A Comparison of Web Development Technologies: WebObjects vs. ASP.NET

By: Adnan Al-Ghourabi

Chairman: Dr. Axel Schreiner
Reader: Dr. James Heliotis

Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623

1. Introduction

One of the major disadvantages of html is the fact that it is static; any modification requires editing the html file. This is a cumbersome task, as websites grow bigger in content and complexity. The advent of server-scripting languages was targeted at solving the deficiencies of html by providing dynamic web content that is usually database-driven.

There are several languages that have been tailored for web development: JSP, which is based on Sun's Java language; ASP, Microsoft's early server-side scripting language; PHP and others. These languages hold considerable shares of web application development projects for both the Windows and Unix/Linux platforms.

With the introduction of the .NET framework in late 2000, ASP has been tremendously improved to take advantage of object oriented programming, and have recently become the dominant development language of choice for web applications on windows platforms. Johnson & Johnson, for example, has recently launched two websites developed in ASP.NET, changing a long-followed trend of using JSP. Intranet applications have also been developed in ASP.NET.

There is also the less heard about: WebObjects. WebObjects is a system that consists of a number of development and deployment tools, and a set of frameworks to develop server applications. WebObjects applications are Java based and therefore can be deployed on a variety of platforms[1]. WebObjects was first created by NeXT in late 80's and was based on Objective-C. It was later acquired by Apple in 1995. The latest version available is 5.2.

Until three years ago, a WebObjects development license used to cost $50,000, limiting its use to big projects such as the Dell Computers website and The United States Postal Service call center. With a $699 a license ($99 for students), an increasing number of developers has been using WebObjects in the coming years.

There have been various studies for comparing Web development languages. ASP.NET decisively outperforms most of existing web development languages. For example, in a .NET version of Sun's Pet Shop application, which was developed in JSP, the lines of code in ASP.NET were less by a staggering 75% [1]. Other comparative studies have also shown similar results comparing ASP.NET to ASP and PHP in terms of performance. These impressive results explain the momentum ASP.NET has been gaining since the launch of the .NET technology [2].

[1] This is true as of version 5.1

The lack of a comprehensive study comparing WebObjects to other available technologies has incited the idea to provide a comparison between ASP.NET and WebObjects[2]. It is therefore the focus of this project to highlight the differences between the two technologies in terms of system architecture, frameworks, mutability, reusability, scalability, development environment and tools, portability, security, and implementation.

In order to provide a fair, realistic, and a productive comparison, this project will focus on studying an existing application in ASP.NET and providing a counterpart application using WebObjects. The IBuySpy Web portal will be used as a sample ASP.NET application. The differences between the two applications will be highlighted at the different stages of development with respect to the above criteria.

2. Web Portals

As web development matured, web portals have been introduced to further increase the efficiency of website development and maintenance.

Web portals offer online administration and dynamic content management for websites. This concept has made website maintenance a fast, easy, and inexpensive process. In contrast to static pages and traditional dynamic-content pages, the development effort is a one-time process; once the website has been developed, any further modifications to the content can be applied online without any programming effort.

The extendibility of the website content depends on the capabilities that are built-in. Traditional dynamic-content pages often require modifying pages and rewriting code in order to handle any changes in the design and the layout in which data to be presented. This is true for both logic and design changes.

Web portals are usually module and role-based. The main goal of designing role-based web applications is to make website management an easy process. Not only will the authorized user be able to change the format and order of data on the fly, but he will also be able to do the changes remotely. This flexibility cuts down on costs and the overall effort. The authorized user need not be a dedicated webmaster; any user registered under the administrator role, for example, can do the changes himself.

The reason why there has been a shift towards using Web portals is that they are module-based and therefore extendable. More functionality can be added by simply attaching new modules to the application.

2. Overview

3.1 IBuySpy Portal

The IBuySpy Portal is a web portal that has been developed by the asp.net team at Microsoft [3]. The portal demonstrates the best practices for developing ASP.NET applications, using the .NET framework, in terms of modular design, caching, XML serialization, Windows and forms based authentication, caching features, and state management.

The IBuySpy portal runs on IIS and uses SQL Server 2000. It is module and role-based. The available modules can be added, removed, and moved around on the website itself provided the user has the credentials to do so. These modules provide for basic needs such as html text, item lists, documents' uploading, links, threaded-based discussion etc. Depending on the role assigned to the user, he, upon logging in, will only see the modules that are assigned for that specific role.

[2] I have not found a complete comprehensive comparison; only bits and pieces in discussion forums.

3.1.1 Architecture

The IBuySpy portal is a multi-tier application:

A. the presentation layer is represented by Web forms and User Controls that display the portal customizations and allow for data managements. The data is generated dynamically using a variety of built-in Server controls.

B. the Business layer is represented by code-behind files that handle windows/forms authentication, XML serialization, server-side validation, and communicating data back and forth between the presentation and the data layers.

C. the Data Layer is represented by database access classes that invoke stored procedures on SQL Server 2000 using ADO.NET data structures.

3.2. ASP.NET

ASP.NET is the inheritor of ASP [4]. With the introduction of the .NET framework, the functionality of ASP has been dramatically changed to take full advantage of .NET. ASP, as well as other server-side scripting languages such as PHP, is interpreted. This results in pages being interpreted whenever a client requests a page. This is a major drawback in terms of speed and server overhead. Another drawback is the lack of separation between logic and design: the logic is embedded within the html code. Not only does maintenance become cumbersome, but also the whole development process since the design and logic are tightly integrated. Although PHP and JSP support object-oriented programming, they still suffer from some of these drawbacks.

ASP.NET was designed with the intension of avoiding these drawbacks and offering a language that is faster, easier to develop, easier to maintain, and reusable. So how does ASP.NET do it?

ASP.NET is a Web development language with support to more than 25 languages that have access to all the classes available in the .NET framework. This big set of classes offers an enormous advantage to ASP.NET developers. It also makes it easier for application developers to write code for web applications in an environment with which they are familiar and in a language that they master. Currently, C# and VB are the dominant languages of choice.

Pages in ASP.NET are called Web Forms. Web Forms have an .aspx extension and represent the interface to the user, while the logic resides in a code-behind file. Alternatively, one can provide the logic in the same file by including it between script tags, similar to that used for JavaScript. In the latter case, the user does not need to compile the application; the page will be compiled upon request.

Web Forms can include user-customed modules called User Controls. These modules provide an easy way for reusability. As User Controls cannot be requested independently, they need to be a part of a web form. The User Control can be included in a page by declaring a "@ Register" directive that declares a tag prefix attribute by which the control will be referenced, a tag name attribute to associate a name with the control, and a source attribute that specifies the virtual path to the User Control. User Controls can also be added dynamically.

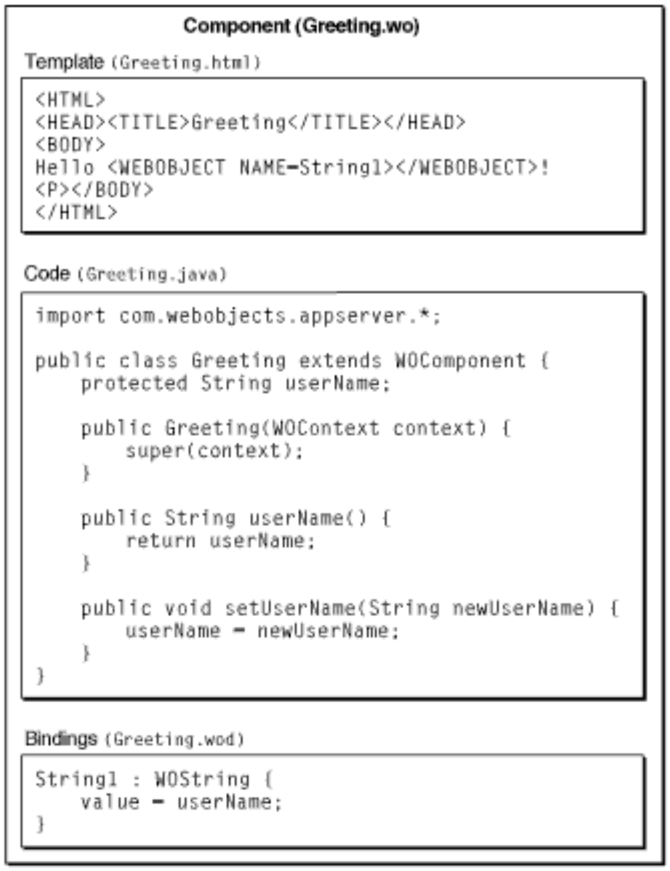
Web Forms and User Controls may include Server Controls [5], which are built-in controls that are available for ASP.NET development. Among these are TextBox, Label, CheckboxList, Buttons, and LinkButtons controls etc.

ASP.NET is a compiled language; once the client issues the first request to a page, the pages, including any User Controls, get compiled and cached in memory, which will make further replies to the client's requests a lot faster; faster by 50% than classic ASP according to Microsoft [4]. ASP.NET also offers developers the choice of whether to cache pages or controls and for how long. This feature has a great advantage especially with pages that consist of controls; if a menu on a page does not change often, the menu control can be cached to speed up loading time for example.

A beta version of ASP.NET 2.0 is available with the beta release of Visual Studio Whidbey. New features in ASP.NET 2.0 include built-in login controls, master template pages, and new server controls such as DataGridViews.

3.3 WebObjects

A Web page in WebObjects is represented by a Web component (.wo), which in turn can include other Web components. A web component consists of the html file (.html) that represents the presentation layer, the Java classes that represent the logic (.java), and a binding file that maps the elements in the presentation layer to the correspondent objects in logic along with any event handling (.wod). Data is generated dynamically through the use of Dynamic Elements that are represented in the .wo file. These elements can be embedded using the <webobject></webobject> tags as shown in the below image [6].



```
<HTML>
<HEAD><TITLE>Greeting</TITLE></HEAD>
<BODY>
Hello <WEBOBJECT NAME=String1></WEBOBJECT>!
<P></BODY>
</HTML>
```

```
import com.webobjects.appserver.*;

public class Greeting extends WOComponent {
    protected String userName;

    public Greeting(WOContext context) {
        super(context);
    }

    public String userName() {
        return userName;
    }

    public void setUserName(String newUserName) {
        userName = newUserName;
    }
}
```

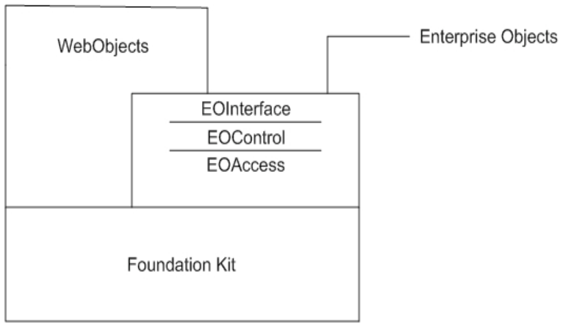
```
String1 : WOString {
    value = userName;
}
```

The WebObjects system consists of a set of frameworks, development and deployment tools.

3.3.1 Frameworks

The frameworks are libraries that comprise of the following parts:

a. WebObjects classes: these classes are included in the com.webobjects.appserver package, which contains WOApplication, WOComponent, WOSession, among other high-level WebObjects Classes [7].



b. Enterprise Objects: these classes instantiate business objects directly from the database by representing its entities as Java objects, creating an abstract layer that hides the underlying data storage from the data layer [8]. JDBC provides the interface between the Java platform and the different databases. This feature allows WO to avoid database-specific data layer by using the EOModeler to create these objects. As the data within objects change, EO will maintain the integrity of data and handles all database-related issues such as locking and referential integrity.

c. The Foundation Kit: a set of classes that provide collection classes (arrays, dictionaries) that were inherited from Objective-C (have been rewritten entirely in Java) and are shared by Cocoa, Apple's desktop application development system [7]. These classes have preserved their prefix "NS", which stands for NeXTStep, an operating system that was offered by NeXT, the company that previously owned WebObjects.

3.3.2 Development Tools

a. Project Builder: the IDE for WebObjects which manages the Java business logic, properties files, Web components, and other files. Other tools can be invoked from within Project Builder.

b. WebObjects Builder: the tool for developing Web components and creating mappings between components in this interface and Java objects (either by typing, or by graphically dragging variables to correspondent elements).

c. EOModeler: the tool for mapping table entities and Java objects in the application. It can also be used to create the database schema and generate the Java code that represents the mappings.

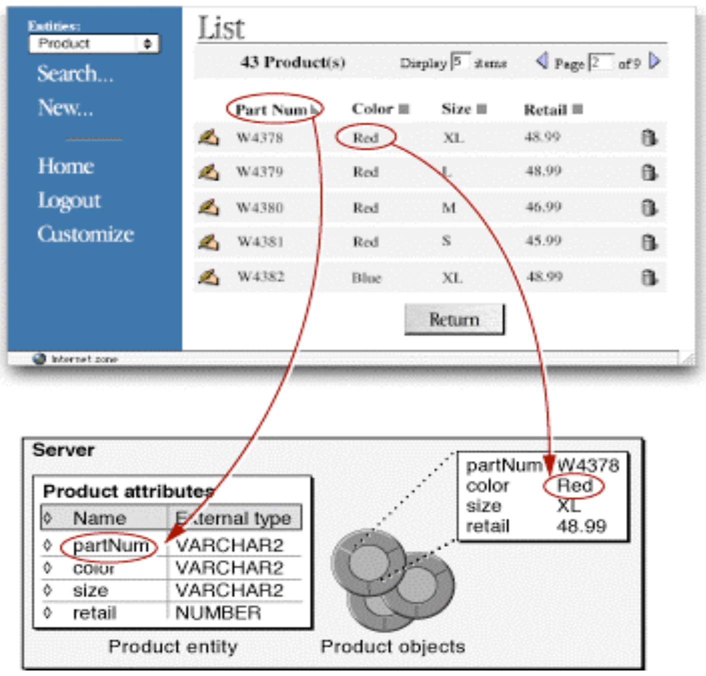
3.3.3 Deployment Tools

a. WebObjects Task Daemon: this is a task manager that communicates between WebObjects applications and the WebObjects adaptor using XML.

b. WebObjects Monitor: a front-end for the task daemon for configuring running WO applications [7].

3.3.4 DirectToWeb

DirectToWeb is a technology that can create a user interface (UI) for a data model created by the EOModeler tool. It generates pages dynamically at run time to allow data access and modification.



Part Num	Color	Size	Retail
W4378	Red	XL	48.99
W4379	Red	L	48.99
W4380	Red	M	46.99
W4381	Red	S	45.99
W4382	Blue	XL	48.99

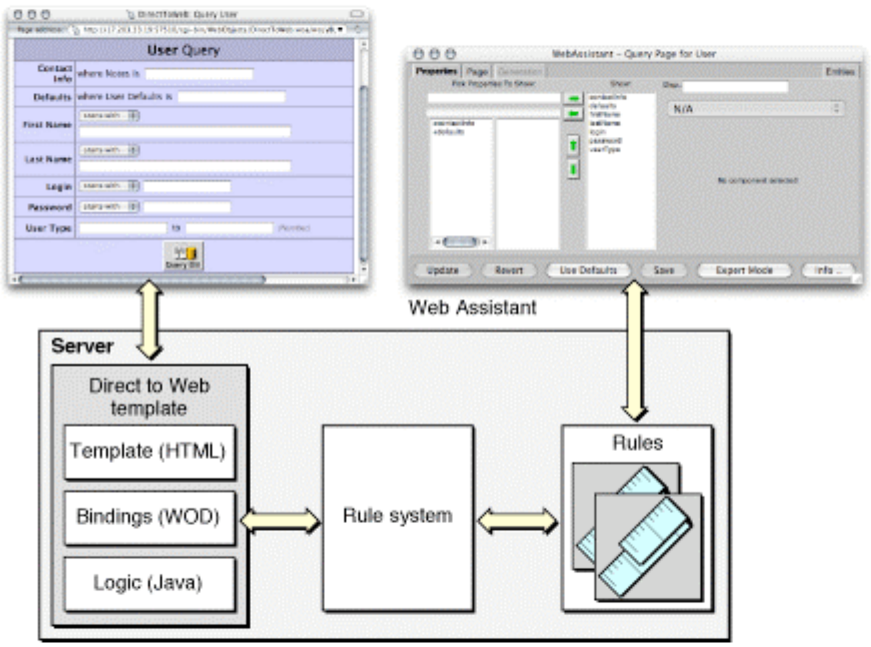
Product attributes

Name	External type
partNum	VARCHAR2
color	VARCHAR2
size	VARCHAR2
retail	NUMBER

partNum	W4378
color	Red
size	XL
retail	48.99

The behavior can be customized through a set of rules and setting different priorities for these rules. These rules can be set using the Web Assistant.

Although pages in DirectToWeb are generated dynamically, one can “freeze” these pages to view the different files compromising the web component or to simply convert the page to a reusable component within a WebObjects project. DirectToWeb also offers dynamic paging: view records in pages based on a page size. It also allows for creating relationships between database entities, allowing developers to create robust UI to databases. This technology resembles the DataGrid server control in ASP.NET, although the latter is limited in its features and requires the developer to hand code events handling and data access and modification.



DirectToWeb has many advantages. Given its capability to generate pages dynamically, developers will spend less time developing applications. It also increases the maintainability of these applications and reduces the possibility of errors during development. And most importantly, developers will focus more on the business logic instead of the presentation layer.

4. Functional Specification

4.1 Comparison Criteria and Objectives

The object of this project is to compare both technologies with respect to the following criteria:-

a. System architecture: although, hypothetically, both systems offer separation of the different application layers, the EOModeler in WO provides another abstraction layer to the data layer by making the data access logic independent of the data source. Also, some implementations in ASP.NET requires providing logic within the .ASPX or .ASCX files to achieve the required functionality. The presentation layer between both technologies differs in terms of available built-in controls (components). For example, ASP.NET 2.0 incorporates portal features such as the login control that have no correspondence in WebObjects. The project will highlight these differences and provide an analysis for the pros and cons for the two architectures at the different layers and how it affects the overall application in terms of the other criteria.

b. Mutability: it is conceivable where requirements might change resulting in the need for modifying the application's logic. Mutability is tightly-coupled to the system architecture and how well the technology adheres to object oriented design and separation of layers. Hypothetical scenarios will be drawn to highlight the differences between WO and ASP.NET. For example, would the data layer in the IBuySpy portal work if the same database schema is moved to a different database system?

c. Reusability: one important characteristic of good design patterns is to separate layers and make common functionality reusable. The project will delve into the extent to which components can be reusable. For example, DataGrids editing capabilities in ASP.NET requires providing event handlers for actions like "edit", "add", "delete", and "cancel" etc., which may not allow the logic to be reused. Editing interfaces can be generated dynamically using WebObjects DirectToWeb, and then turned into a reusable component.

d. Development Environments: one advantage that ASP.NET has over WO is its IDE. The IDE does play an integral role in expediting development by providing a complete set of tool that a developer would need. The tools available for developers of both technologies will be examined, such as debugging and data modeling tools. The comparison will be limited to the tools available from Microsoft and Apple.

e. Portability: being Java-based, WO has the advantage of not being restricted to one platform. No attempt will be made to deploy the two versions on different platforms, but further study will be conducted.

f. Security: ASP.NET offers the ability to use Windows authentication through the use of Microsoft Active Directory Service. ASP.NET 2.0 offers a built-in control for user's authentication. Although it may not be feasible to compare the security offered by both technologies, further research will be conducted to draw a general comparison between both systems.

g. Platforms and Deployment: the WO version of the IBuySpy will be deployed on a Windows XP Professional machine. Deployment details will be documented along with a performance comparison in terms of responsiveness of Web pages. Although it is understood that the native support for .NET in IIS gives ASP.NET and edge over WO, it is expected that WO would perform reasonably. The ease of deployment will also be compared, including the built-in capability of creating installation packages.

h. Learning Curve: one of the main objectives of this project is to learn WO and to take advantage of the available tools in this huge system. WO has a reputation of requiring a high learning curve before being able to grasp all the features that the system offers to developers.

i. Frameworks: ASP.NET takes advantage of the rich set of .NET frameworks. WO has a set of frameworks as well. Along the development process of the WO version, frameworks of both systems will be compared and differences will be highlighted.

5. Analysis and Development

The development process will proceed with a careful analysis of the IBuySpy portal to better understand the architecture, the business logic, and the data layer. This analysis will also aim at pointing out any design deficiencies.

Being a beginner in WebObjects, a good amount of time will be spent learning WebObjects to be able to understand the aspects of this system and provide a reasonable counterpart application.

Initially, a comparison between ASP.NET server controls and WebObjects dynamic elements is needed. The purpose will be to identify how to achieve a similar or better functionality using WebObjects. If there is no existing component in WO that resembles its ASP.NET counterpart, such component will be developed. One of the goals is to learn to think the WebObjects way and to avoid imitating the ASP.NET application design.

In the next phase, the focus will be on the business and data layers and how to reach the goal of layers separation and components reusability with respect to the criteria outlined above. Development will proceed with respect to the outcome of the analysis phase.

The modules in the ASP.NET version offer a variety of functionalities such as file uploading, data editing, rendering XML documents, threaded discussions, etc. Similar modules will be provided for the WO. DirectToWeb will be used to provide the interface for any data access or modification functionality, which may result in changes to the presentation layer.

The UI will comprise of Web components (.WO files). The look and feel of the IBuySpy portal will be preserved.

6. Implementation

6.1 ASP.NET

The IBuySpy portal has been developed with ASP.NET 1.1 and Visual Studio .NET 2003. New features have been introduced with the beta releases of ASP.NET 2.0 and Visual Studio .NET 2004. These improvements will be considered in this study.

The IBuySpy portal currently uses SQL Server 2000. If Yukoon, the future release of SQL Server, will be available during the time of this study, an additional research will be conducted to include any changes in this project.

6.2 WebObjects

The tools that are available for the WO system will be used to develop the IBuySpy portal. The WO IBuySpy portal will run and be tested on IIS 5.1 and IIS 6.0.

Since it is based on Java, WO is purely object-oriented. The architecture and the implementation will try adhering to this programming paradigm to achieve the goals of separating application layers and creating reusable components.

The WO portal will be tested with both OpenBase and SQL Server 2000 using JDBC. Special attention will be made to highlight the abstraction layer that WO offers over the data layer. At attempt will be made to use OpenBase as a database for the ASP.NET portal as a way of comparing both systems.

7. Expectations

In a comparative study, it is always hard to expect the outcome. Both technologies are being used in real life applications. One has been around for over a decade and the other for less than 4 years. Both technologies have claims of their superiority. And therefore, the purpose is to put these claims to the test.

Having been developing ASP.NET applications for over seven months now, the most important expectation is to become fairly experienced in WebObjects, provide a fair analysis of WebObjects compared to ASP.NET, and offer recommendations to students and developers alike based on the experience gained by this project.

8. Deliverables

8.1 Packages

WebObjects, through the customization of “targets”, offer the capability to bundle up files in a jar or a zip file. The WO version of the IBuySpy portal will be provided in both types.

8.2 Documentation

A user manual along with diagrams to explain the installation steps and how to use the application will be included with the setup file.

8.3 Source Code

The source code for the WO version of the IBuySpy Portal will be available publicly on my personal website. The data models and database schemas will be available for SQL Server 2000 and OpenBase.

9. Schedule

May: phase 1 / study WebObjects and analyze the IBuySpy portal

June: phase 2 / design the WebObjects portal

July: phase 3 / development + testing

August: phase 3 / development + testing

September: phase 4 / comprehensive testing and improvements

October: implementation and deployment on different systems

November: project defense

10. Implementation Issues

1. The End-User License agreement (EULA) for the IBuySpy Portal permits modification and redistribution of code for testing and development purposes.

2. Platform: The WO version of the IBuySpy portal will be tested on a Windows XP Professional machine running IIS 5.1, and on a Windows Server 2003 running IIS 6.0. Both SQL Server 2000 and/or OpenBase will be used as the database backend.

3. Being a beginner in WO, different resources will be consulted along the development process to ensure that the portal architecture in WO is taking advantage of the features that WO offers.

11. References

1. Author, Leake, G., & Author, Duff, J., Microsoft (May, 2003). *Microsoft .NET Pet Shop 3.x: Design Patterns and Architecture of the .NET Pet Shop*. Retrieved February 6, 2004, from: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/petshop3x.asp

2. Microsoft. *Why ASP.NET?* Retrieved February 12, 2004, from: http://www.asp.net/whitepaper/whyaspnet.aspx?tabindex=0&tabid=1

3. Vertigo Software (2002). *IBuySpy Application Series: Portal*. Retrieved January 8, 2004, from: http://www.asp.net/ibuyspy/portalpaper.aspx?tabindex=5

4. Microsoft. *ASP.NET Home*. Retrieved February 6, 2004, from:

http://msdn.microsoft.com/asp.net/

5. Microsoft. *Including a User Control in a Web Form Page*. Retrieved February 29, 2004:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconincludingpageletcontrolinanotherwebformspage.asp

6. Apple. *Web Applications: A Programmer's View*. Retrieved April 12, 2004, from:

http://developer.apple.com/documentation/WebObjects/WebObjects_Overview/WebApplications/chapter_4_section_2.html#//apple_ref/doc/uid/TP30001008-CH204-BGFHHGEG

7. Marker, J. (2004). WebObjects 5 for Mac OS X. Berkeley, CA: Peachpit Press.

8. Apple. *WebObjects Overview*. Retrieved April 7, 2004, from:

http://developer.apple.com/documentation/WebObjects/WebObjects_Overview/index.html#//apple_ref/doc/uid/TP30001008