WebObjects Tools and Techniques

by Theresa Ray of Tensor Information Systems, Inc.

Sponsored by Apple Computer, Inc.
Apple Developer Connection

media

WebObjects Tools and Techniques

by Theresa Ray of Tensor Information Systems, Inc.

Whether you are creating a new WebObjects application or maintaining an existing one, proficiency in using the ProjectBuilder, EOModeler and WebObjects Builder tools is essential. This survival guide enumerates tips and techniques that will allow you to enhance and tailor these tools to suit your programming needs and personal style.

The documentation provided with WebObjects provides a wealth of information regarding tool use and techniques. I recommend you read the documentation provided with each release to learn the features present in the current version. When looking through the documentation, don't forget to check the online documentation. There are many gems of information provided in the documentation installed with WebObjects that are not contained within the bound materials provided with the standard developer releases. To access this documentation on Mac OS X Server, view the following file in your web browser: file:/System/Documentation/Developer/DevelopersHomePage/DevelopersHomePage.html. On NT, select WOInfo from the WebObjects Program Group under the Start menu.

This survival guide supplements the documentation provided with WebObjects by providing additional tips and techniques, and by highlighting some of the key techniques that are also provided in the standard documentation.

Project Builder

Auto Indenting

Project Builder is capable of automatically indenting your code as you write it. This feature saves the developer time when writing new code, and helps to assist with code readability. A group of developers should use the same settings so that the code produced by all will have a consistent look and feel. To set up your preferences to produce code like this:

```
- (void) foobar
{
   id results = [ EOEditingContext objectsWithFetchSpecification:
   fetchSpecification ] ;

   if ( [ results count ] )
   {
      bar = foo ;
   }
}
```

Follow these steps:

1) In Project Builder, choose the "Edit" menu and select "Preferences".
2) From the pulldown list at the top, select "Key Bindings". Select the radio button for the option titled "Indent Always". The width specified below this option is not important.

3) From the pulldown list at the top, select "Indentation". In the first section titled "Auto-Indent Characters", check the boxes next to "Newline" and "{". Make sure the other boxes are unchecked.
4) In the second section titled "Spaces to Indent", change the settings for "Per level:" and "Wrapped line:" to 3 (or whatever your preference is). Make sure you press the "Enter" or "Return" key after typing each setting into the text field.
5) If you prefer braces matched and on a line by themselves as shown above, change the setting for "Solo left brace" to 3 (or whatever your preference is). Make sure you press the "Enter" or "Return" key after typing this setting into the text field.
6) If you prefer braces to begin at the end of the current line of code as shown below, change the setting for "Solo left brace" to 0. Make sure you press the "Enter" or "Return" key after typing this setting into the text field.

```
Else if (you like this style) {
}
```

7) If you like to have the matching right brace automatically inserted for you, select the line in the scrollview showing "Auto-Insert matching right brace".
8) Select the line in the scrollview showing "Indent wrapped lines".

The automatic indentation of newline and "{" characters, and the automatic insertion of a matching right brace take effect immediately after changing the preferences. Selecting text and pressing the "Tab" key will reformat code to the specified paramemters. However, the wrapped line preferences do not take effect until you create a new class.

Code Completion

By default, ProjectBuilder maps the F5 function key on Mac OS X Server and the F2 function key on NT to a completion feature. Begin to type a string and press F5 (or F2 for NT). ProjectBuilder will attempt to automatically complete the string based on similar strings already used.

To see a list of possible completions, begin to type the string and press Option+l (this feature is only available on Mac OS X Server) - that is a lowercase L, not a 1 or an I. A window will show up listing all possible completions. From this window you can double click on the appropriate completion or highlight the desired completion and select OK.

Comments and ProjectBuilder's Indexer

If you put braces, brackets or parentheses in comments in your code, be sure to keep them balanced. The indexer treats these as if they were in actual source. Mismatching these characters within comments will confuse the indexer, and prevent you from being able to double click on a bracket or brace to find its matching partner.

Hand Editing of ProjectBuilder Files

Hand editing of Project Builder's configuration files - PB.project and Makefile - is not recommended or usually necessary. Nearly all customization required for a project (such as modifying the compile flags or loading an additional object file) can be achieved by editing the Makefile.preamble or Makefile.postamble.

For example, to modify the compile flags for profiling, you could add the following to your project's Makefile.preamble:

```
OTHER_CFLAGS += -pg
```

Or to load an additional object file, you could add the following to your project's Makefile.preamble:

```
OTHER_OFILES += MyAdditionalFile.o
```

In another example, you can use your project's Makefile.postamble to customize the environment for installing compiled code by using the following:

```
INSTALL_AS_USER = $(LOGNAME)
INSTALL_AS_GROUP = other
```

If you choose to edit the PB.project file or Makefile by hand, be aware that ProjectBuilder will overwrite these files the next time that you change the project using ProjectBuilder, and you are likely to lose your hand edited changes.

ProjectBuilder automatically updates the PB.project and Makefile when you do any of the following:

1) Add a new class, file or framework
2) Rearrange your classes, files or frameworks
3) Change a header file's Project/Public Header setting

Listing Classes and Other Suitcases in Alphabetical Order

To reorder files in ProjectBuilder, click on the file name you wish to move, hold the Cntrl key down and drag the file to its desired destination. This does not work on NT.

Listing Methods in Alphabetical Order

As you browse a class in ProjectBuilder, the methods in the browser view are listed in the order in which the method appears in your code. To switch this to alphabetical order, choose the "Edit" menu and select "Preferences". From the pulldown list at the top, select "Indexing". Select the radio button next to either "Symbol name" or "Symbol type and name" to reorder your methods alphabetically.

Search Key Shortcuts

When searching for text, you can highlight the text you wish to find again and choose Cmd+E on Mac (or Cntrl+E on NT) to copy that text to the Find Panel. Cmd+G (Cntrl+G on NT) searches forward for the next occurrence, while Cmd+D (Cntrl+d on NT) searches backward for the next occurrence.

Another quick find option (available only on the Mac) is to choose Cntrl+s. A simplistic find panel will appear at the bottom of the window. As you type in this text field, a find is automatically executed.

SQL Generation by an EOF Application

Add the following command line option within the launcher to see the SQL generated in an EOF application:

```
- EOAdaptorDebugEnabled YES
```

Tool Bar Suppression

For those of you developing on a laptop or other computer where screen real estate is at a premium, you can suppress Project Builder's display of the toolbar icons by choosing the "Tools" menu and selecting "Hide Tool Bar". The following keyboard shortcuts may be used to access the equivalent toolbar functions:

Mac OS X Server Shortcuts:
Build - Cmd+Shift+B - automatically brings up the build panel and builds the project.
Find - Cmd+Shift+F - brings up the find panel.
Inspector - No keyboard shortcut.
Launcher - Cmd + Shift + L - brings up the launcher panel.
Class Browser - No keyboard shortcut.
Help - Cmd+? - Opens Project Builder Help files

NT Shortcuts:
Build - Cntrl+Shift+B - automatically brings up the build panel and builds the project.
Find - Cntrl+Shift+F - brings up the find panel.
Inspector - Alt+T then I - brings up the inspector panel.
Launcher - Cntrl + Shift + L - brings up the launcher panel.
Class Browser - Alt+T then use arrows to select Class Browser -> Show Class Browser
Help - Cntrl+? - Opens Project Builder Help files

Debugging Within ProjectBuilder

GDB

Debugger Crash Prevention

Before executing the debugger's "po" command on an argument, execute a "p" command first. If you try to use the "po" command on a non-object, you stand a good chance of raising an exception within the debugger itself.

Debugging Exceptions

When debugging your code, set a break point on [NSException raise] so that you can view the exception reason and userInfo, or print a backtrace.

Debugger Auto-Execution of Commands

Using the commands feature within the debugger, you can add code that is to be automatically executed whenever a break point is reached (for example you might want to execute a backtrace and po self). The procedure for using the commands feature is as follows:

1) Start the debugger and set the breakpoints for your project as usual.
2) Within the debugger, type "commands" followed by the breakpoint number(s) for which the commands should be executed (by default only the most recent breakpoint is used).
3) Type the commands to be executed when the specified breakpoints are reached - one per line.
4) Type "end".

If you specify "silent" as the first command, the standard message about "Breakpoint 1, -[Main sayHello], etc will be suppressed.

Using the command feature and ending each command set with "continue" allows you to easily see the status of your program at various points within its execution without interrupting the flow of the program and without having to include logWithFormat: or printf statements in your code.

Frequently Used Commands

You can define frequently used GDB commands by editing the gdb.ini file (.gdbinit on UNIX). To define a frequently used command, use the following syntax:

```
define commandName
set $tmpVar = gdbCommand
po $tmpVar
end

document commandName
   Line of documentation goes here
end
```

These commands will then be available as an extension of the default set of gdb commands. To see what commands are defined, type "help user-defined" within gdb. The commandName and the line of documentation you included will be displayed.

The following example defines a new command that prints the class name of a specified variable.

```
define nameOfClass
$tmpClass = [$arg0 class]
po $tmpClass
end

document nameOfClass
   Prints the class name of the receiver
End
```

JDB

Debugging Exceptions

When debugging your code, don't forget to use the try/catch exception statement so that you can use printStackTrace() (for Throwable exceptions) and getStackTrace() (for NSExceptions) to view information regarding the exception.

Printing Variables in JDB

To print variables using JDB, add the following to the project's Makefile.preamble:

```
OTHER_JAVATOOLS_FLAGS = -g
```

or

```
OTHER_JAVAC_FLAGS = -g
```

Also, don't forget to build with the debug option if you want to print variables using JDB. If you forget to build for debug, you will likely see the error message "'values' is not a valid local or class name".

Mixed Language Debugging

Debugging Using JDB and GDB Simultaneously

This topic is addressed in depth through the online documentation *Debugging Java Applications*. However it is worthwhile to reiterate the process for switching between JDB and GDB on the command line. Remember that the Java Virtual Machine (VM) is still running when the Java Debugger is debugging code in the stopped target, whereas gdb can only debug code when all processes involving the target (including the Java VM) are stopped. To switch between debuggers in a session (assumes that the target is executing Java code and both debuggers are active):

1) Choose the "Tools" menu and select Debugger -> Suspend Java VM to use the Java Debugger.
2) Click the continue button on the launch panel to continue execution.
3) Choose the "Tools" menu and select Debugger ->Suspend Process to switch to gdb.
4) Click the continue button on the launch panel to continue execution and re-enable the Java VM.

EOModeler

Changes and Synchronization with ProjectBuilder

Note that changes made to an EOModel are not automatically made to the source code for the updated class (i.e. adding an attribute to an entity through EOModeler does NOT create the corresponding accessor methods in the code for that custom class).

Customization of Generated Source Files

When you choose the "Generate Java Files…", "Generate Client Java Files…" or "Generate ObjC Files…" option in EOModeler, a default set of configuration files are used as the templates in the creation of the code. The default files (respectively named EOJavaClass.template, EOClientJavaClass.template, and EOImplementationFile.template along with EOInterfaceFile.template) can be found in the <APPLE_ROOT>\Library\PrivateFrameworks\EOModelWizard.framework\Resources directory. You can customize the code created by EOModeler by copying the appropriate files, updating them, and placing them in your project's directory. The locally stored files will be used by EOModeler when you generate the source code for each class.

Customization of the View

When viewing the details of a model or entity in EOModeler, use the "Add column…" button at the bottom of the view to customize the attributes displayed. By configuring the attributes displayed to be those with which you work most frequently, you can limit your need to use the inspector panel and reduce unnecessary window clutter and keystrokes.

NSNumber versus int

When generating code from EOModeler, it is easy to become confused trying to predict whether EOModeler will create an int or an NSNumber. For example, creating an attribute with value class NSNumber and an internal representation of integer may result in either and integer or an NSNumber being created in the code generated by EOModeler. Which representation is generated depends on whether you configure the attribute to "allow null" or not. In order to support null values for attributes, you have to use an object type, so EOModeler will generate an NSNumber. If you do not allow a null value for an attribute, EOModeler is likely to generate an integer or double type.

Many to Many Relationship Creation

To quickly create a many to many relationship between two tables, follow these steps:

1) Choose the "Tools" menu and select "Diagram View".
2) Shift+click to select the tables to join in the many to many relationship.

OR

1) Select the Model frisbee in the leftmost browser view (the right view should contain a list of entities for this model).
2) Cntrl+click to select the tables to join in the many to many relationship.

AND

3) Choose the "Property" menu and select "Join in Many-to-Many".

The two tables selected will be automatically joined in a many to many relationship. The flattened property and non-class relationship to the necessary intermediate entity will be automatically created and named.

WebObjects Builder

Binding Rules

When textually entering bindings using WebObjects Builder, the following rules apply:

1) Constant strings (such as "Fred") must be in quotes.
2) Variable and method names must NOT be in quotes.
3) Symbolic constants (such as YES and NO) must NOT be in quotes.
4) Keys must specify the full key path (to bind to the session variable sessionNumber you must specify session.sessionNumber).

Changes and Synchronization with ProjectBuilder

Note that changes made to source code using ProjectBuider are automatically transferred to WebObjects Builder. For example, adding variables and methods to your component's code in ProjectBuilder automatically adds them to the object browser in WebObjects Builder. In order to delete a key or from a component, you must delete it directly in ProjectBuilder. Additionally, only ProjectBuilder can be used to add variables and methods to the session and application classes.

Custom Palettes

Create a custom palette to store items used often, such as graphics or tables - and don't forget to include reusable components such as login panels. To create a new palette, choose "Palettes" from the menu and select "New Palette". Enter the location you wish to save the palette in, then choose OK.

Dragging Items to a Palette on NT

If the palette's background is gray, it is not in an editable mode. To enable editing, choose "Palettes" from the menu and select "Make Editable". The background will be white while the palette is in edit mode. Don't forget to save the palette and disable editing when you are finished.

On NT, you cannot drag an item directly to the palette because the palette window doesn't appear unless the WebObjects Builder window has focus. To overcome this problem, drag the item you wish to add to the palette and hold it over the WebObjects Builder icon on the task bar until the palette window appears. WITHOUT RELEASING THE MOUSE BUTTON, drag the item to the palette and release it.

Dynamic Element Reference

To quickly access the documentation on a particular dynamic element, select the element in WebObjects Builder. Bring up the Inspector window and click on the book at the bottom of the Inspector window. The relevant page from the *Dynamic Elements Reference* will be displayed in your web browser.

Make Static and Make Dynamic

Use the "Make Static" and "Make Dynamic" options in the Inspector to quickly convert a component from a static or dynamic element to its alternate counterpart.

Recognizing File Types

To configure WebObjects Builder to recognize filetypes other than the default set (.gif, .jpeg, .tif, .eps, .bmp, .eomodeld, etc), choose the "Tools" menu option and select "Options". A list of file extensions recognized by WebObjects Builder is displayed. Dragging any file with one of these extensions into WebObjects Builder will result in the file being added to your project. Use the "Add Extension" and "Remove Extension" to add or remove filetypes from this list.

Repetitions or Conditionals and Tables

When you wrap a repetition around a table row, a blue border is displayed around the row, rather than the repetition icon being displayed. Similarly, wrapping a conditional around a table row results in the background for that row changing to blue. You can also wrap a repetition or conditional element around a single cell in a table.

Reusable Subcomponents

Make sure you select the "Partial Document" option in the Page Attributes Inspector when you create a component that is always used as a subcomponent of another page so that WebObjects Builder will not automatically include the <HTML>, <HEAD> and <BODY> tags.

Common Tool Techniques (ProjectBuilder, EOModeler and WebObjects Builder)

Defaults

Applications have many "defaults" settings that can be very useful for optimizing or customizing use of that application. However, these "defaults" settings are rarely documented and difficult to ascertain. One easy method of determining the "defaults" valid for a given application is as follows:

1) Run the application in gdb.
2) Once it is running, break and issue the command:

```
po [ [ NSUserDefaults standardUserDefaults ]
dictionaryRepresentation ]
```

Of course this assumes that the application follows the documented API and uses registerDefaults. Some don't, so this mechanism is NOT foolproof.

Text Key Bindings

By default, many of the function keys on your keyboard are not assigned to a task (F1 is usually assigned to help and F2 or F5 is usually assigned to complete:). You can assign (or bind) key functions for every application by following the instructions in the file TextDefaultsAndBindings.pdf. This file contains instructions for proper implementation of this feature, and lists some possible key binding assignments. The file is installed online at:

Mac OS X Server:
file:/System/Developer/Documentation/YellowBox/TasksAndConcepts/ProgrammingTopics

NT:
<AppleRoot>\Documentation\Developer\YellowBox\TasksAndComcepts\ProgrammingTopics

Version Control Software and Apple Tools

When using version control software (like CVS) and ProjectBuilder, be conscious of the interaction between the version control software and ProjectBuilder. ProjectBuilder caches the files it touches, so if you update a file using the external version control software, ProjectBuilder will not automatically display the changes until you choose the "File" menu and select "Revert". The safest policy is to quit ALL tools (ProjectBuilder, EOModeler and WebObjects Builder) before doing anything with an external application that may cause changes to files in your work area.

Conclusion

Customizing your development environment and becoming familiar with the techniques presented in this guide and those mentioned in the documentation provided with WebObjects is essential for creating a productive development environment. In addition to the information provided in this guide, you will find more tips and techniques in each of the resources listed below.

Resources…

http://developer.apple.com/techpubs/webobjects
WebObjects Developer's Guide
Enterprise Objects Framework Developer's Guide
Enterprise Objects Framework Tools and Techniques (online with developer release)
http://www.omnigroup.com/MailArchive/WebObjects
http://www.omnigroup.com/MailArchive/eof
http://www2.stepwise.com/cgi-bin/WebObjects/Stepwise/Sites
ftp://dev.apple.com/devworld/Interactive_Media_Resources
http://www.apple.com/developer
http://developer.apple.com/media
http://til.info.apple.com/

About the Author…

Theresa Ray is a Senior Software Consultant for Tensor Information Systems in Fort Worth, TX (*http://www.tensor.com)*. She has programmed in OPENSTEP (both WebObjects and AppKit interfaces) on projects for a wide variety of clients including the U.S. Navy, the United States Postal Service, America Online, and Lockheed-Martin. Her experience spans all versions of WebObjects from 1.0 to 4.0, EOF 1.1 to 3.0, NEXTSTEP 3.1 to OPENSTEP 4.2, Rhapsody for Power Macintosh, and yellow-box for NT. In addition, she is an Apple-certified instructor for WebObjects courses.

Tensor Information Systems is an Apple partner providing systems integration and enterprise solutions to its customers. Tensor's employees are experienced in all Apple technologies including OPENSTEP, NEXTSTEP, Rhapsody, EOF and WebObjects. Tensor also provides Apple-certified training in WebObjects, Oracle consulting and training, as well as systems integration consulting on HP-UX. And Oracle.

You may reach Theresa by e-mail: *theresa@tensor.com* or by phone at (817) 335-7770.