset control is bound to a particular database using a database connection, and specifies a particular set of records from that database.

Visual InterDev makes it easy to create Recordset controls by using the data environment. You can also create Recordset controls by using the Toolbox, and then associate these controls with the data environment. For information on how the data environment is related to controls and other database features of Visual InterDev, see Chapter 18, "Database Concepts."

To create a Recordset control using the data environment

1. In your Visual InterDev project, add a data connection to the database whose records you want to display. For details, see "Connecting to a Database" in Chapter 3, "Database Basics."

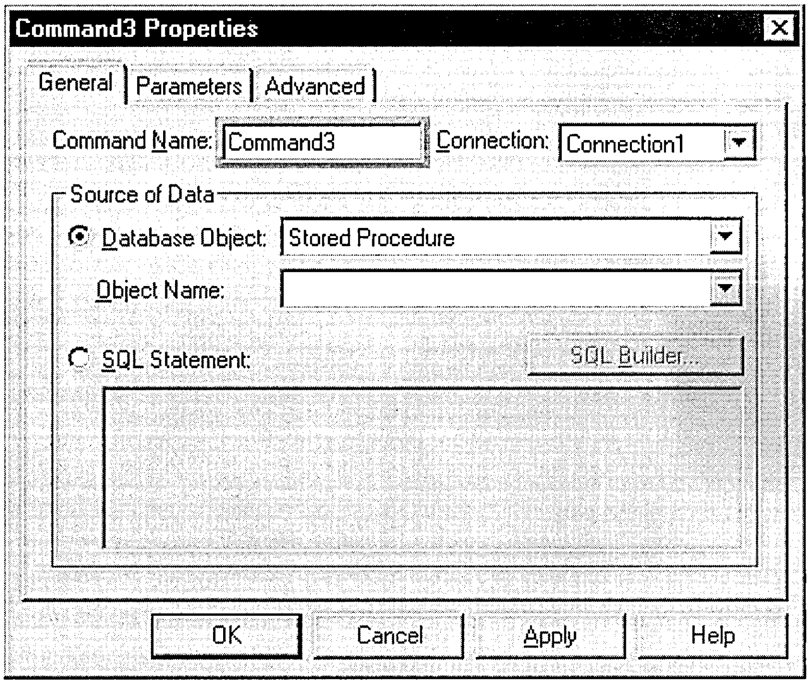
   The data connection is displayed in **Project Explorer** in the **DataEnvironment** folder, underneath the Global.asa file.



Tip You can also right-click the **DataEnvironment** folder and then click **Add Data Connection** to create a data connection. The same set of dialog boxes is displayed that appears when you use the **Add Data Connection** command on the **Project** menu.

2. Right-click the **DataEnvironment** folder and select **Add Data Command**.

   The **Command Properties** page is displayed.



Command3 Properties

General | Parameters | Advanced

Command Name: Command3 Connection: Connection1

Source of Data

(•) Database Object: Stored Procedure

Object Name:

( ) SQL Statement:

3. Enter a name for this Command object in the **Command Name** box, such as CustomersTableRecords.
4. Select the data connection in the **Connection** box. If you selected a data connection in the DataEnvironment folder before you clicked **Add Data Command**, this data connection will already be in the **Connection** box.
5. If you want the Command object to contain the set of records in a table, query, view, stored procedure, or other type of database object, select the button next to the **Database Object** box.

   Then select this type of database object in the **Database Object** box and the name of an object of this type in the database in the **Object Name** box.

6. If you want to use an explicit SQL statement to select the recordset you want, select the button next to the **SQL Statement** box. Then enter the SQL statement in the **SQL Statement** box.

   To construct this SQL statement, you can press the **SQL Builder** button, and use the SQL Builder.

   At any time, you can drag the Command object onto an ASP or HTML page. This creates a Recordset object, which is bound to the specified set of records in the database.

Using the data environment is the preferred method to create Recordset controls. Once the Command object is created using the data environment, modifying the Command object is simple. You only need to update the object once in the data environment. You don't have to continually recreate a Recordset control on different ASP or HTML pages for the same set of records.

You can, however, insert a Recordset control onto an ASP or HTML page from the Toolbox, and then associate this Recordset control with an existing Command object in the data environment.

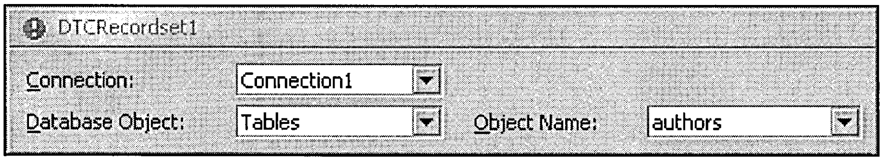
To associate a Recordset control with a Command object

1. Open an ASP or HTML page in the editor.
2. Drag the **Recordset control** from the **Toolbox** onto the page.

   Tip If the Recordset control is not shown in the **Toolbox**, right-click on the **Toolbox**, choose **Customize Toolbox**, and add the Recordset control.

   The Recordset control displays the **Connection**, **Database Object**, and **Object Name** fields. These are properties of the Recordset object.

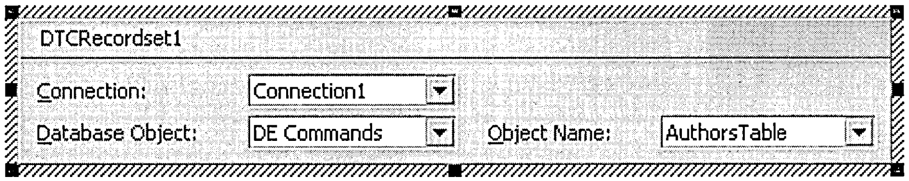
```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0"
<META HTTP-EQUIV="Content-Type" content="text/html">
</HEAD>
<BODY>

  <P><EM>Insert Content Here</EM></P>
```


DTCRecordset1
Connection: Connection1
Database Object: Tables
Object Name: authors

```
</BODY>
<!--METADATA TYPE="EditorGenerated" startspan <COMMENT>
```

3. In the Recordset control on the ASP or HTML page, set the **Connection** property to the name of the data connection for the database whose records you want to see. This is the database that the Command object is connected to.
4. Set the **Database Object** property to **DE Commands**.
5. Set the **Object Name** property to the name of the Command object in the data environment.

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0"
<META HTTP-EQUIV="Content-Type" content="text/html">
</HEAD>
<BODY>

  <P><EM>Insert Content Here</EM></P>
```



DTCRecordset1

Connection: Connection1

Database Object: DE Commands

Object Name: AuthorsTable

```
</BODY>
<!--METADATA TYPE="EditorGenerated" startspan <COMMENT>
```

Now that you've created a recordset and specified its records, you can display data from this recordset on a Web page.

Displaying Data on Your Web Page

Displaying data is easy using Microsoft Visual InterDev. You can connect to a database, specify the set of records to display, and display the records by using data-bound design-time controls, or you can display data directly using script. You can also display data in a grid, showing multiple records from a recordset.

To display data using a data-bound control

1. Create a data environment Command object that specifies a set of records from a database. For details, see "Getting Records," earlier in this chapter.
2. Open an ASP or HTML page in the editor.

3. Drag the Command object from the **DataEnvironment** folder to the page. This creates a Recordset control on the page.
4. Drag one or more fields from the Command object to the page.

   The fields are displayed under the Command object in the **DataEnvironment** folder.

   Each dragged field creates a data-bound control that will display the data from that field in the recordset. Text and numeric fields create Textbox controls. Yes/No or True/False (Boolean) fields create Checkbox controls. For more information, see "The Data Environment" in Chapter 18, "Database Concepts."

You can easily provide navigation (moving from record to record) among the records you display on your Web page with the RecordsetNavbar control.

To provide navigation among records

1. Open the ASP or HTML page in the editor.
2. Drag the RecordsetNavbar control from the **Toolbox** onto the page.

   > **Tip** If the RecordsetNavbar control is not shown in the **Toolbox**, right-click on the **Toolbox**, choose **Customize Toolbox**, and add the RecordsetNavbar control.

   The control is inserted onto your ASP or HTML page.
3. Right-click the control and click **Properties** to open its property pages.
4. Set the **Recordset** property of the control to the name of a Recordset control whose records are displayed on the page.

By default, the RecordsetNavbar control provides Move to First, Next, Previous, and Move to Last buttons. You can use these buttons to move among the records displayed on the page. You can also customize the functionality of these buttons by modifying the script that they create.

Navigating Using Script

You can move between records in the recordset by calling navigation methods in the recordset script object. When you move, the record you move to becomes the current record.

> **Important** When you navigate to another record, the current record is not automatically saved. Be sure that you have copied data to the current record and saved the record before you call a navigation method. For details, see "Updating Records" in Chapter 20, "Modifying Data."

To navigate using script

- Call one of these methods of the recordset script object: moveNext, movePrevious, moveFirst, moveLast, moveAbsolute, or move.

  For example, the following shows the handler you might write for a "Next" button.

  ```
  Sub btnNext_onclick
     rsEmployeeList.moveNext
  End Sub
  ```

If you want to perform a procedure based on navigation, you can write handlers for the onrowenter and onrowexit events. For example, perhaps your application does not simply display only the contents of fields. Instead, you want to use a single textbox to display a combination of first and last names.

To do this, you can write a handler for the onrowenter handler that gets values from the recordset, concatenates them, and puts them in a textbox. The handler might look like this:

```
Function rsEmployeeList_onrowenter()
   firstName = rsEmployeeList.fields.getValue("lastName")
   lastName = rsEmployeeList.fields.getValue("firstName")
   txtFullName.value = lastName & ", " & firstName
End function
```

You can also create a data grid on your Web page using the Grid design-time control.

CHAPTER 20

Modifying Data

In addition to simply displaying data from a database, you can create Web pages that allow users to modify data — update existing records, add records, and delete records. You build data modification into your application by using the Recordset control, which creates a recordset — a virtual table that users can navigate in and update. Modifications are made to the recordset and then passed to the underlying database.

Alternatively, you can display records on your page using a recordset, and then call stored procedures to perform actions such as updating, inserting, and deleting. The method for performing the updates is the same whether you use a recordset or a stored procedure. For details about binding a Recordset control to stored procedures, see "Executing Database Commands Using the Data Environment" in Chapter 21, "Accessing Databases Directly."

No matter what method you use, you must first be sure that it will be possible to update the database at all. Factors to consider are:

- **Permissions** Many databases require explicit permission for updates. Even if you as the developer have permission to make updates, you must be sure that your Web application's users will also.
- **Sufficient data** As a rule, the recordset must include sufficient information for the database to find the correct record to modify — usually the primary key or a unique index value. In addition, only fields that are part of the recordset can be updated. For example, a recordset might include an employee's ID, first name, and last name, even though the employee table has many more fields. In that case, you would only be able to update those three fields or provide only those three fields when adding a new record.
- **Updatable data source** If you are updating via the recordset, it must have a Keyset or Dynamic cursor type and a lock type that is not read-only. In addition, the recordset must be based on a database object that you can update. For example, if the recordset is based on a view that contains no primary key or unique index, it cannot be updated, regardless of what cursor type you have selected. The exact constraints depend on how you generated a result set and on the features of the database you are using.

  Note If you are using stored procedures to update the database, the cursor type of your recordset is less important, because updates are not passed through the recordset.

Getting Values from the Current Record

By default, when the page runs, the recordset script object opens the table or executes the query specified in its data binding properties, and then creates a recordset. A pointer is placed at the first record in the recordset, which becomes the current record.

To display data, you need to get the values from the current record. If you are working with design-time controls and have established data binding for them, the controls are automatically updated to reflect database values.

However, if you are working with non-data-bound controls, you must copy values from the current record to the control. Values from the current record are available in the recordset's fields collection.

To get values from the current record

- Call the getValue method for the fields collection and specify which field you want. For example, the following statement extracts the value of the field "Name" from the fields collection of a recordset object called rsEmployeeList:

  ```
  name = rsEmployeeList.fields.getValue("Name")
  ```

  Tip A good time to get the values from the current record is in a handler for the recordset's onrowenter event.

After getting a value, you typically display it in a control such as a textbox or label control. For example, the following script shows how you would display values from the current record in a set of textbox script objects called txtName, txtAddress, and so on.

```
txtName.value = rsEmployeeList.fields.getValue("Name")
txtAddress.value = rsEmployeeList.fields.getValue("Address")
txtCity.value = rsEmployeeList.fields.getValue("City")
areaCode = rsEmployeeList.fields.getValue("AreaCode")
phone = rsEmployeeList.fields.getValue("Phone")
txtPhone.value = areaCode & " " & phone
```

Note You can only copy information from the recordset to a control if the control and recordset use the same target scripting platform. For more details, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts," and "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

If you are displaying data in editable controls — such as textboxes — users can change the data. To save their changes and write them to the database, see the next section, "Updating Records."

Updating Records

The Recordset control includes features that make it easy for you to allow users to update the current record in a recordset. For example, if you use a RecordsetNavbar control on your page, you can set an option that automatically updates the current record with user changes when a user navigates to another record.

Not all recordsets allow changes to be written to the database. The Recordset object's cursor type must be set to Keyset or Dynamic. You must also have permission to update the database. Finally, queries can produce recordsets that do not contain sufficient information to allow updates. For details, see the beginning of this chapter.

To enable automatic navigation updates

- Set the RecordsetNavbar control's updateOnMove property to True.

  Note Setting this option to True causes the recordset to be updated each time the user navigates, even if no changes have been made. If you anticipate that users will seldom make changes, you might want to use a strategy with less overhead.

If you are working with data-bound design-time controls, they automatically copy their values to the current record before the update is made. However, if you are working with a control that is not a data-bound design-time control, you must manually update the current record before the update is made.

To set values in the current record

- Call the setValue method for the fields collection and specify which field you want.You specify the field to update and the value.

  Tip A good time to get the values from the current record is in a handler for the recordset's onbeforeupdate event.

  For example, the following statement sets the Name field of the current record to the value of a textbox called Name:

  ```
  rsEmployeeList.fields.setValue("Name", txtName.value)
  ```

  Note You can only copy from a control to the recordset if the control and recordset use the same target scripting platform. For more details, see “The Scripting Object Model” in Chapter 23, “Scripting Concepts,” and “Creating Forms with Design-Time Controls” in Chapter 24, “Scripting with Design-Time Controls and Script Objects.”

You can also update records in script. The basic procedure is to call a method that writes the current record to the database.

To update records in scripts

1. Be sure there is a Recordset control on your page. For details, see "Getting Records" in Chapter 19, "Viewing Data." Note the Recordset control's name.
2. If you have controls on the page that are not data-bound design-time controls, copy their values into the current record as described in the preceding procedure.
3. After you have finished setting all the values you need in the current record, call the updateRecord method.

The following example shows a handler for a Save button's onclick event, which saves the current record to the database.

```
Sub btnSave_onclick
   rsEmployeeList.updateRecord
End sub
```

The following example shows the same procedure, but copies values from non-data-bound controls to the current record before saving it.

```
Sub btnSave_onclick
   ' Copying data to the current record is required only for
   ' controls that are not data-bound design-time controls
   rsEmployeeList.fields.setValue("Name", txtName.value)
   rsEmployeeList.fields.setValue("Insured", chkInsured.value)
   rsEmployeeList.fields.setValue("LastUpdate", date)
   rsEmployeeList.updateRecord
End sub
```

To help you trap errors when updating a recordset, you can write handlers for the recordset object's onbeforeupdate and onafterupdate events. Typically you write a handler for the onbeforeupdate event to make sure that data in the current record is correct. You can use the onafterupdate event to determine whether the update was successful.

For more information about writing event handlers for design-time controls, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Adding Records

When you work with a recordset object in script, you can add records to the database in two ways. The first is a two-step process. First you initialize a new, blank record. The user can then fill in the record. When the record is complete, the user can click a Save button (or similar) to write the current record to the database, just as if an existing record had been updated. This strategy is useful when you are working with a form where users can edit existing records or add new ones.

Alternatively, you can create a new record and populate it in a single operation. Adding immediately this way can be more efficient if your target scripting platform is Server, because it avoids a round trip to the server to initialize the new record. This strategy is particularly useful when you are not using a form, but instead creating new records in script — especially if you have to add several records to the database at once.

You might not always be able to create new records. You need the correct settings for your recordset, adequate permissions in the database, and sufficient information to uniquely identify records. For details, see the beginning of this chapter.

Initializing and Then Updating

You typically use the initialize and update strategy when working with a form. The form will usually have a New button and a Save button. By using the initialize and update strategy, you can use the same Save button to update existing records that have been edited as well as to insert new records.

To initialize a new record in the recordset, you use the Recordset object's addRecord method. If you are working with data-bound design-time controls, the recordset automatically clears the controls when you initialize a new record. Then when you save the record, the values the user has entered are automatically copied to the current record before it is written to the database.

If you are working with controls that are not data-bound design-time controls, you must perform some of this work yourself. After initializing a new record, you must manually clear the controls so users can enter new data. Then when the user clicks a Save button or similar, you must be sure to copy values from the controls to the current record before it is written to the database.

To initialize and add a record

1. Be sure there is a Recordset control on your page. For details, see "Getting Records" in Chapter 19, "Viewing Data." Note the Recordset control's name.
2. In script, call the recordset script object's addRecord method to prepare a new, blank record in the recordset. For example, the following event handler for a New button calls the addRecord method.

   ```
   Sub btnNew_onclick
      rsEmployeeList.addRecord
   End sub
   ```

   If your page contains non-data-bound controls, clear them after initializing the new record, as in this example:

   ```
   Sub btnNew_onclick
      rsEmployeeList.addRecord
      txtName.value = ""
      chkInsured.value = false
   End sub
   ```

Note You can only work with controls and recordsets at the same time if both use the same target scripting platform. For more details, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts," and "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

When the user clicks the New button, the user can enter information for the new record. The Recordset control on the page sets a flag that this is a new record. When the user clicks the Save button, the Recordset control actually creates the new record, updates it with the user's data, and writes it to the database.

After the new record has been filled in, it is saved the same way an existing record is (with the recordset object's updateRecord method). For details, see "Updating Records," earlier in this chapter.

Adding Immediately

You typically add records to a database immediately if you are creating them in script. Adding a record immediately to the database is similar to first initializing a record and later saving it. However, because the add and update functions are performed in a single step, only one trip is required to the server.

When you create a record immediately, you must gather the information for the fields and then pass them as parameters when you add the new record.

To add a record immediately

1. Gather the information to be written to the database. You can do this using controls on the page, values derived from calculations, or any other method.
2. Create two arrays, one containing the names of the fields and the other the values. For example, the following script creates an array of fields that track a transaction (transID, transDate, and transTime) and values for each field:

```
Dim fieldsArray(2)
Dim valuesArray(2)
fieldsArray(0) = "transID"
fieldsArray(1) = "transDate"
fieldsArray(2) = "transTime"
valuesArray(0) = getTransactionCounter()
valuesArray(1) = Date
valuesArray(3) = Time
```

3. Call the addImmediate method of the Recordset object to add the record to, passing it the two arrays you just created, as in the following example:

```
rsTransactionLog.addImmediate( fieldsArray, valuesArray )
```

For more information about writing event handlers for design-time controls, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Deleting Records

To delete a record in a database, you can add a control such as a Delete button to the page. The button calls the recordset's deleterecord method to delete the current record.

You might not always be able to delete records. You need the correct settings for your recordset, adequate permissions in the database, and sufficient information to uniquely identify records. For details, see the beginning of this chapter.

To delete records in script

1. Be sure there is a Recordset control on your page. For details, see "Getting Records" in Chapter 19, "Viewing Data."
2. In script, call the recordset script object's deleterecord method, as in this example handler for a button called btnDelete:

```
Sub btnDelete_onclick
   rsEmployeeList.deleteRecord
End sub
```

By default, the script illustrated above causes the page to display the record immediately following the one just deleted. If you prefer that the page display a different record, you can insert a navigation method in front of the return statement. The following example shows how you would move to the preceding record:

```
rsEmployeeList.deleteRecord;
rsEmployeeList.movePrevious;
```

For more information about writing event handlers for design-time controls, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

CHAPTER 21

Accessing Databases Directly

Many database applications do not rely exclusively on controls bound to tables or views in the database. Instead, they perform their data access via stored procedures or queries, which provides several advantages.

Using stored procedures can limit the changes made by users and can enforce validation and other requirements. If you work with queries, you can dynamically change the query as required by your application.

In this section, you can learn about how to create Microsoft Visual InterDev applications that work directly with stored procedures and queries.

Debugging Stored Procedures and Triggers

If you are working with the Enterprise edition of Microsoft Visual Studio, Microsoft Visual InterDev includes a SQL debugger that you can use to debug Microsoft SQL Server stored procedures and triggers in much the same way that you debug other kinds of scripts or programs. However, there are some differences in how you set up SQL debugging and how the debugger runs.

Setting Up SQL Debugging

SQL debugging requires the following:

- You must have the Enterprise edition of Visual Studio.
- You must be running SQL Server 6.5 with Service Pack 2.
- SQL Server must be running Microsoft Windows NT 4.0 or later.
- Your workstation must be running Windows 95 or Windows NT 4.0 or later.

To use SQL debugging, you must properly configure your server and workstation. You should:

- Make sure that you have installed SQL Debugging components on your SQL server.
- Establish a Windows NT user who has administration privileges on the server computer where SQL Server is running.
- Configure Distributed COM (DCOM) on the server for SQL debugging.
- Make sure that the DCOM configuration on the client supports SQL debugging (Windows 95 workstations only).

Installing SQL Server Debugging Components

SQL debugging requires components that you install on your SQL Server. These components are part of Visual Studio, Enterprise Edition.

To install SQL debugging components

1. On the computer where SQL Server is installed, start the Visual Studio Enterprise Edition setup program.
2. The installation wizard presents slightly different choices, depending on whether you have previously installed server components on the computer.
   - If the server is already installed, under **Add/Remove Options**, choose **Server Applications and Tools**.
   - If no other components have been installed, proceed through the setup wizard until you get to the page offering you **Enterprise Setup Options**. Choose **Server Applications**.
3. Proceed through the setup wizard until you get to the page offering you **Enterprise Setup Options**. Choose **Server Applications**.
4. On the next page, select **Launch BackOffice Installation Wizard**, and then choose **Install**.
5. When the BackOffice Business Solutions wizard is displayed, choose **Custom** and then choose **Next**.
6. Proceed until you see the page offering you a list of components to install. Uncheck all components except the following:
   - **SQL Server Debugging**
   - **MS Data Access Components**
   - **Visual InterDev Server**
7. Proceed with the installation.

Setting up a Debug User

To use SQL debugging, you must be able to provide the name and password of a Windows NT user who has administration privileges on the server computer where SQL Server is running.

To set up a user for SQL debugging

1. In the Windows Control Panel on the server, choose **Settings**, and then choose **Services.**
2. Select **MSSQLServer**, and then choose **Startup**.
3. Check the **Log On As** settings. If the option is set to **System Account**, change it to **This Account**, enter the valid domain and user account (in the form *domain\account*) of a user with administration privileges, and then enter the password.
4. If you have changed the setting, restart SQL Server.

Setting up DCOM for SQL Debugging

SQL debugging uses Distributed COM (DCOM) to communicate between your client computer and the server. You must therefore configure DCOM to allow a remote user to attach the debugger to a process there.

By default, the correct DCOM settings are in place when SQL Server is installed on the server. However, because of security considerations for the computer running SQL Server, you might want to restrict access to debugging. Use the following as a general procedure for setting up DCOM for the SQL server computer.

To configure DCOM on the server for SQL debugging

1. From the Windows Start menu on the server, choose **Run**, and then in the **Open** box type **Dcomcnfg.exe** at the prompt.
2. In the **Distributed COM Configuration Properties** window, choose the **Default Security** tab. Under **Default Access Permissions**, choose **Edit Default.**
3. If the group **Everyone** does not already have permissions, choose **Add**, and then add the domain and user account (in the form *domain\account*) of a user with administration privileges.
4. After adding the account, check for **SYSTEM**. If it is not already in the list, add it by choosing it from the list of users in the **Add Names and Groups** dialog box.
5. If you have changed any of the settings described in this procedure, restart SQL Server.

   Note If you added your account to the remote server but the current account on the remote server has not been added, then that account cannot debug even if a user with that account name is running Visual InterDev on the server computer.

Running SQL Debugging

Unlike debugging other types of processes, you don't debug stored procedures or triggers while they are running. Instead, you open the procedure in the editor and debug it from there.

To debug a stored procedure

- In the **Data View** window, right-click the stored procedure to debug, and then choose **Debug**.

  The editor window opens with the stored procedure text in it and the debugging commands enabled on the **Debug** menu.

After you've opened the editor window in Debug mode, you can use debugger commands as usual. For example, you can set breakpoints and step through the procedure. You can view the values of variables and passed parameters in the Locals window. You can also drag expressions to the Watch window to track them as you step through or run the procedure. The results of SQL PRINT statements are displayed in the Output window.

However, the SQL debugger has the following differences from the debugger used for script:

- The Auto and Immediate windows are not functional while you are debugging stored procedures. Although you can display them, the Auto window will be empty and the Immediate window will not allow you to enter expressions.
- You cannot change the execution sequence using the Set Next Statement command.

If you are working with simple SELECT statements — ones that return only a single value — the return value is in a variable that you can inspect in the Locals window. However, if the SELECT statement returns a result set, it is not displayed in the debugger. Instead, you can view the result set in the Output window.

Executing Parameterized Queries

In many applications, you want to work with data sets created using conditions supplied in the application. For example, your application might display a report of all the employees in a department. You can set up a form to prompt users for the name of the department, and then execute a query based on the value they enter. This type of query is called a *parameterized query*.

For parameterized queries, you use a Recordset design-time control as you would for a table or other database object. The difference is that the Recordset control is bound to a stored procedure or an SQL statement instead of a table.

You can bind the control directly to the procedure or statement, or you can bind it to a data command that points to one of these types of object. For details about using data commands, see "Getting Records" in Chapter 19, "Viewing Data."

To create a parameterized query

1. Drag a Recordset design-time control onto the page. For details, see Chapter 19, "Viewing Data."

2. Right-click the control and choose **Properties**. In the **General** tab, set the control's name and connection.

3. Under **Source of data**, specify the binding:

   - If you are binding to a data command, choose **Database object**, and then select **DE Commands**. From the **Object name** list, choose the name of the data command to use.

   - If you are binding directly to a stored procedure, choose **Database object**, and then select **Stored Procedures**. From the **Object name** list, choose the name of the stored procedure to use.

   - For SQL queries, choose **SQL statement**, and then enter the SQL text. Choose **SQL builder** to launch the Query Designer. In the SQL statement, use a question mark (?) to indicate parameters in your query. For example, the following SQL statement creates a query in which the department name is a parameter:

     ```
     SELECT * FROM employee WHERE department = ?
     ```

     Note Do not use named parameters.

4. Choose the **Parameters** tab. Under **Values for parameters** you will see a list of the parameters specified for your query. For each parameter where the value of **Type** is **In** (a parameter that is being passed into the stored procedure or query), specify a value to be passed to the stored procedure. Values can be:

   - **Literals** Enter character values in single quotation marks and numeric values without quotation marks.

   - **Variables** Enter the name of a variable defined in server code that will contain the value you want to pass.

   - **Object references** Enter an object reference and property value, such as `Textbox1.value`. The object must be one that is available in server script. Be sure to use correct capitalization, because the expression will be evaluated as a JavaScript expression.

   - **Expressions** Enter any combination of literals, variables, object references, and function calls. The expression is evaluated as a JavaScript expression, so use JavaScript conventions, including single quotation marks for character literals and a plus sign (+) for concatenation.

You must make sure that the value for the parameter can be evaluated successfully when the query is run. By default, a Recordset design-time control will execute the query when the page is first loaded. In that case, the parameter value cannot be a value that is gathered or evaluated only after the page has been displayed. If you are passing the value of a variable as a query parameter, you can use two events that are processed before the recordset is opened:

- The onenter event for the page.
- The onbeforeopen event for the recordset.

Alternatively, you can also specify that the Recordset control does not automatically open the recordset when the page opens. This strategy is useful if you are using the same page for the data-entry form and the results.

To prevent the Recordset control from automatically opening a recordset

- Right-click the Recordset control and choose **Properties**. In the **Implementation** tab, clear the **Automatically open the Recordset** option.

You can initially display the page without the recordset and prompt for a value using a form. When the user fills in the form and clicks a button, you can open the recordset and pass it the value of a variable or control.

To open the recordset

- Call the recordset script object's open method.

For example, you might create a form with a textbox design-time control and a button design-time control. The user can fill in the text box with the value to search for, and then click the button.

In the Parameters tab of the Recordset control's Property Pages window, specify the textbox script object's value as the parameter using this expression, substituting the name of the textbox for *textbox1*:

```
textbox1.value
```

The onclick event handler for the button then simply calls the recordset script object's open method, as in this example:

```
Function Button1_onclick
   Recordset1.open()
End Function
```

Executing Dynamic Queries

If the recordset script object is bound to an SQL query, you can change the query at run time and re-execute it. This is particularly useful if you want to run a query that is based on information provided by the user.

The current text of your SQL command is available by calling the recordset script object's getSQLText method. You can set the SQL text by calling its setSQLText method. After setting the SQL text, you re-execute the query by calling the recordset script object's open method.

The following example shows how you can dynamically change the sort order of the records in your recordset. It gets the SQL text, appends a new ORDER BY clause onto the end of the command, updates the recordset again, and runs the query. After the query has been run, the handler restores the original query text.

```
Sub btnByFName_onclick()
   oldSQL= rsEmployeeList.getSQLText
   newSQL = oldSQL & " ORDER BY firstName"
   rsEmployeeList.setSQLText(newSQL)
   rsEmployeeList.Open    ' Executes the new query
   rsEmployeeList.moveFirst
   ' Restores the original query
   rsEmployeeList.setSQLText(oldSQL)
End Sub
```

You can determine when the query has finished by writing a handler for the ondatasetcomplete event. For example, you might wait until the query has finished before you display data from it and before you enable the controls on the page. A handler to do that might look like this:

```
Sub rsEmployeeList_ondatasetcomplete()
   ' Enables controls
   btnNext.disabled = False
   btnPrevious.disabled = False
   btnSave.disabled = False
   btnNew.disabled = False
   ' Refreshes the fields on the page
   txtLastName.value = rsEmployeeList.fields.getValue("au_lname")
   txtFirstName.value = rsEmployeeList.fields.getValue("au_fname")
End sub
```

Executing Database Commands Using the Data Environment

The most convenient way to add database access to your Web pages is to use Recordset controls and data-bound controls. These tools are easy to add to your pages, and they expose a rich object model that provides powerful access to database functionality. For details, see "Design-Time Controls" in your Visual InterDev online documentation.

In some cases, however, you might want to create script that executes database commands more directly. Doing so offers a few advantages: your page is smaller than if you use a Recordset control. You might also want to script database access directly if are creating your own user interface and don't want to rely on the design-time controls.

You can script database access that goes through the server, or you can access databases directly from client script. For a discussion of server and client access to data, see "Data Binding" in Chapter 18, "Database Concepts."

To script server-based database access, you rely on the data environment for connection information and database commands, as described later in this topic. For client-based database access, you use Remote Data Service (RDS). For information about RDS, see the Microsoft RDS Web site at http://www.microsoft.com/data/rds/.

Scripting the DE Object

To script server-based database access, you work with a special object — the DE object — that exposes an object model for executing commands and managing their results. The DE object model is an easier-to-use version of the ActiveX Data Objects (ADO) model.

Before scripting database access, you must add a data connection to your project's data environment. When you do, Microsoft Visual InterDev adds script to the Global.asa file that creates a DE object incorporating the connection information. When you write data access script in your pages, you can reference this DE object, and Visual InterDev will automatically know how to connect to the database.

After adding a data connection, you add one or more data commands to the data connection. Each data command provides you with access to a database object, which can include a table or view, an SQL command, or a stored procedure.

Executing a Command

You can execute database commands by referencing the corresponding command object in your script.

To execute a database command

1. Be sure that the command you want to execute is defined as a data command in the data environment. For details, see "Getting Records" in Chapter 19, "Viewing Data."
2. In your Web page, use server script to reference the DE object. If you have enabled the scripting object model for the page, use this script:

   ```
   <%
      thisPage.createDE()
   %>
   ```

   If you are not using the scripting object mode, use the following script:

   ```
   <%
   Set DE = Server.CreateObject("DERuntime.DERuntime")
   DE.Init(Application("DE"))
   %>
   ```

3. In server script, execute the command you want to use by referencing its name, using this syntax:

   ```
   DE.commandObjectName
   ```

   For example, to execute the SQL query associated with a command object called Authors, use this script:

   ```
   <%DE.Authors%>
   ```

4. If the command takes parameters (such as a command that references a stored procedure or parameterized query), you can pass the parameters when you call it, using this syntax:

   ```
   DE.commandObjectName (parameter1, parameter2, [...])
   ```

   You can pass parameters as variables, literals (character strings should be passed using the correct quotation marks for the language you are scripting in), or any other valid expression.

   For example, to execute a stored procedure that takes two parameters, reference its corresponding command object with script such as this:

   ```
   <%DE.updateAuthors ("A101", txtAuthorLname, 100.00, "London")%>
   ```

5. If the command returns a value, you can assign it to a variable when you execute the command, using script like this:

   ```
   <%iRetValue = DE.updateAuthors ("A101", txtLname, 100.00)%>
   ```

Working with Result Sets

All commands return a result set, although in some cases (as with update queries) the result set is empty. But if your command returns a useful result set, you can navigate in it and extract its contents to display on a page.

To provide access to the result set, the DE object dynamically creates a result set object based on the command object, named using this syntax:

```
DE.rscommandObjectName
```

You can set a variable to this object, and then reference it in your scripts.

When a result set is generated, it includes a pointer indicating which record is the current record. Any operations that you perform, such as displaying data, navigating, or updating, are always performed with respect to the current record. To work with a different record, you must first navigate to that record.

The procedures outlined below allow you to work with a result set that is used entirely on one page. For example, you can use the procedures here to create a page that lists all the records from a result set in a table in one page.

To extract the contents of a result set

1. After creating the result set, set a variable to point to the DE result set object. The DE result set object is named after your command object, but has the "rs" prefix. For example, the following two statements create the result set, and then set a variable to point to it:

   ```
   <%
   DE.Authors
   Set rs = DE.rsAuthors
   %>
   ```

2. Extract individual values from the Fields collection of the result set object, using syntax such as the following:

   ```
   <%
   DE.Authors
   Set rs = DE.rsAuthors
   lname = rs.Fields("au_lname")
   %>
   ```

To move to a different record, you use navigation methods on the record set.

To navigate in a result set

1. Call the moveNext, movePrevious, moveFirst, or moveLast methods of the result set object.
2. To determine whether you are at the end or beginning of the result set, test the EOF or BOF properties.

The following script shows a complete example of how to script the DE object to get information from an Authors table. The script opens a recordset based on a data command object called "Authors." It then navigates from the beginning to the end of the result set, displaying for each record the contents of the `au_lname` field.

```
<%
Set DE = Server.CreateObject("DERuntime.DERuntime")
DE.Init(Application("DE"))
DE.Authors
set rs = DE.rsAuthors
rs.moveFirst
Do While Not rs.EOF
%>
   Next name = <%=rs.Fields("au_lname")%><br>
   <%
   rs.moveNext
Loop
%>
```

Paging Records

If you want to display a single record from the result set on a page and then provide navigation controls to page between records, the situation is more complex, as described in "Data Binding" in Chapter 18, "Database Concepts."

If that is your goal, you will find it more convenient to use a combination of a Recordset control, data-bound controls, and a RecordsetNavbar control.

For details, see "Displaying Records" in Chapter 3, "Database Basics."

For additional information about designing paging, see the Microsoft Visual Studio Web site at http://www.microsoft.com/vstudio/.

CHAPTER 22

Managing Database Projects

If you're managing enterprise databases in Microsoft Visual InterDev, database projects give you a big advantage: complete control over your database, its structure, and its data. Database projects make this possible through SQL scripts, files that you can use to build the database and populate it with data. These files and the folders that contain them in the database project can be put under source control for even greater control.

When you create a database project and add a data connection, the database is displayed in the Data View window, and you can manipulate the objects in the database using the Microsoft Visual Database Tools (the Database Designer and Query Designer).

You can use the same data connection in a Web project and a database project. This allows you to maintain control over databases whose data you want to display on Web pages.

Using database projects in addition to Web projects gives you the following benefits:

- Data connections aren't required in a database project until you want to run a SQL script against a database. In addition, you can rename data connections in a database project.
- You can use source control to ensure that the database and its structure are under the administrator's control.
- You can easily deploy the database to other projects.

Populating a Database Project

A database project can contain one or more connections to databases. Each database contains database objects such as tables, queries, views, relationships, and indexes, in addition to the database's data.

You can create and modify the database objects and the data in three ways:

- Using SQL scripts you add to the database project.
- Editing the database objects directly in Data View.
- Using the Microsoft Visual Database Tools (Database Designer and Query Designer).

You use SQL scripts to give you complete control over the database, its objects, and its data. SQL scripts are files containing SQL statements that create and modify databases and database objects.

To populate a database project using SQL script files

1. Add SQL script files to your database project.
2. Execute the SQL script files against databases to which you've made a data connection. For information on how to do this, and what SQL script files contain, see "Working with SQL Scripts" in the Visual Database Tools online documentation included with Visual InterDev.

   Tips Microsoft Visual InterDev supplies a number of templates you can use to add SQL script files to your database project. When you right-click on a database project or one of its data connections and select the **Add SQL Script** command, the following templates are available: **New SQL Script, New Stored Procedure Script, New Table Script, New Trigger Script**, and **New View Script.**

   The **New SQL Script** template adds a generic SQL script file to your database project, which you can use to create or modify any type of database object. The other templates create SQL scripts specific to SQL Server database objects.

   You can also right-click a SQL Server database object in the Data View window and choose the **Copy SQL Script** command. You can then paste this SQL script into the SQL editor or directly into a database project. This creates an SQL script, which, when run against the database, will create the database object you selected.

   You can also select multiple database objects in the Data View window and use the **Copy SQL Script** command to create an SQL script that you can add to a database project and use to create all the selected database objects. In addition to SQL Server database objects, the **Copy SQL Script** command also works with Oracle views, triggers, stored procedures, and functions, but not with Oracle tables and synonyms.

When you add a data connection to a database project, the data connection and database objects are displayed in Data View. You can edit the database objects directly. This is the easiest way to create and modify database objects and data, but you don't have as much control over the database as you do with SQL scripts.

To populate a database project using Data View

1. Select the database object you want to modify, or choose a command to create a new object.
2. Edit the data in the object directly, or use the SQL editor to modify the SQL statements in the object.

The Query Designer and Database Designer allow you to create and modify queries, tables, and objects associated with tables such as indexes, constraints, and relationships.

To populate a database project using the Database Designer

1. In Data View, select the data connection containing the database.
2. Select the database object (table or database diagram) you want to modify and choose the **Design** command from the shortcut menu.

   – or –

   Create a new table or new database diagram from the shortcut menu.
3. Modify and save the database object.

To populate a database project using the Query Designer

1. In the Project Explorer, select the data connection containing the database.
2. Select the query you want to modify and choose the **Design** command from the shortcut menu.

   – or –

   Choose the **New Query** command from the shortcut menu.
3. Modify and save the query.

Using Database Projects and Web Projects Together

Database projects and Web projects appear in separate nodes in a solution because they are different types of projects. Use Web projects to create dynamic Web pages and display information (including data from databases). Use database projects to create, manage, and deploy enterprise databases.

You can use a database project and a Web project together, however, if you want to maintain control over and deploy a database whose data you also want to display on a Web page. You can create a data connection in the Web project, then add this connection to the database project. The SQL scripts in the database project control the structure and data in the database.

To use a Web project data connection in a database project

1. Create a Web project. For details, see "Creating a Web Project" in Chapter 1, "Web Project Management."
2. Add a data connection to the Web project. For details, see "Connecting to a Database" in Chapter 3, "Database Basics."
3. Create a database project. For details, see "Creating a Database Project" in the Visual Database Tools online documentation included with Visual InterDev.
4. Add the data connection from the Web project to the database project.

5. Create SQL scripts in the database project to manage the database and its objects.
6. If you want, deploy the database project to other solutions.

   Note You can't drag and drop database objects between Web projects and database projects.

Adding Source Control to a Database Project

You can maintain control over a database and its structure by adding your database project files to source control. Database projects consist of folders and SQL script files. Once you've placed these folders and files under source control, you can manage these folders and files, just as you do with the folders and files in Web projects. For information on how to use source control with folders and files, see Chapter 8, "Working with Multiple Developers."

Note When you add folders and files to a database project, they are physically added to your hard disk. In addition, when you put a database project under source control, there is a one-to-one correspondence between the files and folders in the database project and the files and folders in the source control project.

You can also rename a data connection in a database project. Because connections are represented as folders on your hard disk and in the source control project, this will also rename those folders.

To add a database project to source control

1. In Project Explorer, select the database project.
2. Right-click the database project name, and from the shortcut menu, choose **Add to Source Control**.

   The database project and its files are now under source control.

   If you want to remove them from source control later, you can use the **Remove from Source Control** command on the shortcut menu.

In the Data View window, you can put stored procedures under source control. This gives you additional control over these database objects. In a database project, a stored procedure might be contained in an SQL script; for Web projects they appear in Data View as stored procedures. For more information, see "Adding a Stored Procedure to Source Control in Data View" in the Visual Database Tools online documentation included with Visual InterDev.

If you are working locally, databases on a Web server or database server are available. However, if you are offline, you must have the database on your local computer. There's no way to merge any changes you make to the data while you are offline in Microsoft Visual InterDev.

PART 5

Editing and Scripting

Part 5 provides information about editing pages, the programming model, and debugging script.

Chapter 23 Scripting Concepts

This chapter covers the Visual InterDev editor and the concepts behind "Scripts in Web Applications," the "Scripting Object Model," "Document Elements," and the "Script Debugging Process."

Chapter 24 Scripting with Design-Time Controls and Script Objects

This chapter covers creating a user interface for your Web application using the Visual InterDev design-time controls and extending the Visual InterDev scripting object model.

Chapter 25 Scripting with HTML Elements

Even if you do not use design-time controls in your Web page, you can create fully functional Web applications by creating and scripting HTML elements. This chapter describes writing script to interact with native HTML elements.

Chapter 26 Debugging Your Pages

This chapter covers debugging both client and server script in Visual InterDev.

Chapter 27 Packaging Script as Objects

Microsoft Scripting Components (scriptlets) provide you with an easy way to create powerful, reusable controls and COM components. This chapter discusses creating and using scriptlets.

CHAPTER 23

Scripting Concepts

Editing Modes

The default Microsoft Visual InterDev HTML editor allows you to work with Web pages in different modes:

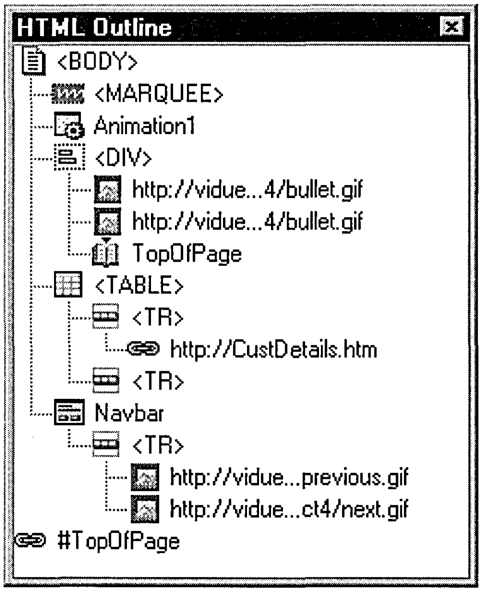
- **Design view** The editor displays text with character and paragraph formatting, much like a word processor.
- **Source view** The editor shows HTML tags, text, and script, and highlights the HTML tags and text.
- **Quick view** The editor displays .htm files as they will appear in Microsoft Internet Explorer.

Navigating in the Editor

When you are using the HTML editor, Visual InterDev allows you to display various document outlines that help you move quickly and easily through your document.

- In Design and Source view, the HTML Outline window shows a hierarchical view of the HTML elements and objects in your page.

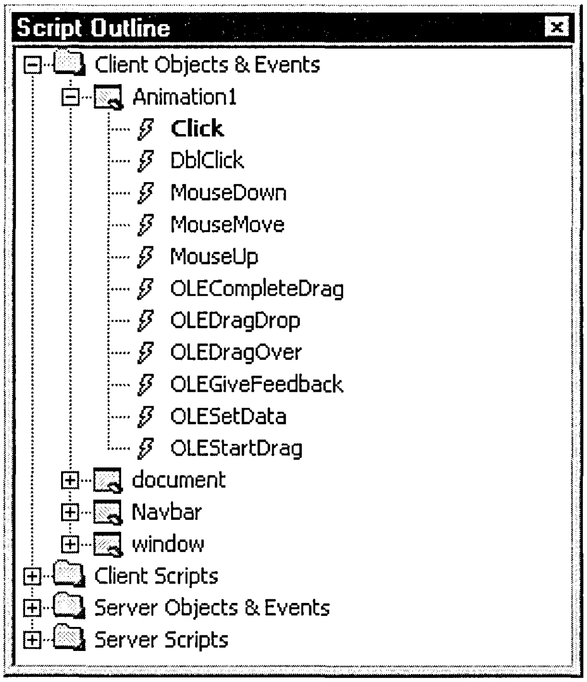
HTML Outline Window



```
<BODY>
  <MARQUEE>
  Animation1
  <DIV>
    http://vidue...4/bullet.gif
    http://vidue...4/bullet.gif
    TopOfPage
  <TABLE>
    <TR>
      http://CustDetails.htm
    <TR>
  Navbar
    <TR>
      http://vidue...previous.gif
      http://vidue...ct4/next.gif
#TopOfPage
```

- In Source view, the Script Outline window shows a tree view of all scriptable elements on the page, and for each element, the events for which you can write scripts. Existing scripts are indicated in bold.

Script Outline Window



When you select an element in either outline window, you move to that element in the page.

The Toolbox

When you work in the editor, you can add objects to your pages by dragging them from the Toolbox and dropping them onto a page. You can use the Toolbox in Design view or Source view. The Toolbox displays a preselected set of controls currently available on your computer, including standard HTML controls (such as text boxes and buttons), Visual InterDev design-time controls, ActiveX controls, and server objects.

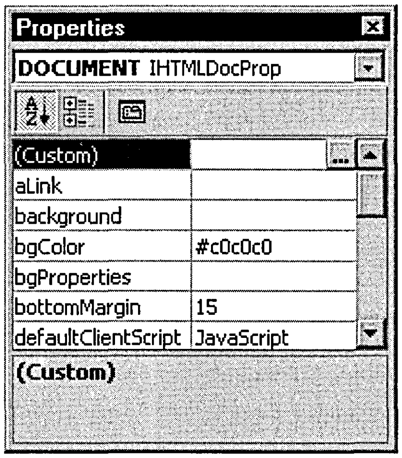
The Toolbox is more than just an alternative means of adding objects, however. You can use it to store any text from the editor, including scripts, HTML text, and so on. These bits of text you store in the Toolbox are referred to as *fragments*. In addition, you can edit the tabs in the Toolbox, adding, deleting, and reorganizing them to suit your development requirements.

Design View

Design view makes it easy to view and edit HTML text in a format similar to what it will look like in a browser. You can add and work with HTML text, as well as elements such as images, tables, design-time controls, HTML controls (forms, buttons, and so on), ActiveX controls, and Java applets. Your page is displayed with all the character and paragraph formatting that you've specified, much like in a browser. If you are using a cascading style sheet (CSS) or have added style information to HTML tags, Design view reflects these as well.

As you work in Design view, you can set properties for elements on the page. If the Properties window is displayed, it displays properties for the currently selected element on the page. Some HTML elements such as tables allow you to set attributes using custom properties dialog boxes as well.

Properties Window in Design View


Properties

DOCUMENT IHTMLDocProp

(Custom)	
aLink	
background	
bgColor	#c0c0c0
bgProperties	
bottomMargin	15
defaultClientScript	JavaScript

(Custom)

Note In Design view, the Properties window displays style information that is not displayed in Source view.

Although Design view is similar to how a page will look in a browser, it differs in the following ways:

- Character and paragraph formatting might appear different, because each browser can implement formatting differently.
- Client scripts do not run.
- Links are not live.
- Some elements, such as scripts, comments, and unrecognized HTML tags, can be displayed as glyphs, so that you know they are in the page.
- You can display a border around elements that are invisible when displayed in the browser so you can see where they are. For example, even if a table is defined with no border, you can choose Visible Borders from the Design toolbar to have the editor display a one-pixel border around the table so you can work with the table more easily.

For more details about Design view, see "Design View, HTML Editor" and "Design Toolbar" in the online documentation.

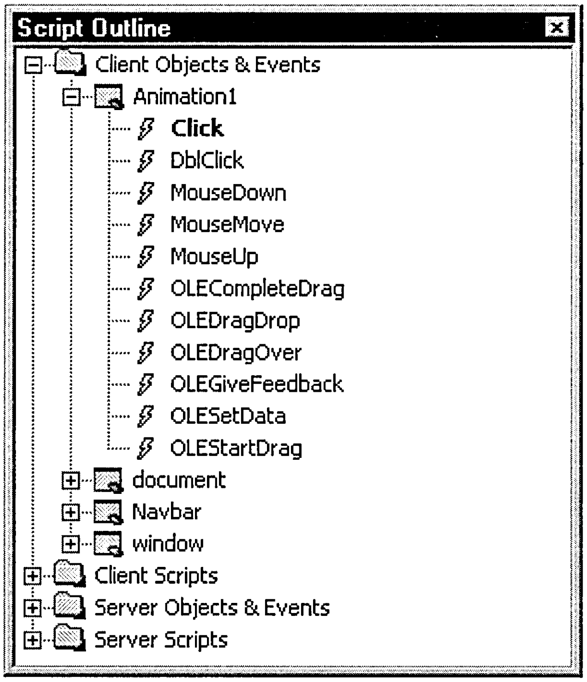
Source View

Source view allows you to see and edit the raw HTML of your page and work directly with script. You can also work with either graphical or text representations of Visual InterDev design-time controls, ActiveX controls, and Java applets. Text is colored so you can easily distinguish script keywords, HTML tags, attributes, comments, and so on. These features make it easy for you to develop a Web page while giving you complete control over its contents.

To edit objects or HTML elements, you can select them, and then set their properties using the Properties window or a custom Property Pages window. Changes you make in the Properties window or in Property Pages dialog boxes are reflected in the HTML source code for those objects.

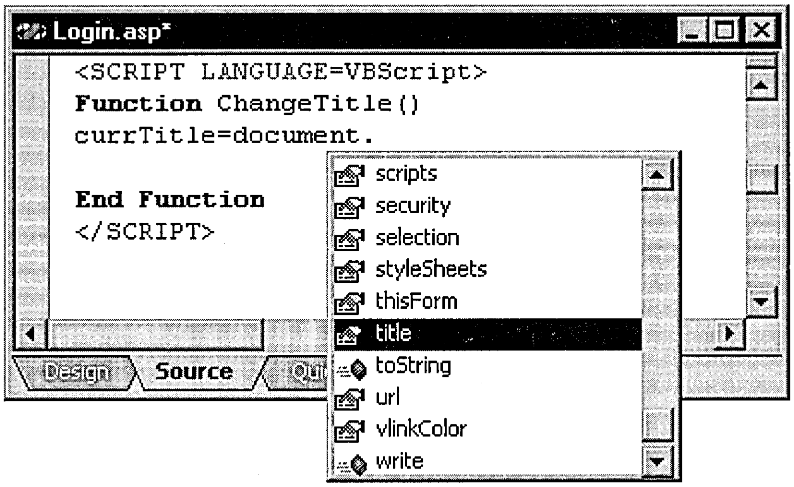
If your page contains scripts, you can see the source code of the scripts in Source view. While in Source view, you can use the Script Outline window to view scriptable elements in your page and to see what scripts are already created.

The Script Outline Window



As you are typing statements, IntelliSense helps you complete statements by displaying a list of properties and methods for objects that you are using.

IntelliSense Statement Completion



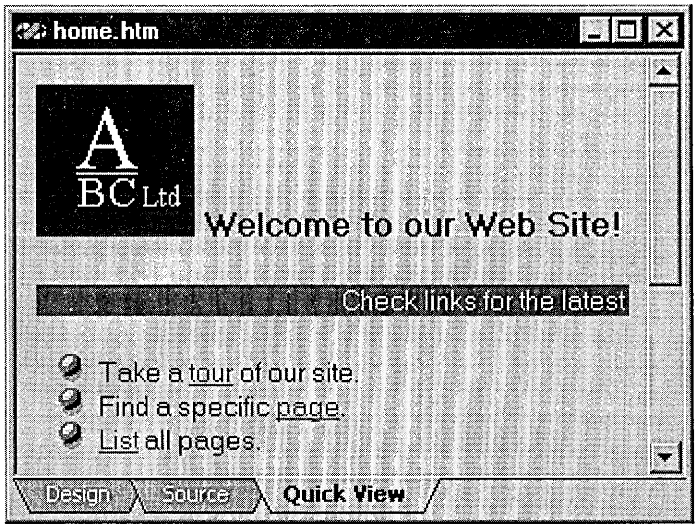
```
<SCRIPT LANGUAGE=VBScript>
Function ChangeTitle()
currTitle=document.

End Function
</SCRIPT>
```

While you are editing a script in Source view, you can invoke a debugger that allows you to set breakpoints, step through scripts, and perform other typical debugging tasks. For more information, see Chapter 26, "Debugging Your Pages."

Quick View

To see a preview of what an .htm file will look like in Microsoft Internet Explorer, you can use Quick view. Images and links work as if the document were in its location on the target Web server.

Quick View



Note Quick view does not process the page through the server, so it cannot provide an accurate view of what ASP files will look like. If you want to preview a document containing server elements, use the **View in Browser** command from the **View** menu.

Because Quick view shows you a page in a browser, you cannot edit or debug the page in that view.

Scripts in Web Applications

To display text, images, or links in a page, you add text and format it with HTML tags. However, to control the way a page behaves, you create script, or programs that you embed in a Web page to perform specific functions, such as:

- Controlling what happens when a user clicks a button, enters text, or submits a form.
- Navigating to a specific page based on a condition such as user preference.
- Collecting and storing user information in order to customize Web applications dynamically.
- Querying a database and displaying results.

You can create script in different ways:

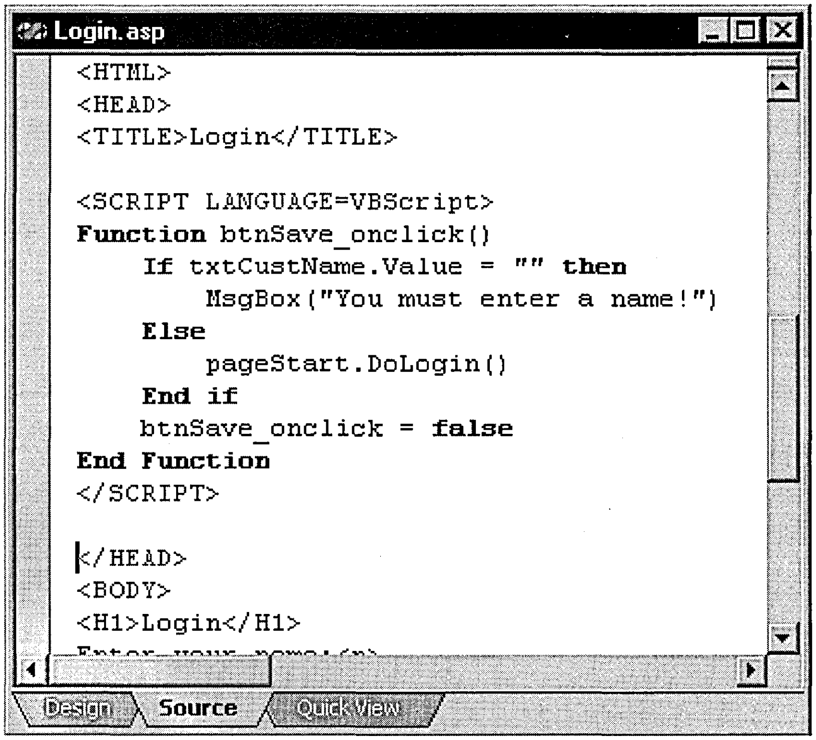
- Use design-time controls, which allows you to set property values and enter values in dialog boxes, and then generates script for you.
- Write your script in Source view of the HTML editor.

Microsoft Visual InterDev supports a complete scripting object model that allows you to use standard object-oriented techniques for creating Web pages. For details, see "The Scripting Object Model" later in this chapter.

How Scripting Works

The script's source code appears in the page, as shown in this example.

Script Source Code



```
<HTML>
<HEAD>
<TITLE>Login</TITLE>

<SCRIPT LANGUAGE=VBScript>
Function btnSave_onclick()
    If txtCustName.Value = "" then
        MsgBox("You must enter a name!")
    Else
        pageStart.DoLogin()
    End if
    btnSave_onclick = false
End Function
</SCRIPT>

</HEAD>
<BODY>
<H1>Login</H1>
```

When the page is requested, the script is read along with all the other text on the page. The server or browser reads the scripts and checks for errors, and then runs the script.

Because scripts are simply blocks of text, you can write a script in one page, and then include it in multiple additional pages. For details, see "Writing Reusable Script" in Chapter 25, "Scripting with HTML Elements."

Scripting Languages

You can write scripts in any scripting language that you are comfortable with. Common scripting languages include Microsoft Visual Basic, Scripting Edition (VBScript) and ECMAScript, a standard scripting language. Popular implementations of ECMAScript are Microsoft JScript and JavaScript.

Because scripting languages are interpreted, you must be sure that when the user requests a page, the user's browser (and server, if you are writing server script) can use the language in which you have scripted. For example, if you write all your scripts in VBScript, you must be sure that the user's browser can interpret VBScript. Microsoft Internet Explorer supports VBScript, but not all browsers do. For details on determining the capabilities of a browser, refer to the documentation for your browser, and see "Creating Portable Script" in Chapter 25, "Scripting with HTML Elements."

You can use different scripting languages on the same page if necessary. This might be the case, for example, if you are adding your own scripts to scripts generated by a design-time control. When you use design-time controls, you specify a target scripting platform (client or server), which determines by default what language to script in. For details, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Scripting Objects."

If you are scripting outside of design-time controls, you often work in only one language, so you can specify a default language for each new page that you create, and if you need to, for individual scripts. For details, see "Choosing a Scripting Language" in Chapter 25, "Scripting with HTML Elements."

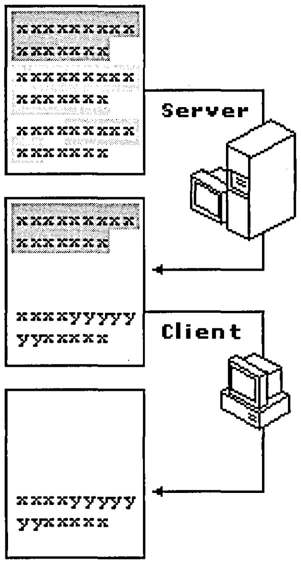
Client and Server Scripts

If your server is Microsoft Internet Information Server (IIS), you can create ASP files that contain both *client* and *server scripts*. Both types of scripts can appear in the same page.

Client scripts are part of a page, and are sent to and run by the browser when a user requests the page.

Server scripts are also part of a page, but are not sent to the browser. Instead, they are run by IIS after the page is requested but before it is passed to the browser. When the page is sent to the browser, the server has already run the server script and removed it from the page.

Client and server script execution



The ability to specify that a script runs on the client or on the server is an important feature of Web scripting, which allows you to specify the right run-time environment for the task you want to perform. For example, the following are tasks typically performed using client scripts:

- Change the text or appearance of a page at the time it is loaded in the browser or in response to an event such as a button click.
- Perform validation on data entered into an HTML form before it is sent to the server, such as making sure that an employee ID number contains the correct number of digits. In contrast, verifying data against a database is typically a server-side task.
- Display information in response to a user event such as a button click.

In contrast, these are tasks that you would typically perform using server scripts:

- Query a database and feed the results to an HTML page. For details, see Chapter 19, "Viewing Data."
- Redirect a user's request to a specific page based on a condition, such as password lookup. For in-depth information about using scripts to move between pages, see "Extending the Scripting Object Model Across Pages" in Chapter 24, "Scripting with Design-Time Controls and Script Objects," and "Navigating Conditionally" in Chapter 25, "Scripting with HTML Elements."
- Process the information entered by a user on an HTML form. For in-depth information about how to create scripts for processing HTML forms, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects," and "Gathering Information with HTML Forms" in Chapter 25, "Scripting with HTML Elements."

Although the tasks listed above are typical uses for client or server scripts, they are not rigid rules. For example, you can use Visual InterDev design-time controls to create server script that responds to a client event such as a button click. As another example, if your users use Microsoft Internet Explorer 4.0 or another browser that supports Dynamic HTML (DHTML), you can write an application that accesses a database from the client browser.

The decision to use client or server script therefore depends not just on the task you are accomplishing, but the environment in which your application runs, specific constraints (such as performance), and so on.

Client Script Processing

Client scripts are processed by a browser such as Microsoft Internet Explorer, which calls the appropriate run-time module to execute the script. Client scripts are enclosed between <SCRIPT> and </SCRIPT> tags. The following simple example shows a script that prints the current time on the page:

```
<SCRIPT LANGUAGE="VBScript">
   Document.Write time
</SCRIPT>
```

Your page can contain as many script blocks as you need. You can put multiple functions and subroutines into a single script block, or put each in a separate script block.

Client scripts are processed at different times, depending on how they are written:

- Statements can appear in a script block but not as part of a procedure (function or subroutine). These are called *global* or *inline* scripts, and are processed in order when the browser reads the page. For example, the following shows a global script:

  ```
  <SCRIPT LANGUAGE="VBScript">
     Document.Write time
  </SCRIPT>
  ```

- Statements can appear as part of a procedure, such as a function or subroutine. These are not executed immediately. Instead, they are parsed when the page is run and checked for syntax errors. However, they are not run until the procedure is called.
- Event-handling procedures are not executed immediately. Instead, they are executed when the user performs the event that triggers the script, such as clicking a button. For example, the following script illustrates an event-handling subroutine that is executed only when the user clicks the Submit button on a form:

  ```
  <SCRIPT LANGUAGE="VBScript">
  Function btnTest_onclick
     If Len(Document.frmTest.txtName.value) < 1 then
        Alert("You must enter a name!")
     End If
  End Function
  </SCRIPT>
  ```

Scripts for event-handling procedures, subroutines, or functions can appear anywhere in a page, because they will be processed only when needed. However, it is common to put these types of scripts in the header of a page.

When you write client scripts, you can access objects on the page to get their properties or write event handlers for them. The exact list of objects available for you to work with depends on the type of browser that your users will be using. For more information, see "Document Elements" later in this chapter, and Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Client scripts can directly interact with the user by posting message boxes for output and by using dialog boxes or forms for input. For example, if a user makes an error when entering information in a form, a client script can display an error message (as shown in the preceding example).

For more information about displaying information, see "Displaying Information to the User" in Chapter 25, "Scripting with HTML Elements." For information about using forms from both client and server scripts, see Chapter 24, "Scripting with Design-Time Controls and Script Objects," and "Gathering Information with HTML Forms" in Chapter 25, "Scripting with HTML Elements."

Server Script Processing

In Visual InterDev, you put server script in Active Server Pages (ASP files). The ASP extension on the file alerts IIS that the page can contain server script. When IIS reads the page, it looks for server script and processes it. After the server script in an ASP file has been processed, it is removed from the file, which is then sent to the browser (including any client script that might be in the file). The browser treats the ASP file as it does an ordinary .htm file.

Server script can modify any aspect of a page before sending it to the browser. Typically this involves performing tasks and incorporating the output of the tasks into the HTML text of the page. However, server script can just as easily create client script, because client script is nothing more than additional text on the page.

A special case of ASP file processing is the Global.asa file. This file contains scripts that respond to application-wide events: each time the ASP application is started or closed, and each time a new user starts a session.

You can create server script in the Global.asa file for these events, which is useful for tasks such as storing application settings (for example, the default scripting language); initializing application-wide variables; maintaining counters; and so on. For details about creating event handlers in the Global.asa file to store global information, see "Sharing Dynamic Information" in Chapter 25, "Scripting with HTML Elements."

When you write server script in an ASP file, you distinguish it from other text (including client script) in one of two ways:

- Within the delimiters `<%` and `%>`. Any text between these two tags is processed as inline server script by IIS. The `<% %>` delimiters are often used to enclose expressions that are evaluated and inserted into the HTML text of a page. For example, the following server script displays the current time on a page:

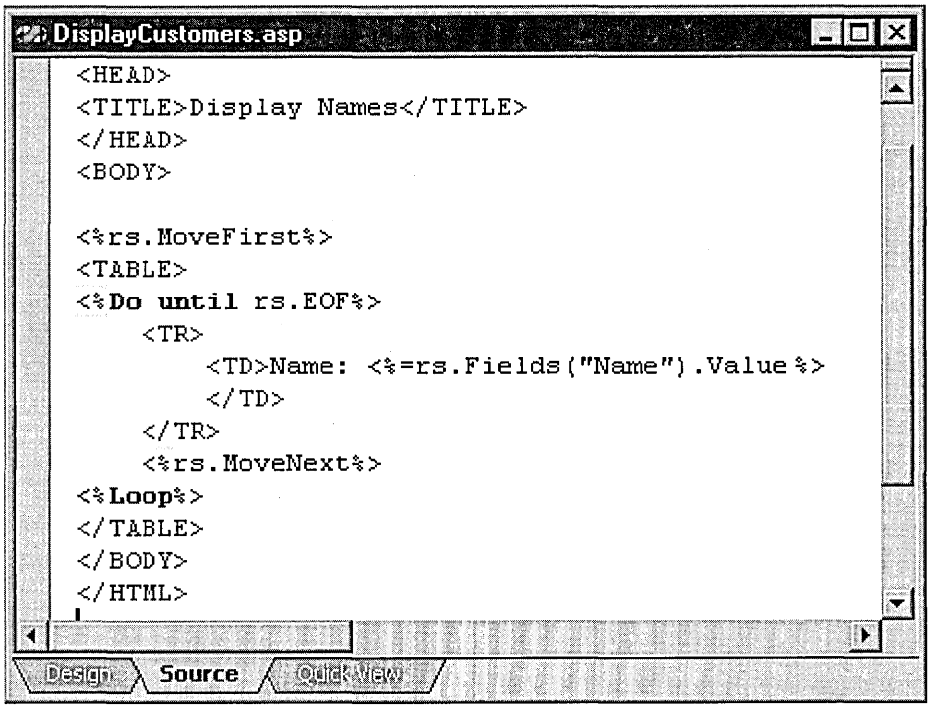
  ```
  <% response.write time %>
  ```

- In a <SCRIPT> tag (as with client scripts), but with the RUNAT=SERVER property, which is used to enclose stand-alone procedures such as functions and subroutines. The following example shows the RUNAT property:

  ```
  <SCRIPT RUNAT=SERVER>
     Function GetDate
        [some script lines here]
     End Function
  </SCRIPT>
  ```

In the Visual InterDev HTML editor, server script appears in yellow to distinguish it from client script. The following illustration shows a page that includes both server script and HTML text.

Editing server script



```
<HEAD>
<TITLE>Display Names</TITLE>
</HEAD>
<BODY>

<%rs.MoveFirst%>
<TABLE>
<%Do until rs.EOF%>
    <TR>
        <TD>Name: <%=rs.Fields("Name").Value%>
        </TD>
    </TR>
    <%rs.MoveNext%>
<%Loop%>
</TABLE>
</BODY>
</HTML>
```

Server script is generally not event-driven. Exceptions are the event handlers in the Global.asa file and server event handlers created by design-time controls. Instead, when the ASP page is requested, the server reads the page and processes all server script from top to bottom. The script performs whatever calculations and database access you write, and evaluates all expressions and variables. Stand-alone procedures are called as needed.

Because the script is running on the server, it has access to the objects available on the server. For example, a server script running on IIS can reference the ASP Application, Session, Request, and Response objects. A server script could not, however, make use of the objects available in the browser — for example, a server script could not use the Internet Explorer document or window objects.

When you write server script, you must be careful to use only objects that are available in the context of the server. For more information about objects that you can use in server scripts, see "Document Elements" later in this chapter, and Chapter 24, "Scripting with Design-Time Controls and Script Objects."

If the server script produces some output — for example, if you want to display the value of a variable, or display some records retrieved from a database — you can place the output on the page using the `Response.Write` method (or using the abbreviated form, the "=" operator) in inline script. For example, the following simple page shows server script that calculates the current time and puts it into a variable. Later in the page, the value of the server script variables are integrated into some HTML text:

```
<%
vDate = date
vTime = time
%>

<HTML>
<BODY>
When the script ran, it was <%Response.Write vTime%> o'clock on <%=vDate%>.
</BODY>
</HTML>
```

When the page is processed, the server evaluates the expression following Response.Write or "=", and its value is placed at that point in the HTML page stream. When the page is displayed in the browser, it will look something like this:

```
When the script ran, it was 10:46:30 o'clock on 12/31/97.
```

If you look at the source of the page, it would look the same as the page output. You would not see the expressions <%Response.Write vTime%> or <%=vDate%> in the source, because those expressions would have been evaluated by the server before the page was sent to the browser.

Server scripts can produce any type of output, including not just values for variables or expressions, but HTML tags and text and even client scripts.

The Scripting Object Model

To make Web application development faster and easier, Microsoft Visual InterDev provides the *scripting object model,* which simplifies Web application development by introducing a familiar object-oriented programming model to HTML and script programming. The model also greatly reduces the complexity and quantity of scripting required to write applications involving interaction between client (browser) and server.

The Visual InterDev scripting object model defines a set of objects with events, properties, and methods that you can use to create and script your application. You can create the visual interface for your application using design-time controls, and then write script to control the application using familiar object-oriented techniques.

The scripting object model allows you to create Web applications in much the same way you create applications in environments such as Microsoft Visual Basic and Microsoft Access.

The scripting object model is easiest to understand if you compare the scripting object model with how Web applications are created using the combination of ASP and HTML. To create a form, for example, you place HTML elements on a page, including text boxes, list boxes, and buttons. One of the buttons is typically a Submit button, which causes the form to be sent to the server, and which specifies an ASP page containing server script to process the form. Scripts on the destination page must manually examine the state submitted by the browser, and there is no association with the object that created the state.

Note In Dynamic HTML (DHTML), controls on a page (and the page itself) are part of a document object model, but that model applies only to client scripts; it does not include server-side processing. In addition, the DHTML document object model can be used only with browsers that support DHTML.

In contrast, the scripting object model allows you to work with controls and with the page using standard object-oriented techniques. For example, rather than use the complex form submission process required by ASP and HTML, you can simply place a button on the page and write a handler for its onclick method to process the form. The scripting object model abstracts events such as onclick so that you can write handlers for them in either client script or server script.

Advantages of the Script Object Model

The scripting object model provides these advantages:

- **Rapid application development using a familiar object-oriented model** You can apply standard object-oriented techniques to developing Web applications. Because you have to write less script and can use simpler script, you can create applications more quickly and with fewer errors.
- **Browser and platform independence** You can use the scripting object model no matter what browsers will access your application. It works virtually the same whether you design your applications for server script (for maximum reach) or client script (for a better user experience, such as less flashing and fewer round trips to the server).
- **Simplified forms** You can create forms by dragging design-time controls onto a page in the same way you do in environments such as Visual Basic, and change their appearance and behavior using properties. You can process forms by writing standard event handlers and by calling methods — the scripting object model handles the complexities of posting the form and dispatching to the correct form-handling logic.
- **Isolated application logic** You can more easily isolate your application logic in discrete procedures — including procedures on other pages — rather than mingling it with user interface and navigation elements.

- **State maintenance** The scripting object model provides a mechanism for retaining the state of objects as control is transferred between the client and server. You do not have to manually manage state in hidden fields or session variables.
- **Data binding** The scripting objects can be bound to database fields so you can use them for data-entry and data-editing forms.
- **Simplified page navigation** The scripting object model provides a mechanism that allows you to navigate to another page by name rather than specifying the URL of the target page, and that can jump directly to any procedure on the page.
- **Remote scripting** You can execute server scripts from a page without navigating off the page. As in traditional client/server applications, this allows browser scripts to ask the server for a set of results, which can then be used in the browser for further calculations or to alter the page through DHTML.

Components of the Scripting Object Model

When you use the scripting object model, you work with a variety of objects. You use the same objects whether you are writing server-based applications (in ASP files) or pages with script that will be executed on the client browser (.htm files).

Note The scripting object model is implemented using script stored in the Script Library. Do not alter the contents of the library, or components in the scripting object model will not work properly.

Enabling the Scripting Object Model

Before you can use the scripting object model, you must enable it, which constructs the scripting object model framework for the page.

Note Visual InterDev design-time controls require the scripting object model. If you add a design-time control to a page that does not already have the scripting object model enabled, Visual InterDev prompts you to enable it.

To enable the scripting object model for a page

1. Right-click anywhere in the page away from an object or control, choose **Properties**, and then choose the **General** tab.
2. Under **ASP settings**, choose **Enable scripting object model**.

   The Visual InterDev editor adds the scripting object model framework to the page at the top and bottom. You should not alter the content of these tags.

Design-Time Controls and Script Objects

To create the user interface for your application, you can use the Visual InterDev *design-time controls.* A design-time control is a control used when you are working in the editor to create Web application's functionality. Design-time controls generate HTML text, script, and sometimes other components to implement at run time the functionality you are designing.

The Visual InterDev design-time controls include standard user-interface elements — text box, label, check box, list box, command button, and so on — that produce run-time text based on the scripting object model. Using the design-time controls makes it easy to create forms and bind to data by taking advantage of the features of the scripting object model. For details about design-time controls, see the topic "Design-Time Controls" in the Visual InterDev online documentation.

Design-time controls have two levels of interaction. At design time, they are like controls that you might put on a form in an environment such as Visual Basic. You can interact with them visually in the HTML editor and set their properties to specify their appearance and behavior. Their actual purpose, however, is to generate script to be executed when the page is running. Therefore, when you set properties of a design-time control, you are actually changing the code that the design-time control generates.

When the generated code for a design-time control runs, it dynamically creates a *script object.* The script object is the object you write script against, setting its properties, calling its methods, and responding to its events.

The properties you set for a design-time control while working in the editor differ from those available at run time in the corresponding script object. However, they are related. For example, a design-time control's ID property is used to create the ID property of a script object at run time.

While designing your user interface, you can set properties for design-time controls in the Properties window or in custom property pages. Script object properties are not displayed in the Properties window, because they are run-time properties. However, they do appear in the IntelliSense statement completion drop-down list when you are scripting.

The design-time control also has no methods or events, because it never operates at run time. Script objects have predefined methods and events. As with script object properties, you can see a list of methods and events in the IntelliSense statement completion drop-down list. In addition, you can see the predefined events for a script object in the Script Outline window.

In some instances, a design-time control has a visual appearance at design time, but the corresponding script object has none. For example, the Recordset control, which controls access to data, is visible at design time so you can set its properties. But when the page runs, the script object created by the Recordset script object has no visible component, even though it has properties, methods, and events just like any other script object.

One of the advantages of design-time controls and script objects is that you can work with them the same way whether you want them to run on the server or the client. For example, if you want to target many different browsers, you can set the platform for the controls to "server," and all the design-time controls' generated code will run on the server. If you target the client as platform, the generated code will run on the client (though you must make sure that the client is capable of running it). But your interaction with the script objects, such as calling their methods, is virtually identical. The complex task of using server script to respond to a client event, such as a button click, is built into the script objects.

Tip An easy way to assemble a form is to use the FormManager. For details, see "Simplifying Data Entry Pages" in Chapter 5, "Walkthroughs," and "Creating Event-Driven Forms" in Chapter 3, "Database Basics."

Page Objects

The scripting object model allows you to use Web pages as objects that you can reference in your scripts like any other object. By default, the current page is available as an object called `thisPage`.

Page objects can be referenced from other pages. This is useful for page navigation, especially if your application requires you to process a specific script on another page. For example, in your application you might process a form by navigating to another page that contains scripts for validation, database updates, and so on. In the HTML model, you post your form to the other page, and then write script on the destination page that would need to determine how the page had been called and how to proceed. By using a page object instead, you can call a method on that page object, and the scripting object model performs all the navigation and dispatching for you.

For example, in a page object called `EmployeeList`, write a function called `DoQuery`, and then expose it as method. From another page, you can execute this function by simply calling a method, as in the following:

```
EmployeeList.navigate.DoQuery("Accounting")
```

Using page objects also provides the means for remote scripting. Remote scripting allows you to call a script function process on an ASP page (to be run on the server) from client script without leaving the page. (The usual model is that you would have to leave the current page to execute a procedure at the server, even if the procedure is on the current page.) The client page therefore preserves its state, including where the script was executing before the call to the server was made.

Because remote scripts can also run asynchronously, you can provide a richer user experience. For example, you can perform database lookups to validate a user entry while the user is still working with a form. For more details, see "Executing Server Script Remotely" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Document Elements

When you write scripts, you can manipulate various objects to perform application tasks. For example, in a client script you might:

- Test a check box, and then set the value of a text box.
- Navigate to another page.
- Display a specific graphic in a Java applet when a page is first displayed.
- Create multimedia effects by moving text across the screen, resizing text or graphics, and so on.

In a server script you work with different objects, which you might manipulate in order to:

- Fetch the information that a user entered into a form.
- Establish a connection with a database, and then run a query against it.
- Determine whether the current browser supports specific features that your application requires.

The objects that you can use in a script depend on the context in which the script will run. If you are working in server script, you can use only the objects available on the server. Conversely, if you are writing a client script, you can only use objects that are part of the page, are available to the browser, or that you know exist on the client computer.

> **Note** If you use the scripting object model and design-time controls, you can use the same script objects whether you are working with client script or server script. For details, see Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Client Objects

In client scripts, you can get properties and call methods for objects on the page and write event handlers for them. For example, the following small script displays the title of the current document using the alert method of the window object and the title property of the `document` object:

```
<SCRIPT LANGUAGE="VBScript">
   alert(document.title)
</SCRIPT>
```

Each object has its own methods and properties, which you can use as required. In client scripts, most objects also support events for which you can write handlers. For example, you can write a handler for a button's onclick event to control what happens when the button is clicked.

The exact objects and events that are available in a client script depend on the object model available in the browser your users will be using. For example, in some browsers you can write a script that changes the text on the page, but not all browsers support that feature. In general, you can rely on the following:

- Most browsers, including Microsoft Internet Explorer 3.0, support at least HTML level 3.2. In the HTML 3.2 object model, you generally cannot alter the appearance of objects already on the page. However, you can write event handlers for HTML controls such as buttons, forms, Java applets, ActiveX controls, and for the document as a whole. A list of scriptable objects in HTML 3.2 appears in the next table.
- Some browsers, including Microsoft Internet Explorer 4.0, support Dynamic HTML (DHTML), which provides full complement of run-time properties, methods, and events for any named object on the page. You can also write a handler for a timer event to move or resize text or objects at specific intervals. DHTML gives you about the same level of control over an HTML page as you have over forms in Visual Basic or documents in Visual Basic for Applications (VBA).
- Visual InterDev supports a scripting object model and design-time controls that allow you to script controls on the page using standard object-oriented techniques. For details, see Chapter 24, "Scripting with Design-Time Controls and Script Objects." You can use the scripting object model and design-time controls with any browser, as long as you use Microsoft Internet Information Server as your server.

When you write script, it is a good idea to limit yourself to the object model supported by your users' browsers. For example, if you know that most of your users will be using Internet Explorer 3.0 (or another browser that supports HTML 3.2), you should not make your application dependent on DHTML features. Alternatively, you can write your application to test for a particular browser and then expose features based on what browser the user has. For information about testing browser capabilities, see "Creating Portable Script" in Chapter 25, "Scripting with HTML Elements."

The following table lists common client objects available in HTML 3.2.

Object	Description
`window`	Browser object that allows you to determine information about the browser, prompt the user for information, and display messages. The two commonly used events onload and onunload allow you to perform initialization tasks when a document is loaded.
`document`	Browser object that allows you to set document colors, determine the URL of the current and referring document, get the document's title, and write text into the page. For more information about writing text, see "Displaying Information to the User" in Chapter 25, "Scripting with HTML Elements."

(continued)

(continued)

Object	Description
`form`	Browser object that allows you to determine information about a form's method and action, and enumerate the elements in the form. You can write handlers for the form's submit event in order to specify what happens when the user clicks the submit button. For more information about using forms, see "Gathering Information with HTML Forms" in Chapter 25, "Scripting with HTML Elements." **Tip** Visual InterDev provides design-time controls and a FormManager control that allow you to create forms easily without writing scripts for the HTML <form> element. For details, see Chapter 24, "Scripting with Design-Time Controls and Script Objects," and "Creating Event-Driven Forms" in Chapter 3, "Database Basics."
`element`	Individual form elements such as buttons, text boxes, and so on. You can write handlers for element events such as being clicked, getting or losing focus, and changing. For more information, see "Gathering Information with HTML Forms" in Chapter 25, "Scripting with HTML Elements." **Tip** You can use Visual InterDev design-time controls instead. For details, see Chapter 24, "Scripting with Design-Time Controls and Script Objects."
Java Applets, ActiveX controls	Objects created externally and added to the page. Applets are placed onto a page in an <APPLET> block, and objects in an <OBJECT> block. Applets and ActiveX controls support their own set of properties, methods, and events for which you can write handlers.

Server Objects

If you are writing a script that will run on Microsoft Internet Information Server, you can use objects that are intrinsic to the server, such as the server's Request and Response objects. You can also use components that are bundled with the server, but not an inherent part of it, such as the AdRotator component, the BrowserCapability component, and ActiveX Data Objects (ADO). Finally, you can use any other object that is registered on the server, including components that you create and register yourself.

Using server objects and components is a very powerful feature of Visual InterDev. When you create Web applications, you usually cannot control what browser the user has or what controls are registered on the user's computer. By installing controls on the server and using them in server script, you make the features of those controls available to any user, no matter what browser is in use.

When you are writing server script, you cannot directly manipulate client objects such as the browser window, an HTML form, Java applets, or DHTML objects, because these objects do not reside on the server.

If you are working with the server's intrinsic controls such as the Request or Response objects, you can simply reference the objects in your script, as in this example:

```
<%Server("starttime") = time%>
```

However, for all other objects, you create the object. You can do so by using an <OBJECT> tag in which you specify the attribute RUNAT=SERVER. Creating an <OBJECT> tag allows you to reference the object in any server script on the page, and adds the object and its members to the IntelliSense statement completion drop-down list. For example, the following creates an object reference to an ADO connection object:

```
<OBJECT RUNAT="Server" ID=cn PROGID="ADODB.Connection">
</OBJECT>
```

In your script, you can reference this object using the name you assigned in the ID attribute. The following statements use the object defined in the <OBJECT> element:

```
cn.Open Application("ConnectionString")
' other processing here
cn.Close
```

Alternatively, you can create objects with the CreateObject method of the Server object to create an instance of the object, as in the following statement:

```
<%
   Set Ad = Server.CreateObject("MSWC.Adrotator")
   Ad.GetAdvertisement("/ads/adrot.txt")
%>
```

The following table lists common server objects and components.

Object	Description
`Request`	Intrinsic IIS object that provides access to any information passed into the script through an HTTP request, such as form information, search strings, browser information, and information stored in cookies. For more information and examples of using the Request object, see "Gathering Information with HTML Forms," "Sharing Dynamic Information," and "Creating Portable Script" in Chapter 25, "Scripting with HTML Elements."
`Response`	Intrinsic IIS object that sends information to the user by writing information into a Web page stream. For more information and examples of using the Response object, see "Navigating Conditionally" and "Sharing Dynamic Information" in Chapter 25, "Scripting with HTML Elements."
`Session, Application`	Intrinsic IIS objects that allow you to set and get values that persist between pages in your Web application. For more details, see "Sharing Dynamic Information" in Chapter 25, "Scripting with HTML Elements."
`Server`	Intrinsic IIS object that allows you to create instances of objects that are registered on the server, including bundled components and objects that you create.
`AdRotator, BrowserCapability, TextStream, and NextLink`	Components that are bundled with IIS, and allow you to display a changing set of images, store and get information about specific browsers, read and write to text files, and create an ordered path through pages.
`ActiveX Data Objects (ADO)`	Bundled components that allow you to connect to and query databases.

The Script Debugging Process

You can use the Microsoft Visual InterDev debugger to test scripts written in Microsoft Visual Basic Scripting Edition (VBScript) and Microsoft JScript. Debugging Web pages can be different than debugging in traditional development environments in these ways:

- Most Web applications consist of scripts that run on the client (in .htm files) and on the server (in ASP files, the Global.asa file, and .cdf files).
- Scripts can be in different languages.
- Scripts can be mixed with HTML text.
- Many Web applications include not just scripts, but Java components such as applets and COM objects.

The Visual InterDev debugger allows you to debug in all of these scenarios. You can debug client script running in your local version of Microsoft Internet Explorer.

> **Note** It is highly recommended that you do not use Active Desktop mode of Microsoft Internet Explorer 4.0 when you are debugging.

To debug script running in Microsoft Internet Information Server (IIS) 4.0, you can run the debugger on your computer and attach it to a script running on the server. If the server is running on another computer, you can use *remote debugging* to debug script running there.

> **Note** For information about debugging Java components on your Web page, see the Java documentation on debugging.

Types of Errors

Debugging is about finding errors. When you work with script, you might encounter the following types of errors that require debugging:

- A *syntax error* occurs if you mistype a keyword, forget to close a multiline command (such as DO … LOOP), or introduce a similar syntax error. If a script includes a syntax error, the script will not execute and an error message is displayed as soon as the browser or server processes the page.

  > **Note** In some programming environments, syntax errors are referred to as preprocessor, compilation, or compile-time errors.

- A *run-time error* occurs when a command attempts to perform an action that is invalid. For example, a run-time error occurs if you try to perform a calculation using a variable that has not been initialized. If a run-time error occurs, the script either stops or performs an exception routine.
- A *logic error* occurs when a script executes without syntax or run-time errors, but the results are not what you intended. For example, a script might prompt the user for a password, but then allows access to the application even if the password is wrong.

Working in the Debugger

The basic process of debugging scripts consists of these tasks:

- Start debugging by running the document you are currently working with in your Web project, or by attaching the debugger to a document that is already running in a browser or on a server. You can also launch the debugger in response to a script error, which is called *just-in-time* debugging.
- Stop script execution by issuing a break command. You can also set a *breakpoint* in the script where the debugger will stop automatically. When you stop a procedure, its source is displayed.
- Inspect the state of the script by examining the values of variables or properties and the list of running procedures (the *call stack*).
- Control the execution of individual statements or procedures (*stepping*) and watch the effect both in your application and by watching the values of variables or properties.
- Skip over (*step over*) or walk through (*step into*) procedures called by the current procedure. If multiple procedures or threads are active, you can move to another one and proceed from there.

  Note You can't work with the debugger in Design view or Quick view of the HTML editor. To debug, switch to Source view.

To allow you to perform these tasks, the debugger includes these commands and windows:

- The Processes dialog box allows you to attach the debugger to a document that is already running in a browser or on a server, which can include scripts in pages that are not part of your Visual InterDev project. You can debug scripts running locally on your computer or attach to processes running on remote computers, such as an ASP file running on a server.
- The Running Documents window allows you to view a list of documents that are available to the debugger.
- Source view in the HTML editor displays source code of the script or component you are debugging. If the script or component you are debugging is part of your current Visual InterDev project, you can fix errors, and then rerun the document to test it again.

- In the Visual InterDev HTML editor, you can set and clear breakpoints. After the debugger has reached a breakpoint, you can use commands on the Debug menu to step into individual lines in your script. If you reach a point in your script that calls another procedure (a function, subroutine, or applet) you enter (*step into*) the procedure or run (*step over)* it and stop at the next line. At any point, you can jump to the end (*step out*) of the current procedure and proceed to the next breakpoint.
- In the Locals window you can see the values of variables within the scope of the current procedure. You can also specify that you want the debugger to display the values of specific expressions, such as properties, by setting up *watch expressions* in the Watch window.
- To set and change values, you use the Immediate window. You can evaluate any expression in the window and can enter script commands and see their effect. You can also view and change values in the Watch window.
- Using the Call Stack window, you can move to any currently active procedure.

For details about using debugger commands and windows, refer to the Visual InterDev online documentation.

Understanding Script Processing

Understanding how client scripts are processed and how errors are handled can help you debug client scripts successfully.

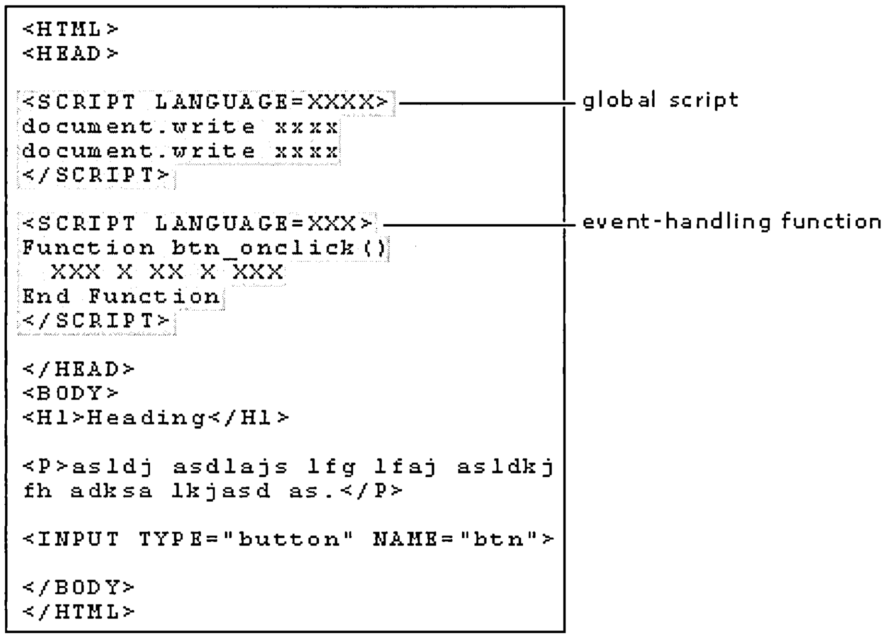
Processing Client Script

Client script is processed by Microsoft Internet Explorer. The browser calls the appropriate run-time module to process VBScript scripts or JScript scripts.

Client scripts are initially parsed when the Web document is loaded into the browser. During this parsing phase, the browser reports any syntax errors that it finds.

After parsing a section of script, the browser executes it. *Global* or *inline* scripts, which are scripts that are not part of an event-handling subroutine or function, are executed immediately. Event-handling subroutines or functions, and procedures that are called by other procedures, are parsed immediately but are not executed until triggered by an event or called by another procedure.

Client script processing



```
<HTML>
<HEAD>

<SCRIPT LANGUAGE=XXXX>
document.write xxxx
document.write xxxx
</SCRIPT>

<SCRIPT LANGUAGE=XXX>
Function btn_onclick()
  XXX X XX X XXX
End Function
</SCRIPT>

</HEAD>
<BODY>
<H1>Heading</H1>

<P>asldj asdlajs lfg lfaj asldkj
fh adksa lkjasd as.</P>

<INPUT TYPE="button" NAME="btn">

</BODY>
</HTML>
```

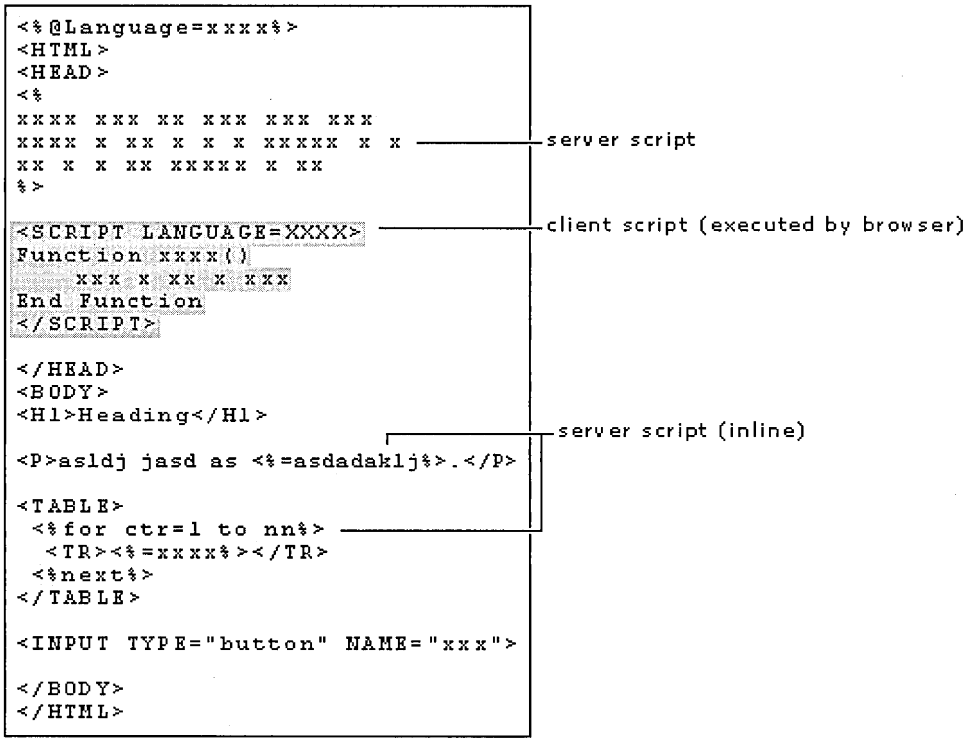
If a run-time error occurs when a client script is executed, an error message is displayed and the script containing the error stops. Other client scripts in the document can still run (unless you start the debugger). If the script containing the error is called again, the error message is displayed again.

Depending on the language you are using, you might be able to include statements in your scripts to trap run-time errors and run your own error procedures. For example, in VBScript, you can use the ON ERROR statement to establish error trapping. For more details, see the documentation for your scripting language.

Processing Server Script

Most server script is not event-driven. Instead, when an ASP file is requested, the server reads the page and processes all server script from top to bottom. This includes script that is inline with HTML text, as shown in the following diagram.

Server script processing



```
<%@Language=xxxx%>
<HTML>
<HEAD>
<%
xxxx xxx xx xxx xxx xxx
xxxx x xx x x x xxxxx x x
xx x x xx xxxxx x xx
%>

<SCRIPT LANGUAGE=XXXX>
Function xxxx()
    xxx x xx x xxx
End Function
</SCRIPT>

</HEAD>
<BODY>
<H1>Heading</H1>

<P>asldj jasd as <%=asdadaklj%>.</P>

<TABLE>
 <%for ctr=1 to nn%>
  <TR><%=xxxx%></TR>
 <%next%>
</TABLE>

<INPUT TYPE="button" NAME="xxx">

</BODY>
</HTML>
```

Not all server script is executed immediately. As with client script, server script can include functions and subroutines that are executed only when they are called from other procedures.

Global.asa files are a special case. The Application_OnStart and Session_OnStart procedures in these files are executed only once for an application and for a session. Therefore, to debug these events easily, you must embed debugging statements in the file. For details, see "Debugging a Global.asa File" in Chapter 26, "Debugging Your Pages."

CHAPTER 24

Scripting with Design-Time Controls and Script Objects

To create the user interface for your application, you can use the Microsoft Visual InterDev scripting object model and Visual InterDev design-time controls. The Visual InterDev design-time controls include standard user-interface elements — text box, label, check box, list box, command button, and so on.

When your Web pages run, design-time controls create script objects based on the scripting object model. You can add functionality for your application by writing scripts that work with these script objects.

Creating Forms with Design-Time Controls

One of the primary advantages of using the scripting object model is that it simplifies the script required to process the information in Web page forms. By using the scripting object model and design-time controls, you can create the user interface for your application in much the same way you create forms in Microsoft Visual Basic and Microsoft Access.

You can create a user-interface using the design-time controls listed in the following table.

Button	Textbox	Listbox
Label	Grid	Checkbox
OptionGroup	Recordset	RecordsetNavBar

Note Design-time controls are implemented using script stored in the Script Library. Do not alter the contents of the library, or the controls might not work properly.

Selecting a Target Platform

Before you add controls to your page, you must decide what platform you will be using for your application: server or client. Your choice determines how the design-time controls are created and where events occur. Use the following table as your guide.

Target platform	Result	Choose if
Server	• Your page is an .asp file. • Script objects are created in server script. Their properties, methods, and events are available only in server script. • Scripting object model events (such as onclick) are processed in server script. For more information, see "Writing Script for Script Objects" later in this chapter. • Data binding occurs on the server.	Your application will be used by a variety of browsers. Advantages include: • The application will run the same on all browsers. • You want to be able to control access to the application logic and database connection string information.
Client	• Your page can be either an .asp or .htm file. • Script objects are created using client script. Their properties, methods, and events are available only in server script. • Events are processed in client script. For more information, see "Writing Script for Script Objects" later in this chapter. • Data binding can occur on either the client or the server.	The application will be used only by browsers that support Dynamic HTML (DHTML) such as Microsoft Internet Explorer 4.0. The advantage is typically a better user experience. In addition, your application can use the DHTML object model.

It is important to understand that the target platform not only determines where a script object is created, but also how you can write script for it. For example, if the target platform for a control is Server, its corresponding script object is created in server script. You can only access the script object's properties, methods, and events in server script — the script object is not available to client script. The reverse is also true: if a control's target platform is Client, the properties, events, and methods of the corresponding script object are not available in server script.

Choosing a target platform does not mean that all design-time controls generate script for that platform. The scripting platform for an individual control, and for individual scripts, will depend on how the control is used. For example, the target platform for a Save button in a form might be Server so that the script can save information to a database. But another script, even for the same button, might run on the client in order to validate the data before it is saved.

You can choose a target platform for an individual page or as the default for your project.

To specify a target platform for a page

1. Switch to **Source** view.
2. Right-click the page and choose **Properties**.
3. In the **Property Pages** window, choose the **General** tab.
4. Under **DTC scripting platform**, choose **Server** or **Client**.

   Note If you change target platforms, existing event-handling scripts are not automatically changed to reflect the new setting, but you can do this manually. For example, when changing the target from Server to Client, event handlers in server <SCRIPT RUNAT="Server"> blocks need to be moved to client <SCRIPT> blocks. For details, see "Writing Script for Script Objects" later in this chapter.

If you know you want to use the same platform all the time, you can set the default target for a project. All pages you create after that will reflect your default setting.

To specify a target platform for a project

1. In the Project Explorer, right-click the project, and then choose **Properties**.
2. In the **Property Pages** window, choose the **Editor Defaults** tab.
3. Under **DTC scripting platform**, choose **Server** or **Client**.

   Note If you already have pages with the design-time controls on them, changing the project default does not change the settings for those pages.

By default, all design-time controls on the page inherit the target platform from the current page. However, you can change the target platform for an individual control. For example, some of the design-time controls on your page might be bound to a database and use the server as a target. Others might not be data-bound and use the client as a target.

Note Data-bound design-time controls must use the same target platform as the recordset object that they are bound to.

To specify a target platform for an individual control

1. Switch to **Source** view.
2. Right-click the control, and then choose **Properties**.
3. In the **Property Pages** window, choose the **General** tab.
4. Under **Scripting Platform**, select **Client** or **Server**.

Adding Controls to a Page

You add design-time controls to the page by dragging them and then setting their properties.

To add design-time controls

1. Make sure that you have set options to view controls graphically. From the **View** menu, choose **View Controls Graphically**.

   Note To set this option as default, use the **HTML** node of the **Options** dialog box.

2. Drag the design-time control from the **Design-Time Controls** tab of the Toolbox to your page.

   If the target platform for a page is Server and the scripting object model is not already enabled for the page, you are prompted to enable it now. You must enable the scripting object model for design-time controls to work properly.

   Note Do not define forms using the HTML <FORM> tag. The scripting object model framework creates a form for you.

3. Right-click the control, and then choose **Properties**. The custom **Properties** window for that control appears.

4. Set options for the control as required.

   Note When you print a page in the HTML editor, some design-time controls might not be displayed with the values you set. If you put more than one instance of a design-time control on a page — for example, if you put multiple Textbox or Button controls on a page — the second and subsequent ones will display their default values in the printout.

By default, design-time controls are displayed using their graphical representation in both Design view and Source view. For example, the Button design-time control appears as a button.

You can choose to view the controls as text. However, while the control is in text mode, it cannot communicate with other controls on the page, which can cause the control not to function properly.

Note ASP pages do not render properly in Quick view, because Quick view does not run server script. Therefore, Quick view does not allow you to preview script objects whose target platform is server.

You can cut, copy, and paste design-time controls as well. However, you can do this successfully only if you copy the graphical version of a control, not the text version.

To make it easier for you to specify options for design-time controls, you can set their properties using custom property pages.

To set properties for design-time controls

- Right-click the control, and then choose **Properties.**

  – or –

  In the Properties grid, move to **(Custom)** and then click the button in the value field for that property.

Each design-time control supports different options. For details about the properties you can set for each, press F1 in the Property Pages dialog box.

The design-time controls generate script in the page to implement the corresponding script object. You can preview the generated text for a control.

To preview generated text

1. Switch to **Source** view.
2. Right-click the design-time control, and then choose **Show Run-time Text.**

You can permanently convert a design-time control to run-time text, which leaves the script required to create the script object, but strips the information used to communicate with other controls and to display the control graphically.

You might do this if you wanted to customize the generated text, but this is not the recommended way to work with design-time controls.

To convert a design-time control permanently to text

- In Source view, right-click the design-time control, and then choose **Convert to Run-time Text.**

  Warning Converting to run-time text is a one-way operation. You cannot undo the operation to return to a graphical view. You should make this conversion only if you are comfortable customizing the control's run-time text.

Creating a Form with Design-Time Controls

When you use the scripting object model and design-time controls, you do not create forms using the HTML <FORM> tag. Instead, you drag the design-time controls that you need onto the page.

Tip An easy way to assemble a form with design-time controls is to use the FormManager. For details, see "Simplifying Data Entry Pages" in Chapter 5, "Walkthroughs," and "Creating Event-Driven Forms" in Chapter 3, "Database Basics."

To define specific behavior for individual controls, you can write event handlers in script. Rather than adding an HTML Submit button, you can add a Save button that calls a forms-processing method, as in the following VBScript example:

```
Sub btnSave_onclick
   ProcessForm()
End sub
```

The forms-handling procedure can be anywhere else on the page. It can extract values from the form by requesting value of the controls on the page. The following example tests the value of a text box called txtName:

```
Sub ProcessForm()
   If txtName.value = "" then
      txtName.value = "Please enter a name!"
   Else
      ' etc.
   End if
End sub
```

Writing Script for Script Objects

One of the primary benefits of using the scripting object model and design-time controls is the ease of writing scripts that define your application's behavior. The scripting object model makes your Web pages work like forms in Microsoft Visual Basic or Microsoft Access. After dragging design-time controls onto the page, you can use familiar scripting techniques to set their properties and write handlers for events.

However, there are some differences between working with the scripting object model and working with other environments. In the sections that follow, you will find information about how to write script for script objects, including information about where you need to be aware of differences from other environments.

Writing Script Appropriate for the Target Platform

The target platform specifies where scripts run, and therefore dictates what your scripts can do. When the target platform is server, you can use the scripting object model and the ASP programming model, including Microsoft Internet Information Server (IIS) objects. Conversely, if your target platform is client, the scripting object model extends the document object model provided by Dynamic HTML (DHTML).

As you write script, you must be clear where it will run, so that you do not attempt procedures that are not appropriate for the context. For example, if your target platform is server, do not try to display messages directly to the user with functions like MsgBox or with the alert method. Even if the functions would work properly (generally, they result in an error), they would display the message on the server rather than to the user.

Changing Target Platforms

If the target platform is server, the RUNAT attribute of the script block will be set to SERVER. If the target platform is client, there is no RUNAT attribute.

If you change target platforms for a page, you must manually add or remove the RUNAT attribute of affected script blocks. For example, if you initially added design-time controls to a page when the target platform was set to client, a script block might look like this:

```
<SCRIPT LANGUAGE="VBScript">
```

If you then change the target platform to server, you must add a RUNAT attribute, as in this example:

```
<SCRIPT LANGUAGE="VBScript" RUNAT=SERVER>
```

This might not be the only adjustment required. Other platform-specific features to be aware of include:

- Client script can display messages to the user. If you switch platforms to server, you must remove or replace MsgBox or alert calls.
- Client script can reference objects from the DHTML object model, including window, document, and others. If your script uses these objects, you must alter it when changing to the server platform.
- Server script must be written in the default page language (the language specified by the @ LANGUAGE=attribute at the top of the page) or must include a LANGUAGE attribute in the <SCRIPT> block.
- Server script cannot access return values for certain calls, such as the onkeyup event.

Testing Script Objects

Test your page in the right place. Quick View in the HTML editor runs locally, so it does not process server script. Therefore, if your target platform is server, no script objects will appear in Quick View. Instead, use the browser to view the page from the Web server.

To test using the browser

- In the Project Explorer, right-click the name of the file to test, and then choose **View in Browser**.

Referencing Script Objects and Properties

In your script, you can reference a script object using the name that you assigned to it when you created the design-time control. You read and write most properties as usual, by adding their name onto the object reference with a dot (.).

For example, if you have dragged a Textbox design-time control onto the page and named it txtName, you can read its current contents by getting the value of its value property, as in the following:

```
<SCRIPT LANGUAGE="JAVASCRIPT">
function getLName
{
   fname = txtName.value;
}
</SCRIPT>
```

Note Object and property names are case-sensitive in JScript and JavaScript.

By default, when you type in the name of a script object followed by a dot (.), the IntelliSense popup will display all the properties and methods appropriate for the script object you are working with.

When the scripting object model is enabled, you can reference the current page using the object name thisPage, or if you have specified it as a page object, by its page object name. For example, the following statement sets the value of the current page's cancelEvent property:

```
thisPage.cancelEvent = True
```

Note The thisPage object does not appear in the Script Outline window or IntelliSense drop-down unless you add a PageObject design-time control to the page. However, the thisPage object is available at run time and you can write scripts against it even if you have not used a PageObject control.

Some values are accessible as pairs of methods: a get method to get the property's value, and a set method to write it. For example, you can check the state of a Checkbox script object by calling its getChecked method, and set it using the setChecked method. The following example illustrates this type of property:

```
Function toggleCheckBox
   If checkbox1.getChecked() then
      checkbox1.setChecked(0)
   Else
      checkbox1.setChecked(1)
   End if
End Function
```

For a list of properties supported by each script object, see the topic "Script Objects" in the Visual InterDev online documentation.

Calling Methods and Functions

You call methods for script objects the way you do with any object. For example, if you have dragged a Listbox design-time control onto the page, you can populate it by calling its addItem method, as in the following script:

```
<SCRIPT LANGUAGE="VBSCRIPT">
Function LoadListBox()
   ListBox1.additem "Paris"
   ListBox1.additem "London"
   ListBox1.additem "Cairo"
End Function
</SCRIPT>
```

Note Method names are case-sensitive in JScript and JavaScript.

Responding to Script Object Events

Each script object can generate a predetermined (or *implicit*) set of events. For example, the script object for a Button design-time control can generate a click event, and the script object for a Textbox design-time control can generate an onchange event.

Note You can also extend the set of events that an object can respond to so that it can make use of any events that the browser generates. For details, see "Extending Events for an Object" later in this chapter.

To write a handler for a script object, create a procedure using the object's name and the event to handle. You can write event handlers in any scripting language supported by the browsers used for your application.

For example, if you create a button called btnDisplay, you can write a handler for its onclick event that might look like this in VBScript:

```
<SCRIPT LANGUAGE="VBSCRIPT">
Sub btnDisplay_onclick()
   TextBox1.value = "Button has been clicked"
End Sub
</SCRIPT>
```

Tip Use the Script Outline window to create event handlers. In the Script Outline window, expand the node for the object you are working with, and then double-click the name of the event you want to write a handler for. For details, see the topic "Script Outline Window" in the Visual InterDev online documentation.

Note Script object events occur only in response to a user action, not in response to programmatic changes. For example, if a user selects an item in a Listbox script object, the onchange event is fired. However, if you change the selection using a script statement, no event is fired.

An important feature of design-time controls is that you write event handlers for script objects in either client or server script, according to the target scripting platform. When a user clicks a button, the event actually occurs on the client. However, if your target scripting platform is server, you write a handler in server script that responds to that event as if it had occurred on the server.

In reality, if your scripting platform is server, the scripting object model wraps your page in a form. To process events in server script, the scripting object model performs a round trip to the server.

First, the event is captured and that, along with information such as other script object values, is sent as part of the post. Then at the server, an additional scripting object model procedure determines that it is processing an event and calls your event handler. When the server script is finished, the refreshed page is sent back to the browser. A similar round trip is performed for every event processed in server script.

For the most part, this process is invisible to the user and to your scripts, with these exceptions:

- Processing events in server script is slower than doing so in client script, because it involves a round trip to the server.
- You must be aware of where the event handler script is running, so that you do not attempt procedures that do not work on the server or on the client.

Extending Events for an Object

Each scripting object has a predetermined, or implicit, set of events that it can respond to. However, you might want to take advantage of other events that the browser generates and use them with your script objects.

Most typically, if you are using design-time controls and your target scripting platform is client, you might want to take advantage of the events available in the DHMTL document object model.

For example, the textbox script object supports an implicit onchange event that you can write handlers for. However, on the client you might also want to write events for the onkeypress event and others.

You can extend the set of events available to an object by *advising* for an event, or registering the object to be notified when the event occurs. After you have advised for an event, you can write event handlers for that event (for that object) as you would for any other event. You can cancel event notification by unadvising for the event.

Each object supports an advise method that allows you to register a specific event. When you advise, you specify the name of the event and the name of a function that will be called when the event occurs — in effect, the name of the event handler for the event.

> **Tip** In general, advising for events is practical only when your scripting target platform is client. If your platform is server and you advise for an event such as onkeypress, you will cause a round trip to the server each time the event occurs (in this case, with each keystroke).

You can advise and unadvise for an event at any time. A common time to do so is when a page is loaded. For client scripting targets, you create a handler for the window object's onload event and call the advise method there.

The following example shows how you would advise at page initialization time to have a DHTML onkeypress event sent to the text box called Textbox1. When the onkeypress event fires for Textbox1, it will call the function checkkeys. The result of the advise method is an advise object called objAdviseTextbox1 that you can use later if you need to unadvise for the event.

```
<SCRIPT LANGUAGE="VBScript">
Function window_onload()
   objAdviseTextbox1 = Textbox1.advise("onkeypress", "checkkeys()")
End function
</SCRIPT>
```

The function you specify in the advise method works like any event handler. If the event passes parameters, you can get those using the DHTML window object's event method. The following shows the handler for the onkeypress event in the previous example. It examines each keystroke that occurs in the Textbox1 object and copies only the numbers to the object Textbox2.

```
Function checkkeys()
   character = Chr(window.event.keycode)
   If character >= "0" and character <= "9" then
      Textbox2.value = Textbox2.value & character
   End if
End function
```

When you no longer need notification of the event, cancel it by calling the object's unadvise method. The unadvise method requires the advise object returned by the advise method as well as the name of the event. The following shows an example of calling unadvise:

```
Textbox1.unadvise("onkeypress", objAdviseTextbox1)
```

Trapping Events on the Client

If the target platform for a page is server, event handlers are executed as server script. In some instances, however, you might want to trap the event — handle the event first in client script — before it is passed to the server. For example, you might want to trap a Save button onclick event in client script in order to validate user information before it is sent to the server.

Trapping events is useful only if the target platform is server. If events are being processed on the client already, there is no need to trap the event, as it will not be passed to the server.

To trap an event on the server

- Write a JavaScript handler for the current page's onbeforeserverevent event. This event is fired just before the event is forwarded to the server. The syntax of the handler is as follows:

  ```
  function thisPage_onbeforeserverevent( objectName, eventName )
  ```

  The *objectName* parameter contains the name of the object that fired the event, and the *eventName* parameter contains the name of the event that is being forwarded to the server.

In your handler, you can cancel the event (prevent if from being forwarded to the server).

To cancel an event on the server

- Set the cancelEvent property to true.

The following is an example of how to trap the button click event for a Delete button. The script prompts the user to confirm the deletion before proceeding.

```
<SCRIPT LANGUAGE="Javascript">
function thisPage_onbeforeserverevent( obj, event ){
if (obj=="btnDelete"){
      if(evnt=="onclick"){
         if (confirm("Are you sure you want to Delete?")){
            alert("Deleted per your request");
         }
         else {
            alert("Delete cancelled");
            thisPage.cancelEvent = true;
         }
      }
   }
}
</SCRIPT>
```

Converting Parameter Values

Some event handlers and methods receive parameters with the event call. If your target scripting platform is server, and if a procedure call is the result of a round trip to the server, parameter data types are converted to strings. A round trip occurs when:

- An action on the client (such as a button click) is processed in server script.
- Procedure in client script calls a method that is implemented in server script.
- You call a page object method on another page. For details, see "Extending the Scripting Object Model Across Pages" in the next section.
- A client makes a remote scripting call to an ASP page. For details, see "Executing Server Script Remotely" later in this chapter.

In any of these cases, the procedure that receives the parameter must convert it to the appropriate data type as necessary. For example, Boolean values are converted to the strings "true" or "false." If you write a procedure that receives a Boolean parameter, you should test it using script such as the following:

```
Sub TestValue( boolFlag )
   dim flag
   If boolFlag = "true" then
      flag = True
   Else
      flag = False
   End If
   If flag then
      ' processing here
   End If
End Sub
```

A Script Object Example

The following example shows a complete script block that includes properties, methods, and events. The page is a simple calculator with two text boxes, named txtNumber1 and txtNumber2, into which the user types numbers. The user selects an operator from lstOperators. When the user clicks btnCalculate, the result of the calculation is shown in lblResult as follows:

```
<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>
Sub thisPage_onenter()
   If thisPage.firstEntered then
      txtNumber1.value = ""
      txtNumber2.value = ""
      lstOperators.addItem "+", 10
      lstOperators.addItem "-", 20
      lstOperators.addItem "÷", 30
      lstOperators.addItem "x", 40
      lstOperators.selectByValue(10)
   End if
End Sub
Sub btnCalculate_onclick
   Dim Result
   Dim Value1, Value2
   Value1 = CInt(txtNumber1.value)
   Value2 = CInt(txtNumber2.value)
   Select Case CInt(lstOperators.getValue())
   Case 10:
      Result = Value1 + Value2
      Operation = " plus "
   Case 20:
      Result = Value1 - Value2
      Operation = " minus "
   Case 30:
      Result = Value1 / Value2
      Operation = " divided by "
   Case 40:
      Result = Value1 * Value2
      Operation = " times "
   End Select
   lblResult.setCaption(Value1 & Operation & Value2 & " is " & Result)
End Sub
</SCRIPT>
```

Extending the Scripting Object Model Across Pages

Using design-time controls and the scripting object model allows you to create and script a Web page using standard object-oriented techniques. However, Web applications differ from other development environments in an essential way: they are usually constructed out of a collection of pages.

In a simple application, you can simply add links to other pages. When the user clicks a link, the browser follows the link, reads the page, and displays it. More complex applications, however, use more sophisticated linking techniques. For example, your application might require links that:

- Jump conditionally to different pages.
- Process a specific script on another page when the user jumps to it.
- Navigate to pages that provide server processing (such as database lookups) for your applications.

The scripting object model extends across pages to help you design applications with these types of requirements. You can use the scripting object model to designate pages as *page objects*.

A page object is an ASP page that contains server script that you use in your application. The procedures — functions or subroutines — on the page can become methods for the page object.

For example, you might have an ASP page in your application that you call from a form to display a list of employees. Procedures on the target page construct different queries for displaying the list — in different order, using different fields, or with different selection criteria. When you convert the page to a page object, you can specify each of these procedures as a method. You can then invoke the methods from other pages in your application.

Page objects also allow you to create properties, which maintain state over multiple round trips to the server.

Page objects give you:

- Simplified navigation. You can navigate to other pages in your application using standard object references, without having to track the URL of the page.
- An easy way to execute specific script on another page. By exporting procedures as methods, you can jump directly to a specific procedure on another page without writing script to parse hidden form elements or query strings.
- A means of maintaining state information. You can define properties on a page object that maintain their value for the duration that you specify — page lifetime, session, or application.
- A way to execute server script from a page displayed in the browser.

The basic steps in creating a page object are these:

- Specify a page as a page object.
- Export procedures on the page as methods.
- Define the properties that the page object will support.
- Reference any other page objects that you will use in your scripts, on that page and in other pages.

Specifying a Page as an Object

You can specify any ASP page as a page object. To do so, you use the Page Object design-time control.

> **Note** Page objects are implemented using script stored in the Script Library. Do not alter the contents of the library, or page objects might not work properly.

To specify a page as an object

1. Create or open an .asp file in the HTML editor.
2. Enable the scripting object model for the page.

   Make sure that you have set options to view controls graphically. From the **View** menu, choose **View Controls Graphically**. To set this option as the default, use the **HTML** node of the Options dialog box.
3. From the **Design-Time Controls** tab of the Toolbox, drag a PageObject control onto your page. You can drag the control anywhere on the page, although it must be inside the framework of the scripting object model blocks.
4. In the **Name** box on the PageObject control, type a name for the page object. This will be the name that you can use to reference the object in script.

The name you give your page object is registered in your Microsoft Visual InterDev project so that it is available to any other page. Even if you move the page to another location, its page object name remains the same.

Exporting Procedures as Methods

One of the primary benefits of using page objects is that you can expose your application's tasks as methods that you can easily call from script. Doing so greatly simplifies the organization of your code and eliminates the time and effort required to transfer variables between pages, dispatch to the correct routine, and so on.

Page objects support two types of methods:

- *Navigate methods* are called by a client page that wants to jump to the ASP page and run a procedure there, and then perhaps jump somewhere else. A common use for navigate methods is to process a form.
- *Execute methods* are called by a client page that wants to use remote scripting to execute a script function. A common use for execute methods is to validate a user-entered value by looking it up in a database, or to perform a database query to populate a control on a page.

All page objects have a default method called `show()`, which displays the contents of the page. This is the method that is called after any other methods on the page have been processed.

To create a method for a page object

1. If the page does not already have one, add a PageObject control to the page and give the PageObject control a name.
2. Write the procedures (functions or subroutines) in a script block that has the attribute RUNAT=SERVER. The procedure can take any number of parameters, but all are passed by value.

   For example, the following function creates a query string:

   ```
   <SCRIPT LANGUAGE="VBScript" RUNAT="Server">
   Sub ListEmployees(sortOrder)
      sqlText = "SELECT lname, fname FROM Employees"
      sqlText = sqlText & " ORDER BY " & sortOrder
      DoQuery(sqlText)
   End sub
   </SCRIPT>
   ```

   Note Parameters are converted to strings when you call a page object method so that they can be successfully passed across the Web. In your page object scripts, you should convert parameter values to the appropriate data type as required.
3. Right-click the PageObject control, and then choose **Properties** to display the **Property Pages** dialog box.
4. Determine whether the method will be available via navigation or execution. Then in the list under either **Navigate methods** or **Execute methods**, find the first blank line. From the drop-down list, select the procedure that you want to expose as a method.

   Note It is possible to expose the same method as both a navigate and execute method. However, the method's underlying procedure must be written to accommodate both types of calls.

Defining Properties for a Page Object

Page objects allow you to expose properties, which are essentially global variables. You can scope a property to three levels: page, session, and application.

- **Page-scope properties** make a property available to scripts anywhere on the page until you navigate to another page.
- **Session-scope properties** are available to any page you navigate to in your project. Session values use IIS session variables to store values.
- **Application-scope properties** are available to any user of your application. Application values use IIS application variables to store values.

You also specify whether properties are available in client scripts, server scripts, or both. If you specify a property as a client property, you can further specify that it be either read/write or just read-only. Server properties are always read/write.

To create a property for a page object

1. If the page does not already have one, add a PageObject control to the page and give the PageObject control a name.
2. Right-click the PageObject control, choose **Properties** to display the **Property Pages** dialog box, and then choose the **Properties** tab.
3. In the **Name** column, find the first blank line, and then enter the name of the property you want to create.
4. Select the characteristics for the new property from the remaining columns:
   - **Lifetime** Select whether the property will be available until the page is exited, or as a session or application value.
   - **Client** Select whether the property will be read-only or read/write in client script. If you choose **None**, the property will not be available in client script.
   - **Server** Select **Read/Write** to make the property available in server script, or **None** if it will not be available.

To make properties accessible to your scripts, page objects implement get( ) and set( ) methods for them. For example, if you define a property called Color, you can read its value using the method getColor( ) and set it using the method setColor( ). For more details, see "Accessing Page Object Properties" later in this chapter.

Referencing Other Pages

You can call methods and use properties on the current page using the default page object name of thisPage. However, if you want to access the methods or properties of another page object, you must first create a reference to that page on the current page.

To reference another page object

1. If the page does not already have one, add a PageObject control to the page and give the PageObject control a name. If your scripting target is Server, the scripting object model must be enabled for the page.
2. Right-click the PageObject control, choose **Properties** to display the **Property Pages** dialog box, and then choose the **References** tab.
3. In the **Name** column, click the three-dot button to display the **Create URL** dialog box.
4. Select the .asp file that you want to reference as a page object. Enter options for how the page object should be called, and then click **OK**. For assistance with the options, press F1 in the **Create URL** dialog box.

Calling Page Object Methods

You can call methods on page objects in two ways: by navigation and by execution. You call a method by navigation when you are finished with one page and want to jump to another page and process script there. This is similar in HTML scripting to navigating to a URL: a one-way jump, except that using a page object saves the current page's state before navigating. For example, you might collect query parameters from the user on one page, and then jump to another page where you actually create and execute the query.

In contrast, calling methods by execution is similar to a familiar object-oriented method call — you call script somewhere else that performs a task and then returns. However, methods called by execution run asynchronously. The page that calls the method remains in the browser and the user can continue to work with it.

You can call methods by execution only if you are running client script and the method to be called is on a server page. For example, while the user is filling in the form, a client script might call a method on a server page to perform a database lookup.

To call a page object method

1. Create a reference on the current page to the page object you want to use. For details, see "Referencing Other Pages" earlier in this chapter.
2. In your script, call the page object method using one of these types of calls:

   ```
   pageObject.navigate.methodName(parameters)
   pageObject.execute.methodName(parameters)
   ```

   Where:

   - *pageObject* is a reference you created earlier to a page object.
   - navigate and execute are child objects of the page object that determine how the method call will be processed.
   - *methodName* is the name of the method on pageObject that you want to call.
   - *parameters* is a list of parameters you are passing to the method. All parameters are passed by value.

     Note Parameters are converted to strings when you call a page object method so that they can be successfully passed across the Web. In your page object scripts, you should convert parameter values to the appropriate data type as required.

The following script shows some simple form processing. When the user clicks a List Now button on the client page, the onclick handler for the button extracts the values of a list box on the page, and then calls a method on the page object poListEmployees for processing. In this instance, the user information is passed as a parameter to the page object's CreateList method.

```
<SCRIPT LANGUAGE="VBScript">
Sub btnListNow_onclick()
   department = lstDept.gettext()
   poListEmployees.navigate.CreateList(department)
End Sub
</SCRIPT>
```

Accessing Page Object Properties

When you define a property for a page object, the scripting object model creates a get( ) method and a set( ) method that you use to access the property. For example, if you have defined a property called UserName, you can read the value of the property using the method getUserName, and set it using setUserName, as shown in the following example:

```
newUser = PageObj1.Navigate.getUserName()
PageObj1.Navigate.setUserName(txtUserName.Value)
```

When working with properties, you need to be aware of their lifetime. If you have defined the property's lifetime as "page," for example, you can get and set its value only until you leave the page and display another one. (Calling the same page again to execute a method retains property values scoped to the page.) However, after you navigate to another page, the property is reset.

Executing Server Script Remotely

If you have created ASP pages as page objects, you can call methods on those pages remotely — that is, you can execute a method on an ASP page while a client page is loaded in a browser and without navigating away from the client page.

Because you do not leave the client page, its values are preserved and client scripts can continue processing without complex strategies to save values between pages. In the meantime, the server script can execute server-appropriate procedures, such as database lookups. When a procedure is finished, the server can send just the results to the client, rather than reformatting and resending an entire page. This reduces server load.

You can execute server script two ways:

- **Synchronously** Your script calls the remote procedure and waits for it to return. This is useful if you need the results of the remote procedure before you proceed.
- **Asynchronously** Your script makes the call to a remote script, and then continues processing. The page remains available for users to work with. Asynchronous calls are useful in Web applications because a remote procedure can take a long time while the request goes to the server and back.

Making Remote Procedure Calls Synchronously

Remote scripting uses the technology of ASP page objects. The page to be called is an ASP page object on which you have exposed methods. The client page, which can be either an .htm file or .asp file, contains client scripts that call the page object's methods.

To make remote scripting calls from a client page to a server page, you use the page object's execute child object. The execute child object does not return a single value from the method you call. Instead, it returns a *call object*, which is an object containing return and status information about the called procedure.

The most commonly used property is the call object's return_value property, which contains the single value calculated or looked up by the remote procedure. Other call object properties allow you to get more information about the state of the remote procedure call.

To make a synchronous remote procedure call

1. Create a reference on the current page to the page object you want to use. For details, see "Referencing Other Pages" in "Extending the Scripting Object Model Across Pages" earlier in this chapter.
2. In your client script, call the procedure using syntax such as this:

   pageObject.execute.*methodName*

   Note Remote scripting is implemented using script stored in the Script Library. Do not alter the contents of the library, or remote scripting might not work properly.

For example, the following function is called in client script to validate a credit card. To perform the validation, it calls the method validate in the page object poSignIn. The value returned by the remote procedure is available in the return_value property of the call object valid.

```
<SCRIPT LANGUAGE="JAVASCRIPT">
function checkCreditCard(){
   retObj = poSignIn.execute.validate(txtCC.value);
   if (retObj.return_value == "OK"){
      alert("Accepted");
      }
      else{
      alert("Rejected");
   }
}
</SCRIPT>
```

Calling Methods Asynchronously

When you call a remote scripting procedure asynchronously, you must include extra processing in the calling script to determine when the remote call has finished. To do this, you specify a *callback procedure*, which is a function that is called when the remote procedure has finished.

To use a callback function, you create an additional function in the client page to process the results of the remote script. You can optionally also create an error handling process in client script that can be called if the remote script encounters an error condition.

To make an asynchronous remote procedure call

- When calling a method using the `execute` object, include the name of a callback function using syntax such as this:

  ```
  co = PageObject.execute.Method(p1, p2, callBack)
  ```
  where:

  co A call object.

  p1, p2 Any parameters required by the called procedure. You can pass as many parameters as required.

  callBack The name of a method that will be called when the remote procedure has finished. When the remote method has finished, it jumps immediately to the process you have specified.

For example, the following illustrates a button whose onclick attribute specifies that it should call a remote procedure for validating a credit card. The call to the remote procedure specifies the function displayResults as its callback.

```
<BUTTON
   ID=""btnValidate"
   TYPE="button"
   LANGUAGE="JavaScript"
   ONCLICK="poSignIn.execute.validate(txtCC.value, displayResults) ">
   "Validate Credit Card"
</BUTTON>
```

The displayResults function accepts the remote procedure call object as a parameter and tests it to determine whether the credit card was valid.

```
<SCRIPT LANGUAGE="JavaScript">
function displayResults(retObj)
{
   if (retObj.return_value == "OK"){
      alert ("Your order has been accepted. ");
   }
   else{
      alert("Invalid credit card number, please re-enter. ");
   }
}
</SCRIPT>
```

Note You can also use remote scripting calls outside the scope of the page object's execute method. For details, and for more information about remote scripting, see the Microsoft Scripting Web site at http://www.microsoft.com/scripting/.

CHAPTER 25

Scripting with HTML Elements

Even if you do not use design-time controls in your Web page, you can create fully-functional Web applications by creating and scripting HTML elements. Doing so is more involved than using design-time controls, but might be useful if you prefer scripting HTML elements yourself or if you are working with pages that might be shared with an application other than Microsoft Visual InterDev.

Creating script for specific tasks usually means that you must understand how different elements of Web pages fit together, and how the browser (or client) and server interact. It is also important to understand the capabilities of different scripting environments — for example, knowing whether the script will run on certain browsers.

Choosing a Scripting Language

Microsoft Visual InterDev allows you to design Web applications using the scripting language you are most comfortable with. Your application can contain a mix of files that use VBScript and JScript. You can even use different languages on the same page, but the script inside a single script block must contain a single scripting language.

To set the scripting language for script block

- In the <SCRIPT> block, set the LANGUAGE attribute to the language you want to use for that script, as in the following example:

```
<SCRIPT LANGUAGE="VBSCript">
   [some scripting statements here]
</SCRIPT>
```

If you do not set a language for a script block, a default is assumed. The default for server script run by Microsoft Internet Information Server (IIS) is established as a server setting (usually VBScript). For client script, the default language is determined by the browser. Most browsers use JavaScript as their default language. Microsoft Internet Explorer assumes JavaScript as its default language, but will reset the default language for the current page to the first LANGUAGE attribute it encounters in a <SCRIPT> block. (It does not reset the default again after the first LANGUAGE attribute.)

Choosing a Language for ASP Files

Because ASP pages can contain inline script that is not in a <SCRIPT> block, they provide a means for resetting the default for the page as a whole. At the top of each ASP page, a <%@ LANGUAGE= %> directive specifies the default scripting language. You can change this setting manually or by setting a property value.

To change the scripting language for an ASP file

1. With nothing selected in the editor, choose **Property Pages** from the **View** menu.
2. In the **Property Pages** dialog box, choose the **General** tab, and then under **Default scripting language**, choose the language in the **Server** list.

Design-time controls also use the LANGUAGE directive to know what language they should use to generate script. If you delete the <%@ LANGUAGE= %> directive in a file, the Visual Interdev controls generate script using the language currently specified in the Properties dialog box for the page. In addition, when these pages are run on the Web server, they will be processed using the language set in the DefaultScriptLanguage parameter under the ASP entry in the Web server registry.

> **Note** If you have pages without the <% @ LANGUAGE= %> directive, make sure your language setting in the Options dialog box is identical to the default language set on your server; otherwise, the controls will generate code in the wrong language.

You can set the default for all new ASP pages you create.

To reset the default scripting language for new ASP pages

1. Right-click a project in the Project Explorer, and then choose **Properties.**
2. In the **Property Pages** dialog box, choose the **Editor Defaults** tab.
3. Under **Default script language**, select the language you want in the **Server** list.

If you change the default scripting language, new files you create will reflect the change, but the specified scripting language in your existing files will not change.

Choosing a Language for the Global.asa file

When you create a new project, the default script language you specify in the Options dialog box is used to define the events in the Global.asa file. You can change the language used for those events.

To change the script language for the Global.asa file

1. In the editor, open the Global.asa file.
2. In each script block, change the LANGUAGE attribute.

Choosing a Language for Generated Scripts

If you write scripts by hand, you can set the scripting language yourself using the procedures above. However, you can also set a default scripting language for script that you generate using the Script Outline window.

To change the scripting language for generated scripts

1. With nothing selected in the editor, choose **Property Pages** from the **View** menu.
2. In the **Property Pages** dialog box, choose the **General** tab.
3. Under **Default scripting language**, choose the default language for client and server scripts.

   Note This change affects only new scripts, not existing ones.

You can also change the default language for script generation for your project as a whole.

To reset the default scripting language for new ASP pages

1. Right-click a project in the Project Explorer, and then choose **Properties.**
2. In the **Property Pages** dialog box, choose the **Editor Defaults** tab.
3. Under **Default script language**, select the default language for client and server scripts.

   Note This change affects only new pages, not existing ones.

Handling Events with HTML Elements

You can specify how HTML elements on a page behave by writing script that responds to events. For example, you can write client script that initializes variables when a document is loaded (the window object's onload event), when a user enters text into a text box (the text box's onchange event), or when a user clicks a button (the button's onclick event).

> **Note** The easiest way to add objects to your application and write script for them is to use Microsoft Visual Interdev design-time controls. For details, see "Creating Forms with Design-Time Controls" and "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

You write event handlers as procedures (functions or subroutines) that appear in <SCRIPT> blocks in your page. To specify that the procedure is a handler for a specific event, you can:

- Create an implicit handler by naming the procedure after the event that it is a handler for.
- Explicitly relate the procedure to the event using object or script attributes.

If you are using Microsoft Visual Basic, Scripting Edition, you can create the procedure as either a subroutine or function. You would use a function if you want to return or set a value for the procedure, which you do in some instances to cancel the effect of the event.

Creating Implicit Handlers

An implicit handler uses a naming convention to link the handler to the object and event.

To create an implicit handler

1. In the **Script Outline** window, expand the node containing the object you want to script.

   Note Objects appear in the Script Outline window only if their ID or NAME attribute is set. Objects inside a form appear only if the form's ID or NAME attribute is set.

2. Expand the name of the object.
3. Double-click the name of the event you want to write a handler for.

   The editor creates a skeleton handler for you in this format:

   ```
   function objectName_event
   ```

   For example, if your page contains a form that has a button called btnNext, the handler will be created with the name btnNext_onclick. If you create a JavaScript handler, the editor also adds an event attribute to the control you are scripting. For details, see the next section, "Using Attributes to Create Handlers."

   Handlers are created in one of the following <SCRIPT> blocks, depending on whether you are creating handlers for client or server objects, and what the default language is for the object:

   - clientEventHandlersJS
   - clientEventHandlersVBS
   - serverEventHandlersJS
   - serverEventHandlersVBS

   For details about setting a default language for generated script, see "Choosing a Scripting Language" earlier in this chapter.

After the skeleton has been created, you can fill it in. For example, the following client script function for a text box named txtName is called when the user tabs or clicks away from it.

```
Function txtName_onblur()
   If frmMyForm.txtName.value = "" then
      alert("Name is required!")
   End if
End Function
```

If you are scripting in VBScript, you can cancel the effect of some events. For example, you can write a handler for a form's onsubmit event, which occurs when the user clicks the Submit button. Your script can check that the data is valid, and if it is not, cancel the onsubmit event. For details about whether you can cancel an event, see the documentation for the event you are working with.

To cancel the effect of an event

- Create the VBScript procedure as a function, and set the function's return value to False, as shown in the following script:

```
<SCRIPT LANGUAGE="VBSCRIPT">
Function frmMyForm_onsubmit()
   If frmMyForm.txtName.value = "" then
      alert("You must enter a name.")
      frmMyForm_onsubmit = false
   End if
End function
</SCRIPT>
```

Using Attributes to Create Handlers

Another way to link a procedure to an event is to set attributes of the object or of the script that explicitly link the two.

Setting Object Attributes

When you create an object, you can set an attribute to assign a handler to it.

To use attributes to assign a handler to an event

- In the tag that creates the object, name the event and assign the procedure to it, using this syntax:

```
<INPUT TYPE="ObjectType" LANGUAGE="language" EventName="Procedure">
```

– or –

```
<FORM LANGUAGE="language" NAME="FormName" EventName="Procedure">
```

The language attribute is not required, but guarantees that the handler will work the way you want. If no language is specified, the default language for the page is assumed.

For example, the following page contains a form with two buttons. Each button definition explicitly links the onclick event to a procedure:

```
<FORM NAME="Form1">
   <INPUT TYPE="button" NAME="btnVB" VALUE="VB" onClick="pressed"
      LANGUAGE="VBScript">
   <INPUT TYPE="button" NAME="btnJS" VALUE="JS" onClick="pressed2()"
      LANGUAGE="JavaScript">
</FORM>

<SCRIPT LANGUAGE="VBSCRIPT">
Sub Pressed()
   alert ("Pressed the VBScript button")
End Sub
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript">
   function pressed2() {
      alert("Pressed the JavaScript button.");
   }
</SCRIPT>
```

In client script, for an object such as the window object, which is not explicitly created using a tag, use the <BODY> tag, as shown in the following example:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub newWindow
   'script statements here
End Sub
</SCRIPT>
</HEAD>

<BODY LANGUAGE="VBScript" onLoad="newWindow">
   [...]
</BODY>
</HTML>
```

Setting Script Attributes

Another way to link objects and handlers is to set attributes of the script that assigns it to an event.

To use script attributes to assign it to an event

- In the <SCRIPT> tag, use the FOR attribute to identify the object and the EVENT attribute to identify the event, using this syntax:

  ```
  <SCRIPT LANGUAGE="language" FOR="object" EVENT="EventName">
  ```

You can use this method for any named elements and for any elements inserted using the <OBJECT> tag. You do not need to create the script as a subroutine or function, because the script attributes specify when the script will run.

The following example is similar to the previous script example, but it uses a different syntax:

```
<SCRIPT LANGUAGE="VBScript" FOR="Button1" EVENT="onclick">
   alert("Button has been pressed")
   document.Form1.Button1.value="Pressed"
</SCRIPT>

<FORM NAME="Form1">
   <INPUT TYPE="button" NAME="Button1" VALUE="Click">
</FORM>
```

Gathering Information with HTML Forms

A common way to prompt users for information is to put a form on a page. Users can enter information or select choices using text boxes, option buttons, drop-down lists, and checkboxes. They then submit the form by clicking a button, and the information becomes available to your application.

You can define forms in these ways:

- Use Microsoft Visual Interdev design-time controls and then write handlers for their events. This method allows you to work visually with your form's controls and use object-oriented techniques for scripting. It also makes it easy to bind form controls to a database.
- Create an HTML form. This is the traditional HTML way of creating forms. You might use this method if you thought that the page might be edited using tools outside of Visual Interdev.

Using design-time controls is easier and more powerful, because it allows you to use the scripting object model for designing forms. For information about using design-time controls and the scripting object model, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

The sections below describe how to create and work with HTML forms in the traditional HTML manner.

Defining HTML Forms

You create an HTML form on a page with the <FORM> tag.

> **Note** Do not add an HTML form to a page that contains Visual InterDev design-time controls, because the page already contains a form used to manage the controls. For details, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

To create an HTML form

1. Select the text that you want to enclose in a form, and then from the **HTML** menu choose **Form.**

   The editor puts <FORM> and </FORM> tags around your selection. The form's ID and NAME attributes are automatically assigned unique values, which you can change if you want.

   > **Note** If you enclose existing HTML controls in a form, they appear inside the form's node in the Script Outline window.

2. Specify the name of an .asp file that will be used to process the form by assigning it to the ACTION attribute:

   ```
   <FORM NAME="FormName" ACTION="ASPFileName.asp">
   ```

3. Specify how the form information will be sent to the server using the METHOD attribute:

   ```
   <FORM NAME="FormName" ACTION="ASPFileName.asp" METHOD="method">
   ```

   To send the information as part of the HTTP header (so it can be extracted using the Forms collection of the server's Request object), set METHOD to "POST."

   To send the information as a search string to the .asp file (so it can be extracted using the Request object's QueryString collection), set METHOD to "GET."

After defining the form, you can define controls such as text boxes, buttons, and so on.

> **Note** You can use design-time controls to create the user interface for your application. For details, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

To create controls in the form

- Drag controls from the HTML tab of the Toolbox onto the form.

  The control's ID and NAME attributes are automatically assigned unique values. You can change these values if you want.

  Note When you drag HTML controls into an HTML form, they appear inside the form's node in the Script Outline window.

For example, a form that contains a text box and a Submit button might look like this after you dragged the controls from the Toolbox and edited them:

```
<FORM NAME="Form1" NAME="frmMyForm" ACTION="process.asp" METHOD="post">
   Enter your name:
   <INPUT TYPE=text NAME=text1 ID=Text1>
   <INPUT TYPE="submit" VALUE="Submit" ID=Submit1 NAME=Submit1>
</FORM>
```

Processing Form Information Before Submitting

In simple forms, no processing is required on the client: the user clicks the Submit button, and the browser handles the tasks of gathering the data and sending it to the server. However, you can process forms by writing client scripts to handle button clicks and so on.

A typical example is to write a script that validates user input before the form is posted. But in client script you can have access to the events fired for all the form's controls, so you can write handlers for any purpose you need, including replacing server processing altogether.

Note The scripting object model allows you to script design-time controls easily. For details, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

To process forms in a client script

- Write an event handler for the form's onsubmit event or for the events associated with individual controls, such as a button's onclick event or a text box's onblur event.

To determine the value of a specific control, you reference it using an object hierarchy that goes from document to form to control to property.

The following illustrates a simple validation routine in a form's onsubmit event handler:

```
<SCRIPT LANGUAGE="VBScript">
Function form1_onsubmit()
   If Len(document.form1.text1.value) < 1 then
      MsgBox("You must enter a name!")
      form1_onsubmit = False   ' This cancels the submission
   Else
      ' Information automatically sent to server
   End If
End Function
</SCRIPT>
```

To write handlers for other events, you can use the Script Outline window. For more information about writing scripts for the events associated with HTML form controls, see "Handling Events with HTML Elements" earlier in this chapter.

Processing Forms on the Server

In Visual InterDev, forms are usually handled by an .asp file on the server.

To process a form in server script

- On the target ASP page, use the Forms collection of the server Request object to extract values from the form.

  The Forms collection is like an array that holds the values of all the controls in the form. You can extract the values of individual controls by referencing them by name in the Forms collection.

For example, the following script gets the values of several fields on the form using the names assigned in the form to the <INPUT> tags, and puts them into Session object variables to be used elsewhere in the application:

```
<%
Session("LastName") = Request.Form("LastName")
Session("BirthDate") = Request.Form("Birthdate")
%>
```

Creating Dynamic Information in a Hidden Field

If you want to pass information that is not directly entered by a user, you can use a hidden field.

> **Note** If you use the scripting object model and design-time controls, information is automatically available in server scripts. For details, see "Creating Forms with Design-Time Controls" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

To create information in a hidden field

1. In a form, create an <INPUT> tag and set its TYPE attribute to HIDDEN, as in the following example:

   ```
   <FORM NAME=frmEmployee METHOD=POST ACTION=process.asp>
      [...]
      <INPUT TYPE="Hidden" NAME="ExtraInfo">
   </FORM>
   ```

2. Put information into the hidden field using a client script.

   For example, you can load the hidden field just before the form is submitted by writing a handler for the form's onsubmit event. The following simple example puts the current time into the hidden field:

```
<SCRIPT Language="VBScript">
Function frmEmployee_onsubmit
   document.frmEmployee.ExtraInfo = time
End Function
</SCRIPT>
```

When the form is submitted, you can get the information out of the hidden field using server script exactly the way you get information out of any other form element, as in the following example:

```
<%Session("UpdateTime") = Request.Form("ExtraInfo")%>
```

Posting Information to the Same File

Instead of always creating a separate .asp file to process the contents of a form on an .htm file, you can put both on the same ASP page: the form and the script that processes the form. This can simplify your application and make it easier for users.

For example, it is typical to have three or more pages associated with a form: the page with the form, another with the script that processes the form, and a third with an error message. However, if the form is posted to the file it appears on, you can send informational messages along with the context of the form and you only have to create one page.

To post information to the same file

1. Create the form in an .asp file.
2. In the ACTION attribute of the form, specify the name of the file as the target URL.
3. In the server script at the top of the .asp file, use the Request object to check whether information is being passed to the file. If so, the form has been submitted, and you can process it. Otherwise, assume the form is being displayed for the first time.

The GetEmail.asp file below is an example of this. The script determines whether the user has entered an e-mail address, and if so, whether it is valid. In each case, the script produces an appropriate message, which then appears beneath the form as feedback to the user.

```
<HTML>
<BODY>
<!-- GetEmail.asp -->
```

```
<%
   If IsEmpty(Request("Email")) Then
      Msg = "Please enter your email address."
   ElseIf InStr(Request("Email"), "@") = 0 Then
      Msg = "Please enter an email address" & _
      " in the form username@location."
   Else
      ' In a real application, the following message
      ' would be replaced by actual processing.
      Msg = "This script could process the " & _
      "valid Email address now."
   End If
%>
<FORM METHOD="POST" ACTION="GetEmail.asp">
<PRE>
Email: <INPUT TYPE="TEXT" NAME="Email" SIZE=30
↳ VALUE="<%= Request("Email")%>">
<%= Msg %><P>
<INPUT TYPE="Submit" VALUE="Submit">
</PRE>
</FORM>
</BODY>
</HTML>
```

Displaying Information to the User

Web pages are filled with information for users, but most of it is static. You can use scripts to display dynamic information to the user.

Displaying Information from Client Scripts

Because client scripts run on the browser, they give you flexibility in how you want to display information to the user. One way is to display message boxes, the way stand-alone applications on a computer often do.

To display message boxes from client scripts

- In the client script, call the window object's alert method:

  ```
  <SCRIPT LANGUAGE="VBScript">
     window.alert("Hello, World.")
  </SCRIPT>
  ```

  To reference the current window, you can omit the reference to the window object.

  If you are writing in VBScript, you can use the MsgBox function, which allows you to specify particular buttons.

You can also display information from client scripts directly on the page, intermingled with the HTML text.

To display information in the page from client scripts

- Call the document.write method, which puts text at the location in the page where the script is executing. For example:

```
<SCRIPT LANGUAGE="VBScript">
   document.write "The current time is " & time
</SCRIPT>
```

If your page contains an HTML text box or text area control, you can change the contents of the box to display information.

To display information inside an HTML control

- Set the value property of the text box or text area control, as in the following example:

```
<SCRIPT LANGUAGE="VBScript">
Function btnShowTime_onclick()
   document.Form1.txtTime.value = time
End Function
</SCRIPT>
```

If your application will be running in browsers that support Dynamic HTML (such as Microsoft Internet Explorer 4.0), you can directly set the text of any tag that has a name or ID.

To set the text of a tag using DHTML

- Set the tag's innertext property. The tag must have an ID or a name that you can reference in the script. The following page illustrates a tag and how to set it.

```
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Function btnChangeText_onclick()
   para1.innerText = "The new time is: " & time
End Function
</SCRIPT>

</HEAD>
<BODY>
<P ID=para1>This text will be replaced.</P>
<P><INPUT TYPE="button" NAME="btnChangeText" VALUE="Change text"></P>
</BODY>
```

To format the text you are displaying, you can include HTML tags, as in the following example:

```
<SCRIPT LANGUAGE="VBScript">
   document.write "<P>The current time is <B>" & time & "</B></P>"
   document.write "<P>The current date is <B>" & date & "</B></P>"
</SCRIPT>
```

If the information you want to display includes characters that are reserved in HTML — such as < and > — you cannot directly include them in the string to display.

To display reserved characters

- Use the HTML syntax for ASCII characters, such as `&lt;` or `&#60;` for the opening angle bracket (<):

```
<SCRIPT LANGUAGE="VBSCRIPT">
   document.write "&lt;Click here&gt;"
</SCRIPT>
```

Displaying Information from Server Scripts

To display information to a user from a server script, you usually make it part of the page that is sent to the browser.

To display information on a page from a server script

- Call the Response.Write method, passing it the information you want to display:

```
<% Response.Write "The current time on the server is " & Time %>
```

– or –

- Use the "=" operator, which is a shorter version of the same method:

```
<% = "The current time on the server is " & Time %>
```

– or –

- If the information is not in a variable or calculated in an expression, make it part of the page. To display text, make sure it is outside the <% %> delimiters and is not part of a <SCRIPT RUNAT="SERVER"> block. You can easily include HTML tags this way, as in the following example:

```
<BODY>
<H1>The Time Page</H1>
<% t = time %>
<P>
The current time at the server is <B><%=t%></B>
</P>
</BODY>
```

The server cannot directly display a message box, but a server script can create a client script that displays one.

To display a message box in a server script

- Create a client script, and then enclose it in server script that shows it conditionally. In the following example, the script block is sent to the browser only if the error flag on the server is true.

```
<%if fError = True then%>
   <SCRIPT Language="VBSCRIPT">
      alert("An error occured on server.")
   </SCRIPT>
<%End If%>
```

If the information you want to display includes characters that are reserved in HTML — such as < and > — you cannot include them in the string to display.

To display reserved characters

- Call the HTMLEncode method to convert the characters to HTML syntax for ASCII characters:

```
<%= Server.HTMLEncode("The paragraph tag: <P>") %>
```

Navigating Conditionally

In static Web pages, you link pages together using the <A> tag, as in the following example:

```
Click <A HREF="home.asp">here</A> to return to the home page.
```

However, by using scripts, you can navigate through the application dynamically, specifying target pages based on conditions in the application. For example, a page in your application might ask the user for a password. If the user provides a correct password, the application displays a welcome page, otherwise it displays an error message.

Note You can use the scripting object model and page objects to design navigation for your Web pages. For details, see “Extending the Scripting Object Model Across Pages” in Chapter 24, “Scripting with Design-Time Controls and Script Objects.”

Navigating from Client Script

If you are writing client script, you can control the page that the browser displays next by calling a method or setting a property.

To navigate to another page in client script

- In VBScript, call the navigate method of the browser's window object, passing it the URL of the page to go to.

  – or –

- In any scripting language, set the window object's location.href property to the URL of the target page.

  – or –

- In any scripting language, call the window object's open method, passing it the URL of the page to go to, along with other parameters.

  Calling this method allows you to specify an existing window or frame in which to display a page, start a new instance of the browser, and control the appearance of the browser.

For example, the following client script reads the value of a check box in a form, and depending on the user's preference, navigates to one of two possible pages:

```
<SCRIPT LANGUAGE="VBScript">
Sub NextPage()
   ' The following reads the value of a checkbox called Checkbox1
   fReviewIntro = document.frmNavBar.Checkbox1.checked
   If fReviewIntro then
      window.navigate("http://MyServer/intro.htm")
   Else
      window.navigate("http://MyServer/page2.htm")
   End If
End Sub
</SCRIPT>
```

The following is a similar example, but sets the window object's href property to navigate:

```
   If fReviewIntro Then
      window.location.href = "http://MyServer/intro.htm"
   End If
```

Navigating from Server Script

You can also control the next page displayed in a browser from server script. You use server script to navigate if the condition for moving to a specific page depends on information only available to the server — for example, information from a database, from a Session or Application object variable, and so on.

To navigate in server script

- Call the Response.Redirect method, passing the URL of the page to go to.

For example, you might want to make sure users log in to your application before viewing pages. If they attempt to navigate to a page deeper in your application without logging in first, you want to detect that and send them to the login page instead. In the following example, a Session object variable contains a flag indicating whether the user has already logged on:

```
<%
If Not Session("Been_to_Home_Page") Then
   Response.Redirect "homepage.asp"
End If
%>
```

Buffering Response

By default, when the server sends a page to the browser, it streams the output as soon as it is composed. However, you can have the server buffer the output. If you do, the page is completely processed at the server before being sent. The advantage is that you can then call a method such as Response.Redirect to send a different page instead. If the page is not buffered, the server reports an error if you try to redirect a page after some of the current page has already been sent.

To set buffering

- Set the Response object's Buffer property to True or False, as in this example:

  ```
  <% Response.Buffer = True %>
  ```

Managing Sequential Page Navigation

If you are navigating in server code, you can use a bundled component that can manage sequential navigation through a list of pages. Rather than maintaining URL references in a number of ASP pages, you can specify the sequential organization of pages in a single, easy-to-edit text file.

To create sequential page navigation

1. Create a text file that contains a list of pages, one per line.
2. Create a NextLink object.
3. Use the NextLink object's methods to move between pages in the list.

The following example reads the link order from a text file and creates a table of contents on a single page.

```
<% Set NextLink=Server.CreateObject("MSWC.NextLink") %>
<% count=NextLink.GetListCount("/Nextlink.txt") %>
<% i=1 %>
<UL>
<% For i = 1 to count %>
   <LI><A HREF="<%= NextLink.GetNthUrl("/nextlink.txt",i) %>">
   <%= NextLink.GetNthDescription("/nextlink.txt",i) %></A>
<% Next %>
</UL>
```

Sharing Dynamic Information

Many times in your Web applications you will need to use the same information on two or more pages. For example, your application might:

- Prompt the user for a name, and then pass it to another page to be displayed there.
- Display the time and date when a user last visited your site.
- Require users to log on. When they do, the application assigns them a security level that is checked throughout your application.

  Note You can use the scripting object model and page objects to maintain state information automatically. For details, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts."

To share dynamic information

- Store information in global variables maintained in the Application and Session objects. Information in Application and Session variables persists between pages.

  – or –

- Pass a query string from one page directly to another as part of the URL in a link.

  – or –

- Store information in a database, which allows you to store data permanently, share it with other applications, and use database capabilities to filter and analyze application data. For more information, see Chapter 20, "Modifying Data."

  – or –

- Store information on the user's computer in a cookie. Because you can read and write cookies, you can use them to maintain user-specific information.

  – or –

- Store information in ASP objects.

Each method has different uses, depending on factors such as how many pages might need the information, and whether you need to keep it permanently.

Storing Information Globally

You can define global variables with two Microsoft Internet Information Services (IIS) objects:

- **Application object** The server maintains one Application object for each ASP-based application. The object is initialized when the application is accessed by the first user and persists until the application is shut down.

  When you define variables in the server's Application object, their values are available to all pages in the application. A typical example is an application hit counter that is updated each time another user starts a session.

- **Session object** A Session object is initialized when a user first requests a page from the server. The object persists until the session is abandoned, when the user closes and reopens the browser, manually (by calling the Session.Abandon method), or when the user's browser has not requested a page from the application for a specified period of time.

  Session objects are global for any one user. You might use a session object to maintain a user's security level.

To get and set values in Application and Session object variables

- In a server script, use syntax such as this:

```
Application("VariableName") = value
Session("VariableName") = value
variable = Applicaton("VariableName")
variable = Session ("VariableName")
```

To initialize Application and Session object variables

- In the Global.asa file, write handlers for the Application_OnStart and Session_OnStart events, setting the initial values for each variable you want to use.

For example, you can maintain a hit counter in an Application object variable. You can initialize the counter in the Application_OnStart event using a script such as this:

```
Sub Application_OnStart()
   Application("counter") = 0
End Sub
```

Each time a user starts a new session, the counter is updated. You then also make a local copy of the counter in a Session object variable for the user. The best place to do this is in the Session_OnStart event of the Global.asa file, using script such as this:

```
Sub Session_OnStart()
   iCount = Application("counter")
   iCount = iCount + 1
   Application("counter") = iCount' Global counter
   Session("counter") = iCount ' User's copy of counter
End Sub
```

The opening page of your application could display the counter in this way:

```
<BODY>
<H1>Welcome</H1>
You are visitor number <%=Session("counter")%>
out of <%=Application("Counter")%>.
</BODY>
```

When the user sees the opening page the first time, the two numbers are the same. If the user revisits the page, the "out of" number (the Application object counter) might change if other users have started sessions since the user first started a session.

Application and session object variables generally store dynamic information only. If you want to keep the information permanently, you must devise a way to save the information between applications and sessions. One way is to write out values in Application_OnEnd or Session_OnEnd handlers in the Global.asa file.

Adding Query Strings to Links

You can pass information directly to another page as part of a link (<A> tag). For example, a page might display a list of employee names, each of which is a link. When the user selects one of the names, the link calls a page and passes it the corresponding employee ID.

To add a query string to a link

- In a link definition (<A> tag), add a query string onto the end of the target URL, using "?" as a delimiter between the URL and the search string. The syntax is:

  ```
  <targetURL> ? <parm1>=<value> & <parm2>=<value> & ...
  ```

The following example shows a page that displays a list of employee names. Each name is a link. All links go to the same page, but each one passes a different employee ID.

```
<BODY>
<H1>List of employees</H1>
<P>Click the name of an employee to see information about that employee.</P>
<A HREF=EmplInputForm.asp?empid=1>Ann</A><BR>
<A HREF=EmplInputForm.asp?empid=2>John</A><BR>
<A HREF=EmplInputForm.asp?empid=3>Susan</A><BR>
<A HREF=EmplInputForm.asp?empid=4>Michael</A><BR>
</BODY>
```

In the EmplInputForm.asp file, you can use the QueryString collection of the server Request object to determine what value was passed:

```
<% vEmplID = Request.QueryString("empid") %>
```

You can create more sophisticated dynamic links by using server script to supply values. For example, the following is a single line in which both the employee ID and employee name are supplied from a database query:

```
<A HREF=EmplInputForm.asp?empid= <%=RS.Fields("EmpID")%> > _
   <%=RS.Fields("EmpFName")%> </A>
```

Storing Information on the User's Computer

A convenient way to maintain information about a user is to use a cookie. When a session is first started, the server sends a cookie to the client browser. Each time the client browser requests a page from that server, it sends the cookie back to the server, which can then read the cookie and identify the client browser.

You can use cookies to store your own application information, such as user preferences. Cookies are available until the date specified in the cookies' Expires attribute.

> **Note** Because cookies can potentially write to the user's hard disk, most browsers usually allow users to disable them or to display a warning before accepting a cookie. For some applications — such as public applications that might be accessed by users with a wide range of browsers and security settings — cookies can be impractical. If you do use cookies, your application must provide an alternative way to maintain dynamic information if the user's browser refuses a cookie.

To store information in a cookie

- To put information into the cookie, use the Response.Cookies collection. To get information out of a cookie, use the Request.Cookies collection. For example:

  ```
  <% Response.Cookies ("FavoriteColor")="Red" %>

  <% txtFavorite = Request.Cookies("FavoriteColor")%>
  ```

Cookies can store multiple values. Each value in the cookie is assigned a key by which you identify it.

To store a value in a cookie, you use the Response object, specifying the name of the cookie to update, the key to update, and the value. If the cookie does not already exist, the Response object creates it. For example, the following updates a cookie setting the key FavoriteColor to the value "Red":

```
<% Response.Cookies("Preferences")("FavoriteColor")="Red" %>
```

> **Note** You must update cookies in the <HEAD> section of an .asp file, or an error will result.

If an existing cookie has key values but the Response.Cookies method does not specify a key name, then the existing key values are deleted. Similarly, if an existing cookie does not have key values but the Response.Cookies method specifies key names and values, the existing value of the cookie is deleted and new key-value pairs are created.

To retrieve the value from a cookie, you use the `Request` object with similar syntax:

```
<% vColor = Request.Cookies("Preferences")("FavoriteColor") %>
```

To see how you can use cookies to store information, refer to the User Preferences sample in the Visual InterDev online documentation.

Reading from and Writing to Files

In server scripts, another way to maintain information is to store it in a text file on the server.

To read and write text files

1. In a server script, create a `TextStream` object.
2. Use the object's CreateTextFile, WriteLine, and ReadLine methods to manage the information in the file.

The following example creates a new file and writes a single line of text to the file:

```
<HTML>
<BODY>
<H3>Textstream test</H3>
<%
  Set OutStream = Server.CreateObject("MS.TextStream")
  OutStream.CreateTextFile "tsworks.txt", , True
  OutStream.WriteLine "This line is written to the file."
%>
</BODY>
</HTML>
```

Writing Reusable Script

In most Web applications, you'll want to display blocks of HTML content on multiple pages or process script in multiple files.

To use content in multiple files

- Use the server-side #INCLUDE directive to dynamically insert the contents of other files into your file.

You can use the #INCLUDE directive to:

- Share HTML content between pages.
- Share script libraries.

  Note When you insert the contents of other files into your page, the Script Outline window does not reflect their contents. Similarly, objects that are defined in the included file do not appear in the IntelliSense statement completion drop-down list.

Sharing HTML Content

Your application might contain HTML elements that you want to use on multiple pages: page navigation buttons (Next, Previous, Home), copyright information, a company logo, and so on.

Instead of copying and pasting the text into multiple pages, you can place the information into files, and then reference the files that contain the desired text at the appropriate places in other your pages. Then if you need to update the text, you can change it in the file and all the pages that include the file automatically display the updated text.

To reference information that is in another file

- Use the server commands #INCLUDE FILE (for a path relative to the current file) or #INCLUDE VIRTUAL (for a path relative to the virtual root).

For example, the following lines include a file named Header.inc at the top of a page and a file named Footer.inc at the bottom of a page:

```
<!-- #INCLUDE FILE="Header.inc" -->

Normal HTML text and script goes here.

<!-- #INCLUDE FILE="Footer.inc" -->
```

When the file is processed by the Web server, the entire contents of the Header.inc and Footer.inc files are inserted into the file at the location of the #INCLUDE directive. Any script in Header.inc and Footer.inc is processed after all includes have been merged into the main file. You can even nest included files by including a file that contains another #INCLUDE directive.

You can update an included file on your production server without stopping the Web server. However, you cannot include an #INCLUDE in a loop and you cannot recursively include files. For example, file A.inc might have the following line:

```
<!-- #INCLUDE FILE="B.inc" -->
```

In this case, you cannot include the following line in file B.inc

```
<!-- #INCLUDE FILE="A.inc" -->
```

An attempt to do so will result in an error.

Sharing Script Libraries

In ASP pages, you can share script libraries using the #INCLUDE directive. This allows you to share subroutines and functions between multiple pages. For example, the following file contains a function that returns a string delimited with single quotation marks ('), and a subroutine that takes an array argument and displays the array values in an HTML table:

```
<!-- shared.inc -->
<%
Function SQLString(cString)
   SQLString = "'" & cString & "'"
End Function
%>
```

```
<%
Sub Array2Table(aArray)
If IsArray(aArray) Then %>
   <TABLE>
   <% For i = 0 to UBound(aArray,1) - 1%>
      <TR>
      <% For j = 0 to UBound(aArray,2) - 1 %>
         <TD> <%= aArray(i,j) %> </TD>
      <% Next %>
      </TR>
   <% Next %>
   </TABLE>
<%End If
End Sub %>
```

Instead of repeating this script in many of your pages, you can use the following line to make the subroutine and function available:

```
<!--#INCLUDE FILE="shared.inc" -->
```

Make sure that the #INCLUDE line comes before any calls to subroutines or functions in the file.

Later, if you want to change some formatting options in the Array2Table subroutine (such as giving the table a border of 1 or a width of 100%), you can make the change in the subroutine in Shared.asp and all your pages will instantly display the table with the new characteristics.

In client script, you can include references to other files in the <SCRIPT> block. Use the SRC attribute to specify the name of a file that contains script you want to include. For example, the following <SCRIPT> block includes a reference to a page containing error message routines.

```
<SCRIPT SRC="Errmsg.htm"></SCRIPT>
```

After you have included a reference to a file in this way, you can call script on the page as if it were in the current page.

Creating Portable Script

If you are writing a Web application for use on a corporate intranet or for another known environment, you can usually rely on the features of a specific Web browser. However, if your application will be accessible by users who can choose their own browser, you must be careful to write an application that works well with as many browsers as possible.

When creating applications that are as browser-independent as possible, you must be able to:

- Identify the browser that the user is using and what capabilities it has.
- Create scripts that run on different browsers. Alternatively, you can provide a graceful way to accommodate browsers that cannot support all features in your application.

  Note When you create a new Web page in Microsoft Visual InterDev, one of the properties you can set for the page is the TargetBrowser property. This property does not perform any checking or otherwise prevent you from creating scripts that are incompatible with a specific browser.

  The TargetBrowser property just alerts the editor to whether you are writing scripts for Microsoft Internet Explorer 4.0, and if so, it tells the editor to display Internet Explorer 4.0 objects when completing script statements.

Identifying Browsers and Browser Capabilities

You can use various properties and objects to get information about what browser the user is currently using.

To get browser information in client script

- Query properties of the window's navigator object.

  For example, the following client script displays the name of the current browser:

```
<SCRIPT LANGUAGE="VBScript">
Sub Button1_OnClick()
   MsgBox "Current browser is " & window.navigator.appname
End Sub
</SCRIPT>
```

In server script, you can get basic information about a browser from the HTTP header and find details by using a bundled component.

To get basic browser information from the HTTP header in server script

- Use the ServerVariables collection of the server's Request object, and query the value of HTTP_USER_AGENT.

The value of the HTTP_USER_AGENT variable is a string that lists the compatibility, name, and version of the browser. For example, this script displays the capability of the browser that requested the page on which it appears:

```
<% browser = Request.ServerVariables("HTTP_USER_AGENT")%>
<H1>Browser Identification</H1>
<P>Your browser identifies itself as:</P>
<%=browser%>
```

To get information about specific capabilities of a browser in server script

- Create an instance of the BrowserType object, and then query its properties.

For example, you can determine whether a browser supports frames, or whether it supports VBScript. The following example uses the BrowserType object component to display the current browser's capabilities.

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
Browser: <%= bc.browser %><BR>
Version: <%= bc.version %><BR>
Supports frames?
   <% If (bc.frames = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
Supports tables?
   <% If (bc.tables = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
Supports background sounds?
<% If (bc.BackgroundSounds = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
```

```
Supports VBScript?
   <% If (bc.vbscript = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
Supports JavaScript?
   <% If (bc.javascript = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
```

Information about specific browser types is maintained on the server in the Browscap.ini file. The content of the Browscap.ini file determines what properties are available to the BrowserType object.

Creating Browser-Independent Scripts

Unless you know what types of browsers your users will be using, you should anticipate a wide variety of browsers and create scripts that run on as many browsers as possible.

To make your scripts browser-independent

- Avoid using features, such as specific objects, that are available on only certain browsers.

  – or –

- Allow users to specify what content they want to see.

  – or –

- Test for specific browser capabilities and use branches around sections of script that a specific browser might not support.

  Tip A simple way to target specific browsers is to create two (or more) versions of your Web pages. Based on querying the user's browser (or by asking the user explicitly), you can provide separate pathways through your application, with each pathway containing pages compatible with the user's browser.

Avoiding Browser-Specific Features

Specific browsers make different object models available, although there is often overlap in the object models between browsers. For example, Microsoft Internet Explorer 4.0 supports Dynamic HTML (DHTML), which allows you to add animation and text effects to your Web pages. However, DHTML features are not necessarily available in all browsers, so if you use these features, you must ensure that users with different browsers don't try to view pages with DHTML pages.

For details about what objects you can use with specific browsers, see the documentation for your browser.

Allowing Users to Specify Content

You can ascertain the capabilities of a user's browser in your script, but you cannot determine other factors, such as modem speed, that can affect a user's experience with your Web site. Allowing users to specify the type of content they want to receive has the advantage of putting control in the users' hands.

To allow users to specify content

1. Use a form or dynamic link to query the user's preferences. For details, see "Gathering Information with HTML Forms" and "Sharing Dynamic Information" earlier in this chapter.
2. Store the preferences in a server session variable or cookie. For more information, see "Sharing Dynamic Information" earlier in this chapter.
3. Use branches to display or include user-specific information. For more information, see "Writing Reusable Script" earlier in this chapter.

For example, the following two links allow a user to specify a display preference.

```
<A HREF="setpref.asp?type=basic">Basic Display<A>
<A HREF="setpref.asp?type=rich">Rich Display<A>
```

The script in Setpref.asp sets a Session object variable to the store the user's preference:

```
<% Session("type") = Request("type")
Response.Redirect "home.asp" %>
```

Script in your pages can check the setting of the Session object variable to determine how to display your content:

```
<% If Session("type") = "rich" Then %>
        Display Frames, Tables, and Images.
<% Else %>
        Display text only
<% End If %>
```

Testing for Browser Capabilities

You can also use logic within a script to make specific features available. For example, the following script tests whether a browser supports JavaScript. If so, it inserts a small script that jumps to the home page. Otherwise, it inserts HTML text that displays a jump for the user to click.

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
<% If bc.JavaScript = true then %>
   <SCRIPT LANGUAGE="JavaScript">
      top.location.href = "home.asp"
   </SCRIPT>
<%Else%>
   Click <A HREF="home.asp">here</A> to return to the home page.
<%End If%>
```

You can combine a test of the browser with a server #INCLUDE directive to display entirely different pages to the user. The following script tests the browser. If the browser is compatible with Microsoft Internet Explorer 3.0 or higher, it includes a page with various color options. Otherwise, it includes a more generic page.

```
<% browser = Request.ServerVariables("HTTP_USER_AGENT")%>
<%If browser = "Mozilla/2.0 (compatible; MSIE 3.0B; Windows NT)" Then%>
   <!--#INCLUDE FILE="/myapp/ColoredTable.asp"-->
<%Else%>
   <!--#INCLUDE FILE="/myapp/PlainTable.asp"-->
<%End If%>
```

CHAPTER 26

Debugging Your Pages

As you write script for your Web application, you can test it to find errors. To help you, Microsoft Visual InterDev includes a full-featured debugger you can use to trace errors in client and server script.

Debugging Client Script

You can debug client script in any of these ways:

- Run a page containing the script to debug from within your Microsoft Visual InterDev solution.
- Attach the Visual InterDev debugger to a process (page) already running in Microsoft Internet Explorer.
- Respond to a syntax or run-time error in a script, called *just-in-time* debugging.
- Include a statement in script that starts the debugger.

  Note To debug client script in Internet Explorer, you must be using Internet Explorer 4.0. Debugging must also be enabled in Internet Explorer (this is the default).

  It is also highly recommended that you do not use Active Desktop mode of Internet Explorer when you are debugging, or set the option in Internet Explorer to launch each new instance of the browser in a new process.

If a Web page contains a mixture of client and server script, you can use the Visual InterDev debugger to debug both. For details, see "Debugging Mixed Client and Server Script" later in this chapter.

Enabling Client Script Debugging for ASP Pages

Before you can debug client script in ASP pages, you must enable debugging. You can manually enable debugging for your ASP application as described under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting." Alternatively, Visual InterDev can automatically enable debugging on the server as needed.

Note If you intend to work exclusively with client script in .htm files, you do not need to enable server debugging or perform the following procedure.

To automatically enable client script debugging in ASP pages

1. In the Project Explorer, right-click the project and choose **Properties** to display the **Property Pages** dialog box.
2. Choose the **Launch** tab.
3. Under **Server script**, make sure that **Automatically enable ASP server-side debugging on launch** is checked.

   Note To debug script in ASP pages, you must be running version 4.0 or later of Microsoft Internet Information Server (IIS).

When this option is set, each time you start a debugging session Visual InterDev checks that the server is configured for debugging. This includes:

- Setting the IIS application to run in its own memory space (in COM terms, it runs "out of process").
- Enabling the IIS application's debugging options.
- Setting up a Microsoft Transaction Server (MTS) package to allow you to attach the debugger to the Web application. The package's identify is set when you first start the debugging session by asking you to provide your name and password.

When you quit your debugging session, Visual InterDev restores the server debugging settings and out-of-process setting to their previous values.

Debugging Client Script within a Solution

If you are working in a Visual InterDev solution, you can debug a file by launching the debugger.

To debug a script from within a solution

1. In Visual InterDev, open the project containing the page you want to debug, and load the page into the editor.
2. Make the page your project's start page. In the Project Explorer, right-click the page and choose **Set as Start Page**.
3. Set a breakpoint in the script to debug. For details, see "Setting Breakpoints" in Chapter 5, "Walkthroughs."
4. From the **Debug** menu, choose **Start**.

5. If debugging for client script in ASP pages has not been enabled as described above, Visual InterDev displays a message. Your options are:
   - If you are working with client script in an .htm file (not an .asp file) and you will not be navigating to an ASP page during your debugging session, choose No to proceed with debugging.
   - If you are working in an .asp file, or if you will be navigating to an .asp file during your debugging session, choose Yes to have Visual InterDev automatically enable client-side ASP debugging.
6. If client debugging in ASP pages is enabled, and if this is the first time you have launched a debugging session since opening the project, you are prompted to provide user information that identifies the debugging process on the server. (You are prompted even if you are currently working with an .htm file, in case you navigate to an ASP page.) Enter your domain and name in the form *domain\name* and your password.

   Note You might experience a delay the first time you launch a debugging session for the current project while Visual InterDev establishes the proper debugging configuration on the server.

   Visual InterDev launches Internet Explorer and loads the page into it. When Internet Explorer reaches the breakpoint, it stops and displays the source code in the editor window. If the breakpoint is in an event handler script, you must trigger the event to reach the breakpoint.
7. If you find an error, fix it, and then save the file. If you do not have a working copy of the file, right-click the name of the file in the Project Explorer and choose **Get Latest Version** before you make modifications.
8. From the **Debug** menu, choose **Restart**.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

Debugging Client Script in a Running Document

If a client script is already running in Internet Explorer and you detect a problem, you can stop the script and debug it on the spot. You can debug a running document from within Visual InterDev or from within Internet Explorer.

Note If you are working with an ASP page, debugging must be enabled on the server. For details, see "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

You can attach to a running document only if attaching is enabled.

To enable just-in-time debugging

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Debugger**.
3. Under **Script**, check **Attach to programs running on this machine**.

You can attach to a document directly in Visual InterDev.

To debug a running document in Visual InterDev

- In Visual InterDev, choose **Processes** from the **Debug** menu. In the **Processes** dialog box, choose **Microsoft Internet Explorer**, choose **Attach**, and then choose the script you want to debug.

  – or –

  If you have already attached to a running document, in Visual InterDev, choose **Break** from the **Debug** menu. The debugger will stop at the next script statement that is executed.

If Visual InterDev is not already running, you can launch the debugger from Internet Explorer.

To debug a running document from Internet Explorer

1. In Internet Explorer, choose **Script Debugger** from the **View** menu, and then choose **Break at Next Statement**. In the browser, trigger the event that calls the script you want to debug.

   – or –

   In Internet Explorer, choose **Script Debugger** from the **View** menu, and then choose **Open**.
2. A new instance of Visual InterDev is launched and you are prompted to open a project. If Visual InterDev is already running, a second instance is launched. Open the project containing the file to debug.
3. If the project is already open in another instance of Visual InterDev, you cannot open it again, so Visual InterDev creates a new solution and project instead.
4. The page to debug is loaded into the editor. If necessary, get a working copy of the page. If the project was already open, the page is loaded as read-only file in the new project.

If you make changes to the file, save it and re-deploy it to the server. Refresh the file in Internet Explorer before running the script again.

> **Note** For details about using debugger commands, refer to the Visual InterDev online documentation.

Debugging Client Script in Response to an Error or Debugger Statement

If Internet Explorer encounters a syntax or run-time error, you can use just-in-time debugging to find and fix it. You can also include a statement in your script, such as a Stop statement in Visual Basic, Scripting Edition (VBScript) or a debugger statement in JScript, to launch the debugger from within a script.

You can launch the debugger in response to an error or debugger statement only if just-in-time debugging is enabled.

To enable just-in-time debugging

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Debugger**.
3. Under **Script**, check **Just-In-Time debugging**.

To debug in response to an error or debugger statement

1. When Internet Explorer encounters an error or a statement that starts the debugger, it displays an error message prompting you to debug. Choose **Yes**.
2. A new instance of Visual InterDev is launched. If Visual InterDev is already running, a second instance is launched.
3. Open the project containing the file to debug. If the project is already open in another instance of Visual InterDev, you cannot open it again, so Visual InterDev creates a new project instead.

   The page to debug is loaded into the editor. If necessary, get a working copy of the page. If the project was already open, the page is loaded as a read-only file in the new project.

   Note If you are debugging a client script generated by an .asp file, the line numbers reported in error messages refer to lines in the HTML document currently displayed in the browser. These usually do not correspond to line numbers in the original .asp file, because server script does not appear in the HTML output of an .asp file. For more information, see "Debugging Mixed Client and Server Script" later in this chapter.

If you make changes to the file, save it and re-deploy it to the server. Refresh the file in Internet Explorer before running the script again.

Note For details about using debugger commands, refer to the Visual InterDev online documentation.

Debugging Server Script

From Microsoft Visual InterDev you can debug server script that executes on Microsoft Internet Information Server (IIS). If IIS is running on your computer, you can debug server script in much the same way that you debug client script. If the server is on another computer, you can use *remote debugging* from your computer to find errors in the server script. For details, see "Debugging Remotely" in Chapter 29, "Integration Tasks."

You can debug server script in any of these ways:

- Run a page containing the script to debug from within your Microsoft Visual InterDev solution.
- Attach the Visual InterDev debugger to a process (page) already running in Microsoft Internet Explorer.
- Respond to a syntax or run-time error in a script, called *just-in-time* debugging.
- Include a statement in script that starts the debugger.

  Note To debug script in ASP pages, you must be running version 4.0 or later of Microsoft Internet Information Server (IIS).

If a Web page contains a mixture of client and server script, you can use the Visual InterDev debugger to debug both. For details, see "Debugging Mixed Client and Server Script" later in this chapter.

Enabling Server Script Debugging for ASP Pages

Before you can debug client script in ASP pages, you must enable debugging. You can manually enable debugging for your ASP application as described under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting." Alternatively, Visual InterDev can automatically enable debugging on the server as needed.

To automatically enable script debugging in ASP pages

1. In the Project Explorer, right-click the project and choose **Properties** to display the **Property Pages** dialog box.
2. Choose the **Launch** tab.
3. Under **Server script**, make sure **Automatically enable ASP server-side debugging on launch** is checked.

When this option is set, each time you start a debugging session Visual InterDev checks that the server is configured for debugging. This includes:

- Setting the IIS application to run in its own memory space (in COM terms, it runs "out of process").
- Enabling the IIS application's debugging options.

- Setting up a Microsoft Transaction Server (MTS) package to allow you to attach the debugger to the Web application. The package's identify is set when you first start the debugging session by asking you to provide your name and password.

  Note You can perform the first two steps manually on the server. For details, see "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

When you quit your debugging session, Visual InterDev restores the server debugging settings and out-of-process setting to their previous values.

Debugging Server Script within a Solution

If you are working in a Visual InterDev solution, you can debug server script by launching the debugger.

Note Before debugging server script, make sure debugging is enabled as described above, or under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

To debug server script from within a solution

1. In Visual InterDev, open the project containing the server script you want to debug.
2. Make the page your project's start page. In the Project Explorer, right-click the page and choose **Set as Start Page.**
3. Set a breakpoint in the server script you want to debug.
4. From the **Debug** menu, choose **Start** to launch the project.

   Visual InterDev attempts to attach the debugger to the document running on the server.
5. If this is the first time you have started the debugger since opening the current project, you are prompted to provide user information used to identify the debugging process on the server. Enter your domain and name (in the form *domain\name*) and password.
6. The browser opens, and you can proceed with debugging. When server script execution reaches the line with the breakpoint, the debugger displays the page in the editor with that line highlighted.
7. Fix any errors, save the file, and then from the **Debug** menu choose **Restart.** If you do not have a working copy of the file, right-click the name of the file in the Project Explorer and choose **Get Latest Version** before you make modifications.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

Debugging Server Script in a Running Document

If debugging has been enabled on the server for your project and you detect an errorwhile the application is running, you can attach the debugger to it. For details about enabling debugging on the server, see “Enabling ASP Debugging on the Server” in Appendix A, “Troubleshooting.”

You can attach to a running document only if attaching is enabled.

To enable just-in-time debugging

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Debugger**.
3. Under **Script**, check **Attach to programs running on this machine**.

To debug a running script

1. In Visual InterDev, choose **Processes** from the **Debug** menu. In the **Processes** dialog box, choose **Active Server Pages**, and choose **Attach**.
2. In the **Running Documents** window, select the script you want to debug.
3. Set breakpoints, and then choose **Restart** from the **Debug** menu, or refresh the document in the browser.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

Debugging Server Script in Response to an Error or Debugger Statement

If debugging is enabled for an IIS application on the server and the server encounters a syntax error or run-time error in a server script, you can use just-in-time debugging to find and fix it. You can also include a statement in your script, such as a Stop statement in VBScript or a debugger statement in JScript, to launch the debugger from within a script. For details about enabling debugging on the server, see “Enabling ASP Debugging on the Server” in Appendix A, “Troubleshooting.”

> **Note** If a debugger is installed on the server computer, the server does not pass error information through to the client. Instead, it displays an error message on the server computer’s monitor. For more information, see “Just-in-Time Debugging of Server Pages” in Appendix A, “Troubleshooting.”

You can launch the debugger in response to an error or debugger statement only if just-in-time debugging is enabled.

To enable just-in-time debugging

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Debugger**.
3. Under **Script**, check **Just-In-Time debugging.**

To debug server script in response to an error or debugger statement

1. When an error message appears prompting you to start the debugger, choose **Yes**.
2. A new instance of Visual InterDev is launched and you are prompted to open a project. If Visual InterDev is already running, a second instance is launched.
3. Open the project containing the file to debug. If the project is already open in another instance of Visual InterDev, you cannot open it again, and Visual InterDev creates a new solution and project instead.

   The page to debug is loaded into the editor. If necessary, get a working copy of the page. If the project is already open, the page is loaded as read-only file in the new project.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

If server debugging is not enabled for the application, errors are displayed in the browser as text in the page. In that case, open the project containing the page in Visual InterDev and start the debugger there, as described above.

Debugging Mixed Client and Server Script

Many ASP pages contain both client and server script. You can set breakpoints in both client and server script, and as each script executes, it can call the debugger at breakpoints.

If the server is not running on your computer, you use *remote debugging* to debug the server script in the page. For details about how to set up remote debugging, see "Debugging Remotely" in Chapter 29, "Integration Tasks."

Note It is highly recommended that you do not use Active Desktop mode of Microsoft Internet Explorer 4.0 when you are debugging.

Enabling Debugging for ASP Pages

Before you can debug script in ASP pages, you must enable debugging. You can manually enable debugging for your ASP application as described under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting." Alternatively, Microsoft Visual InterDev can automatically enable debugging on the server as needed.

Note To debug script in ASP pages, you must be running version 4.0 or later of Microsoft Internet Information Server (IIS).

To enable script debugging in ASP pages

1. In the Project Explorer, right-click the project and choose **Properties** to display the **Property Pages** dialog box.
2. Choose the **Launch** tab.
3. Under **Server script**, make sure **Automatically enable ASP server-side debugging** is checked.

When these options are set, each time you start a debugging session Visual InterDev checks that the server is configured for debugging. This includes:

- Setting the IIS application to run in its own memory space (in COM terms, it runs "out of process").
- Enabling the IIS application's debugging options.
- Setting up a Microsoft Transaction Server (MTS) package to allow you to attach the debugger to the Web application. The package's identify is set when you first start the debugging session by asking you to provide your name and password.

When you quit your debugging session, Visual InterDev restores the server debugging settings and out-of-process settings to their previous values.

Debugging Pages Containing Client and Server Script

When you set a breakpoint in client script in a page that contains both server and client script, the debugger tracks the breakpoint even though the position of the breakpoint can change significantly in the file after server script has executed. If server script causes the client script to be written several times into the page, the debugger tracks each breakpoint separately.

You can set breakpoints in server script, client script, or both. If you set breakpoints in both, the debugger will stop at the server script breakpoints first. When you continue to run, the page is sent to the browser, and the debugger will then stop at breakpoints in client script.

To debug pages containing both client and server script

1. In Visual InterDev, set breakpoints in the lines of client and server script that you want to debug.
2. Make the page your project's start page. In the Project Explorer, right-click the page and choose **Set as Start Page**.
3. From the **Debug** menu, choose **Start**.

   Visual InterDev attempts to attach the debugger to the document running on the server.

4. If this is the first time you have started the debugger since opening the current project, you are prompted to provide user information used to identify the debugging process on the server. Enter your domain and name (in the form *domain\name*) and password.
5. Proceed with debugging. When server script execution reaches the line with the breakpoint, the debugger displays the page in the editor with that line highlighted.

   When you step out of server script, the page will continue executing until it gets to another server script. When the debugger has finished with server script, the server sends the page to the browser, which displays it.
6. If necessary, trigger the event (such as a button click) that will run the client script you want to debug.

   The debugger stops at the breakpoint and displays the version of the page that is being processed by the browser.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

When you are debugging pages that contain both server and client script, remember that server script can potentially insert HTML text and client script into the page. For example, a few lines of server script can dynamically create a large table out of database information, or can write or rearrange client script on the page.

After processing the page, the server removes the server script. As a consequence, the page being processed by the browser can look quite different than it does in the Visual InterDev editor with both server and client script in it. For details about how server script is processed, see "Understanding Script Processing" in Chapter 23, "Scripting Concepts."

Debugging Embedded Server Script

A special case occurs when you want to debug server script that appears inside a block of client script. In the following, server script first extracts a value from a form and stores it in a variable. Later, inside the block of client script, a block of embedded server script dynamically inserts the value of the variable into a client statement:

```
<%loginName = = Request.Form("loginName")%>

<SCRIPT LANGUAGE="VBScript">
Sub ShowWelcome
   txt = "<%=loginName%>"
   MsgBox("Welcome, " & txt)
End sub
</SCRIPT>
```

Note When you write server script inside of client script, the editor does not follow color-coding convention for server script.

If you set a breakpoint on the line with the embedded server script, it is not clear whether you want the debugger to stop on the server script (<%=loginName%>) or later in the client script (txt =). Visual InterDev therefore offers you two options:

- **Client breakpoint only.** The breakpoint is interpreted as a client breakpoint, and the debugger does not stop when the server is processing the embedded server script.
- **Server and client breakpoint.** The debugger stops twice — the first time when it processes the server script, and then again when it reaches the same line in client script.

The default is client breakpoint only. To stop both times, you enable a debugger option.

To enable breakpoints for embedded server script

1. From the **Tools** menu, choose **Options**, and then open the **Debugger** node.
2. Choose the option **Insert breakpoints in Active Server Pages for breakpoints in client script**.

When you are running the debugger and it breaks on embedded server script, it does not stop directly on the line containing the server script. Instead, it stops on the line of client script or HTML that immediately follows the preceding line of server script. In some cases, this can cause the breakpoint to appear many lines before the embedded server script.

For example, if you set a breakpoint on the embedded server script (<%=loginName%>) in the following example, the debugger will stop on the <HTML> tag.

```
<%loginName = Request.Form("loginName")%>
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub ShowWelcome
   txt = "<%=loginName%>"
   MsgBox("Welcome, " & txt)
End sub
</SCRIPT>
</HEAD>
```

This behavior occurs because when the server is processing the page, it ignores anything that isn't server script. The server doesn't "see" the lines of HTML and client script that precede the embedded server script. To the server, the beginning of the line containing the embedded server script is therefore the one immediately following the preceding line of server script.

To move to the embedded server script, use the debugger's **Step Into** command.

Debugging a Global.asa File

Debugging a Global.asa file differs from debugging .asp files in these ways:

- The Global.asa file cannot be a start page. To debug the Global.asa, you must request an ASP page. When the ASP page is requested, the server processes the Global.asa page.
- Procedures in a Global.asa file are event-driven, unlike inline script in .asp files.
- Procedures in a Global.asa file usually run only once per application or once per session:
  - The Application_OnStart procedure executes the first time that any page in an ASP-based application is accessed.
  - The Application_OnEnd procedure executes only when the application is shut down.
  - The Session_OnStart procedure executes only once per user session.
  - The Session_OnEnd procedure executes only when a user's session times out or when a script explicitly calls the Session object's Abandon method.

Enabling Debugging in the Global.asa File

Before you can debug script in the Global.asa file, you must enable debugging for ASP pages. You can manually enable debugging for your ASP application as described under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting." Alternatively, Visual InterDev can automatically enable debugging on the server as needed.

To automatically enable script debugging in ASP pages

1. In the Project Explorer, right-click the project and choose **Properties** to display the **Property Pages** dialog box.
2. Choose the **Launch** tab.
3. Under **Server script**, make sure **Automatically enable ASP server-side debugging on launch** is checked.

When this option is set, each time you start a debugging session Visual InterDev checks that the server is configured for debugging. This includes:

- Setting the Microsoft Internet Information Server (IIS) application to run in its own memory space (in COM terms, it runs "out of process").
- Enabling the IIS application's debugging options.

- Setting up a Microsoft Transaction Server (MTS) package to allow you to attach the debugger to the Web application. The package's identify is set when you first start the debugging session by asking you to provide your name and password.

  Note You can perform the first two steps manually on the server. For details, see "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

When you quit your debugging session, Visual InterDev restores the server debugging settings and out-of-prucess settings to their previous values.

Debugging Scripts in the Global.asa File

To debug the Global.asa file, you set up a debugger call from script in the file, and then start an ASP application.

To call the debugger from the Global.asa file

1. Open the Global.asa file in the editor, and then set a breakpoint in a script.

   – or –

   Include a statement that starts the debugger explicitly, such as Stop in VBScript or debugger in JavaScript. Place the statement at the beginning of the procedure, before any statements that you will want to step through.
2. Request an ASP page from the current project in the browser. This causes the IIS to run the Global.asa file.

Debugging the Global.asa File in Response to an Error

If there is an error in a Global.asa file (either a syntax error or run-time error), the server stops the procedure containing the error. If debugging is enabled for that ASP application, the server prompts you to start the debugger and displays an error message. If debugging is not enabled for the ASP application, an error message is sent to the client browser. In either case, the procedure containing the error stops.

Note If a debugger is installed on the server computer, the server does not pass error information through to the client. Instead, it displays an error message on the server computer's monitor. For more information, see "Just-In-Time Debugging of Server Pages" in Appendix A, "Troubleshooting."

You can launch the debugger in response to an error or debugger statement only if just-in-time debugging is enabled.

To enable just-in-time debugging

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Debugger**.
3. Under **Script**, check **Just-In-Time debugging**.

To debug Global.asa script in response to an error

1. When an error message appears prompting you to start the debugger, choose **Yes**.
2. A new instance of Visual InterDev is launched and you are prompted to open a project. If Visual InterDev is already running, a second instance is launched.
3. Open the project containing the Global.asa file to debug. If the project is already open in another instance of Visual InterDev, you cannot open it again, and Visual InterDev creates a new solution and project instead.

   The Global.asa file is loaded into the editor. If necessary, get a working copy of the page. If the project was already open, the Global.asa file is loaded as read-only file in the new project.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

If server debugging is not enabled for the application, errors are displayed in the browser as text in the page. In that case, open the project containing the page in Visual InterDev and start the debugger there, as described above.

Restarting the Global.asa File

You cannot stop and restart the script by refreshing a Global.asa file. To rerun Application_OnStart or Session_OnStart procedures, you must refresh the file or trigger the events again or otherwise restart the application.

To rerun all procedures in the Global.asa file

- In the editor, change the Global.asa file, and then deploy it to the server.

  – or –

  Stop and restart the Web server.

  This restarts the application and session, which runs the procedures in the Global.asa file again.

CHAPTER 27

Packaging Script as Objects

Microsoft Scripting Components (scriptlets) provide you with an easy way to create powerful, reusable controls. You create scriptlets using a Web scripting language such as Microsoft Visual Basic Scripting Edition (VBScript) and ECMAScript (a standard language based on JScript 2.0 and JavaScript 1.1). After creating your scriptlet, you can register and use it the way you would any ActiveX control.

Scriptlets:

- Provide script authors the ability to create controls.
- Provide script authors with access to a broad range of system services.
- Are easy to create and maintain.
- Are small and efficient.

Types of Scriptlets

You can create two types of scriptlets:

- **DHTML scriptlet** A Web page based on Dynamic HTML (DHTML) that you can use as a control in any application that supports controls, such as Microsoft Visual Basic or Microsoft Internet Explorer 4.0.
- **Server scriptlet** A scriptlet that you can use as a COM component. Server scriptlets contain only script (no HTML or other user interface elements). You can use server scriptlets as COM components in applications such as Microsoft Internet Information Server (IIS), Microsoft Windows Scripting Host, and any other application that can support COM components.

  Note Server scriptlets are not discussed in this documentation. For information on building and using server scriptlets, visit the Microsoft Scripting Web site at http://www.microsoft.com/scripting/.

DHMTL scriptlets are used as visual controls in an application, and are suitable for display in client applications such as Internet Explorer and Visual Basic. With DHTML scriptlets, you can use the graphical and hypertext capabilities of Web pages as a visually rich interface for an application, such as a calendar control that you can display in a Web page, in Visual Basic, or in another COM-based environment.

How DHTML Scriptlets Work

If you are creating a DHTML scriptlet, there are two aspects to your scriptlet. The first is the visual interface. You create this as you would for a Web page using DHTML.

The second aspect is the scriptlet's control interface: the properties, methods, and events that allow you to use the scriptlet as a control. You create these control elements using script that follows certain conventions for how methods and properties are exposed, and how events are trapped and forwarded to the host application.

For example, a DHTML scriptlet might define animation that moves and resizes text on the page. You use DHTML to define the text and set its color, size, and location. You then use scripts to define properties that another application can use to set the speed, and direction of the animation text, and methods that allow another application to start, stop, and pause the animation.

> **Note** To learn more about using Dynamic HTML in your Web pages, you can view the documentation for the Internet Client SDK located at http://www.microsoft.com/msdn/sdk/inetsdk/help/default.htm/.

After creating a DHTML scriptlet, you can use it as you would any control. You can add it to toolbars, draw it on a form, or add it to a Web page. In the host application, the DHTML scriptlet is hosted by a *scriptlet container object.* The container object creates a window for the DHTML scriptlet and provides a way for the host application to specify where the scriptlet displays, at what size, and so on. The scriptlet container object also provides the interface for you to set and get the DHTML scriptlet's properties, call its methods, and respond to its events.

Creating a DHTML Scriptlet

A DHTML scriptlet is an HTML file that contains ordinary DHTML to define the scriptlet's look and feel. The DHTML scriptlet also contains scripts to define how the DHTML scriptlet exposes properties, methods, and events.

To create a DHTML scriptlet

1. Create a new file.
2. Use DHTML to define the visual interface for your scriptlet.
3. Define the properties, method, and events for your scriptlet. For details, see "Defining Properties and Methods in DHTML Scriptlets" later in this chapter. You can also define event handling for your scriptlet. For details, see "Exposing Events in DHTML Scriptlets" later in this chapter.

After you have created the DHTML scriptlet, you can use it as you would any control. Microsoft Visual InterDev includes a feature that allows you to add the scriptlet easily to the Toolbox. For more details, see "Adding DHTML Scriptlets to Your Application" later in this chapter.

Defining Properties and Methods in DHTML Scriptlets

DHTML scriptlets can expose any number of properties and methods. You can do so in two ways:

- **Create a JavaScript public_description object** The object's constructor function explicitly defines what properties and methods the scriptlet exposes.
- **Use default interface descriptions** You expose properties and methods by simply creating functions that follow specific naming conventions.

In general, using a public_description object is associated with using JavaScript, and using default interface descriptions with using VBScript. However, the only strict language requirement is that if you use a public_description object, the public_description object itself must be created in JavaScript. Any additional functions, such as those to define properties and methods, can be written in any scripting language. Similarly, although using default interface descriptions is associated with VBScript, you can actually write the functions in any language.

Using a public_description object has several advantages. You can use any names for variables and functions that you want to expose as properties and methods, because you assign them public names in the public_description object. In addition, using the public_description object provides you with a convenient way to summarize and document the properties and methods that the scriptlet exposes.

Creating a public_description Object

A public_description object is a JavaScript object that provides run-time access to the properties and methods defined by the object's constructor function. Any behavior that is not explicitly declared in the constructor is not available.

Note You must use the name public_description in lowercase characters as shown, or the scriptlet's properties and methods will not be exposed properly.

A skeleton public_description with its constructor function looks like this:

```
<SCRIPT LANGUAGE="JavaScript">
   var public_description = new CreateScriptlet();
   function CreateScriptlet(){
      // statements here to define properties and methods
   }
</SCRIPT>
```

Note You do not use the constructor function to define events. For details, see "Exposing Events in DHTML Scriptlets" later in this chapter.

When you create the public_description object, the constructor function that you assign to it can have any name, as long as the corresponding function appears somewhere in the scriptlet. In the constructor, declare the properties and methods you want to expose using the syntax listed in the following table.

Declaration in constructor	Creates
this.*PropertyName* = *expression*;	Property as an expression. When the user gets the property, the value of *expression* is passed to the calling container. When the user sets the property, the value of *expression* is updated.

this.get_*PropertyName* = *function;*
this.put_*PropertyName* = *function;*
	Property as a function. The function called by the property definition can be in any active scripting language. To make a property read-only, do not provide the put_ function declaration; to make it write-only, do not provide the get_ function declaration.

this.*method* = *methodFunction*;	Method.

For example, the following scriptlet includes a marquee that exposes properties and methods for setting its text and color. The constructor defines two properties: version, a simple value, and marqueetext, created as a set of functions that illustrate how to get and set property values conditionally.

The constructor also defines the method togglecolor, which calls a function in the scriptlet called setcolor:

```
<HTML>
<HEAD>
<TITLE>Scriptlets!</TITLE>
<SCRIPT LANGUAGE="JavaScript">
// public_description object used to declare scriplet
var public_description = new scriptletobject();

// general object description
function scriptletobject()
{
   this.version = "1.0";              //property
   this.get_marqueetext = readtext;   //property (read)
   this.put_marqueetext = writetext;  //property (write)
   this.togglecolor = setcolor;       //method
}

function readtext(){
   if (mrql.innerText == ""){
      return "No text";}
   else{
      return mrql.innerText;}
}
```

```
function writetext(newtext){
   if (newtext != ""){
      mrq1.innerText = newtext;}
}

function setcolor(){
   if (f1.color == "#ff0000"){
      // red, make it blue
      f1.color = "#0000ff";}
   else{
      // not red, make it red
      f1.color = "#ff0000";}
}
</SCRIPT>
</HEAD>

<BODY>
<B>Sample scriptlet</B><br>

<FONT ID="f1" COLOR="red">
<MARQUEE ID="mrq1">Scriptlets are fun and easy!</MARQUEE>
</FONT>
</BODY>
</HTML>
```

Creating Default Interface Descriptions

If there is no public_description object defined in the scriptlet, the scriptlet container object exposes properties and methods using variables and functions in the scriptlet that follow certain naming conventions.

To expose scriptlet properties and methods, use these conventions:

- To create a read/write property, declare a variable scoped at the page level (that is, not defined inside a function) and give it a public_ prefix.
- To create a readable property as a function, define a function with the prefix public_get_.
- To create a writable property as a function, define a function with the prefix public_put_.
- To create a method, define a function with the prefix public_.

  Note When a property is exposed, its name in the host application does not have the `public_` prefix. For example, if you define a property called public_MyTitle in the scriptlet, its name in the host application is MyTitle.

The following table shows examples of variables and functions in a scriptlet, and the resulting interface that they expose in the host application.

Example	Exposed as	Used in container
`var public_Color = "red"`	Property	`vColor = SC1.Color`
		`SC1.Color = "blue"`
`function public_get_C()`	Property (read)	`x = SC1.C`
`function public_put_C(param)`	Property (write)	`SC1.C = "test"`
`function public_look(param)`	Method	`SC1.look(param)`
`function look()`	Not available (no `public_` prefix)	
`function get_C()`	Not available (no `public_` prefix)	
`var Color = red;`	Not available (no `public_` prefix)	
`var get_Color = red;`	Not available (no `public_` prefix)	

The following example shows a simple scriptlet containing a paragraph named "p1." The script block following the paragraph exposes a property called text and another called color, which is defined using get and set functions that show how to set a property conditionally. The scriptlet also exposes a method called settext.

```
<HTML><HEAD></HEAD>
<BODY>
<FONT ID="f1" color="black">
<P ID="p1">This is a paragraph of text.</P>
</FONT>

<SCRIPT LANGUAGE="JavaScript">
var public_text = p1.innerText;
function public_get_color(){
   return f1.color;
}
function public_put_color(color){
   f1.color = color;
}
function public_settext(newtext){
   p1.innerText = newtext;
}
</SCRIPT>
</BODY></HTML>
```

Note The scriptlet reserves the function name prefixes public_get_ and public_put_ to define properties. For example, if the scriptlet contains a function named public_get_MyText, it will be treated as a property called MyText. If you attempt to call the function public_get_MyText as a method using the syntax SC1.get_MyText(), an error will result, because the function itself is exposed only as if it were a property named MyText.

Exposing Events in DHTML Scriptlets

When you use a DHTML scriptlet in your host application, the application can be notified about events that occur in the scriptlet. A DHTML scriptlet can expose two types of events:

- **Standard DHTML events**, such as the onclick event and the onkeypress event.
- **Custom events**, which are events that you define or DHTML events not provided as standard events. For example, the scriptlet can fire an event when a property value changes.

Handling Standard Events

A DHTML scriptlet can expose these standard DHTML events:

onclick	onkeypress	onmousemove
ondblclick	onkeyup	onmouseup
onkeydown	onmousedown	

Note For information about scriptlet events, properties, and methods, refer to the Visual InterDev Reference in the online documentation.

Events that occur in the scriptlet are not automatically sent to the host application. To trap standard events in the host application, you must write handlers in two places: one in the scriptlet to trap and forward the event, and another in the host application to capture the event.

To pass an event from the scriptlet to the host application

1. Attach an event handler script to the event that you want to pass.
2. Within the event handler script, call the bubbleEvent method to forward the event to the host application.

   Note Before passing events to the container object, you can check the scriptlet's frozen property to be sure that the container object is ready to handle events.

If the scriptlet does not include an event handler for a specific event, that event will not be passed to the host application. Similarly, if the scriptlet includes a handler for the event but does not call the bubbleEvent method, the event will not be visible to the host application.

Note The scriptlet container object exposes all standard events at design time, even if the scriptlet does not contain a script that passes the standard event to the application. For example, in Microsoft Visual Basic, the code window for the scriptlet container lists all standard events, even if not all are available in a specific scriptlet.

The following scriptlet script shows how you can pass a text box's onkeyup event to the host application:

```
<INPUT TYPE=text ONKEYUP="passKeyUp()" NAME="t1" VALUE="">
<SCRIPT LANGUAGE="JavaScript">
function passKeyUp() {
   // script statements here if required
    window.external.bubbleEvent();
   // further script statements here if required
}
</SCRIPT>
```

Note When a standard event occurs, the host application sees the corresponding event triggered for the scriptlet container object as a whole. For example, the scriptlet might contain several buttons. If a user clicks one of the buttons (and if the scriptlet is forwarding the event), the host application receives only a general click event.

To pinpoint which control in the scriptlet triggered the event, you can create a custom event in the scriptlet that contains additional information about the event. For details, see "Creating and Handling Custom Events" below.

Additional information about a standard event, such as the location of the mouse pointer or the state of keys at the time the event was triggered, is available in the script container object's event property. For example, the following Visual Basic subroutine shows how you would capture the scriptlet's onkeypress event and display the key code of a character typed in a scriptlet textbox:

```
Sub ScriptContainer1_onkeyup()
   MsgBox "The character typed was " & ScriptContainer1.event.keyCode
   MsgBox "The shift state was " & ScriptContainer1.event.shiftKey
End Sub
```

In Microsoft Internet Explorer, the following script does the same thing:

```
<SCRIPT LANGUAGE=JavaScript FOR=document EVENT=onkeyup>
   alert("Key code = " + window.event.keyCode)
   alert("Shift status  = " + window.event.shiftKey)
</SCRIPT>
```

Creating and Handling Custom Events

You can expose custom events in either DHTML or automation scriptlets. Custom events allow you to:

- Send more detail about a standard event that occurred — for example, which of several buttons in the scriptlet was clicked.
- Notify the host application about non-standard changes in the scriptlet, such as when the value of a property changes.

- Notify the host application about DHTML events that are not among the standard events handled by the bubbleEvent method.

As with standard events, you must send the event from the scriptlet and capture the event in the host application.

To send a custom event in the scriptlet

- Call the scriptlet's raiseEvent method.

  Note Before passing events to the container object, you can check the scriptlet's frozen property to be sure that the container object is ready to handle events.

For example, the following shows how you can send a custom event called oncolorchange whenever the scriptlet's backgroundColor property is reset. The second parameter is an object reference to the window.event object, which will allow the container handler to get information about the object that fired the event.

```
<SCRIPT LANGUAGE="JavaScript">
function public_put_backgroundColor(value)
{
   window.document.bgColor = value;
   window.external.raiseEvent("event_onbgcolorchange",window.event);
}
</SCRIPT>
```

To handle a custom event in the host application

- Create an event handler for the onscriptletevent event.

The following is an example in Visual Basic that shows how you can determine what control triggered an event:

```
Sub ScriptletContainer1_onscriptletevent( ByVal txtTitle As String, _
    ByVal eventData As Variant)
   objName = eventData.srcElement.id
   MsgBox "The event " & txtTitle & " occurred in " & objName
End Sub
```

If your host application is Internet Explorer, use a script such as the following to capture the scriptlet event:

```
<SCRIPT LANGUAGE="JavaScript"
      FOR="s1"
      EVENT="onscriptletevent (event, obj)">
   msg = "Event fired in the scriptlet was " + event
   msg = msg + "\nand the ID of the object was " + obj.srcElement.id
   alert(msg);
</SCRIPT>
```

You can use a Select Case structure in the onscriptletevent event to take different actions based on different events.

Adding DHTML Scriptlets to Your Application

After you have created a DHTML scriptlet, you can use it in your applications. You must first create a scriptlet container object. This object creates a window for the scriptlet and provides a way for the host application to specify where the scriptlet displays, at what size, and so on.

To add a DHTML scriptlet to a host application

1. Create a scriptlet container object in your application and set its Name property.
2. Set the scriptlet container object's url property to the URL of the scriptlet you want to use.

   Important If you are adding the scriptlet to a Web page, do not set the url property to the URL of the current page. Doing so causes a recursive call to the page and will cause the browser to stop functioning.

If you are working with a Web page, you can use the <OBJECT> tag to reference the scriptlet. You can add a scriptlet to the Microsoft Visual InterDev Toolbox.

To add a DHTML scriptlet to the Toolbox

- In the Project Explorer, right-click the scriptlet's .htm file, and then choose **Mark As Scriptlet.**

  An <OBJECT> tag containing a pointer to that scriptlet is added to the Scriptlet tab of the Toolbox. (If this is the first scriptlet on the Toolbar, the Scriptlet tab is created.) You can then drag the scriptlet from the Toolbox onto another page and automatically create the <OBJECT> tag necessary to implement the scriptlet.

  Note When you add a scriptlet to the Toolbox, it includes the scriptlet's absolute URL. After you drag a scriptlet onto your page, you might need to modify the <OBJECT> tag's URL property in the Properties window or in Source view to make the link relative.

Alternatively, you can create an <OBJECT> tag yourself that references the scriptlet.

To refer to a DHTML scriptlet in an <OBJECT> tag

- Create an <OBJECT> tag with the following syntax, substituting the scriptlet's URL and name for *url/scriptletName*:

```
<OBJECT ID="MyScriptlet" TYPE="text/x-scriptlet" WIDTH=300 HEIGHT=200>
   <PARAM NAME="url" VALUE="url/scriptletName">
</OBJECT>
```

Working with the Scriptlet Container Object

In your application, the DHTML scriptlet appears within a scriptlet container object. This object provides you with an interface to the scriptlet. For example, when an event occurs in the scriptlet (and if the scriptlet is forwarding the event), your application sees the event occurring in the container object.

You can manage the appearance of the scriptlet by setting properties of the scriptlet container object. In some cases, you can set properties within the scriptlet's scripts to manage its own appearance.

You can also resize the container object from within the scriptlet by using a script to set the DHTML script object's pixelHeight and pixelWidth properties. For example, the following example shows how you can resize the scriptlet container when the scriptlet is first loaded:

```
<HTML ID="MyPage">
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub window_onload()
   MyPage.style.pixelHeight = 300
   MyPage.style.pixelWidth = 400
End Sub
</SCRIPT>
</HEAD>
```

If you change the .htm file after creating the control, the display in the control is not updated until the next time the page is read. This occurs when the application is run or if you change the url property of the control again.

> **Note** After the scriptlet has been initialized, the F5 key, used to refresh a page in Microsoft Internet Explorer, is not active in the scriptlet container object.

After creating an instance of the scriptlet, you can write scripts for it as you would for any other control. The object you are using to work with properties and methods is the scriptlet container object; the exact properties and methods you can use are defined by the scriptlet identified in the container's url property. If you are working in an environment that can display an object's properties and methods, such as Microsoft Visual Basic, you will not see the properties, because these are not exposed to the development environment.

For example, the following code in a Visual Basic form sets a property and calls a method in the page referenced by the ScriptContainer1 control:

```
Sub cmdColor_Click()
   ScriptContainer1.BackgroundColor="red"
   ScriptContainer1.UpdateText (Text1.Text)
End Sub
```

> **Note** In Visual Basic, you must pass a parameter to a scriptlet method even if the method does not require one, or errors can occur. For example, the following statement passes a placeholder parameter of zero to a scriptlet method that does not require parameters:
>
> ```
> ScriptContainer1.ToggleColor (0)
> ```

Before getting a scriptlet's properties or calling its methods, you must be sure that the scriptlet has been fully loaded by testing the container object's onreadystatechange event and readyState property, and the scriptlet's frozen property.

PART 6

Building Integrated Solutions

Part 6 provides information about remote debugging, deploying Web applications, and using Microsoft FrontPage and Visual InterDev together to develop Web applications.

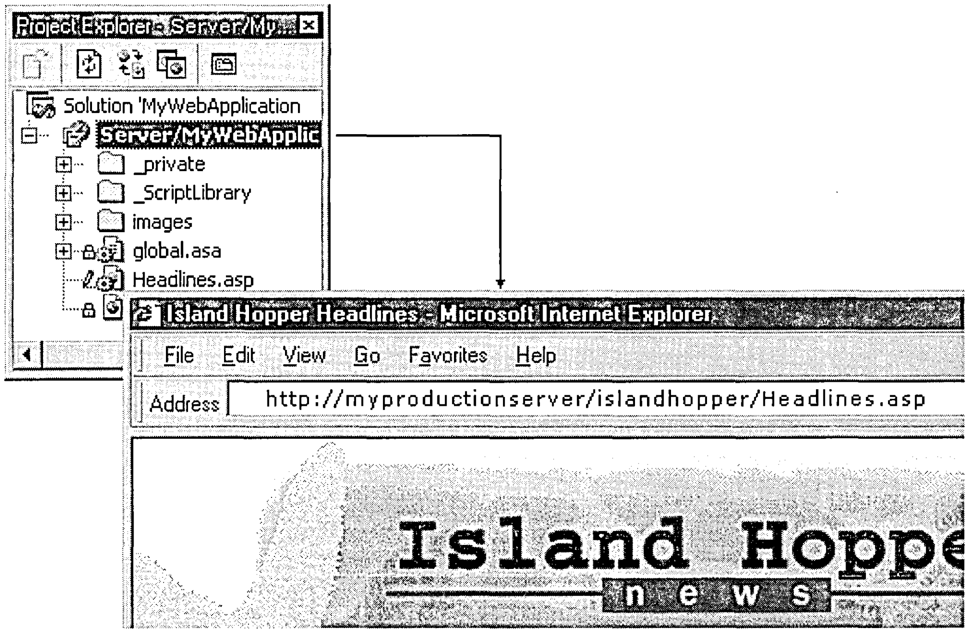
CHAPTER 28

Web Application Deployment

When your Web application has been tested, you are ready to deploy the application from the development server to the production server.

The production server is where your users see the live Web application. Deployment ensures your end users have access to a properly functioning version of the Web application. The figure below shows a Web application in a project that becomes the deployed version of the application in a Web browser.

Moving from the Development to the Deployment Environment


Project Explorer - Server/My...
Solution 'MyWebApplication
Server/MyWebApplic
_private
_ScriptLibrary
images
global.asa
Headlines.asp
Island Hopper Headlines - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address
Island Hoppe
news

http://myproductionserver/islandhopper/Headlines.asp

Considerations for successful deployment include the following:

- Production Server Capabilities
- Deployment Activities and Results
- Tips for Easy Deployment

Production Server Capabilities

Your production server and system capabilities determine how you deploy your Web application. One typical system includes multiple servers handling a combination of purposes. For example, one system dedicated to master Web application and database development, one dedicated to live Web applications, and one dedicated to production databases. You might need to determine if each has everything your Web application requires to run properly.

To be ready for deployment, your production server requires Microsoft Internet Information Server or another Web server installed. Depending on your Web application, the server may also need the following:

- Microsoft FrontPage Server Extensions to use the Microsoft Visual InterDev deployment features
- Microsoft Posting Acceptor 2.0 to use the Microsoft Visual Studio, Enterprise Edition deployment features
- Microsoft Transaction Server to use transaction packages

These server components are available from the Visual InterDev installation CD.

Deployment Activities and Results

Deploying your Web application accomplishes the following tasks:

- Specifying an application root on the Web server.
- Making and saving a copy of each file used by the application.
- Registering components marked as server components.
- Creating transaction packages for use by Microsoft Transaction Server.
- Performing the copy using the Secured Socket Layer.

Visual InterDev makes deployment of your Web application as easy as a simple copy. To do everything manually, you would need to copy files and folders and use the Web server administrator, as well as set up the Web server's application root.

You can use a variety of methods to deploy your application, but the quickest, if your production server has FrontPage Server Extensions, is copying to the production server through Visual InterDev.

Visual InterDev uses the FrontPage Server Extensions to communicate with the Web server. Your Web application is ready to run as soon as the operation is complete. For more information, see Chapter 30, "Deploying and Maintaining Web Applications."

If your production server does not have FrontPage Server Extensions installed, you can use the Posting Acceptor 2.0.

Tips for Easy Deployment

Before deploying your application, you can ensure that your application runs smoothly once deployed through proper preparation.

In preparing for deployment, you need to consider the following areas:

- Protect Links
- Ensure Data and Data Connection Portability
- Verify Production Web Server Capabilities
- Include All Web Items in the Web Project
- Mark Components for Server Registration and Microsoft Transaction Server Packages, If Appropriate

Protect Links

- **Use relative paths to specify links between application pages.** Relative paths for links between pages that reside on the same Web server are portable. You can move your application to any Web server and the links do not break because the Internet Host name or Web server is not specified in the link.

  For example, you would specify links to other pages and to images using the path relative to the current page as in the following: `<a href ="mypage1.asp"` and `<img src ="../images/myimage.gif">`.

- **Use absolute paths to link to external sites.** Absolute paths specify the protocol and the Internet Host name or Web server name.

  For example, if you link to the Visual InterDev site on Microsoft.com, you would use `<a href = "http://www.microsoft.com/vinterdev/default.asp">`.

Ensure Data and Data Connection Portability

- Make sure the production database is properly configured, deployed, and connected to the production Web server.
- Make sure the current data connection points to the production database you want to use during run time.

  If using a file Data Source Name (DSN), make sure the Use Connection String option was specified. If using a machine DSN, make sure a matching machine DSN has been specified on the production Web server.

  If you did not use the data environment to specify recordsets, you may need to update the connection information for each page referencing data. For more information about data connections, see Chapter 18, "Database Concepts."

Verify Production Web Server Capabilities

- Check that the production Web server has all of the necessary software for the Web application to run correctly, such as FrontPage Server Extensions and ODBC drivers.
- Verify that the security settings in the operating system and Web server are appropriate for your Web application. For general Web security, see "Security" in Chapter 6, "Web Project Concepts."
- If the Web server is Internet Information Server, you need to make sure the Identity is set to one of the administrators and not the Interactive User.
- If your Web project includes components that are part of a package used by Microsoft Transaction Server, you need to make sure that Microsoft Transaction Server is installed on the production server.
- If your Web project includes server components that you want to register on the server, you need to make sure you have permissions to register the components.

Include All Web Items in the Web Project

- Ensure that your Visual InterDev project contains all of the files the application needs to run properly. For example, it's easy to remember the pages that bring the core functionality, but you should also include any downloadable documents and other items that are offered on the Web pages.

  If you have files that are used by the Web application but not included in the Web project, you need to deploy them manually.

Mark Components for Server Registration and Microsoft Transaction Server Packages, If Appropriate

- Verify that application components that need to run on the server are marked as server components in the project. For more information, see "Deploying an Integrated Web Solution" in Chapter 30, "Deploying and Maintaining Web Applications."
- Ensure that you have appropriate permissions on the production Web server to register server components.
- Make sure any components requiring Microsoft Transaction Server have been added to a package in the project. For more information, see "Deploying an Integrated Web Solution" in Chapter 30, "Deploying and Maintaining Web Applications."

CHAPTER 29

Integration Tasks

Once you have a working Microsoft Visual InterDev project, complete with database access and scripting, you can take advantage of the integration features available to extend your project even further as you see fit.

Visual InterDev works in conjunction with other products in the Microsoft Visual Studio family and beyond to make your projects even more powerful.

Using FrontPage and Visual InterDev to Create Web Sites

Developers, writers, and designers can all work on the same Web project by using Microsoft FrontPage and Microsoft Visual InterDev in conjunction.

Visual InterDev provides developers with a robust set of tools for developing Web applications, while FrontPage provides a WYSIWYG environment for editing pages that doesn't require programming knowledge.

When you create a Web application in Visual InterDev, the same Web application files can be opened in a FrontPage project to edit, view, and preview files. Visual InterDev users can also open a FrontPage Web and work on the files using Visual InterDev tools.

The following sections contain issues to be aware of when using both FrontPage and Visual InterDev on the same Web application:

Server Extensions

Adding Navigation Bars

Applying Themes

Using Source Control

To open a Visual InterDev Web project in FrontPage

1. From the FrontPage **File** menu, choose **Open FrontPage Web**.
2. In the **Getting Started** dialog box, select **Open an Existing FrontPage Web** and choose the appropriate Web from the list.

   – or –

   From the **File** menu, choose **Open FrontPage Web**.

3. In the **Getting Started** dialog box, select **More Webs.**

   The **Open FrontPage Web** dialog box appears.

4. In the **Select a Web server or disk location** box, type the name of the master Web server and select **List Webs.**

5. In the **FrontPage Webs found at location** list box, choose the name of the Web project and select **OK.**

6. In the **Getting Started** dialog box, select **OK.**

When you work on Web project files in FrontPage, you are working in master mode. Once you save a file, any changes you make are available to all users. Visual InterDev, however, allows you to work in master mode or local mode.

Local mode allows you to make changes to Web project files on your local machine and later update the master Web server with your changes. Visual InterDev also allows you to work offline, which allows you to work on Web project files without being connected to a Web server.

Visual InterDev also uses icons in the Project Explorer to indicate the current state of a file in the Web project. For more information, see "Specifying a Project Mode" and "Working Offline" in Chapter 7, "Working Locally."

You can also open a FrontPage Web as a Web project in Visual InterDev. You must, however, specify in FrontPage that the Web project can run applications. Check the "Allow programs to be run" option available from the General Tab of the Properties dialog box.

To open a FrontPage Web in Visual InterDev

1. From the Visual InterDev **File** menu, choose **New Project.**

2. In the **New Project** dialog box, select **Visual InterDev Web Projects** and then choose **New Web Project.**

3. Select **Open**. The **Open or Create Web Project** dialog box appears.

4. In the **Web Server URL** text box, type the name of the FrontPage Web server.

5. In the **Web Name** text box, type the name of the FrontPage Web.

6. In the **Options** area, choose **Open existing Web** and select **OK.**

   Warning Opening .asp files in FrontPage can change the .asp file contents. Do not open .asp files in FrontPage.

Server Extensions

Visual InterDev and FrontPage use the same server extensions for Web projects.

> **Note** Make sure you are using the same version of the FrontPage Server Extensions for FrontPage Web servers and Visual InterDev Web servers in order to open, edit, and save Web project files successfully.

To identify Web server extension versions in Visual InterDev

1. In the **Project Explorer**, right-click the root of the Web project and select **Properties**.
2. Select the **Master Web Server** tab.
3. In the **Server properties** area, note the **Extensions version** information.

You can also easily find the FrontPage Server Extensions version information in FrontPage.

To identify Web server extension versions in FrontPage

1. In the FrontPage Explorer, select **Web Settings** from the **Tools** menu.
2. Select the **Configuration** tab.
3. Note the **FrontPage Server Extensions Version** information.

Adding Navigation Bars

You can use both the FrontPage Navigation Bar Bot and the Visual InterDev PageNavbar Design-Time Control to generate navigation links. Visual InterDev provides greater flexibility for creating navigation bars. For more information, see Chapter 13, "Designing Site Navigation."

Both Visual InterDev and FrontPage share the same navigation structure for a Web application. In FrontPage, you create and maintain navigation bar links in Navigation View. Each FrontPage Web has a single Navigation View and Navigation View can only have a single tree.

In Visual InterDev, you create and maintain navigation bar links in a site diagram. Each Visual InterDev Web project can have multiple site diagrams, and each site diagram can have multiple trees.

> **Note** In a single page, you should use only the FrontPage Navigation Bar Bot or the Visual InterDev PageNavbar control. Do not use both types of navigation bar tools in the same page.

Applying Themes

Visual InterDev uses a slight variation of FrontPage themes. For example, Visual InterDev themes do not use the active graphics and color options available in FrontPage.

Visual InterDev also applies themes differently than FrontPage. Visual InterDev uses cascading style sheets (.css files).

When you open a page in FrontPage that has a Visual InterDev theme, FrontPage will not recognize that a theme has been applied to the page. This can cause the page to appear differently in FrontPage and in some browsers.

If you open a page with a FrontPage theme in Visual InterDev, Visual InterDev will prompt you to replace the FrontPage theme with a Visual InterDev theme. For more information, see Chapter 17, "Customizing Page Appearance."

> **Note** For a consistent visual approach, use either the FrontPage themes or the Visual InterDev themes for a Web project. Do not use both types of themes in the same Web project.

Using Source Control

Visual InterDev and FrontPage allow you to use source control software with your Web projects.

In FrontPage, you can only check out files exclusively. Another user cannot check out and edit a file that you have already checked out. Visual InterDev, however, allows multiple users to check out the same file. For more information, see "Working Under Source Control" in Chapter 8, "Working with Multiple Developers."

Debugging Remotely

In addition to debugging scripts and processes that are running on your computer, you can debug those that are running on another computer. This is referred to as *remote debugging.* In Microsoft Visual InterDev, remote debugging is useful for debugging server scripts running in ASP pages on Microsoft Internet Information Server (IIS).

Ordinarily, to debug a server script in an ASP page, you would have to install Visual InterDev on the server and then debug scripts locally on that server. However, with remote debugging, you can attach the debugger running on your computer to a script running on the server and issue debugging commands across the network.

> **Note** If IIS and Visual InterDev are running on the same computer, you can debug server scripts using that copy of Visual InterDev, without remote debugging.

For the most part, remote debugging is similar to debugging locally, except for these differences:

- You must perform some extra setup steps before you can use remote debugging.
- Only one user can use remote debugging on a server at a time.

The information below explains how to set up remote debugging, and then how to establish a remote debugging session.

Setting Up Remote Debugging

To use remote debugging, you must perform several setup steps. The first step is to be sure that the proper debugging components have been installed on the server. A full server installation of the Visual InterDev will normally load the proper components. However, if you did not perform a full server installation, or if you are not sure, you can follow the procedure below.

To install debugging components on the server

1. On the server computer, start the Visual Studio Enterprise Edition setup program.
2. Under **Add/Remove Options**, choose **Server Applications and Tools**.
3. On the next page, select **Launch BackOffice Installation Wizard**, and then choose **Install.**
4. When the BackOffice Business Solutions wizard is displayed, choose **Custom** and then choose **Next.**
5. Proceed until you see the page offering you a list of components to install. Uncheck all components except the following:
   - **Remote Machine Debugging**
   - **Visual InterDev Server**
6. Proceed with the installation.

To configure DCOM on the server for remote debugging

1. From the Windows Start menu on the server, choose **Run**, and then in the **Open** box type **Dcomcnfg.exe** at the prompt.
2. In the Distributed COM Configuration Properties window, select Machine Debug Manager, and then choose Properties to display the **Machine Debug Manager Properties** dialog box.
3. In the **Security** tab, choose **Use custom access permissions**, and then choose **Edit.**
4. In the **Registry Value Permissions** dialog box, verify that **Allow Access** is selected in the **Type of Access** list, then choose **Add**

5. In the **Add Users and Groups** dialog box, select the server's name from the **List Names From** list. The server name typically is in the form \\server and appears at the top of the list.
6. Under **Names**, choose **Administrators**, choose **Add**, and then choose **OK**.
7. Return to the **Security** tab, choose **Use custom launch permissions**, and then choose **Edit**. Verify that **Allow Launch** is selected in the **Type of Access** list, choose **Add**, and then add administrators as you did in Steps 4 through 6.

   Return to the Distributed COM Configuration Properties window, select **MTS Client Export**, and then choose
8. **Properties.** Repeat Steps 4 through 7.
9. Return to the **Distributed COM Configuration Properties** window, select **Catalog Class**, and then choose **Properties**. Repeat Steps 4 through 7.

Enabling Debugging for ASP Pages

Before you can debug remotely, you must enable ASP debugging. You can manually enable debugging for your ASP application as described under "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting." Alternatively, Visual InterDev can automatically enable debugging on the server as needed.

To automatically enable script debugging in ASP pages

1. In the Project Explorer, right-click the project and choose **Properties** to display the **Property Pages** dialog box.
2. Choose the **Launch** tab.
3. Under **Server script**, make sure **Automatically enable ASP server-side debugging on launch** is checked.

When this option is set for your project, each time you start a debugging session, Visual InterDev checks that the server is appropriately configured for debugging. This includes:

- Setting the IIS application to run in its own memory space (in COM terms, it runs "out of process").
- Enabling the IIS application's debugging options.
- Setting up a Microsoft Transaction Server (MTS) package to allow you to attach the debugger to the Web application. The package's identify is set when you first start the debugging session by asking you to provide your name and password.

  Note You can perform the first two steps manually on the server. For details, see "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

When you quit your debugging session, Visual InterDev restores the server debugging settings and out-of-process setting to their previous values.

Running the Debugger Remotely

After the server has been configured for remote debugging, you can debug on the server in much the same way you do locally.

> **Note** It is highly recommended that you do not use Active Desktop mode of Microsoft Internet Explorer 4.0 when you are debugging.

There are two ways to start remote debugging:

- **Launch a project using the debugger** This works just like debugging locally, except that Visual InterDev automatically attaches the debugger to the server and starts remote debugging.
- **Attach to a process that is already running on the server** This is useful if you become aware of a server script problem while the application is running. You can detect an error in server script if the server returns error text instead of your page, or if the browser appears hung when you request an .asp file.

If you launch a project using the debugger, you can start debugging at the first line of script. Otherwise, you will only be able to debug script that runs after you attach to the process. In addition, if you launch a project, you will be able to edit the files you are debugging. When you attach to a process, you can see the files you are debugging, but cannot edit them there.

> **Note** Because remote debugging ties up process threads on the server, it is recommended that while a remote debugging session is in progress, other users avoid using the server.

Launching Remote Debugging

You launch remote debugging from within Visual InterDev.

> **Note** Before debugging server script, make sure you have enabled debugging as described above.

To launch a project using the debugger

1. In Visual InterDev, open the file to debug, and if needed, set breakpoints.
2. Make the page your project's start page. In the Project Explorer, right-click the page and choose **Set as Start Page**.
3. From the **Debug** menu in Visual InterDev, choose **Start**.

   > **Note** If another user is already debugging on the server, Visual InterDev displays an error message and you will not be able to start the debugger.

4. If this is the first time you have started the debugger since opening the current project, you are prompted to provide user information used to identify the debugging process on the server.

   Enter the domain and name (in the form *domain\name*) and password of a user with administrative privileges. For more details, see "Setting Up Remote Debugging" earlier in this chapter.

5. After the debugger starts and attaches to the document, proceed with debugging as normal.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

Attaching to a Running Process

If debugging is already enabled on the server for your project and you detect an error while the application is running, you can attach the debugger to it.

Note For information about setting server debugging options, see "Enabling ASP Debugging on the Server" in Appendix A, "Troubleshooting."

To attach to a running process

1. If it is not already open, start Visual InterDev.
2. From the **Debug** menu, choose **Processes.**
3. In the **Processes** dialog box, type the name of the computer to attach to. If you don't know the computer name, choose **Machine** and use the **Browse for Computer** dialog box to locate the server where you want to debug.
4. In the machine processes list, select your project, and then choose **Attach.**

   The process you selected appears in the **Debugged Processes** list.

5. Close the **Processes** dialog box and use the **Running Documents** window to select the ASP page to debug.

   You can step through scripts and test variables and expressions normally. However, to run a page you must navigate to it in your browser. In addition, you cannot edit the pages that you are debugging.

   Note For details about using debugger commands, refer to the Visual InterDev online documentation.

CHAPTER 30

Deploying and Maintaining Web Applications

When you are done developing and testing your Web application, you can make the tested version of your application available to end users on the production server.

Deploying a New Web Application

You can choose from a variety of methods for making your Web application available to your users. Your choice may depend on the software available on the production server. The following three methods help you choose what might work best for you.

- **Copy Project** is the easiest method and is found on the Project menu of Microsoft Visual InterDev. This method requires the Microsoft FrontPage Server Extensions on the production Web server. For more information about this feature, see "Deploying through FrontPage Server Extensions" later in this section.
- **Web Post** does not require the FrontPage Server Extensions.
- **Manual Deployment** is possible through the Windows Explorer and your Web server administration software. However, if you have all your files included in your Web project, Visual InterDev can set up everything for you.

For any method you use, you need to provide some basic information so that the application is copied to the correct Web server with the correct application name. The information you provide when you copy your Web application to the production server includes the elements of the URL for the Web application.

Typically you can't specify a server unless it already exists, but you can rename your Web application when you copy it to the server. For example, if you used a code name for your application during development, you can rename the application when you copy it. When deploying, you can use a new name for the application's URL that is easy for a user to remember.

Deploying through FrontPage Server Extensions

If your production server has FrontPage Server Extensions installed, you can quickly deploy your application by copying the project. The extensions are provided with Microsoft Internet Information Server.

To deploy through FrontPage Server Extensions

1. In the **Project Explorer**, select the project that points to the Web application you want to deploy.
2. From the **Project** menu, choose **Web Project**, and then **Copy Web Application**.
3. In the **Copy Project** dialog box, choose the copy of the application you want to deploy. Typically, you deploy the master version.
4. In the **Server name** box, enter the name of the destination Web server.
5. In the **Web project** box, enter the name you want the users to type for the URL.
6. Clear the **Copy changed files only** checkbox.

   Note If you want to use secure transmissions while deploying to the production server, select the **Connect using Secured Socket Layer** check box.

A new application root is created on the destination Web server and the files in the Web application are copied to that new folder. The name you specified in the Copy Web dialog box becomes part of the application's URL. You can now perform a final verification of the application.

Deploying without FrontPage Server Extensions

If your production server does not have FrontPage Server Extensions or if your application requires special deployment configurations, you can use the Web Post feature available with the Visual InterDev Enterprise Edition.

Deploying Manually

You might prefer to manually deploy some of your files. Because the detailed steps depend on the tools you actually use, they are not covered here. However, here is a list of tasks you should consider.

- Specify an application root in the Web server administrator.
- Set Web folder permissions.
- Make and save a copy of each file used by the application.
- On the production server, register components marked as server components.
- Create transaction packages for use by Microsoft Transaction Server.

Maintaining Existing Applications

Once your Web application has been deployed, you can èasily add or update individual pieces of the application. You might want to update your Web application with new pages, images, or other functionality. Or perhaps you made changes to files that are already deployed. You can copy over the currently deployed files with new versions.

Typically, even while the application files are browsed by an end user, you can make changes without affecting the user. Users are protected because a temporary copy of the file was downloaded to the user when it was requested. The new version is available when the user refreshes the file in the Web browser.

Note If your application includes server components and they are in use when you make the upgrade, you might want to make the application unavailable through the Web server, otherwise the changes may generate errors. For example, if a user is browsing a page that used a .dll or .ocx files, the files for the .dll or .ocx can be locked so you can't replace it while the page is in use.

If your application uses data connections, you need to disable the data connection while you make changes to the page and then enable the connection after you are done. For more information about data connections, see "Connecting to a Database" in Chapter 3, "Database Basics."

To upgrade pages in an existing Web application

1. In the **Project Explorer**, select the project that points to the Web application you want to deploy.
2. From the **Project** menu, choose **Web Project,** and then **Copy Web Application**.
3. In the **Copy Project** dialog box, choose either the copy of the application on the master Web server or the local Web server.
4. In the **Server name** box, enter the name of the Web server you want to use.
5. In the **Web project** box, enter the name you want the users to type in for the URL.
6. Make sure the **Copy changed files only** option is selected.

Only those pages that have been added or modified since the last deployment of the application are copied to the application root. You can then test the application on the production server.

Deploying an Integrated Web Solution

If your Web application uses components created using other Microsoft Visual Studio tools, you can add their outputs to your Web project and deploy them along with your Web files.

For example, a banking application may include a .dll, .ocx, or .class file created in another type of project. You can include the project's output as part of your Web application by adding the component to the Web project. If it is a server component, you need to mark it as such. If you want the component to execute within the scope of a Microsoft Transaction Server package, you can specify a package.

In deploying an integrated solution, you might perform the following tasks:

- Registering Server Components
- Packaging Components for Microsoft Transaction Server
- Deploying an Application with Components

Registering Server Components

If your component is designed to run on the server and not on the end user's computer, you need to make sure it is registered on the production server. You also need to register it on your master server during testing and design time.

You can manually register components or specify the component as a server component within your Web project for registration when you copy them.

Note In order to register components through Visual InterDev on a remote Web server, the Web server must have Microsoft Internet Information Server installed with the Visual InterDev RAD Remote Deployment Support option selected during custom installation.

To specify a server component

1. In the **Project Explorer**, add the component to your Web project.
2. Select the component you want to register on the server.
3. In the **Properties Grid**, select **Custom.**
4. In the **Component Installation** tab of the **Custom** property page, select **Register on server,** and then choose **OK.**

   Note If your Master Web server and local project are on the same computer the registry for that machine only notes the component once — not twice for the master and local version. If you remove the local copy, the registry entry for the component is also removed even though the component is marked as a server component in the project.

When you use the Copy Web Application feature, the component is automatically registered on the server. If the registration fails, you need to check your permissions on the server.

Packaging Components for Microsoft Transaction Server

If your Web application uses components or business objects that you want to be controlled using Microsoft Transaction Server, you can deploy them to the server using Visual InterDev.

For example, in your banking application, three business objects work together to transfer money from one account to another. The Transfer Object calls the Debit and the Credit Objects to transfer the money. For the transfer component, you need something that makes sure both of the other objects run successfully before either committing the changes or rolling the entire transaction back in the case that one part of the transaction fails.

Microsoft Transaction Server can manage your components during transaction processing. All you need to do in Visual InterDev is specify the components that make up a Microsoft Transaction Server Package and deploy them on the server.

To package components for MTS

1. In the **Project Explorer**, right-click the component you want to add to the package.
2. Choose **Properties** from the short-cut menu.
3. In the **Component Installation** tab, choose **Add to Microsoft Transaction Server package.**
4. In the **Package name** box, type the name of the package to add the component to.
5. Select the **Transaction support** option appropriate for your component.

   Note Typically the component's objects inherit the transaction specified by the client. You can set these options to specify otherwise.

To	Select
Set the component's objects to execute within the scope of a transaction regardless of whether the client has a transaction	Requires a transaction
Create a new transaction for the component's objects to execute within regardless of whether the client has a transaction	Requires a new transaction
Set the component's objects to execute within the scope of the client's transaction	Supports transactions
Set the component's objects to run without a transaction regardless of whether the client has a transaction	Does not support transactions

When you deploy the application, the package is provided to Microsoft Transaction Server.

Deploying an Application with Components

Once you have completed marking your components within the project, you can deploy your Web application.

If your production server has FrontPage Server Extensions installed, you can perform the following task.

To deploy a Web application with components

1. In the **Project Explorer**, select the project that points to the Web application you want to deploy.
2. From the **Project** menu, choose **Web Project**, and then **Copy Web Application**.
3. In the **Copy Project** dialog box, choose the copy of the application you want to deploy.

   Note If you work on a team, you typically deploy the master version because it includes the updated files from the team members.
4. In the **Server name** box, enter the name of the Web server you want to use.
5. In the **Web project** box, enter the name you want the users to type for the URL.
6. Select **Register Server Components.**
7. Choose **OK.**

A new application root is added on the destination Web server and the files in the Web application are copied to that new folder. The name you specified in the Copy Web Application dialog box becomes part of the application's URL. You can now test the application on the production server.

APPENDIX A

Troubleshooting

While working in Microsoft Visual InterDev, you may encounter unexpected behavior or results. You can use the following summary of common issues and concerns to understand Visual InterDev's features more fully and get the results you want. The major areas covered are:

- Web Projects
- Databases
- Editing and Scripting
- Debugging

Web Projects

This section covers issues with files, diagrams, site navigation, builds, and Microsoft Internet Information Server (IIS).

Working with Solutions and Projects

Typically, when you first create a project, you automatically create a solution with the same name. After you have created a solution, you can add additional projects. If you do not want the initial project and the solution to have the same name, you can rename the solution after it is created.

To rename a solution

1. In the Project Explorer, select the solution.
2. From the **File** menu, choose **Rename**.
3. In the Project Explorer, enter the new name for the solution.
4. The solution file (.sln) has the new name and the Web project file (.vip) remains the same.

Authentication Error Occurs During Creation or Copying of a Web Application.

You may encounter an authentication error if you are entering the number for the IP address instead of the computer name when you specify a server for a Web application. If your computer uses a proxy server only for external addresses, this error occurs because the system attempted to access the IP address through the proxy server even though it is actually a local or an intranet address.

You can avoid this error by performing one of the following tasks:

- Use the computer name instead of the number for the IP address when creating a project or copying a Web application.
- If your Web project is not using the system Internet settings, change the proxy options for your Web project by including the number for the IP address in the List of hosts without proxy in the Web Project option settings.
- Change the system Internet settings by including the number for the IP address in the List of hosts without proxy in the Internet properties for the operating system.

For more information about specifying the list of hosts, see "Connecting to a Proxy Server" in Chapter 9, "Adding Security."

Adding Personal Files to a Project

If your project has personal files in it, you can update the master Web application to include those files. If you are trying to update the master by selecting multiple files in the Project Explorer, select only the personal files. If you include other files that already exist on the Master server, the Add to Master command is not available.

If someone removes master files using Microsoft Windows Explorer or some way other than through the Visual InterDev project, the file will appear as a personal file in your project. When you refresh your project, you will be prompted to delete the file from your local Web application as well. If you want to recover the file, you need to choose "No." Use the Add to Master command to include the file in the Web application.

For more information, see "Updating the Master Web Application" in Chapter 7, "Working Locally."

Site Diagram Issues

- In local mode, if you add a new file to a site diagram and then save the site diagram, Site Designer automatically adds the new pages to the master Web application on the master Web server.

- In local mode, if you add a personal file to a site diagram and then save the site diagram, Site Designer automatically adds the personal file to the master Web application on the master Web server.
- Netscape Navigator and Netscape Communicator versions 4.0 through 4.03 implement cascading style sheets (CSS) differently than Microsoft Internet Explorer. Navigator and Communicator interpret relative URLs as relative to the document rather than to the linked .css file. To fix this:

  - Copy, but do not move, all of the images from the directory of the theme that you are using to the directory that contains the document that references the .css file.

    – or –

    Change the relative URLs in your CSS files to the appropriate absolute URLs.

Navigation Bars, Layouts and Themes Issues

- The PageNavbar design-time control (DTC) allows you to use custom HTML to generate navigation bar links for your Web pages. If you use custom HTML for navigation bar links, you must set the Orientation property to *horizontal* to specify a custom orientation for the navigation bar on the page.
- The text **[FrontPage VINavbar Component]** appears in place of navigation bar links if you do not have the proper server components installed on both your local machine and the master Web server. To use the PageNavbar control, you must have the VINavbar component installed on your local machine and on the master Web server. The VINavbar component is installed with the Microsoft FrontPage 98 server extensions.

To verify that the VINavbar component is installed

- On your local machine, search for the `vinavbar` folder. The `vinavbar` folder is installed by default in *drive*:\Program Files\Microsoft FrontPage\version3.0\bots\.

If you cannot find the `vinavbar` folder on your local machine, you must install the FrontPage 98 Server Extensions. See the Visual InterDev Readme file, Readmevi.htm, for installation instructions.

After you have installed the FrontPage 98 Server Extensions, you must recalculate links on your Web project.

To recalculate links for a Web project.

1. In the Project Explorer, right-click the root of the project.
2. From the shortcut menu, choose **Recalculate Links**.

- The Layout design-time control (DTC) header and footer text indicates where you should add content in a page that has a layout applied to it. In the actual layout template file, Layout.htm, the Layout DTC header and footer text means the opposite.
- Changes made in site diagrams or the PageNavbar control may not immediately appear on the navigation bars when you preview your pages. To fix this:
  - From the **Project** menu, choose **Web Project** and then choose **Recalculate Links**.

  The Recalculate Links command updates the navigation bar with the latest navigation structure information.

  Note Changes made in site diagrams or the PageNavbar control may also not immediately appear on the navigation bars if you have unselected the **Always update navigation structure information on local Web server** option on the General Tab (Project Properties Dialog Box).

- If you customize a theme file, such as a .css file, and later apply a theme from FrontPage, your customizations may be discarded. To prevent this:
  1. Rename the theme folder containing the customized theme files in the _Themes directory. For example, if you are using the Nature theme, rename the Nature folder to MyTheme.
  2. Rename the .inf file for the customized theme. For example, rename `Nature.inf` to `MyTheme.inf`.
  3. In the renamed .inf file, change the `title=` text to the new theme folder name. For example, change `title=Nature` to `title=MyTheme`.
  4. Reapply the theme to all the pages previously using the customized theme.
- If you specify a parameter or bookmark in the URL for an HTML page in a site diagram, the PageNavbar control will not generate navigation bar links for the .htm file. The PageNavbar control does not support parameters or bookmarks in the URL for .htm files. To prevent this:
  - If you have specified a parameter, create a new .asp file in the Project Explorer. Remove the .htm file from the site diagram and add the new .asp file using the **Add Existing File** command to specify the parameter for the .asp file.

    For more information, see "Adding Pages to a Site Diagram" in Chapter 12, "Designing a Web Site."
  - If you have specified a bookmark, create an .asp file with a parameter that redirects the browser to the bookmark in the .htm file. Remove the .htm file from the site diagram and add the new .asp file.

Link Diagram Issues

- In local mode, Link View displays link information for your Web project as if you have already updated the master Web server with your changes. The link diagram may therefore display link status that conflicts with the actual status of the Web application.

  Note In local mode, Link View may not immediately display changes to the link information in your files if you have unselected the **Always update navigation structure information on local Web server** option on the General Tab (Project Properties Dialog Box).

- If you attempt to view links for items contained in directories that start with the underscore character _, such as _myfiles, the items may appear broken, the items may not display the proper in links and out links, or the items will not appear in the diagram. To avoid this, do not create directories that use an underscore character _ as the first character in its name.

  Caution By default, Visual InterDev creates four directories that use an underscore character as the first character in its name: _Layouts, _private, _ScriptLibrary, and _Themes. Do not rename these directories.

- When you create a link diagram using the View Links command, you create a design-time view of your links. Dynamically generated links, such as links generated by ASP script or from a database, do not appear. To view dynamically generated links, use the View Links on WWW command and enter the full URL for the file. The View Links on WWW command provides a run-time view of the links in a Web application.

  For more information, see "Link Verification" in Chapter 11, "Site Design."

Broken Links Report Issues

- For best results, you should recalculate the links for the Web project before you run the Broken Links Report. Recalculating links ensures that you have the most current link information for the Broken Links Report. For more information, see "Repairing Links" in Chapter 16, "Maintaining Links."
- If you have created a directory that starts with the underscore character _, such as _myfiles, files contained in these directories may be excluded from the Broken Links Report or may be listed as unused in the Broken Links Report. To fix this, rename the directory so that it does not use an underscore character_ as the first character in its name.

Unable to Connect Remotely to Windows 95 or Windows 98 Web Server

Microsoft Windows 95 and Windows 98 support the Web servers installed with FrontPage or the Windows NT Option Pack. These Web servers are intended for local development. If you want to use a machine as a remote Web server, install Windows NT Server and the Windows NT Option Pack which includes Microsoft Internet Information Server.

IIS Administration Operations Fail with MTS Installed

Attempts to perform Web administration operations, such as setting application roots, when no user is logged on the Web server will fail under the following circumstances:

- If you are using Microsoft Transaction Server (MTS)

  – and –

- If during installation of MTS, you accepted the default option, Interactive User, as the option to run MTS sessions.

To set "allow Web administration operations" without someone logged onto the server

1. Open the Microsoft Master Console for the server.

   Note To open the Microsoft Master Console: From the Start menu, choose Windows NT 4.0 Option Pack, Microsoft Internet Information Server, and then Internet Service Manager.

2. Expand **Microsoft Transaction Server**.
3. Expand **Computers**.
4. Expand **My Computer**.
5. Expand **Packages Installed**.
6. Right-click **System** and from the shortcut menu, choose **Properties**.
7. On the **Identity** tab, choose a user that has administration privileges on the server.
8. Close the Microsoft Master Console.

For more information, see the documentation for Internet Information Server or Microsoft Transaction Server.

Build Order and Visual InterDev Projects

If you include your Web project in the Build Order dialog box, the project is automatically added to the end of the list so that it is built last. Visual InterDev projects must be built last. If you manually move the Visual InterDev project up in the list, your build may fail.

Visual SourceSafe Explorer and Visual InterDev

When using Visual SourceSafe with Visual InterDev, use the Visual SourceSafe Explorer only to view history or to rollback to a previous version. If you use the Visual SourceSafe Explorer to set options for checking files in and out, you may get unexpected results.

For example, you can set an option in Visual SourceSafe Explorer to remove local copies of a file on check in. If you choose that option, Visual SourceSafe considers the master copy to be the local copy and will remove that file from the Web server.

If you want to remove local copies after checking them in, use the options available from within Visual InterDev.

To remove local copies

1. From the **Tools** menu, choose **Options**.
2. In the **Options** dialog box, choose **Projects** and then **Web Projects**.
3. In the **Project Options** area, select **Remove local copies when checking in files**.
4. Choose **OK**.

Databases

This section covers issues with connecting to databases through ODBC, data binding and Recordset controls.

Adding a Data Connection

You can use the Add Data Connection command to add a data connection to a project. This command is available from within the Project Explorer when you select a project or the data environment.

The Add Data Connection command allows you to create a data source name (DSN) and define a data connection using this data source, as described in “Connecting to a Database” in Chapter 3, “Database Basics.”

You can also use the ODBC Data Source Administrator, available in the Control Panel, to create DSNs. These DSNs will be available in the Select Data Source dialog box, and you can use them to make data connections in Visual InterDev.

However, you must do this using the Add Data Connection command; the DSNs created using the ODBC Data Source Administrator don't become data connections until you use the Add Data Connection command to create the connection from the data source.

For more information about using data connections with Visual InterDev, see "Connecting to a Database" in Chapter 3, "Database Basics," and Chapter 18, "Database Concepts."

Data Binding

If you want your Web page to display data from a database, you must place a Recordset design-time control on the page, and set its properties to display a set of records from a database to which you've made a data connection. For more information, see "Data Binding" in Chapter 18, "Database Concepts."

Using Quoted Identifiers

There is a "Use quoted identifiers" check box in the Create New Data Source ODBC dialog box, which is on by default. If you turn this option off and have database object names containing quotes, problems will arise. If the identifiers you use for database objects may contain quotes, make sure to leave this option on.

However, if you select this option, you could have unexpected results when dealing directly with database objects, such as stored procedure or trigger code. If you select this option, be careful how you use strings in your script. It might be a good idea to use single quotes (') when possible.

Tooltips Show Errors for Recordset, PageObject, and FormManager Design-Time Controls

If you have an error in a Recordset, PageObject, or FormManager design-time control while you're designing it, you can use the mouse pointer to display a Tooltip describing the error.

The Tooltip is displayed when you move the mouse pointer over the control in Design view, over the fields in its property pages, or over the property fields displayed on the face of the Recordset control.

For example, if you create a row in the Actions Performed Before Transition table on the Action tab of the FormManager control's property pages, but don't fill in values for all the fields in this row, you'll generate an error for the control. This error ("Errors were found in the 'Action' property page of this DTC") will be displayed in a Tooltip if you move the mouse pointer over the FormManager control in Design view.

Exclamation Point Appears on Recordset Control

When you see a red circle containing an exclamation point on a Recordset design-time control (DTC), it means there is a data problem. Select the Recordset DTC so that it has the focus, and then move the mouse over the red circle and Visual InterDev will display a Tooltip that explains the problem.

Some typical causes are:

- The red circle appears in the Object Name list box when you change the Connection or Database Object setting. If you change the connection or database object, select a new object name.
- The red circle appears to the left of the Recordset name at the top left of the DTC when there's a problem with the data connection. Areas to investigate include:
    - Whether a connection can be made to the database server.
    - The data connection in the DataEnvironment under the Global.asa file in the Project Explorer.
    - The connection on the General tab of the Recordset Design-Time Control Property Pages dialog box.

Name of Recordset Appears as Red on DTC's Property Page

You bind data to a data-bound design-time control by setting the DTC's Recordset property to one of the Recordset DTCs that exists on the same page. If you set a data-bound DTC's Recordset property to an existing Recordset DTC, and then delete that same Recordset DTC from the page, the name of the deleted Recordset will appear in red on the General property page of the data-bound DTC.

To fix the problem, you must either add a Recordset DTC to the page that uses the same name as the deleted Recordset DTC, or change the setting of the data-bound DTC's Recordset property to an existing Recordset DTC.

Errors when Moving Quickly Through Database Records

If you are working with an ASP page containing a data-bound form and try to navigate quickly from one record to the next, you might see the following error:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e21'
```

This error results from an attempt to get the next record before the previous database request has completed. You can avoid this error by buffering the page containing the form, so that it does not display in the browser until the data has been completely fetched and the page composed. To set buffering, add the following command to the top of the page:

```
<% Response.Buffering=true %>
```

Editing and Scripting

This section covers issues with using the HTML editor, using design-time controls, and other controls.

Enabling the Scripting Object Model

Design-time controls function properly only if the scripting object model is enabled for the page on which they appear. If you drag a design-time control onto a page that doesn't already have the scripting object model enabled, a message box prompts you to add it. However, if you choose "No" and proceed, the control will not communicate with other controls on the page, and it will not display properly at run time. For details, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts."

Scripting with Design-Time Controls and Script Objects

When writing script to interact with script objects created by design-time controls, keep the following tips in mind:

- While scripting, press CTRL+J at any time to display the statement completion drop-down list. If you are in the context of a particular object, that object's properties, methods, and events are included in the list. If you are not in an object context (for example, you are on a blank line), the drop-down list contains a complete list of objects and their members.
- Treat the names of properties, methods, and events as case sensitive even if you are scripting in VBScript. In many cases, the names you use are evaluated by the scripting object model framework (which is largely in JavaScript), and are therefore case sensitive.

  Tip When you are working with JavaScript, statement completion will not recognize an object unless you have used correct capitalization.

- When you are working on an ASP page with the scripting object model enabled, you can refer to the current page using the object thisPage. If there is a PageObject design-time control on the page, the object thisPage appears in the Script Outline window and is available for statement completion. If there is no PageObject control on the page, you can still use the thisPage object in your scripts, but it will not be visible in the Script Outline window or available for statement completion.

  In client script, the object thisPage is available only if you have a page object on the page. But in client script, you use the thisPage object only for accessing the properties and methods for page objects. For properties, methods, and events of the current page in client script, you can use the DHTML document object.

- When working with the names of methods and functions, you must be careful to use parentheses correctly. If you use statement completion to write method calls, parentheses are not added to method calls.

  If you are scripting in JavaScript, you must provide parentheses if the method is being executed as a function, if the method call returns a value, or if the method returns an object reference. The following two examples illustrate this use:

  ```
  str = Textbox1.value()   // returns a value
  // In the following getPagingNavbar() returns an object,
  // and hide() executes a function
  Grid1.getPagingNavbar().hide()
  ```

  If you do not include parentheses, the method call is treated as a function pointer, which is used to pass the name of a function as a parameter. For example, if you call an object's advise method, you pass the second parameter (the function name parameter) as a function pointer without parentheses, as in the following example:

  ```
  <SCRIPT LANGUAGE="JavaScript">
  function setAdviseMethods(){
     Btn1.advise("onmouseover", changecaption);  //no (), function ptr
  }
  </SCRIPT>
  ```

  If you are working with VBScript, you must provide parentheses if the method call returns a value; otherwise, they are optional. For example, in VBScript, both of the following statements will work:

  ```
  Textbox1.hide()
  Textbox1.hide
  ```

  Be careful to include parentheses when invoking a method that returns an object reference, as in the following example:

  ```
  Grid1.getPagingNavbar().hide
  ```

  If you call an object's advise method in VBScript, you pass the name of the second parameter (the function name parameter) as a string, including the parentheses:

```
<SCRIPT LANGUAGE="VBScript">
Function setAdviseMethods()
   Btn1.advise("onmouseover", "changecaption()") ' note quotes and ()
End function
</SCRIPT>
```

Tip Remember that because the scripting object model uses JavaScript, you should treat method names as case sensitive, even in VBScript.

Using the Properties Window

The Properties window uses a grid to display all properties for the selected control or object. The following lists some tips for using the Properties window:

- **Check the selection** The drop-down list at the top of the window displays the current element's type. If you are not seeing the properties you expect, check this. In a few cases (such as when you are working with HTML table elements), the list displays a hierarchy of elements that you can use to select the correct one.
- **Categorize properties** Buttons at the top of the window allow you to display properties alphabetically or by category. Grouping properties by category makes it easier to find properties, such as inline style attributes.
- **Use the Custom option** Some elements, such as design-time controls, provide custom property pages that make it easier to set property values. If a custom property page is available, the (Custom) property appears first in the alphabetical list, and you can click the button in the text box for that entry to display the window.
- **Use Help** Help is provided for most individual properties. To see Help, select the property in the grid, and then press F1.

Previewing with the Quick View Tab

The Quick view tab of the HTML editor processes only the client portions of a page: HTML text and client scripts. Although you can preview .asp files in Quick view, the editor does not perform a round trip to the server, therefore:

- No server script is executed.
- Design-time controls whose target platform is Server will not appear.

Cutting and Pasting HTML Text

When you cut or copy text to the Windows Clipboard from an HTML source — such as Microsoft Internet Explorer, Design view of the Visual InterDev editor, or the Visual InterDev Help system or sample applications — two versions of the text become available: an HTML version and a text version.

The HTML version uses HTML escape sequences for reserved characters such as <, >, and quotation marks. For example, if you copied "<MARQUEE>" to the Clipboard, the HTML version would be "&lt;MARQUEE&gt;". The text version contains an exact copy of the original text you cut or copied.

When you paste, you can choose either version. To paste the HTML version, choose Paste from the View menu or the right-click menu. To paste the text version, choose Paste as Text from the right-click menu. In general:

- In Design view, choose Paste if you want to see the actual text such as "<MARQUEE>". Choose Paste As Text if you want to create the tag. Tags do not appear in Design view, so the text you see might not appear to be the same as what you cut from another source.
- In Source view, choose Paste as Text.

Cutting or Copying in Design View of the HTML Editor

If you cut or copy an element from Design view to the Windows Clipboard, it contains additional information used to manage that element. If you paste the element anywhere but in Design view, it might be pasted with an extra <DESIGNTIMEP> tag in it. Always check the results of a paste operation if the source was Design view.

Distinguishing Design-Time Controls and HTML Controls

Be careful to distinguish design-time controls and HTML controls on the Toolbox. The controls have similar names — for example, there are text boxes and buttons on both tabs. It can be particularly difficult to distinguish the controls when working in Design view, because HTML controls are displayed graphically there.

You will find it easier to distinguish them in Source view. There, HTML controls are displayed as simple HTML tags such as <BUTTON> or <INPUT>. Design-time controls are displayed either graphically or as <OBJECT> tags with additional information.

Working with Text View in Source View

You can view controls (design-time controls, Java applets, and other controls) in Source view as text, which is useful when you want to see the exact contents of a page, and for making changes quickly to a series of controls. The following tips will help you when working with text view.

- Viewing a control as text applies only in Source view. Design view and Quick view always render the graphical view of a control (if possible).
- You can set the default view for controls using the HTML tab in the Options dialog box.

- You can select text view for an individual control by right-clicking it, and then choosing Always View As Text. When you do, the VIEWASTEXT attribute is added to the control's <OBJECT> tag.
- You can choose also the View Controls As Text on the View menu to temporarily view controls in text view. You can then use the View Controls Graphically command to redisplay controls graphically (except those whose <OBJECT> tag contains the VIEWASTEXT attribute).
- If you choose Refresh from the View menu, the editor displays all controls in the current default view (as set in the Options dialog box) except those whose object tag contains the VIEWASTEXT attribute.

Using text view can affect how controls work on your page. Visual InterDev design-time controls do not function properly if they are displayed as text, because they cannot communicate with the scripting object model framework. Before you add a design-time control to a page, make sure that you have set options to view controls graphically. If you do inadvertently add a control to the page while the text view option is set, the HTML editor cannot create an instance of the control. You will see only the HTML <OBJECT> tag for the control, not the control itself.

Printing Pages Containing Design-Time Controls

When you print a page in the HTML editor, some design-time controls might not be displayed with the values you set. If you put more than one instance of a design-time control on a page — for example, if you put multiple Textbox or Button controls on a page — the second and subsequent ones will display their default values in the printout.

Creating Controls Inside Script Blocks

Do not drag controls — design-time controls, ActiveX controls, or any other objects — into a <SCRIPT> block in Source view. A <SCRIPT> block is always assumed to contain only script, so the editor does not parse for controls, and therefore cannot create instances of them inside the block.

If you do drop a control into a <SCRIPT> block, the editor creates an <OBJECT> block containing information about the control such as its class ID. However, no additional information is created, such as parameters or script, so the control cannot be instantiated and will not function correctly.

Non-Functional ActiveX Controls on the Toolbox

In some instances, you might see controls listed on the Toolbox ActiveX Controls tab that are not available. For example, you might see the options WalletAddress and WalletPayment on the Toolbox, but when you try to drag them onto a page, an error is displayed such as "can't show control graphically."

This can occur if you are working in Microsoft Windows NT and chose the "Browser Only" option when installing Microsoft Internet Explorer 4.0. This minimal installation does not copy the controls to your computer, so they are not available. However, a Windows registry entry for the control is created, so it appears on the Toolbox.

Using Server Objects from the Toolbox

The Server tab of the Visual InterDev Toolbox lists a number of objects that are commonly available on Microsoft Internet Information Server (IIS). These server objects are part of IIS, but are used heavily in ASP pages as part of Web solutions built in Visual InterDev. For example, Visual InterDev developers frequently create server-based database access using ActiveX Data Objects (ADO).

The list on the Server Objects tab is predefined — it is not constructed based on objects that are actually installed on a particular server. This allows the Toolbox to display the same list no matter what server you are working with at the moment. However, it also means that it is possible to add a control from the Toolbox to an ASP page that might not be available on the server when the page is run. You should always test pages with server objects on the production server.

To use server objects, drag them from the Toolbox onto your Web page. (Do not drag them into <SCRIPT> blocks.) When you do, the editor creates an <OBJECT> block with the correct RUNAT and PROGID attributes, and with an ID attribute. Some of the objects created this way require additional information that you typically supply using <PARAMETER> tags inside the <OBJECT> block.

When the page runs, the server creates an instance of the object in the <OBJECT> block. You can reference the object in your script using the object's ID. For example, if you drag a Browser Capabilities object onto your page, the editor creates an object tag that looks like this:

```
<OBJECT RUNAT=server PROGID=MSWC.BrowserType id=OBJECT1></OBJECT>
```

Tip Change the object's ID to something meaningful. For example, in this case you might change it to oBType.

After the object has been instantiated with the <OBJECT> block, you can reference it in your script using its ID. The following excerpt from a page shows how you might use the object created above:

```
Your browser is <%=oBType.browser%>, version <%=oBType.version%>.
<p>
Your browser supports these programming languages:
<UL>
<%If oBType.javascript Then%>
   <LI>JavaScript</LI>
<% End If %>
<%If oBType.VBScript Then%>
   <LI>VBScript</LI>
<% End If %>
</UL>
```

Note If the object you are working with is available on the computer where Visual InterDev is installed, you will get statement completion for the object's properties, methods, and events when you are scripting. For example, if Visual InterDev is running on the same computer as IIS, Visual InterDev has access to the objects on the Server Objects tab.

You can find information about using server objects and about individual objects in the Microsoft Developer Network and elsewhere, as listed in the following table.

For information about	See this location in MSDN and elsewhere
Using server objects	Tools and Technologies Active Server Pages Using Scripting Languages Using Components and Objects
ADO Command ADO Connection ADO Recordset **Note** Server ADO objects are not part of the Visual InterDev data-binding model. To bind Visual InterDev design-time controls to a database, use the Recordset design-time control.	Database and Messaging Services Microsoft Data Access SDK Microsoft ActiveX Data Objects ADO Programmer's Reference Getting Started with ADO Getting Started With ADO 2.0

(continued)

For information about	See this location in MSDN and elsewhere
Ad Rotator Browser Caps Content Linking My Info	Platform SDK Internet/Intranet/Extranet Services Active Server Pages Installable Components for ASP See also "Creating Portable Script" in Chapter 25, "Scripting with HTML Elements." For an example using the Ad Rotator object, see the Random Ad Sample in the Visual Interdev online documentation.
Dictionary FileSystem	Visual Studio Documentation Reference Language References VBScript Language Reference Objects
CDONTS NewMail CDONTS Session	Platform SDK Internet/Intranet/Extranet Services Active Server Pages Installable Components for ASP Collaboration Data Objects for NTS Component
Index Server Utility Index Server Query	These objects are described in the Index Server documentation. To display the documentation, follow this path from the Windows Start menu on the server where IIS is installed: Programs Windows NT 4.0 Option Pack Product Documentation In the documentation, follow this path: Microsoft Index Server Building Search Forms Active Server Pages
MSMQ MSMQ Mail	Platform SDK Networking and Distributed Services Microsoft Message Queue Server (MSMQ) MSMQ Reference MSMQ ActiveX Components MSMQ Mail ActiveX Components

Passing Parameters for Page Object Methods

When you pass parameters to method on page objects, and if the call results in a trip to or from the server, the parameters are converted to strings. This is required in order for the parameters to be passed via http protocol between pages.

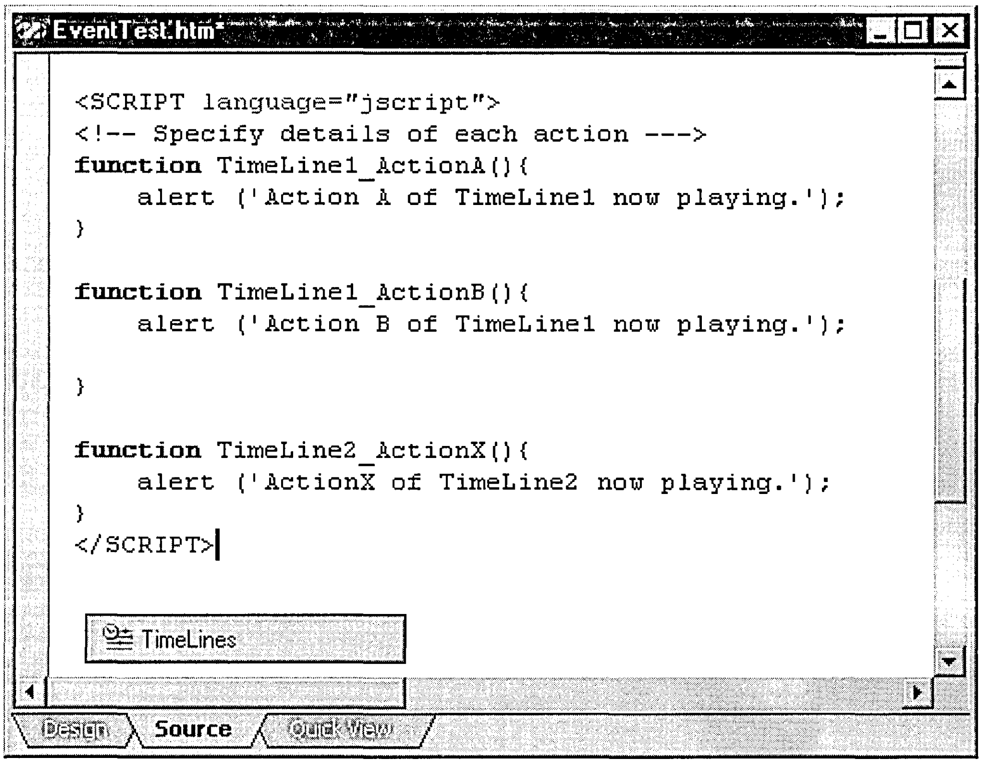
If you write a method that takes parameters that require a particular data type, you should always check the data type and convert it if necessary. For example, the following function in a page object takes a parameter that must be treated as a number. The function therefore converts it before using it.

```
Sub SubmitBid( BidAmount )
   errorCode = ""
   iBidAmount = CInt( BidAmount )
   If iBidAmount < 0 then
      errorCode = INVALIDBID
   End if
   ' etc.
End Sub
```

For more details, see "Writing Script for Script Objects" in Chapter 24, "Scripting with Design-Time Controls and Script Objects."

Specifying Timeline Events

After adding the Timelines control to a page, you need to add script that specifies the events you named in the control's property pages. For example, the control in the figure below has two timelines specified. TimeLine1 has two events, ActionA and ActionB. TimeLine2 has only ActionX. To simplify the example, the events cause an alert to appear.



```
<SCRIPT language="jscript">
<!-- Specify details of each action --->
function TimeLine1_ActionA(){
    alert ('Action A of TimeLine1 now playing.');
}

function TimeLine1_ActionB(){
    alert ('Action B of TimeLine1 now playing.');

}

function TimeLine2_ActionX(){
    alert ('ActionX of TimeLine2 now playing.');
}
</SCRIPT>
```

For more information, see "Timelines Design-Time Control" in online Help.

Testing Page Transitions

Since a page transition plays when you move from one page to another, you can only view the transitions in a Web browser. Quick View of the editor does not play transitions.

To view the results of enter transitions, use your Web browser to start with a page and then open the page with the transition.

To view the results of exit transitions, open the page with the transition and then open another page.

For more information, see "PageTransitions Design-Time Control" in online Help.

Debugging

This section includes information about debugging client and server script as well as notes about SQL debugging.

Enabling ASP Debugging on the Server

You can debug script in an ASP page only if debugging is enabled for your application (project) on the IIS server. If you start a debugger session by launching the page from within your project, Visual InterDev can automatically enable debugging on the server as described under "Enabling Server Script Debugging for ASP Pages" in Chapter 26, "Debugging Your Pages."

However, setting these options might not allow you to debug in these situations:

- You want to attach to a document (process) already running on the server.
- You want to launch the debugger in response to an error detected at run time (just-in-time debugging).

To debug in these situations, you must manually enable debugging for your application your application on the server.

To manually enable debugging for an IIS application

1. On the server, start Microsoft Management Console (MMC) using this path from the Windows Start menu: Programs\Windows NT 4.0 Option Pack\Microsoft Internet Information Server\Internet Service Manager.
2. Open the node for your server.

Right-click your project (application) and then choose **Properties**.

In the **Directory** tab under **Application Settings**, check **Run in separate memory space**, which causes your application to run out-of-process. Choose **Apply**. This might take a short while.

3. Choose **Configuration**, and then in the **Application Configuration** dialog box, choose the **App Debugging** tab.
4. Under Debugging Flags, check **Enable ASP server-side debugging** and **Enable ASP client-side debugging**, and then close all dialog boxes.

Disabling debugging is the reverse of Steps 3 through 5.

Do not leave debugging options on permanently, because they can affect performance. For details, see the section "Performance Issues while Debugging Server Script" later in this appendix.

Browser Displays Wrong Page when Debugging

If you launch the debugger and the browser displays the wrong page — that is, not the page you want to debug — return to Visual InterDev and make sure that the page you want to debug is set as the project's start page.

Just-In-Time Debugging of Server Pages

If a debugger is installed on the server, the server will not pass server script syntax or run-time errors through to the client. Instead, it will stop page processing and display a message on its own computer's monitor prompting to launch the debugger. The server can launch either the Script Debugger installed with Microsoft Internet Information Server (IIS), or if it is installed, the Visual InterDev debugger.

If you can watch the monitor on the computer running the server, you can respond to the message and debug locally on the server. However, if you cannot watch the server computer's monitor, you might prefer that the server pass error information through to the client so you can use remote debugging to find and fix the error.

To do so, you must remove all debuggers on the server. Uninstall the Script Debugger if it is installed. Similarly, make sure that Visual InterDev and other Visual Studio tools are not installed on the server. However, you must be sure that the Remote Debug Manager remains installed, or you will not be able to debug on the server at all.

Evaluating Expressions in the Debugger

While debugging, you can evaluate any expression dynamically by selecting it in the editor and then pointing the mouse cursor at it. The expression is evaluated and the results are displayed in a Tooltip. Alternatively, you can right-click the expression and add it to the Watch window.

However, you must be careful about evaluating expressions that can affect your debug session or even your Visual InterDev session. For example, if you select the expression Session.Abandon, the debugger will evaluate the expression and your current session will be abandoned.

Resetting Debugger Options After Abnormal Termination

If Visual InterDev is set to automatically enable server debugging, it reads the current debugger settings on your Microsoft Internet Information Server (IIS) the first time you debug an ASP page after opening a project. If required, Visual InterDev enables debugging, including moving the IIS application out of process. When you finish your debugging session, Visual InterDev restores the settings to what they were earlier.

However, if Visual InterDev quits while you are debugging — for example, if the computer loses power suddenly — the original settings are lost. When you next start Visual InterDev, it again reads the server debugging settings, but they might reflect the settings that were left when Visual InterDev quit abnormally.

If this has occurred, and you want the settings to be different from how they were left, you must manually reset them in IIS. In particular, if you want to disable ASP debugging and reset the application to run in-process, you must change these options yourself. For details, see "Enabling ASP Debugging on the Server" earlier in this appendix.

Performance Issues while Debugging Server Script

Enabling debugging on Microsoft Internet Information Server (IIS) can affect the performance of your projects in several ways.

Visual InterDev offers options that automatically enable server debugging on your IIS application (your Visual InterDev project). The first time you launch the debugger after opening the project, Visual InterDev checks the IIS debugger options for your project, and if they are not set, sets them. You can reduce this setup time slightly by using the IIS Microsoft Management Console (MMC) application to manually enable debugging for your application, as described in "Enabling ASP Debugging on the Server," earlier in this appendix.

When debugging is enabled on the server, it can affect the server's performance. It is recommended that you enable debugging only when you need it, and that you never enable debugging on a live production server. For details about debugging server script, see "Debugging Server Script" in Chapter 26, "Debugging Your Pages."

Do not use Active Desktop mode of Internet Explorer while you are debugging. Doing so can have two effects. First, the debugger will monitor all applications running from the Desktop, which can affect performance. Second, problems that might arise during debugging that would ordinarily require you only to reboot Internet Explorer might require you to reboot Windows.

Unable to Install SQL Debugging Component

If the BackOffice installation wizard reports that it cannot install SQL Debugging, check that you have installed Service Pack 3 or higher for the SQL server.

Troubleshooting SQL Debugging

If an error occurs while you are trying to use SQL debugging, try one of the following tips:

- If SQL Server is configured to log on as System Account, you will not be able to use SQL debugging.
- Errors on the server are written into an event log. To view the log, from the Windows Start menu on the server, choose Programs, then Administrative Tools, and Event Viewer. Open the Application log, and then look for SQLDebugging98 under Source. It is sometimes also useful to check the System log.
- If you are unsure whether you have the correct setup on your server, look for the file MSSDI98.DLL in the Bin folder of your SQL Server installation.

If you cannot get SQL debugging to work on a Windows 95 workstation, confirm that the client DCOM settings are correct.

To check the DCOM configuration on a Windows 95 client computer

- Using Regedit.exe, check the following settings in the Windows Registry under the key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\OLE:
  - EnableDCOM should be set to Y.
  - EnableRemoteConnet should be set to Y.

  If they are not set correctly, change them, and then restart the computer.

Evaluating SQL Variables in the Debugger

If you are debugging stored procedures or triggers, and if you click a variable with an "@" prefix, the debugger adds it to the Watch window without the @ prefix. Edit the name of the variable in the window to add the @ prefix back in.

Migrating from Visual InterDev 1.0 to Visual InterDev 6.0

You can easily migrate your Microsoft Visual InterDev 1.0 projects to Microsoft Visual InterDev 6.0.

The Web applications you created with Visual InterDev 1.0 will continue to run in Visual InterDev 6.0. In fact, if members of your Web development team are still using Visual InterDev 1.0, they can view the application files even after you have upgraded to the new version of Visual InterDev.

This cross-version compatibility is a key feature of Microsoft Visual Studio components. However, Visual InterDev 6.0 offers a greatly enhanced HTML editor, a new toolbox of design-time controls and integrated debugging that others will also want to use.

For information about important issues when migrating existing Web applications from Visual InterDev 1.0, see the following sections:

- Server Upgrades in Visual InterDev 6.0
- Data Connection Differences between Visual InterDev 6.0 and Visual InterDev 1.0
- Differences between Visual InterDev 6.0 and Visual InterDev 1.0 Design-Time Controls

Server Upgrades in Visual InterDev 6.0

Visual InterDev 6.0 includes enhanced server software capability. All the software needed to upgrade your server is contained on the Visual InterDev 6.0 installation compact disc. Because the upgraded server software is designed to work with Visual InterDev 1.0, you will still be able to open Visual InterDev 1.0 projects after installing the new server extensions.

Note The server extensions upgrade is required to run Visual InterDev 6.0.

Enhanced Server Extensions

To take advantage of many of the new features in Visual InterDev 6.0, you need to run the Visual InterDev 6.0 Server Setup to upgrade your server extensions. Many new and improved features in version 6.0 are dependent upon the latest server extension software. For example, the Visual InterDev 6.0 Scripts in Web Applications, Data Environment, Design-Time Controls, and Data Commands all use the upgraded server capability to simplify Web application development. These features are only supported after you run the Visual InterDev 6.0 Server Setup program.

Note Only the server requires the new server extensions. You do not need to load the server software on the client machine.

Microsoft Internet Information Server 4.0

In addition to the Visual InterDev 6.0 Server Setup, to run the Microsoft Visual InterDev debugger you need to install Microsoft Internet Information Server 4.0. Internet Information Server 4.0 is included with your Visual InterDev software and with Microsoft Windows NT 5.0.

You can use the Visual InterDev debugger to test scripts written in Microsoft Visual Basic Scripting Edition (VBScript) and Microsoft JScript, as well as applications written in Sun Microsystems Java and run using the Microsoft Java Virtual Machine (VM). For more information about the debugger, see Chapter 26, "Debugging Your Pages."

If you have installed an alternative scripting language that supports the Microsoft debugging protocol, such as REXX or Perl, you can also debug scripts in that language.

Data Connection Differences between Visual InterDev 6.0 and Visual InterDev 1.0

A data connection provides your Visual InterDev project with access to a particular database. Once you are connected to a database, you can display or edit data on your Web page in Visual InterDev.

In Visual InterDev, to connect to a database, you first create a data source name (DSN) for the database or choose an existing one. Then, you use the DSN to create a data connection and add it to your project. However, there has been a fundamental change in the way the data connections are stored in the Visual InterDev project with this new release.

In Visual InterDev 1.0, established data connections are identified by session variables within your project. You can see the session variables by opening your project's Global.asa file.

In Visual InterDev 6.0, data connections are identified in application variables within your project. You can see the application variables in your project's Global.asa file. You can also use the new Data Environment to edit the application variables for each data connection. For more information about using the new data environment, see "The Data Environment" in Chapter 18, "Database Concepts."

When you first open a Visual InterDev 1.0 project in Visual InterDev 6.0, the data connections are automatically converted from session to application variables. The session variables that were defined in Visual InterDev 1.0 remain in the Global.asa file, but they are only used with the DTC components that shipped in Visual InterDev 1.0.

Defining data connections as application variables is better, faster, and requires fewer resources than the version 1.0 method. When you use the version 6.0 application variables, each data connection can be shared between all the users accessing your application. This is an improvement over the version 1.0 session variables where each user must recreate the data connection variables.

There are, however, issues you should be aware of when working with Visual InterDev projects in different versions.

- Opening Visual InterDev 1.0 Projects with Data Connections in Visual InterDev 6.0
- Opening Visual InterDev 6.0 Projects with Data Connections in Visual InterDev 1.0

Opening Visual InterDev 1.0 Projects with Data Connections in Visual InterDev 6.0

When you open a Visual InterDev 1.0 project in Visual InterDev 6.0, all the project's data connections are automatically converted from session variables to application variables.

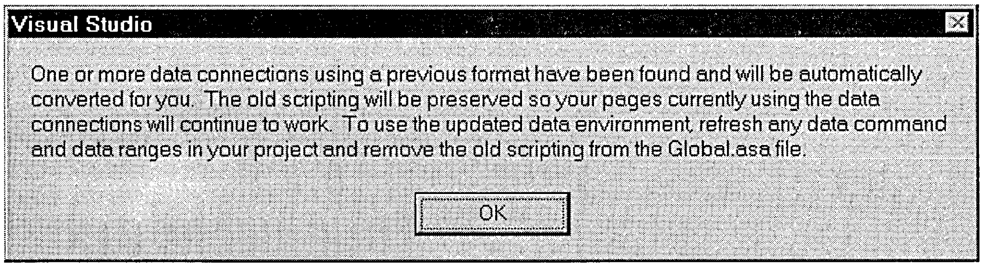
The session variables created in Visual InterDev 1.0 remain visible in your Global.asa file but are not used by Visual InterDev 6.0. Additionally, new application variables defining your data connections are added to your project's Global.asa file.

Therefore, data connection information actually appears twice in the Global.asa file so that both versions will continue to work properly.

In Visual InterDev 1.0, the data connection is defined by session variables in the Global.asa file, as in the script below.

```
Sub Session_OnStart
'==Visual InterDev Generated - DataConnection startspan==
'-Project Data Connection
Session ("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;
   DefaultDir=E:\;Driver={Microsoft Access Driver (*.mdb)};
   DriverId=25;FIL=MSAccess;ImplicitCommitSync=Yes;MaxBufferSize=512;
MaxScanRows=8;PageTimeout=5;SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
Session("NorthWind_ConnectionTimeout") = 15
Session("NorthWind_CommandTimeout") = 30
Session("NorthWind_RuntimeUserName") = "admin"
Session("NorthWind_RuntimePassword") = ""
==Visual InterDev Generated - DataConnection endspan==
End Sub
```

When you open your Visual InterDev 1.0 project in Visual InterDev 6.0, the following message informs you that data connections are being converted into application variables in the Global.asa file.



One or more data connections using a previous format have been found and will be automatically converted for you. The old scripting will be preserved so your pages currently using the data connections will continue to work. To use the updated data environment, refresh any data command and data ranges in your project and remove the old scripting from the Global.asa file.

When you view your project's Global.asa file in Visual InterDev 6.0, your data connection is now defined by application variables, as well as by the original session variables, as it is in the example script below.

```
Sub Application_OnStart
'==Visual InterDev Generated - DataConnection startspan==
'-Project Data Connection
Application ("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;
   ➥ DefaultDir=E:\;Driver={Microsoft Access Driver (*.mdb)};DriverId=25;
   ➥ FIL=MSAccess;ImplicitCommitSync=Yes;MaxBufferSize=512; MaxScanRows=8;
   ➥ PageTimeout=5;SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
Application ("NorthWind_ConnectionTimeout") = 15
Application ("NorthWind_CommandTimeout") = 30
Application ("NorthWind_RuntimeUserName") = "admin"
Application ("NorthWind_RuntimePassword") = ""
==Visual InterDev Generated - DataConnection endspan==
End Sub

Sub Session_OnStart
'==Visual InterDev Generated - DataConnection startspan==
'-Project Data Connection
Session ("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;
   ➥ DefaultDir=E:\;Driver={Microsoft Access Driver (*.mdb)};DriverId=25;
   ➥ FIL=MSAccess;ImplicitCommitSync=Yes;MaxBufferSize=512;MaxScanRows=8;
   ➥ PageTimeout=5;SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
Session("NorthWind_ConnectionTimeout") = 15
Session("NorthWind_CommandTimeout") = 30
Session("NorthWind_RuntimeUserName") = "admin"
Session("NorthWind_RuntimePassword") = ""
==Visual InterDev Generated - DataConnection endspan==
End Sub
```

If you open the Web application again in Visual InterDev 1.0, version 1.0 ignores the new application variables and uses the original session variables whenever a data connection is made.

Visual InterDev 6.0 ignores the existing session variables and uses the new application variables whenever a data connection is made.

Opening Visual InterDev 6.0 Projects with Data Connections in Visual InterDev 1.0

When you create data connections in Visual InterDev 6.0 projects, the data connections are defined in application variables within the project's Global.asa file. These application variables are the only place the data connections need to be defined if you are only going to open the project in Visual InterDev 6.0.

Visual InterDev 1.0 will not automatically interpret the data connections defined in a Visual InterDev 6.0 project.

If you create projects in Visual InterDev 6.0 that you want to open in Visual InterDev 1.0, you must manually redefine your project's data connections so they are Visual InterDev 1.0 compliant. To be Visual InterDev 1.0 compliant, your data connections must be defined in session variables as well as in application variables.

Redefining your project's data connections so they are Visual InterDev 1.0 compliant is easy and can be done in two ways:

- Adding Data Connections in Visual InterDev 1.0
- Copying Application-Defined Data Connections to Session-Defined Data Connections

Adding Data Connections in Visual InterDev 1.0

The simplest way to redefine your Visual InterDev 6.0 data connections so they are Visual InterDev 1.0 compliant is to open the project in Visual InterDev 1.0, and add the data connections as if they were new.

This procedure will automatically generate the session-defined data connections that are required for Visual InterDev 1.0. It will not modify the application-defined data connections that are required for Visual InterDev 6.0.

To add a data connection to your Web project in Visual InterDev 1.0

1. Open your Visual InterDev 6.0 project in Visual InterDev 1.0.
2. Select the project in **FileView**.
3. On the **Project** menu, click **Add to Project**, and then click **Data Connection**.
4. In the **Select Data Source** dialog box, choose the existing DSN that you connected to in Visual InterDev 6.0.
5. Log on to the data server if required.

Your data connection is now available for use with Visual InterDev 1.0.

To verify your data connection properties in Visual InterDev 1.0

1. Close the Global.asa file if it is open in the editor.
2. Expand the Global.asa icon in **FileView**.
3. Right-click the name of the data connection, and click **Properties**.
4. Verify that the data connection properties are the same as the properties you set for the data connection in Visual InterDev 6.0.

Your Visual InterDev 6.0 project now has data connections that will work properly in both Visual InterDev 1.0 and Visual InterDev 6.0.

Copying Application-Defined Data Connections to Session-Defined Data Connections

If you want to make your Visual InterDev 6.0 project work correctly in Visual InterDev 1.0 without first opening the project in Visual InterDev 1.0, you can manually copy the application-defined data connections to session-defined data connections within the Visual InterDev 6.0 Global.asa file editor.

To copy the application-defined data connections to session-defined data-connections

1. Open your project's Global.asa file in the Visual InterDev 6.0 editor.
2. Find the **Application_OnStart** script command that defines the data connection.

   The code will resemble the example below.

   ```
   Sub Application_OnStart
   '==Visual InterDev Generated - DataConnection startspan==
   '-Project Data Connection
   Application ("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;
   DefaultDir=E:\;
      ➥ Driver={Microsoft Access Driver (*.mdb)};DriverId=25;FIL=MSAccess;
      ➥ ImplicitCommitSync=Yes;MaxBufferSize=512;MaxScanRows=8;PageTimeout=5;
      ➥ SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
   Application ("NorthWind_ConnectionTimeout") = 15
   Application ("NorthWind_CommandTimeout") = 30
   Application ("NorthWind_RuntimeUserName") = "admin"
   Application ("NorthWind_RuntimePassword") = ""
   ==Visual InterDev Generated - DataConnection endspan==
   End Sub
   ```

3. Copy the entire application-defined data connection code and paste it back into your Global.asa file, so that the code defining the data connection appears twice.

4. Manually change the highlighted lines in the second instance of the data connection definition to create a session-defined data connection in your Global.asa file, as shown in the example below.

```
Sub Application_OnStart
'==Visual InterDev Generated - DataConnection startspan==
'-Project Data Connection
Application("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;DefaultDir=E:\;
   ↳ Driver={Microsoft Access Driver (*.mdb)};DriverId=25;FIL=MSAccess;
   ↳ ImplicitCommitSync=Yes;MaxBufferSize=512;MaxScanRows=8;PageTimeout=5;
   ↳ SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
Application("NorthWind_ConnectionTimeout") = 15
Application("NorthWind_CommandTimeout") = 30
Application("NorthWind_RuntimeUserName") = "admin"
Application("NorthWind_RuntimePassword") = ""
==Visual InterDev Generated - DataConnection endspan==
End Sub
Sub Session_OnStart
'==Visual InterDev Generated - DataConnection startspan==
'-Project Data Connection
Session ("NorthWind_ConectionString")="DBQ=E:\NorthWind.mdb;DefaultDir=E:\;
   ↳ Driver={Microsoft Access Driver (*.mdb)};DriverId=25;FIL=MSAccess;
   ↳ ImplicitCommitSync=Yes;MaxBufferSize=512;MaxScanRows=8;PageTimeout=5;
   ↳ SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;"
Session("NorthWind_ConnectionTimeout") = 15
Session("NorthWind_CommandTimeout") = 30
Session("NorthWind_RuntimeUserName") = "admin"
Session("NorthWind_RuntimePassword") = ""
==Visual InterDev Generated - DataConnection endspan==
End Sub
```

You have now successfully modified your Visual InterDev 6.0 project so that the data connections are Visual InterDev 1.0 compliant.

Differences between Visual InterDev 6.0 and Visual InterDev 1.0 Design-Time Controls

In Visual InterDev 6.0, several product enhancements combine to create a richer, faster, more robust environment for using design-time controls (DTCs). In Visual InterDev 6.0, you'll find a new set of DTCs that include enhanced design-time user interfaces for run-time effects. In addition, the philosophy behind the DTCs is expanded to allow you to code against the programming model as opposed to just coding script.

Your Visual InterDev 1.0 Legacy DTCs will continue to function when your Visual InterDev 1.0 project is opened in Visual InterDev 6.0, but the approach to using DTCs has changed since the previous version of Visual InterDev. For more information about using DTCs, see "Scripts in Web Applications" in Chapter 23, "Scripting Concepts"; Chapter 24, "Scripting with Design-Time Controls and Script Objects"; and "Installing a Script Library for Design-Time Controls," later in this appendix.

In general, the Visual InterDev 6.0 design-time controls offer you a richer, more visual editing interface for creating data-enriched pages. Data-bound controls make it simple to incorporate data in your ASP or HTML pages to interact with your database.

The controls included with this version of Visual InterDev allow you to target a wide range of browsers, or to narrow your focus to the rich dynamic HTML available in Microsoft Internet Explorer 4.0. For more information about the data-bound controls, see Chapter 24, " Scripting with Design-Time Controls and Script Objects."

The Data Environment

In addition to the increased variety and functionality of DTCs in Visual InterDev 6.0, creating and modifying data-related objects is now collected in one place: the graphical data environment. The data environment will affect the way you use and think about your DTCs.

In the data environment, you can drag and drop objects onto ASP pages to automatically create data-bound design-time controls. The data environment provides a standard interface for creating reusable data-related objects and for placing them on Web pages.

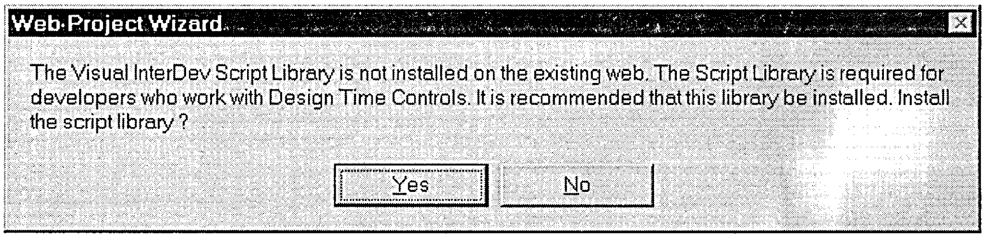
Before creating new projects in Visual InterDev 6.0, and in order to take full advantage of the Visual InterDev 6.0 DTCs, you should become acquainted with the data environment and explore how it will affect your project plans.

Installing a Script Library for Design-Time Controls

The script library is a repository for code generated by Visual InterDev 6.0 design-time controls. Although this library was not part of the Visual InterDev 1.0 release, it is required by the Visual InterDev 6.0 Programming Model.

When using Design-Time Controls in Visual InterDev 6.0 projects, it is important that the script library be present in order for your Visual InterDev 6.0 design-time controls to function correctly.

When you first open a Visual InterDev 1.0 Web application in Visual InterDev 6.0, the following message box will prompt you to install the script library.


Web Project Wizard

The Visual InterDev Script Library is not installed on the existing web. The Script Library is required for developers who work with Design Time Controls. It is recommended that this library be installed. Install the script library ?

Click **Yes** to install the Script Library.

If you choose not to install the Script Library, your project's design-time controls will not be able to find the code they are intended to generate and errors can occur.

If you choose No when asked to install the Script Library, then you later change your mind

1. Create a folder in your Web project folder called _ScriptLibrary.
2. Copy the contents of the Microsoft Visual Studio\VIntDev98\ScriptLibrary folder to the _ScriptLibrary folder.

   Note It is important that the new folder has the name _ScriptLibrary.

Once the new folder is created and the contents of the script library are copied into it, your Visual InterDev 1.0 projects will be able to use design-time controls when opened in Visual InterDev 6.0.

APPENDIX B

Visual InterDev Glossary

A

absolute positioning

The ability to specify the location of an element on an HTML page using x and y coordinates within the current window. In contrast, if an element is not positioned absolutely, its vertical position on the page is determined by where it appears in the HTML text, and its horizontal position is determined by properties such as the <CENTER> tag or the ALIGNMENT attribute.

ADO

ActiveX Data Objects. A cross-language technology for data access that exposes an object model incorporating data connection objects, data command objects, Recordset objects, and collections within these objects. The ADO object model provides an easy-to-use set of objects, properties, and methods for creating script that accesses data in databases.

alternate text

The text that is displayed as a substitute when certain HTML elements cannot be used. For example, if a browser does not display graphics, it will display the alternate text specified for a .gif file.

.asa file

Active Server page containing session global variables.

See *.asp file* and *Global.asa file*.

.asp file

An Active Server Page (ASP) file that includes server script to be executed by Microsoft Internet Information Server (IIS). The .asp extension on the file alerts the server that it should process the file before sending it to the browser. After the server script in an .asp file has been processed by the server, the file is sent to the browser, in the same manner as an .htm file.

B

broken link

A reference to an item that cannot be located because the URL is not valid, the item the link points to doesn't exist, or the server containing the item is busy or having other technical difficulties.

browser

Software that interprets the markup of HTML files posted on the World Wide Web, formats them into Web pages, and displays them to the user. Some browsers can also open special programs to play sound or video files in Web pages if you have the necessary hardware.

C

child page

A page with a parent page in a site diagram.

client script

A script that is executed by the browser on a user's computer. Client scripts are part of a page, and are sent to the browser when a user requests the page. Client scripts typically run in response to an event, such as when the page loads or when the user clicks a button, and are used to change the appearance of the page or to validate information entered by the user.

collapsed item

An item whose in and out links do not appear in the link diagram.

Compare with *expanded item.*

compile-time error

See *syntax error.*

control

An item, such as a button, on an HTML page that can be manipulated by the user to perform an action.

cookie

A small packet of information used to store persistent state information on the user's computer.

.css file

A text file that contains cascading style sheet information.

See *style sheet.*

D

data command

An object in the data environment that contains information about accessing a particular database object. For example, one command object might point to a table, another to a stored procedure, and a third to an SQL command.

Command objects are accessible in script, which allows you to open or execute the underlying data object in script. Data command objects are also reusable, so that if the underlying database changes, you can make a change to the command object, and all pages that reference the object will work properly.

For more information, see Chapter 18, "Database Concepts."

data connection

A collection of information required to access a specific database. The collection includes a data source name (DSN) and logon information. Data connections are stored in the data environment for a project and are activated when the user performs an action that requires access to the database.

For example, a data connection for a Microsoft SQL Server database consists of the name of the database, the location of the server on which it resides, network information used to access that server, and a user ID and password.

For more information, see "Connecting to a Database" in Chapter 3, "Database Basics," and Chapter 18, "Database Concepts."

data environment

A repository in your Microsoft Visual InterDev Web project that holds the information required to access data in databases. The data environment contains one or more data connections. Each data connection can reference one or more data commands that represent a method for querying or modifying the database.

Because data connection and data command information can be shared by multiple Web pages in your project, the data environment is managed as part of the project's Global.asa file.

For more information, see "The Data Environment" in Chapter 18, "Database Concepts."

database diagram

A graphical representation of any portion of a database schema. A schema is a description of a database to the database management system (DBMS), generated using the data definition language provided by the DBMS. A database diagram can be either a whole or partial picture of the structure of a database; it includes objects for tables, the columns they contain, and the relationships between them.

database object

A table, column, stored procedure, or other element of a database that you can work with as a discrete item.

database project

A project you can add to your Microsoft Visual InterDev solution that provides you with a live connection to a database and that includes tools for managing and querying the database. You can use a database project to create and modify tables, columns, views, indexes, queries, and other database objects.

A database project can exist in the same solution as a Web project, and you can be working in both projects at the same time. However, the two types of projects are not connected — to add database functionality to your Web application, you make data connections separately from the Web application's project.

dependent project

A Microsoft Visual Studio project with outputs that are specified as part of a Web project. For example, Microsoft Visual J++ projects can be added to a Web project as a dependent project and built while in Microsoft Visual InterDev.

deployment

The process completed by a developer to copy Web pages and associated files to a publicly available server; publication, propagation, or distribution of Web content. This process can be as simple as copying the files or may include building dependent projects.

For more information, see Chapter 28, "Web Application Deployment."

design-time control

A control in Microsoft Visual InterDev that presents a graphical interface at design time (including properties you can set in the Properties window), but which at run time generates HTML text, script objects, and sometimes other components to implement the functionality you are designing.

Design-time controls include standard user-interface elements — text box, label, check box, list box, command button, and so on, and controls that have no visual appearance but generate a run-time object (such as a Recordset object). Design-time controls also provide data binding capability.

For more information, see Chapter 24, "Scripting with Design-Time Controls and Script Objects."

DHTML

Dynamic HTML. An extension of HTML supported in Microsoft Internet Explorer 4.0 that exposes a Web page and all the elements on it as scriptable objects. DHTML allows you to dynamically change the appearance, content, and behavior of a Web page directly in client script, without running server script.

DHTML is useful in Visual InterDev for creating Web applications that incorporate multimedia effects and client access to databases.

document

A file associated with an application, such as Microsoft Word or Microsoft PowerPoint.

E

executable file

Files representing programs, applications, batch files, scripts, and DLLs.

expanded item

An item whose in and out links appear in the link diagram.

Compare with *collapsed item.*

external icon

The graphical representation of an item in Link View that is not part of the current project. An external icon shows an image of the item type superimposed over a globe representing the World Wide Web.

external item

An item that is not part of the current project.

external page

A page that is not part of the current Web project. External pages cannot have child pages in a site diagram.

extranet

An area of a Web site available only to a set of registered visitors.

F

filter

A means of excluding information that does not match a predefined set of specifications.

G

Global.asa file

A file maintained on a Microsoft Internet Information Server (IIS) for each application. This file is processed automatically by the server when:

- The IIS application starts and stops.
- Individual users start and stop browser sessions that access the application's Web pages.

The Global.asa file typically contains scripts to initialize application or session variables, connect to databases, send cookies, and perform other operations that pertain to the application as a whole.

global navigation bar page

A page that will appear on the global navigation bar as a link. You must have the PageNavbar design-time control (DTC) in the file or a layout applied to the site to get full use of this option.

global script

Client script that is not part of an event-handling procedure or called function, but instead is executed immediately when a page is processed.

H

horizontal layout

A link diagram that displays all *in* links to the left of the expanded item and displays all *out* links to the right of the expanded item.

.htm file

A file containing HTML text and possibly client script. Microsoft Visual InterDev also recognizes files with an .html extension.

See also *.asp file.*

HTML page

A Web page authored using HTML (Hypertext Markup Language) such as an .htm file or an .asp file.

I

in link

A link that points into an expanded item, indicating that the item at the other end of the line links to the expanded item.

independent page

A page with no parent pages or child pages in a site diagram.

inline script

Server script that is interspersed with HTML text or client script in order to feed information from the server into a document.

When referring to client script, “inline” is a synonym for global script.

internal icon

The graphical representation of an item in Link View that is part of the current project. An internal icon shows an image representing the item type such as HTML page, multimedia file, and so on.

internal item

An item that is part of the current project.

item

A resource that makes up a Web site. Link View represents the following types of items with graphical icons:

- HTML pages
- Stylesheets
- Active Server Pages (*.asp)
- Global.asa files
- Active layouts
- Images
- Image maps
- Audio files
- Video files
- Virtual reality files
- Executable files
- Data connections
- Data commands
- Data range headers
- Conditional range headers
- Generic designer controls
- Microsoft Word documents
- Microsoft Excel spreadsheets
- Microsoft PowerPoint files
- Other applications
- Other text files
- Mail to
- News
- Telnet
- Unknown

J

just-in-time debugging

The ability of Microsoft Visual Studio to automatically invoke the debugger when an error is encountered in a script.

For more information, see "The Script Debugging Process" in Chapter 23, "Scripting Concepts."

L

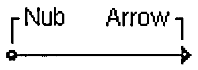
layout

A template for the way that information, navigation bars, and graphics are positioned on a page.

link

The relationship between items in a Web site. Links can be hyperlinks between pages or references to files that are included in a page, such as graphics. Link View determines the links for an item based on the HTML tags and attributes in a page.

Links are represented by a line in a link diagram. The origin of the link is indicated by a nub; the direction of the link is indicated by an arrow.



link diagram

A graphical representation of the structure of a Web site. Link diagrams use icons to represent items, such as HTML pages, in a Web site and lines to represent the links between items.

For more information, see "Laying Out Pages" in Chapter 17, "Customizing Page Appearance."

local file

A read-only copy of the master file, available on the local Web server. A local file is noted in the Project Explorer by a light blue padlock icon as shown in the .htm file example below:

If the file is under source control, it is noted by this icon:

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

local mode

The state of a project in which changes made to files are saved to the developer workstation's copies of the file residing in the local Web application and not to the master Web application. The master Web application must be updated explicitly by releasing the working copy or synchronizing the project. Local mode is noted in the Project Explorer by this icon:

If the project is under source control, you can update the master Web application by checking files in. The project is noted by this icon:

See also *master mode.*

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

local Web application

The collection of Web pages that resides on the developer workstation. These pages are used to create, modify, and test the pages before propagating them to the master Web application located on the Web server accessible to the intranet or Internet.

See also *master Web application.*

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

local Web server

Service, such as Microsoft Internet Information Server, for processing Web pages. Typically, this server is located on the developer workstation and provides the functionality needed to test all types of Web elements before they are propagated to the master Web application.

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

logic error

An error in a script that occurs when a script executes without syntax or run-time errors, but the results are not what you intended. For example, a script might prompt the user for a password, but then allow access to the application even if the password is wrong.

See also *run-time error, syntax error.*

M

marquee

An HTML block that encloses text and other information and scrolls within a defined area.

master file

The primary copy of a project file. Available on the master Web server for all members of a development team. Noted in the Project Explorer as a gray file graphic as shown in the .htm file example below:

If the file is under source control, it is noted by this icon:

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

master mode

The state of a project in which changes made to files are saved to both the local files on the developer's workstation Web application and the master files on the master Web server. Master mode is noted in the Project Explorer by this icon:

If the project is under source control, it is noted by this icon:

See also *local mode.*

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

master Web application

The collection of Web files are saved and stored on the Web server that is accessible to multiple developers and authors or, depending on your system, available to the intranet or Internet audience.

See also *local Web application.*

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

master Web server

Component of your Web development system that stores and processes the primary copies of the Web pages. This server might reside on the developer workstation or on a separate machine.

If you're using Microsoft Visual SourceSafe, you can verify the name and path for your master Web server.

From the Visual SourceSafe Explorer, right-click a file that is checked out, choose **Properties**, and then choose the **Check Out Status** tab. The **Computer** name is the name of your master Web server, and the **Folder** name is the absolute path to that machine.

See also *local Web server.*

multimedia file

An image, image map, audio, video, or virtual reality file.

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

N

navigation bars

A graphical or textual page element that incorporates navigation links to pages that are part of a site's navigation structure. You can design and modify a site's navigation structure in Site Designer.

navigation structure

The hierarchical relationship between pages in a Web site used to determine the links in the navigation bars for a site.

O

ODBC

Open Database Connectivity. A standard protocol for accessing relational databases based around SQL.

You install ODBC drivers for various databases that enable you to connect to the databases and access their data.

offline

Disconnected from the network and the master Web server. Offline mode is noted in the Project Explorer by this icon:

If the project is under source control, offline is noted by this icon:

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts," and "Working Offline" in Chapter 7, "Working Locally."

OLE DB

A database architecture that provides universal data integration over an enterprise's network, from mainframe to desktop, regardless of the data type.

OLE DB is a more generalized and more efficient strategy for data access than ODBC, because it allows access to more types of data and is based around Microsoft COM (Component Object Model).

out link

A link that points out of an expanded item, indicating that the expanded item links to the items it points to.

P

PageNavbar Design-Time Control

A design-time control that automatically generates navigation bar links based on the information stored in the site structure file for the Web project.

page transition

A DHTML special effect on a Web page that controls how one page replaces another visually within the Web browser. The transition occurs when the page is entered or exited.

parent page

A page with one or more child pages in the diagram.

pending item

An item that Link View is in the process of verifying. Link View has not finished searching for the item the link points to on the World Wide Web.

personal file

A file that exists only in the local Web application. Typically, this file has not been added to the master Web application. A personal file is noted in the Project Explorer by a blue flag icon:

production server

The Web processing unit with a Web service installed, such as Internet Information Server, that processes the live Web pages that are made accessible to the intranet or Internet audience.

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts."

protocol

A formal set of rules and formats that computers use to communicate with each other. FTP and HTTP are two examples of protocols used to transfer files between computers connected to the Internet.

proxy server

A Web server that acts as a security barrier between your internal network and the Internet, which keeps Internet users from gaining access to information on your internal network.

For more information, see "Security" in Chapter 6, "Web Project Concepts," and "Connecting to a Proxy Server" in Chapter 9, "Adding Security."

R

radial layout

A link diagram that displays the *in* links or *out* links related to the expanded item in a circle around the expanded item.

relationship

The type of link two or more pages have to one another in the site navigation structure. Generally, you can have parent-child relationships and sibling relationships. You can define the relationship between pages in a site diagram.

run-time error

An error in a script that occurs when a command attempts to perform an action that is not valid. For example, a run-time error occurs if you try to perform a calculation using a variable that has not been initialized. If a run-time error occurs, the script either stops or performs an exception routine.

See also *logic error* and *syntax error.*

S

script

Programs that run when users display your Web page. Script can be executed by the browser (client scripts) or the server (server scripts). The source code for script appears in the Web page itself.

For more information, see "Scripts in Web Applications" in Chapter 23, "Scripting Concepts."

script object

A run-time object created by the script generated with a design-time control. Setting properties of a design-time control establishes values that are then used at run time to create a corresponding dynamic script object. Script objects support properties, methods, and events you can script against using the scripting object model.

For more information, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts."

scripting object model

A set of objects with events, properties, and methods that provides a standard object-oriented programming model for working with Web pages. The scripting object model consists of a document framework and individual design-time controls, which together dynamically generate script objects at run time that you can script against.

The scripting object model provides a consistent event-based model that works transparently across the client and server and provides database-independent data binding.

For more information, see "The Scripting Object Model" in Chapter 23, "Scripting Concepts."

server script

A script in a Web page that is executed by the server before the page is sent to the browser that requested the page. When the page is sent to the browser, the server has already run the server script and removed it from the page. Server script typically performs database lookups, navigates to another page in the Web, or processes information entered by a user on an HTML form.

See also *client script.*

session

The course of a visit a user makes to a Web site.

For more information, see "Session Object" in the Active Server Pages documentation.

sibling page

A page that shares a parent page with another page in a tree.

site diagram

A graphical representation of the navigation structure of a Web site. Site diagrams consist of one or more trees of related pages.

site structure file

The file on the Web server that stores the navigation-related information for pages in a Web project. Use site diagrams to modify the site structure file for a Web project.

site transition

A DHTML special effect on a Web page that controls how one page replaces another visually within the Web browser. Site transitions occur when the page entered or exited from an external page.

solution

A collection of Web projects and dependent projects that organizes a Web application.

source page

The page you drag to create a child or parent relationship to another page in a site diagram.

staging server

The Web processing unit within a Web development system used to test the full functionality of all Web applications and Web elements before propagation to the live production server.

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts."

style sheet

A set of additional tags that describe the appearance of individual HTML tags. The style sheet tags describe the font, color, paragraph alignment, and other attributes for common HTML tags such as headings, paragraphs, and lists. These tags can exist within your document or as a separate text file.

When the style sheet is included with a Web page, all the tags on the Web page that have corresponding entries in the style sheet are automatically formatted according to the style sheet description. If the set of tags exists as a separate text file, it can be applied to one or more HTML documents.

Cascading style sheet (CSS) information is ignored by browsers that do not support this feature.

For more information, see "Site Consistency" in Chapter 11, "Site Design."

syntax error

An error in a script that occurs if you mistype a keyword, forget to close a multiline command (such as DO … LOOP), or introduce a similar mistake. If a script includes a syntax error, the script will not execute and an error message is displayed as soon as the browser or server processes the page.

See also *logic error* and *run-time error.*

T

target page

The page in the site diagram on which you drop the source page to create a child or parent relationship.

template

An HTML or ASP page, or a collection of pages, that acts as a master copy. Templates can be used to generate new pages, saving you the trouble of manually designing the look of the page, and to give a series of pages a similar look and feel. You can also switch templates for an existing page to quickly change its appearance.

When you create new files based on a template, the text in the template is replicated in the new file. A template can also include replaceable parameters that query the user for custom information when the new file is created. For information about creating your own templates, see "Creating Custom Templates" in Chapter 17, "Customizing Page Appearance."

A guide for Web pages in which elements such as heading locations, background image, colors, have already been designed.

theme

A set of graphics, fonts, and other page elements that combine to create a consistent visual design for a page.

For more information, see "Site Consistency" in Chapter 11, "Site Design."

transaction

A server operation that succeeds or fails as a whole, even if the operation involves many steps (for example, ordering, checking inventory, or billing).

tree

A group of related pages in a site diagram.

U

unknown item

An item that Link View has not attempted to verify; the item may or may not exist. Use the Verify command to have Link View determine if the link is valid or broken.

URL

The uniform resource locator, or address, used to identify a resource on the World Wide Web.

V

valid item

An item that Link View has determined exists.

virtual root

A directory that appears to be a subfolder to the root of the server but may actually exist in a subfolder that is any number of levels under the root or elsewhere on the network.

The name of the virtual root can be the same as the actual subfolder or may be an alias specified on the server. For example, in the URL http:/MyWebServer/MyWebApplication, "MyWebApplication" specifies a virtual root on the Web Server, "MyWebServer." The actual folder for MyWebApplication may be file:\\MyWebServer\Default Web Site\MyWebApp.

W

Web

A home page and its associated pages, images, documents, multimedia, and other items stored on a World Wide Web server or on a computer's hard disk.

Web application

A collection of elements that make up a Web site or distinct portion of a Web site organized under a virtual root. In Visual InterDev, Web applications are built from Web projects.

A Web project can reference a single Web application. For example, you can have a Web site with two Web applications: one that defines an online catalog and one that defines pages for administrating the database for the catalog.

See also *master Web application* and *local Web application.*

For more information, see "Project Architecture" in Chapter 6, "Web Project Concepts."

Web element

Anything used in a Web application for displaying or controlling Web content. Web elements can include pages, images, applications, forms, form elements, controls, and script.

Web project

A collection of files that specifies elements of a Web application. You build a Microsoft Visual InterDev Web project into a Web application.

Web project files are stored in two places, the server and the local machine. A project is also part of a larger container, the solution.

For more information, see Part 2, "Creating Web Projects."

Web server

A computer, usually on the Internet, that acts as a host for http and related Web-service software.

Web site

A collection of one or more Web applications organized under a single domain. A Web project can reference a single Web application within a Web site.

working file

A read/write copy of the master file, available on the local Web server. Noted in the Project Explorer by a pencil icon:

✎

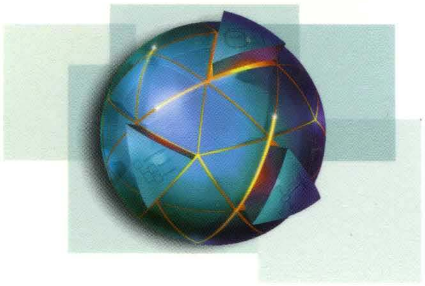
Typically, the file is under source control and has been checked out. In this case, it is represented by a red checkmark icon:

✓

For more information, see "Web File Processing" in Chapter 6, "Web Project Concepts," and Chapter 7, "Working Locally."

Index

Microsoft Visual InterDev 6.0 Programmer's Guide



The complete picture of database-driven Web development.

Microsoft Visual InterDev 6.0 provides a one-stop solution for building dynamic Web applications. The MICROSOFT VISUAL INTERDEV 6.0 PROGRAMMER'S GUIDE shows you how to use that solution to your best advantage. The content is taken from the online product documentation and is provided here in print for convenience and portability.

This comprehensive resource introduces you to the Microsoft Visual InterDev 6.0 environment and guides you in creating Web projects and utilizing databases. Topics covered include:

- **Basics**—Web, database, and editing fundamentals; walkthroughs
- **Creating Web Projects**—project concepts, working locally and with multiple developers, adding security, managing Web projects
- **Designing Sites**—Web concepts, designing site navigation, managing a site diagram, viewing and maintaining links, customizing page appearance, adding multimedia
- **Integrating Databases**—database concepts; viewing, modifying, and filtering data; working with stored procedures; managing database projects
- **Editing and Scripting**—scripting concepts, editing scripts, adding objects, debugging pages, making HTML dynamic, using Active Server Pages as objects, integration tasks and concepts, distributing Web applications

Microsoft Visual InterDev 6.0 is the leader in Web tools and technology, and the MICROSOFT VISUAL INTERDEV 6.0 PROGRAMMER'S GUIDE is the thorough, from-the-source resource that will enable you to unlock its full power.

mspress.microsoft.com

Internet Development/
Microsoft Visual InterDev

Microsoft Press

Part No. 097-0001982