Modern Eyes and WineHangOver.com: Evaluating ASP, JSP, and ASP.NET in the Classroom

By
Randy Connolly
Instructor, Dept. of Computer Science & Information Systems
Mount Royal College
4825 Richard Road SW
Calgary, AB, T3E 6K6
403-440-6061
403-440-6664 (fax)
rconnolly@mtroyal.ca

Abstract

The rapid rate of change within the web development industry has created many challenges for those crafting curriculum in the Internet application area. Given the time it takes to create lectures, lab exercises, assignments, and realistic case studies, choosing the appropriate technology to use is a crucial decision. This paper tries to provide some help for information systems educators in making this choice. It describes the evolution of two different web-application development courses over the past three years in terms of the server-side development technology used during that time and presents the author's analysis of the relevant strengths and weaknesses of ASP, JSP, and ASP.NET for teaching Internet application development.

1. Introduction

Internet application development is now an important topic in information systems and computer science education.[1] Unfortunately, the rapid rate of change within the web development industry has created many challenges for those crafting curriculum in this area. Creating lectures, lab exercises, assignments, and realistic case studies takes a great deal of time and effort. As Lee observes of the typical web development course, "there is an incredible range of different web technologies each of which is constantly changing

and yet requires a significant amount of effort to learn."[2] Recreating all this material every twelve to eighteen months is hardly a palatable choice for most instructors, yet the rate of change within the field of Internet application development has necessitated it. For this reason, some educators have declared the typical web application course to be almost "unteachable."[3]

For those faced with creating or updating a web application development course, deciding upon the appropriate technology to use is a crucial and unnerving decision. One would like to choose a technology that is appropriate for the particular student audience, relevant within the current work world, and which will not be edging towards obsolescence the next time the course is offered. This paper tries to provide some help for educators in making this choice. It describes the evolution of two different web-application development courses over the past three years in terms of the server-side development technology used during that time and presents the author's analysis of the relevant strengths and weaknesses of ASP, JSP, and ASP.NET for teaching internet application development.

The Applied Degree in Computer Information Systems and Business at Mount Royal College has two Internet application development courses. The first, *Building Internet Solutions*, is a second-semester course which covers HTML, web design, JavaScript, as well as an introduction to server-side scripting technology. Since the Winter 2000 semester, the server-side technology used in the course has been Microsoft's ASP (Active Server Pages). The second web-based application development course in the program is a sixth-semester course called *Designing and Implementing Electronic Commerce*. This course has been offered three times since 2002. The course material was originally developed for ASP, but has subsequently used Sun's JSP (Java Server Pages) technology, and more recently, Microsoft's ASP.NET. Both courses use a case study in which students develop a substantial database-driven web application.

2. ASP

The case study for the first course is a fully functioning, database-driven web storefront for a fictional fine arts reproduction company named *The Modern Eye*. Students are provided with the images and an Access database containing product information. Even though these students are only in their first year and have only three weeks instruction in ASP, the students are able to design and create a visually rich site that allows users to browse the images by artist or title, search using up to six different criteria, place or remove items from a shopping cart, and order and "pay" for cart items. For students who have had a bit less than two previous courses of C++ or Java programming, the level of functionality attainable by the student's projects can be startling and seems incommensurate with their first-year standing (see figure 1).

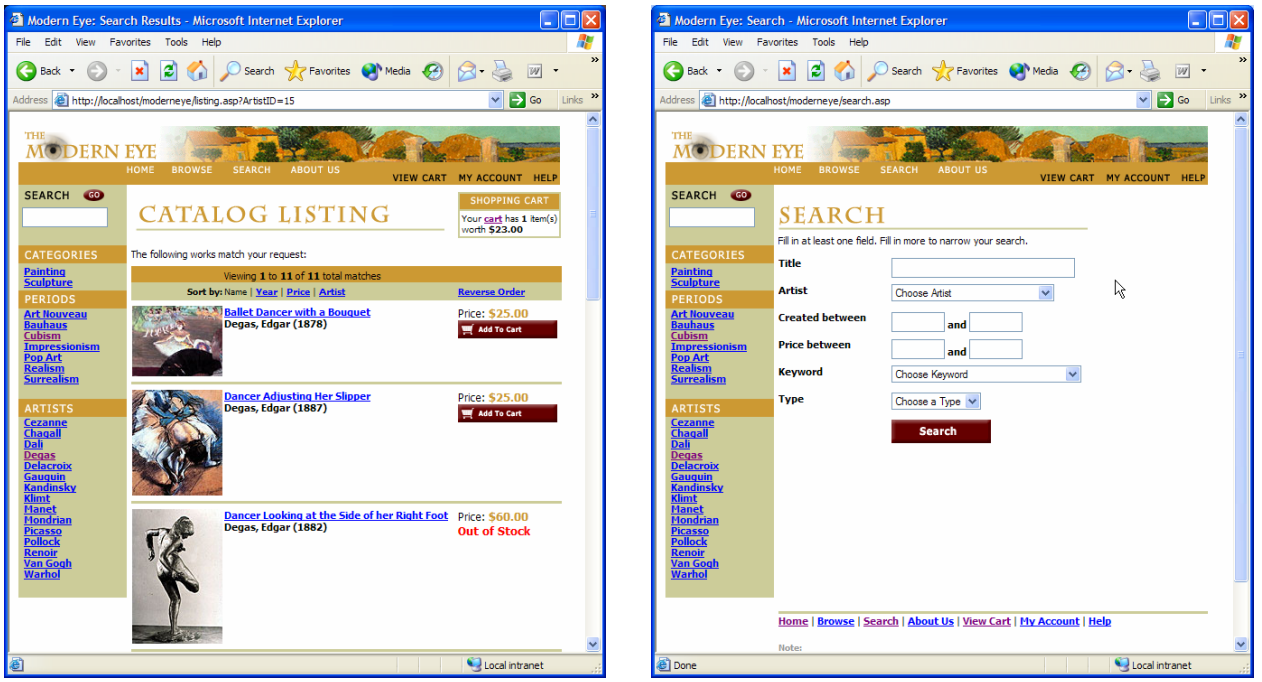

Figure 1

The reason such complexity can be achieved by second-semester students is due, on the one hand, to the "thrill" of creating real-world content for the web, and, on the other hand (and more importantly), to the relative simplicity of ASP. With ASP, one combines programming logic with presentation tags (HTML) together in one file with the extension .asp. The file is uploaded to a server running Microsoft's IIS (or, if your IT department allows it, the student's development machine can have IIS, allowing the students to test pages locally without the upload). The .asp page can then be tested in any browser.

ASP's syntax is quite straight-forward. Standard HTML is supplemented with asp tags (<% … %>). Within these tags are either expressions or programming code written in either VBScript or server-side Javascript (most textbooks use VBScript). If students are already comfortable with conditional logic and loops, one can teach the basics of creating simple dynamic ASP pages within one or two lectures. Another one or two lectures can be spent covering the basic functionality of the core objects in ASP's object model, the `Response` and `Request` objects, neither of which typically causes the student much trouble. Constructing pages from data pulled from a database requires teaching the basics of the ADO object model (usually two lectures), which also does not trouble the student all that much.

What does, then, trouble the student? In short, putting the pieces together. The student's main stumbling block is how to use the simple language and objects together to solve typical web problems. That is, what the students need are algorithms. For instance, a typical web application workflow is to display a page, gather some information from the user, and depending upon the validity of the input, either display another page and pass the data to it (generally via form parameters) or display the original populated form again with an error message (see figure 2).

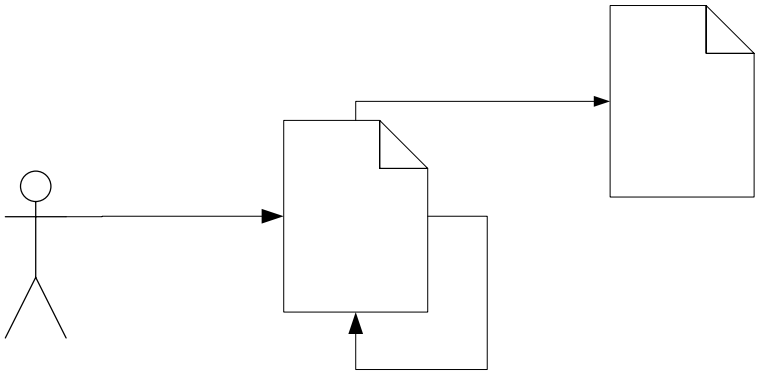

Figure 2

Students typically have some difficulty comprehending and implementing this interaction. The programming itself is not complicated; rather the difficulty lies in the fact that the students must implement temporary state themselves and that their pages must contain code for both the pre-processing and the post-processing together in one .asp page. Due to the scope of the project's requirements, students begin to discover a key drawback to ASP. The logic necessary for implementing these more complex interactions (especially, for instance, the shopping cart and the checkout), typically results in an .asp page containing many hundreds of lines of interspersed presentation (HTML) and logic (VBScript) dealing with security preprocessing, validity testing, flow branching, and state preservation that is beyond some seasoned programmers, much less first-year students. (The current instructors of this course have in fact removed the shopping cart and checkout from the project).

Page A

[if error

Another problem that the students encounter is how to best handle repetitive coding and changes across multiple pages within the site. For instance, within a site, many pages share common user interface elements, such as headers, footers, and navigation systems. As well, most pages typically share a certain amount of functionality, such as security checks (e.g., checking if user is logged in yet), visual state presentation (e.g., displaying

number of items in shopping cart, or “most popular” items lists), and database interactions (e.g., specifying database connection and running SQL queries). In a page scripting language like ASP (or PHP), some developers will simply copy and paste presentation and behavioral elements from one page to another (although both ASP and PHP have an “include file” mechanism that provides a type of modularity to a developer). As students develop their sites, they begin to appreciate the maintenance difficulties with ASP’s less modular page scripting approach to web development.

3. JSP

Sun’s Java Server Pages (JSP) shares many of ASP’s strengths, but avoids some of its weaknesses. Just like ASP, JSP pages can contain both presentation (HTML) and programming logic within <% … %> tags, except, of course, they use the Java programming language. The principal advantage of JSP is that it is possible to separate the presentation from the programming logic via Java Servlets, JavaBeans and/or EJBs (Enterprise Java Beans)/POJOs (Plain Old Java Objects). While this makes large and complex sites more manageable to implement and support, it does require a more sophisticated developer to successfully combine these different coding technologies. For this reason, JSP was introduced in the second web development course in our program, the third-year *Designing and Implementing Electronic Commerce* course. In this course, the students worked with a single case study through multiple implementation iterations. The case was also a fully functioning, database-driven web storefront, this time for a fictional wine retailer named *WineHangOver.com*. The base functionality for this site was not much different than that of the *Modern Eye* site, except for this course, the students were given a set of frequently changing requirements and had to also implement a partial checkout pipeline. Like the first course, the students were supplied with the database (mySQL or SQL Server). Unlike the first course, the students did not have to worry about visual design (see figure three) as all the user interface elements were supplied. Instead, the focus was on web application design principals and patterns.

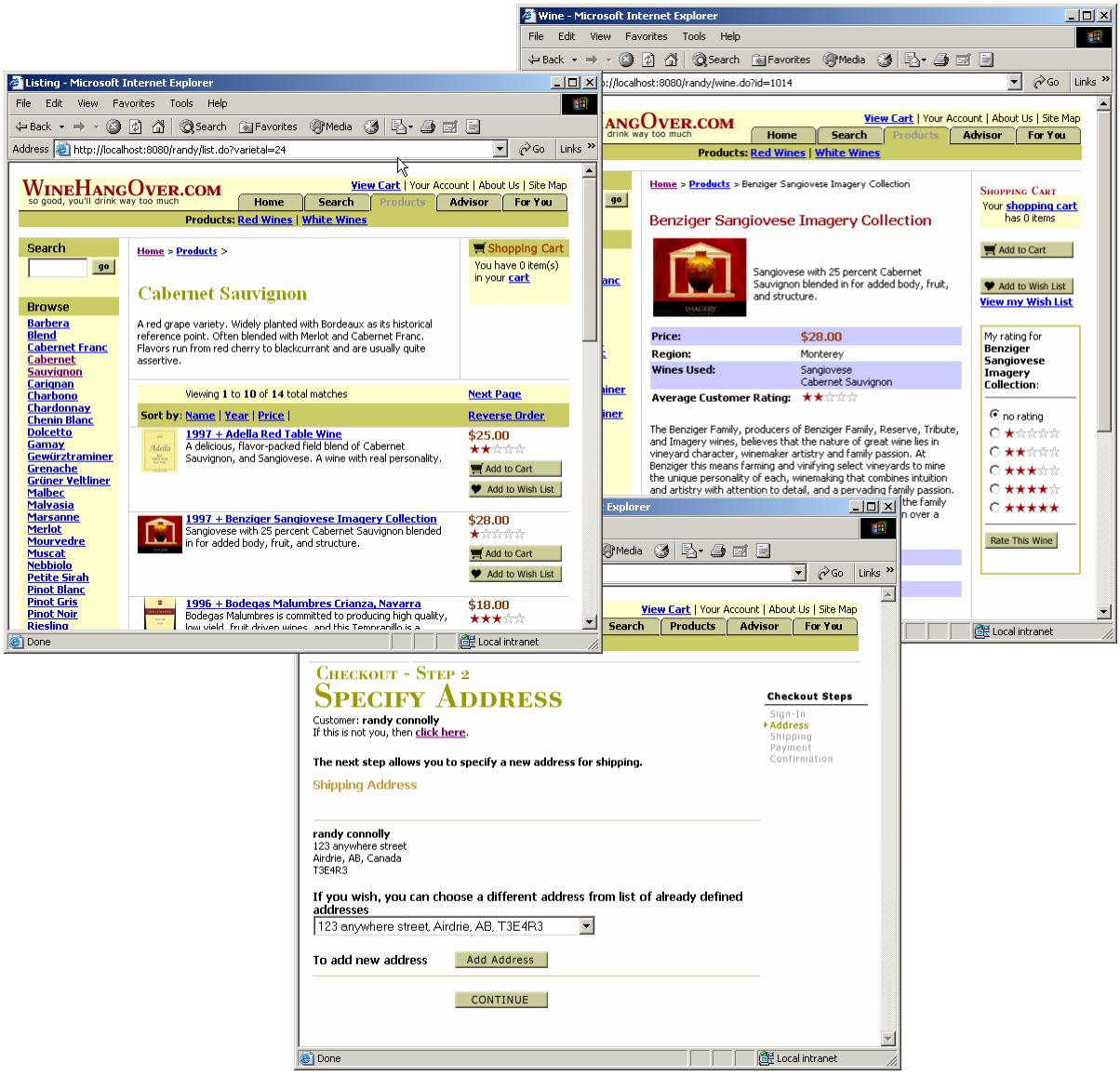

Figure 3

In the first iteration of the case study, students used JSP much like they did ASP, in that application logic was contained within each JSP page. In the next iteration, students used a more complex, but ultimately a more maintainable and object-oriented architecture based on Sun's Front Controller and View Helper enterprise patterns (also referred to as the Model 2 Framework).[4] This architecture follows the Model-View-Controller design pattern by using JSP pages, Servlets, JavaBeans and POJOs to create a functional division of labour between presentation, workflow control, and data handling.

A Servlet is a Java class used to handle HTTP GET and POST requests. In the Model 2 architecture, it acts as a *controller* responsible for processing HTTP requests using various helper classes (which could be EJBs or POJOs) and for instantiating JavaBeans (which are just regular Java classes that follow a naming convention) that act as the *model*. The Servlet then forwards control to the appropriate JSP page, which acts as the *view* and accesses the data in the JavaBeans for eventual display. In this approach, JSP pages typically contain little programming (except for looping through collections). Figure four illustrates the relationships between the elements in the Model 2 architecture.

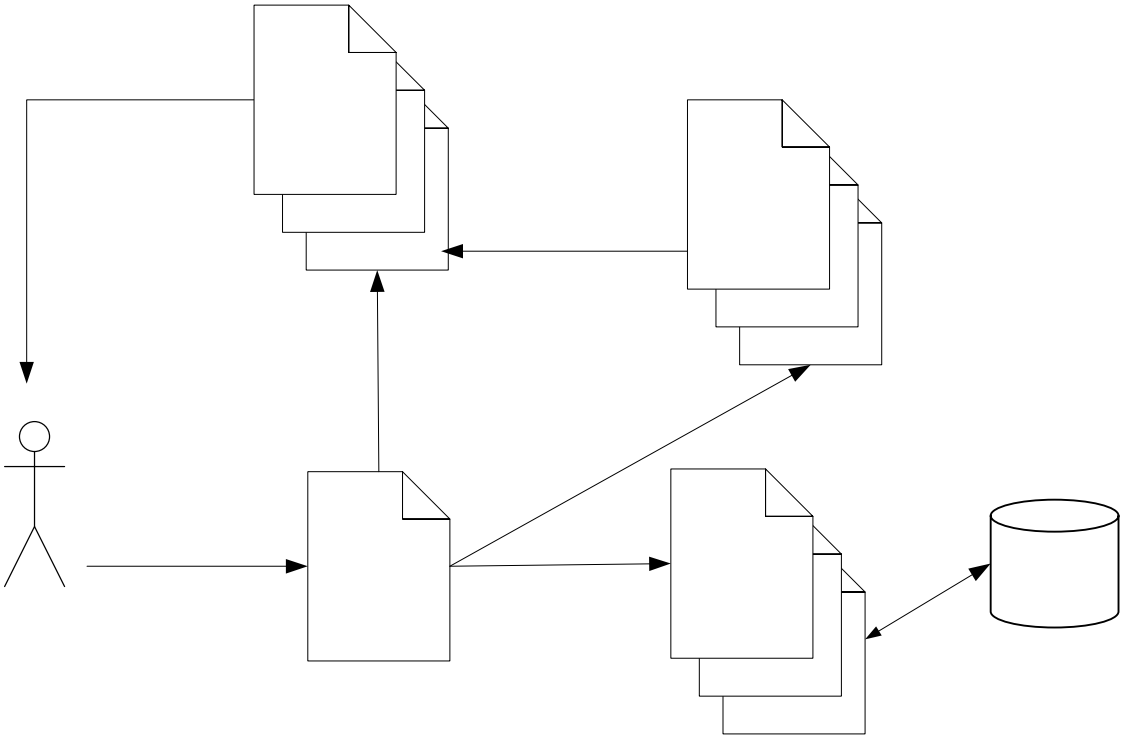

Figure 4

5. user views

JSP

JSP

JSP

4b.

While undeniably more work at first, separating presentation from workflow from business logic and data handling offers a number of important benefits:[5]

- It minimizes the potential impact of changing requirements. Presentation or workflow can be changed without affecting business rules, and vice versa. The project's requirements purposely changed as the students worked on the different iterations of the case study. Students remarked that the changing requirements were more easily managed with the Model 2 approach than with the page-only approach.

- It allows for the potential separation of developer roles. That is, JSP pages can be maintained by designers, while programmers can work with Servlets and the other Java classes. In fact, several of the student groups ended up with precisely this type of division of labour.

- It improves the maintainability of the site by reducing the amount of code duplication. Common data or business behaviors could be localized within classes rather than within pages. However, common user interface elements still require the less than ideal "include file" mechanism (though with enough course time, we could have introduced yet another technology, JSP Custom Tags, to deal with this problem).[6]

While better from a software architecture point-of-view, this integrated JSP, Servlet, and JavaBean approach was significantly more difficult for the students to work with in comparison to the ASP or JSP-only approach, and would be completely inappropriate for first-year students. It took students some time to accommodate themselves to the Model 2 approach. In particular, the students with the most ASP experience found it the most difficult to use an approach whose rewards are not immediately apparent. With the ASP (or JSP only) approach, one simply starts coding the web pages. The reward is that results

can happen quickly; the drawback is that the result can be fragile and difficult to maintain. With the JSP-Servlet-JavaBeans approach, more planning and effort is required to get results. It does, however, create a more maintainable and flexible system.

4. ASP.NET

In the Winter 2003 semester, the technology used in the ecommerce course switched from JSP to Microsoft's new ASP.NET. What was the motivation for the switch? One was the perceived decline in JSP's market share. While it is difficult to get an accurate estimation on market share (since corporate Intranet's and internal web applications are typically hidden from the methods used to ascertain usage),[7] there are some indications that JSP has lost ground to PHP and ASP.NET.[8] Another reason for the switch was instructor curiosity/masochism. But the principal reason for the switch is that ASP.NET, in the opinion of this author, is a better technology for teaching advanced web development.

ASP.NET is a key part of the .NET Framework, Microsoft's new Java platform-inspired architecture. ASP.NET is completely different from regular ASP, and has a considerable learning curve … even longer than that for JSP. It takes at least six to eight weeks before the students can start creating anything substantial. One of the reasons why ASP.NET is more difficult to learn than ASP or JSP is that ASP.NET introduces a host of declarative server controls which encapsulate complex web presentation behaviors.[9] For example, the `DataGrid` tag below declares a templated data grid control which, when programmatically bound to data, displays a list of values from a database table in an HTML table (see figure 5).

```
<asp:DataGrid id="myGrid" runat="server"
   GridLines="none" font-size="x-small" font-name="tahoma"
   cellpadding="4" bordercolor="black" borderwidth="1" >

   <HeaderStyle forecolor="white" backcolor="brown" font-bold="true" />
   <ItemStyle backcolor="palegoldenrod"/>
   <AlternatingItemStyle backcolor="beige" />
</asp:DataGrid>
```

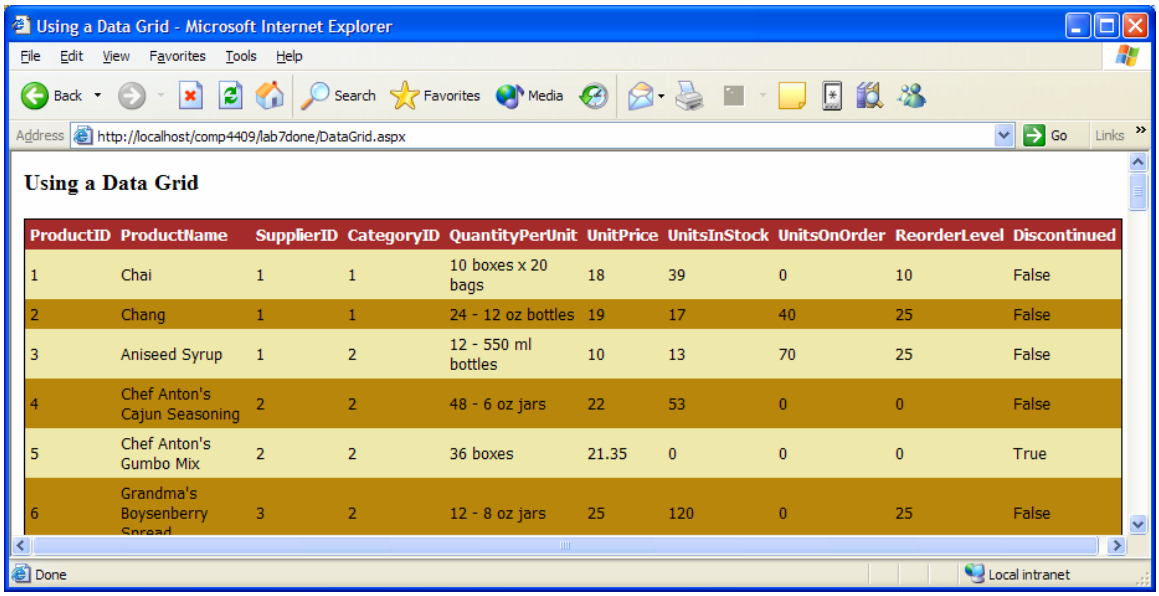

Figure 5

It takes some time and effort to familiarize oneself with most of the controls and their members. As well, the students had to learn a new language (they could use C# or VB.NET) and the new style and interactions between the programming and the presentation. In ASP.NET, presentation logic is formally separated from the presentation display (tags). Presentation tags are contained in one file (.aspx) and presentation logic is contained within a class in a "code-behind" file (.cs or .vb). Knowledge of inheritance is very helpful since the code-behind class is the base class for the .aspx file which in turn inherits from a powerful Page class (see figure 6).

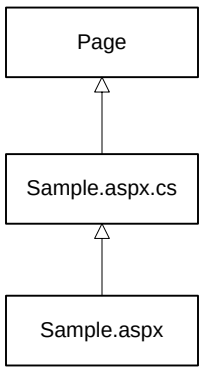

Figure 6

Another level of complexity in ASP.NET is the pseudo event-driven nature of the code-behind class. Server-side processing takes place within server-based events such as `Page_Load` and `Button_Click`. Unlike JavaScript events which are raised and handled on the client, ASP.NET events are generally triggered on the client, but always processed on the server. Given the difficulty new web development students have with programming the more complex web interactions, this extra level of abstraction in ASP.NET would, in this author's opinion, be too difficult for the typical first-year students.

There are, however, a number of benefits once that steep learning curve is surmounted. The powerful data binding syntax of the declarative server controls facilitates the construction of most common data-driven web sites. For instance, the example below shows the programming code to bind the `DataGrid` (shown in figure 5) to a data source. Only a few lines of code are required; the presentation details are described not by coding but via the `DataGrid` tag itself. Most of the server controls in ASP.NET can be bound to any object that implements the `IList` interface (this includes arrays, custom collections, data readers, etc).

```
Protected DataGrid emp;
...
emp.DataSource = someobject;
emp.DataBind();
```

In JSP (without custom tags), this would require a loop similar to the following:

```
<%
number_records = ...
number_fields = ...
%>
...
<table>
  <% for (int i=0; i<number_records;i++) { %>
  <tr>
     <% for (int j=0; j< number_fields; j++) { %>
       <td>
       <%= somefieldvalue %>
       </td>
     <% } %>
  </tr>
  <% } %}
```

```
</table>
```

Like with the JSP-Servlets-JavaBeans approach, a developer can also use other regular C# or VB.NET classes to architect helpful web patterns such as the Front Controller or the Intercepting Filter patterns.[10] In this author's opinion, it is easier for students to create an architecturally-sound web application with ASP.NET than with the JSP, Servlets, JavaBeans combination. With the former, students have only two sets of syntaxes and semantics to master: that of the tags and that of the language used (e.g., C#) for the site's supporting classes; with the later, to achieve the same level of functionality and design, the students have to master JSP, Servlets, JavaBeans, and perhaps EJBs and custom tags as well. Students will have to do more coding and debugging due to the less powerful declarative tags within JSP. (However, if one was to use one of the powerful web development frameworks built on top of JSP such as JavaServer Faces or Jakarta's Struts, then the students would have a development experience comparable to ASP.NET in power and ease of use). As well, the IDE available for ASP.NET – Visual Studio.NET – is, in this author's opinion, significantly more powerful than most available Java IDEs (though perversely, the students were initially encouraged *not* to use it).

5. Concluding Observations

Students who had exposure to both technologies (when the *Electronic Commerce* course switched to ASP.NET, the JSP material moved for one term into another third year course on *Emerging Technology*), expressed a marked preference for ASP.NET. Perhaps more importantly, the students were more consistently able to create a more sophisticated final term project using ASP.NET rather than JSP. The ASP.NET project the students created had almost twice as many functional requirements than the JSP version. It integrated a web services-based credit card checkout system, a fully realized checkout pipeline, and a more complex set of additional requirements (user polls, product reviews and ratings, most popular product highlight, etc.). As such, we are committing to ASP.NET, due to its power and the greater ease of creating enterprise-quality web applications, for our third-year *Electronic Commerce* course. As for our first year course, the page scripting

approach is quite suitable for first-year students in that they can be quite productive quite quickly. We are thus sticking with ASP for next year, even though ASP itself is a technology definitely on the wane.[11] After that, we may switch to PHP as an alternative introductory page scripting technology.

References

[1] – For instance, see IS 2002 Model Curriculum, available http://www.is2002.org (April 2004), or the ACM/IEEE-CS Task Force on Computing Curricula 2001, available http://www.computer.org/education/cc2001/final/index.htm (April 2004).

[2] – Arthur H. Lee, "A Manageable Web Software Architecture: Searching for Simplicity," *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, Vol. 35 No. 1 (January 2003).

[3] – K. Treu, "To teach the Unteachable Class: An Experimental Course in Web-Based Application Design," *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* Vol. 34 No. 1 (2002).

[4] – Deepak Alur, John Crupi, and Dan Malks, *Core J2EE Patterns: Best Practices and Design Strategies, Second Edition* (Prentice Hall, 2003). See also "J2EE Patterns Catalog," available http://java.sun.com/blueprints/patterns/catalog.html (April 2004).

[5] – Inderjeet Singh, Beth Stearns, and Mark Johnson, *Designing Enterprise Applications with the J2EE Platform, Second Edition* (Addison-Wesley, 2002), p. 115.

[6] – The reason why include files are less than ideal is the fact that the run-time environment (in our course it was Tomcat) parses JSP pages into Java classes, compiles them, then executes them. Every time a JSP page is requested, the run-time environment checks the date stamp of the file to see if it needs reparsing and recompiling. Unfortunately, changes to include files are not detected. That is, after changing an include file, one has to resave all JSP files that use it.

[7] – Some statistics seem to show that either PHP or ASP is currently the market leader in server-side technologies (see, for instance those at http://www.netcraft.com). Yet, the netcraft numbers, for instance, only show the number of servers *able* to serve PHP, JSP, etc. Google seems to also provide some indication of relative use. It has indexes on 234 million asp pages, 302 million PHP pages, 33 million JSP, and 21 million ASP.NET pages. However Google, Netcraft and other companies that use spiders or automated surveys to determine market share for a server-side technology are unable to get past the firewalls which protect most intranets. ASP.NET and JSP are arguably used more for intranet applications than for the public pages indexable by Google. As well, these methods can only track pages

using the default extensions (.jsp, .asp, etc); yet, a common feature of the large sites created with content management systems is that they do not use the default extensions.

[8] – Given that ASP.NET was only released in 2002, the growth in its numbers relative to JSP does indicate that it may (or already has) overtaken JSP in the enterprise market; also, the large number of ASP sites are more likely to be upgraded to ASP.NET rather than to JSP. See "ASP.NET Overtakes JSP and Java Servlets," available http://news.netcraft.com/archives/2004/03/23/aspnet_overtakes_jsp_and_java_servlets.html (April 2004). See also Eric Knorr, "Developers Blaze their Own Trail," *InfoWorld* Vol. 25, No. 38 (September 2003).

[9] – JSP Custom Tags are somewhat equivalent to the server controls of ASP.NET.

[10] – After some reluctance, Microsoft now seems to have embraced the importance of enterprise patterns. In the summer of 2003, Microsoft opened a dedicated pattern and best practices site at http://www.microsoft.com/resources/practices. In it, one can even find that Microsoft has adopted Sun's nomenclature for the web presentation patterns!

[11] – Perhaps not in industry use, but textbooks and other online resources are becoming less and less common.