# Working with WebObjects
An Introductory Survival Guide

by Theresa Ray
Tensor Information Systems

Sponsored by Apple Computer, Inc.
Developer Relations Group
for the Apple Media Program

# Working with WebObjects

## An Introductory Survival Guide

by Theresa Ray

Information displayed in HTML is static by nature. When an HTML page is shown to a user in a web browser such as Netscape or Internet Explorer, the information seen is fixed. But life is variable. You want to be able to dynamically react to the environment and make decisions based on what's happening. Web pages should be easily customizable for the user who is viewing it, and a website should be a powerful representation of your business. WebObjects provides this capability by bridging the gap between static HTML information and dynamic objects.

**Why Use Objects?**

Objects model your business by handling the logic (the practices and procedures) that make your business unique. A good object network is reusable for any application that your business needs to create, whether a corporate website or an internal desktop application. An object network has built-in behavior that matches the needs of your business. How does WebObjects use the intelligence in this collection of objects for your website?

A website (in traditional lingo) becomes a WebObjects application that runs on a machine at the web server site. The application is able to determine dynamically what to show the user when a page is requested.

## The Traditional HTML Template

The traditional HTML template is similar to standard static HTML that might otherwise be created for a page, but it contains special `<WEBOBJECT NAME=”SOMENAME”></WEBOBJECT>` tags used as placeholders for the actual value to be displayed. Any HTML editor that allows insertion of special tags can be used to create the template. The tags are replaced at run-time (when the user actually requests the page) with the correct values.

## The Script

The script is where variables are declared and behaviors are added. It is where business logic can be implemented, decisions can be made, data can be pulled from a database or flat file, and where your object network can be accessed (where code is written).

## The Bindings

The third part is the bindings—where the information to fill the `<WEBOBJECT>` placeholders is defined. WebObjects provides a set of dynamic elements similar to the standard HTML elements such as TextField, Browser, PopUpButton, Form, Active Image, and Submit Button. The bindings file for each web page maps a `<WEBOBJECT>` tag from the HTML template with the type of display element. For each element, the bindings file also defines which variable or function (called a *method*) to use for the display value (or *post value* for form elements). WebObjects supports Java components as well as standard HTML elements.

Additionally, each WebObjects application has a *session* object. The session object is automatically managed by the WebObjects application and creates a separate memory space for each user of the website. For instance, the information being submitted by user “Joe” on the Register page is kept in a separate memory location than that being submitted by user “Mary.”

## Powerful WebObjects Websites

Since WebObjects allows you to dynamically determine values at run-time, you can customize the website to fit a particular user or need. You can allow different levels of access to the site based on username or tailor the information presented to the user based on their interests. You can track information about a user's behavior and customize a response on-the-fly. For example, once the user's name is received (from a login page for instance), you could assign that user's name to a variable which other pages could reference. Or you could decide which page to follow based on the information submitted (called *dynamic navigation*). If, for instance, your application was an online order placement system, the user's order can be maintained. If that user quit in the middle of placing the order, the state of the order could be saved when the session timed out. Then the next time that particular user returned, they could have the option of picking up where they left off. Behavior pertaining to that particular user could be saved for use in tailoring special offers that match their interests.

*"With WebObjects Enterprise's powerful database frameworks, you can tightly couple your website to nearly any database, providing even more power and flexibility to your site."*

## Scaleable WebObjects Applications

WebObjects applications are very scaleable. WebObjects has a built-in load balancing system. By simply creating a basic configuration file, the web server can run on one machine, the application can run on a second machine, and the database on a third machine. Many *instances* (or copies) of the same application can even run on several different machines. This can help prevent a single machine from bogging down due to heavy usage.

With WebObjects, development efforts can be focused on particular needs. Provided with WebObjects is the Foundation framework which implements a root set of capabilities such as intelligent strings, numbers, dates, arrays, and dictionaries.

In summary, WebObjects is a tool that allows the development and deployment of highly scaleable, highly flexible, and extremely powerful websites.

## Tips and Tricks

Here is a short list of tips and tricks for those developing applications.

- All variables in webscript are declared as type id.
- To debug webscript, use the following code—it's purpose is similar to the printf function in C:

  ```
  [self logWithFormat:@”Some text you want to see
  with a variable value = %@”,variableToDisplayName];
  ```

  Then manually start the webobjects application (so the output has somewhere to go) by typing the following line at a command prompt:

  ```
  WODefaultApp -d <document_root_directory>
  <application_name>
  ```

  where document_root_directory is the path to the web server's document root directory, and application_name is the path relative to the WebObjects directory for the application (leave off the .woa extension). For example, to start the HelloWorld application, type

  ```
  WODefaultApp -d d:/netscape/server/htdocs
  Examples/HelloWorld
  ```

  The application_name must appear exactly as it will be typed in the browser, so use forward slashes (/). If you have compiled code, use your compiled executable in place of the WODefaultApp.
- When creating an application on NT, *always* manually start your application. Otherwise, WODefaultApp executables will be automatically started which you won't have permission to kill (even if you are the administrator).
- When using webscript, you only need to manually restart the process if the Application script file (Application.wos) has been changed, the information in the database has changed, or the database mapping file (xxx.eomodeld) has changed. Otherwise, reloading the page is *usually* enough to see the changes to the script (caching options can affect this behavior).
- If you're submitting a form and your variables are not being updated with the new values, make sure the dynamic elements are in the form with the submit button or image. Only information contained within the form is submitted.

- When using more than one submit button within a form, make sure the multipleSubmit attribute on the WOForm is set to “1”.
- The error messages displayed by WebObjects are more general at the top and more specific at the bottom. The first few lines generally “clue you in” as to which page (or component) had the error. If you see “error parsing file…” then there is probably a syntax error or typo somewhere in the file. After the portion of the message that specifies which component erred is a section which usually specifies which method in that component had the error. Then the session information is dumped, so if you’ve got a lot of session or page variables, their values will all be displayed. If this isn’t of interest, skip past it to the bottom of the error message. At the bottom of the error message is the specific cause of the problem.
- Put all images for your application in either a general image directory or an application-level image directory. When running a web server on one machine and the application on another machine, the web server requires a separate copy of the images locally with the same relative path from the document root directory. By keeping all images for an application in one place, this becomes a simple task.
- Don’t be afraid to try a test implementation to see what happens. You won’t break anything!

## Resources

When you are new to any technology, knowing where to get started can be the most difficult task. The following resources cab be a big ally in getting started with WebObjects.

The online WebObjects documentation provides an excellent reference. , the chapters titled “Introduction”, “How WebObjects Works”, and “Using WebScript” provide a good overview. The “Simple App Tutorial” provides a hands-on application of this information. Pages detailing how to use the WebObjects Builder tool are also available there, as well as more advanced topics like “Managing State”, “Creating Reusable Components” and Java topics.

http://devworld.apple.com/dev/SWTechPubs/Documents/WebObjects/index.html

Excellent documentation on the process of scaling a WebObject application, can be found at

http://devworld.apple.com/dev/SWTechPubs/Documents/WebObjects/ServingWebObjects/LoadBalancing.html#REF58620.

For information about the frameworks that come with WebObjects and commonly used methods check out this site.

http://devworld.apple.com/dev/SWTechPubs/Documents/WebObjects/DevGuide/Foundation/Foundation.book.html

For more detail, view the documentation installed with WebObjects (on NT, go to the Start menu, choose Programs, then WebObjects, then OPENSTEP Books online. Click on Foundation Reference, choose the Classes tab and click Open. A list of classes appears. Double-clicking on the appropriate class brings up its documentation).

WebObjects installs a host of example programs. The examples are placed in a subdirectory of the document root directory (specified when WebObjects was loaded) called WebObjects/Examples. These examples range from the simple HelloWorld application, to more complex Java applications, to the database-backed DodgeDemo example. Documentation for the examples can be found on NT by going to the Start menu, choose Programs, then WebObjects, then OPENSTEP Books Online. Click on WebObjects Documentation, then click on the URL shown. In the left frame of the web page displayed is an "Examples" option which will take you to the documentation for the installed examples.

Omni Development, Inc. maintains a WebObjects mailing list and archive of questions. This site is frequented by novices and experts alike, and is routinely enhanced by Apple support personnel suggestions.

Training is available for WebObjects. This website shows the current course schedule.

http://www.enterprise.apple.com/Training/OpenEnroll/Schedule.html

For database applications, the book *Enterprise Objects Framework Developer's Guide*, printed by Apple is a good guide for connecting your WebObjects application to the database.

Stepwise maintains a website with links to many resources in the OPENSTEP/WebObjects community including consultants, training providers, and freeware/shareware sites.