Slickr: A Multi-Tiered Web Development Capstone Project Using Databases, Web Services, and AJAX

Mark Frydenberg
mfrydenberg@bentley.edu

Computer Information Systems Department
Bentley College
175 Forest Street
Waltham, MA 02452 USA

Abstract

Current web applications are increasing in popularity because of their browser-rich interfaces and ability to incorporate information from a variety of sources. This paper presents a simple photo sharing application and enhancements to it using Web Services and ASP.NET AJAX in order to illustrate some of the implementation details of Web 2.0 applications on a very small scale. Early in the course, students create web-based applications whose pages have dynamic content obtained from a SQL Server Database. Students later repackage some of their queries to share as web services, and then improve upon the user interface by incorporating AJAX enhancements. The project may serve as a capstone assignment in an undergraduate web application development course, where students use ASP.NET and C# with Microsoft Visual Studio 2005. The paper argues that ASP.NET AJAX provides a new motivation for teaching web services. The paper also discusses pedagogical values and new opportunities as a result of this approach.

Keywords: ASP.NET, Web Services, AJAX, Web 2.0, Multi-Tiered Application Development, capstone, Visual Studio

1. INTRODUCTION

Web 2.0 is has become a buzzword for applications often characterized by the ability to tag and share information, collaborate, “mash-up” content from different web sources in a single application, and interact with rich user interface elements running within a browser that resemble the complexity of interfaces found in traditional desktop applications. (O'Rielly, 2005) Web services enable the sharing of information between applications that may, and often, run on different platforms or servers, through the use of well-defined protocols and standards.

AJAX (Asynchronous JavaScript and XML) is a new combination of existing technologies that enables the creation of sophisticated user interfaces within a web browser. Microsoft’s ASP.NET AJAX Toolkit, released in Fall, 2006, is a framework that simplifies the creation of sophisticated user interfaces for ASP.NET applications through the use of re-usable components.

This paper describes an example that makes use of web services and AJAX as two tools for an incremental capstone project in a multi-tiered application development course. Students create a simple photo sharing application that is data driven, and then enhance it to make use of web services for sharing and searching photos. Finally, students incorporate AJAX elements to enhance the user interface of their applications. In doing so, they interact with current technologies and gain an appreciation for the elements of Web 2.0 applications.

2. TECHNOLOGY OVERVIEW

Web Services

Web services "extend the reach of [web] applications by allowing them to communicate with other applications." (Homer & Sussman, 2006) Through a series of defined protocols and standards, web applications may remotely invoke methods that reside on servers elsewhere. The results of a web service call are always in given in XML (eXtensible Markup Language) format.

Several protocols and standards including SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and Integration) are responsible for accessing, describing, and discovering web services. Along with HTTP (Hypertext Transfer Protocol), these form the "plumbing" between applications that consume web services and the services themselves.

In Microsoft Visual Studio, one adds a web reference to specify a web service that an application will consume. Doing so causes Visual Studio to transparently handle all of the details for creating the associated WSDL documents, SOAP headers, and proxy classes, the technical details of which are beyond the scope of this paper.

By providing methods that are callable from external applications, web services create a new way for organizations to share their data with other applications over the web. Google and Amazon.com make web methods available; other web services are available to obtain current stock quotes, weather information, and news headlines. The web site xmethods.net is a directory of several web services that are freely available.

AJAX

AJAX (Asynchronous JavaScript and XML) has achieved recent popularity as a tool for creating dynamic web pages in which only a part of the page updates at a time. For example, one might call a web service for user authentication from a client script. As a result, it is possible to create sophisticated user interfaces for applications that run within a web browser.

With AJAX applications, JavaScript function calls run independently (asynchronously) of the loading of a web page. The function calls may invoke web services or other JavaScript methods to obtain new information with which to update the page, and only a portion of the page may be updated without having to refresh the entire page. As a result, it is possible to provide a more interactive user experience while maintaining very fast response times.

In Fall 2006, Microsoft introduced the ASP.NET AJAX toolkit, a framework for building AJAX applications within ASP.NET. (Microsoft Corporation) The toolkit includes a number of pre-defined user interface controls such as dependent dropdown lists, accordion controls for animating the display of a page region, and controls for predictive type-ahead (similar to Google Suggest), for creating improved user experience when visiting a web page.

Every ASP.NET AJAX enabled page contains a ScriptManager control and an UpdatePanel control. The ScriptManager manages all of the basic ASP.NET AJAX components on the page. Setting the ScriptManager's EnablePartialRendering attribute to *true* enables the ability to update only a portion of the page. The UpdatePanel contains the content that will be updated without refreshing the entire page. (See Figure 2.)

The ability to invoke an ASP.NET web service from client-side code without having to post back to the server provides a new motivation for students to learn about web services. Controls on the page that receive the updated data from a web service call are refreshed, enhancing the user's experience of the web application. Thus the user interface has much more of a desktop application "look and feel" despite the fact that it is running within a web browser.

3. TEACHING WEB SERVICES

Strategies for Teaching Web Services

While little has been written about integrating web services into the curriculum (Peslak, 2006) they are becoming a standard topic in college level web development courses. In many web application development courses (Humphrey, 2004; Lim, 2006; Peslak, 2006), web services are introduced near the end of the course, after previously covering topics in client/server programming, sessions, security, and web database access techniques. As web service consumers, students often create new web applications that integrate external content available through web services. As web service producers, students learn to repackage existing applications by creating APIs that enable the sharing of functional units with other applications. Web services add a "real world" perspective to application programming as students interact with APIs provided by Google, Amazon.com and others.

Assuncao and Osorio (2005) taught web services as part of a Distributed Systems architectures course. Their course spends four weeks on topics related to web services, introducing WSDL as a service interface description, and SOAP as a way to invoke service operations. Next, they create a simple web service, deploy and test it. They look at proxies from existing web services from Google and Amazon, and then introduce students to Visual Studio as an integrated development environment for creating web services. Their course delves deeper into advanced topics such as generating WSDL files, SOAP headers, state management, and Web Services Interoperability (WS-I) specifications.

Lim, Jong, & Mahatanankoon (2005) present a number of scenarios for integrating web services into CS1 and CS2 courses. In CS1 courses, web services might be introduced as "methods [that] may have been written by others, ... [that are] scattered all over the world but … are callable from the Web." Traditional programming structures of sequence, selection, and repetition may be taught through the creation of mash-ups, programmatic invocations of web methods that may be combined to solve real world problems, such as "plot on a map the United States city with the warmest temperature." In CS2 courses, invoking third-party web services such as Google's Web Services API requires working with complex data structures that contain real data. They argue that these examples "will not only fascinate the students with its interesting collection of activities, but also inspire and prepare them for real-world software development scenarios when they graduate."

Neubauer (2007) has investigated techniques for teaching SOA to non-technical students. "The challenge of teaching SOA is that it requires programmers and … managers to learn to think of what organizations do in terms of business processes, workflows and 'services'" but the notion of services is "unfamiliar and not particularly intuitive." Students created original web services and then applications to consume them in order to better understand how different software components may be combined together to create a functional system.

Teaching Web Services in a Web Development Course

CS 380 is a Multi-Tiered Application Development course for students in the BS in Computer Information Systems program at a New England business college. Topics in the course include data access models, application development life cycle models, creating and integrating a user interface, and application design. Web services are introduced and students create applications that invoke simple web methods.

Students in the course have completed two semesters of object oriented programming (in Java) and at least one database course. They have already learned to create personal web pages in their first year Introduction to Technology courses, and some may have taken an additional World Wide Web course covering HTML and JavaScript.

The goals of the course are to develop individual competency in creating web-based applications using a contemporary integrated, object-oriented development environment. Students apply their programming, modeling, database, and networking skills from previous courses in a business application development and maintenance context, creating new or enhancing existing solutions. Thus it is natural for the capstone development course to require demonstrated individual competence by having students build a small-scale real-world project. The course meets one evening per week for 150 minutes; in each class session, the instructor usually gives a demo and lectures on course concepts, and then students complete a short hands-on activity. There are about five or six implementation assignments over the course of the semester; some later projects may build on solutions completed in earlier ones. Students also complete a "community assignment" in which they must visit one ASP.NET user group meeting or Visual Studio developer's activity at the local Microsoft office during the semester, and write a short paper about what they learned.

CS 380 follows similar steps as Peslak (2006) in teaching web services. An introductory lecture presents basic web services concepts including SOAP, XML, and UDDI. Unless one is teaching an entire course on web services, it may be sufficient in an undergraduate course that is introducing web services to present an overview of the terminology for these standards and protocols. By doing so, students can recognize the kinds of information each of the files contains. A deeper understanding is probably not necessary.

Students first learn to write and test HelloWorld() and other simple web methods that return scalar values (such as multiplying two numbers and returning their product). Then they create simple applications to invoke these methods.

Next, students learn to use real, third-party web services that are freely available at web sites such as xmethods.net. Students examine the methods, test invoking them to see the format of the data returned, and write their own applications to consume them. This gives students experience dealing with real-world APIs, and allows students to see the variety of applications which expose their data as web services.

Finally, students learn to write their own methods that return a list of user-defined objects, and to create applications that invoke them.

One example shown in class demonstrates how to take a previously completed database query or stored procedure and package it as a web service. Doing so gives insight into how a company might implement web services to expose its product information for customers to use on their web sites. For example, an Amazon.com web service includes a method to search the Amazon.com database for a book, given its ISBN number, and return book information and cover images. This example gives students a sense of how such commercial web services might be written.

4. TEACHING AJAX

Teaching AJAX

AJAX technologies have been embraced by several popular web sites: Google Maps, NetFlix movie previews, Flickr photo sharing, and new versions of web-based email clients such as Gmail and HotMail all make use of AJAX. Teaching students to use AJAX in their own applications gives them a sense of what is required to create the more complex applications with which they are already familiar.

ASP.NET AJAX provides a rich set of built-in controls to simplify the process. ASP.NET AJAX creates JavaScript proxy classes for ASP.NET web services automatically. While the implementation is transparent to the user within Visual Studio 2005, the details may be explained at an appropriate level depending on the class.

Another motivation for teaching AJAX using the ASP.NET AJAX Framework and Toolkit is that it provides a new application for web services. Some ASP.NET AJAX controls, such as a CascadingDropDown control, may be populated with results obtained by calling a web service method. When those methods invoke previously written database queries, several course concepts are integrated in one example.

An AJAX Example: Dependent DropDown Lists

A common example of a user interface solution that becomes more elegant with AJAX is the cascading or dependent drop down list. Consider a dropdown list of U.S. states, and another with cities; after selecting a state from the States list, the Cities list should display only those cities in the selected state. Selecting a city displays the City, State pair in a label on the web page.

In a traditional, non-AJAX application, selecting the state would cause a postback to the server, and then the cities list would be populated. The contents of the entire page would be redrawn. Implementing this functionality using AJAX requires only the Cities list and resulting label to be redrawn.

One implementation might query a database containing the appropriate city/state information, placing the results in a SqlDataAdapter, and then copying them into an array of CascadingDropDownNameValue pairs, which will be used, to populate the dropdown lists. The implementation details are omitted. The method signatures are shown in Figure 1.

```
[WebMethod]
    public
CascadingDropDownNameValue[]
GetStates(string
knownCategoryValues, string
category)
    { // code omitted
    }
 [WebMethod]
    public
CascadingDropDownNameValue[]
GetCitiesInState(string
knownCategoryValues, string
category)
    { // code omitted
    }
```

Figure 1. Web Methods that return CascadingDropDownNameValue pairs.

Using the ASP.NET AJAX toolkit, one may place a CascadingDropDown control onto an ASP.NET web form. By providing values for its ServicePath and ServiceMethod attributes, one specifies the names of the path to, and methods in a web service.

The CascadingDropDown controls invoke the GetStates and GetCitiesInState web methods directly through the ServiceMethod attributes, as shown in Figure 2. When the City dropdown list's SelectedIndexChanged event is invoked, that triggers the asynchronous post back trigger to update the label on the page with the selected city/state pair. Error-checking code to ensure a value is selected from both the city and state dropdown lists has been omitted from the ddlCity_ SelectedIndexChanged event handler for the sake of brevity.

```
<asp:ScriptManager
  ID="ScriptManager1"
  EnablePartialRendering="true"
  runat="server" />

State:
<asp:DropDownList
   ID="ddlStates"
   runat="server"  />
<br/>

City:
<asp:DropDownList
  ID="ddlCity"  runat="server"
  AutoPostBack="true"
  OnSelectedIndexChanged=
"ddlCity _SelectedIndexChanged" />

<ajaxToolkit:CascadingDropDown
 ID="cddState"
 runat="server"
 TargetControlID="ddlStates "
 ServicePath="WebService.asmx"
 ServiceMethod="GetStates"
 Category="State"
 PromptText="Select a State"/>

<ajaxToolkit:CascadingDropDown
  ID="cddCity"
  runat="server"
  TargetControlID="ddlCity "
  ParentControlID="ddlStates "
  ServicePath="WebService.asmx"
  ServiceMethod="GetCitiesInState"
  Category="City"
  PromptText="Select a City"
  />

<asp:UpdatePanel
  ID="UpdatePanel1"
  runat="server">

      <ContentTemplate>
      <asp:Label
         ID="Label2"
         runat="server"
         Text=""
      /> <br />
      </ContentTemplate>
```

```
     <Triggers>
      <asp:AsyncPostBackTrigger
   ControlID="ddlCity "
EventName="SelectedIndexChanged"
/>
      </Triggers>
  </asp:UpdatePanel>
```

```
protected void ddlCity
_SelectedIndexChanged(object
sender, EventArgs e)
 {
 string city =
   ddlCity.SelectedItem.Text;
 string state =
   ddlStates.SelectedItem.Text;
 Label2.Text =
   city + ", " + state;
 }
```

Figure 2. ASP.NET Controls ScriptManager and UpdatePanel handle refreshing a portion of the page. The event handler code processes the selected item from the dropdown list on the CascadingDropDown page.

Students can create sophisticated user interfaces with very little coding by using the toolkit controls.

5. Slickr: A CAPSTONE PROJECT USING A SQLServer DATABASE, WEB SERVICES, AND ASP.NET AJAX

Phase 1: Slickr

Overview. CS 380 students use Microsoft Visual Studio with ASP.NET in C# to create data-driven distributed web applications. For their database assignment, students created a simple photo-sharing web application dubbed "Slickr", a simple imitation of the popular Flickr photo-sharing web site. The assignment had students create a Slickr web site on which users may perform the following tasks:

- Create albums and upload photos into specific albums
- Associate a description and one or more user-specified tags with a photo
- Search albums for photos with specific tags, and display matching photos
- Ability to edit and delete album or tag names, and photo descriptions

To fulfill the technical requirements for this assignment, students also had to create C# classes to model all relevant objects, define and invoke at least one stored procedure, and implement a data access layer class. In addition, their application pages had to make use of both SqlDataSource and ObjectDataSource controls a least once.

To encourage individual creativity, students also had to identify an additional feature to implement in their Slickr application. Some students made these enhancements:

- Designating a picture as the "album image" to be displayed along with the album name.
- Rating pictures numerically and displaying an average rating for each.
- Implementing "tag clouds" to display more frequently used tags in larger fonts, and less frequently used tags in smaller fonts.

In a later lesson on ASP.NET security controls, students enhanced the project to create two users with different privileges. An administrative user had privileges to search and view pictures, upload pictures, and edit titles, tags and descriptions. A standard user could only search and view pictures.

Some students added the ability for users to "log in" and access their own set of albums, to be able to perform the functionality above. This required adding a User table, and a composite primary key to the Album table, so that one user might have many albums.

Object Model: While it may be natural to begin by designing the database tables, modeling the objects independent of any database implementation helps to better consider the attributes and methods associated with the objects themselves, and those associated with their data access. This model, shown in Figure 3, separates the information inherent to a particular object from the additional information (primary keys) that will be required to store it in a database.

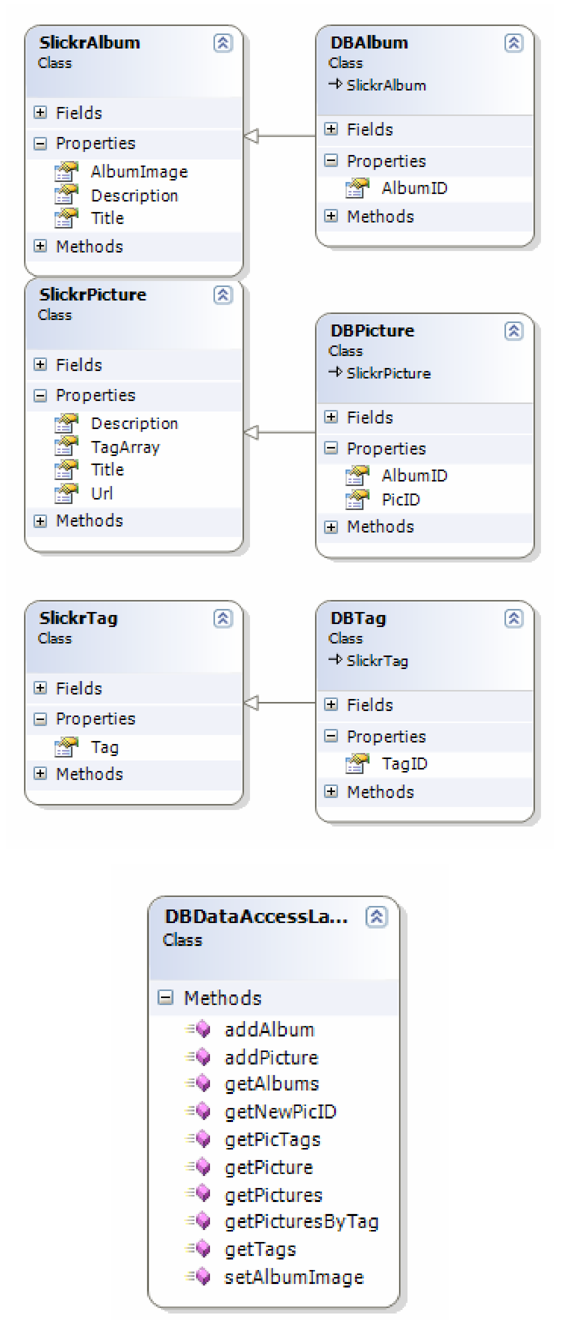


Figure 3. Slickr Object Model

SlickrAlbum is a class describing photo albums; DBAlbum is a subclass containing the database id's for the album and the id for its album image, so that the application may write the information about the album to a database. The other classes are implemented similarly. In this way, the model separates the database information from the object information. This implementation will prove beneficial in the third phase, when only information about the objects will be shared with other applications via web services.

Database Model: The required database tables are relatively straightforward, as shown in Figure 4. The PicTag table, relating many pictures with many tags, is the only table that requires the use of a composite primary key.

Students had to create constraints to handle cascading deletes, so that deleting an album deleted all pictures in it, and deleting a picture deleted the appropriate entries from the PicTag table. For the purposes of this assignment, once a tag is entered into the Tag table in the database, it stays there forever.

Primary keys albumID, picID, and tagID are set up as Identity columns in SQL Server, so that the user may use SCOPE_IDENTITY() to obtain the most recently used primary key value when inserting a new item into the corresponding tables.

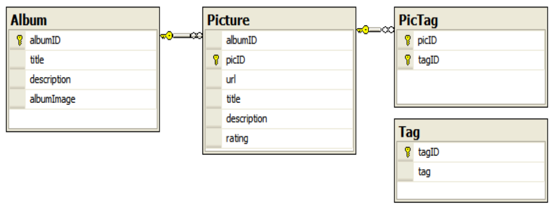


Figure 4. Slickr Database Design

The methods in the Data Access layer, as shown in Figure 5 serve as the middleware between the object classes and the database. Creating a data access layer promotes the concept of a multi-tiered architecture for the application.

Students then implement methods to add albums and pictures to the database, search for pictures by their associated tags, retrieve the album names and tag names, and obtain all of the tag names associated with a given picture.

```
public static class DBDataAccessLayer
{
  public static void addAlbum(
    string title, string description,
    int albumImagePicID) { … }
  public static void addPicture(
    DBPicture picture) { … }
  public static DBPicture
    getPictureInfo (
    int picID) { … }
  public static void setAlbumImage (
    int albumID,
    string albumImageURL) { … }
  public static List<Album> getAlbums
    () { … }
  public static List<SlickrPicture>
    getPictures() { … }
  public static List<SlickrPicture>
    getPicturesByTag(string tag){ … }
  public static List<SlickrTag>
    getTags() { … }
 public static List<SlickrTag>
    getPicTags(int picID) { … }
}
```

Figure 5. Data Access Layer Methods.

The Data Access Layer: Most of the methods require students to create an API for relatively simple database queries. Their implementations are straightforward, and have been omitted here for sake of brevity. The only method that is a bit more involved is addPicture. It must accomplish three tasks:

- Perform an Insert query to add a new picture to the Picture table
- Determine if a tag is already in the Tag table, and if not, add it
- Insert the picID / tagID pair into the PicTag table

It is useful to note that many of the "get" methods return a generic List<> of base class objects. This will be useful in phase 2, when these queries are repackaged for use in methods of a web service.

With this framework in place, students create these web pages for their Slickr projects:

- Create an album
- Upload pictures to a specified album, specifying tags and a description for each photo
- Browse/Search for pictures (view all or by tag, and for a specific album)

Figures (b) and (c) in the Appendix show the New Album and Search pages in one student's completed project.

Phase 2: Web Services Enhancement

This framework provides significant complexity and structure for incorporating web services into the application. In the web services enhancement to the Slickr project, students complete these steps:

- Create a web service with methods to expose their photos
- Rewrite their search and display pages to invoke their web methods
- Create a page to search and display photos from a classmate's Slickr project by invoking the classmate's web service methods

By creating and consuming each others' web services, students act as both web service producers and consumers, and come to understand the need for standard APIs. Students also get a sense of the important role that web services play in making data available for sharing between applications, and the role of XML as a standard for representing that data. The instructor provided a sample web service for those students whose partners had difficulty completing this part of the assignment.

With the relevant queries neatly packaged as methods within the data access layer, repackaging them as web services in the SlickrWebService class is trivial. Each web method invokes a previously written query or stored procedure from the data access layer.

- Get Tags() returns a list of all of the tags from the Tag table in the database.
- GetPicturesByTag(string tagName) returns a list of all pictures from the Picture table that match the specified tag.
- GetPictures() returns all of the pictures from the Picture table.

Students used GetTags() to populate a list box (or in one case, tag cloud) with the names of the available tags, and

GetPicturesByTag() to display those pictures that match the selected tags.

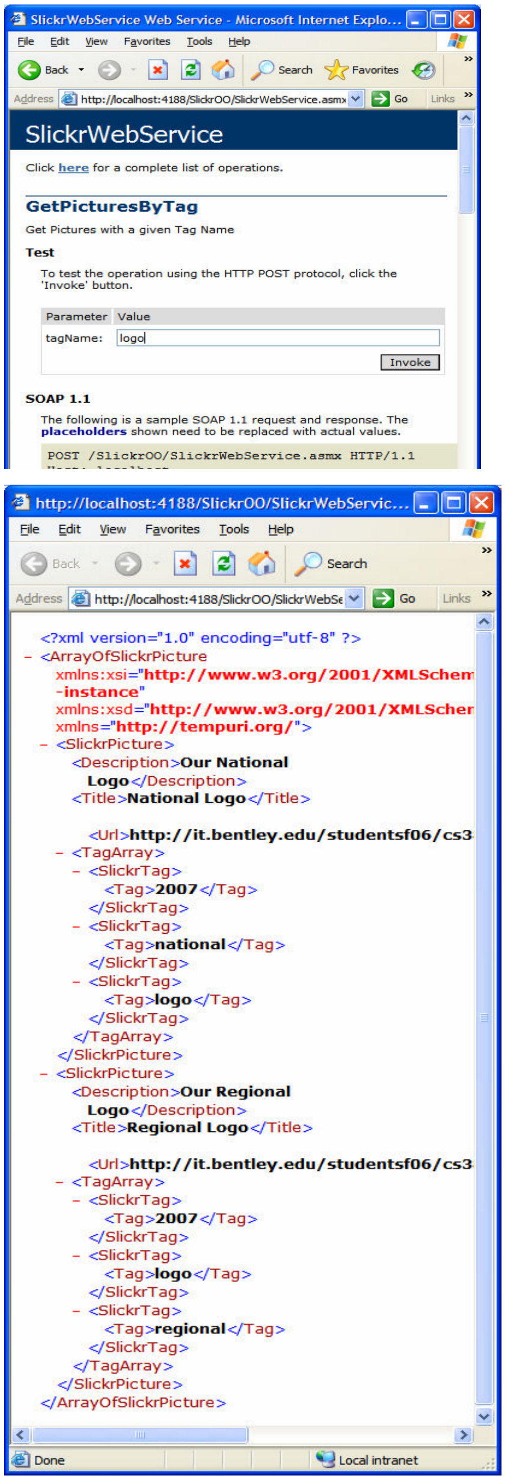
The base classes SlickrTag and SlickrPicture expose only those items that the web methods need to return, as a List<> of these objects. Figure 6 shows the implementation of the associated web methods.

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Collections.Generic;

[WebService(Namespace =
"http://tempuri.org/")]
public class SlickrWebService :
System.Web.Services.WebService {

public SlickrWebService () {}

 [WebMethod(Description = "Returns
list of tags.")]
  public List<SlickrTag> GetTags()
  {
  return DBDataAccessLayer.getTags();
  }

[WebMethod(Description = "Get
Pictures with a given Tag Name")]
   public List< SlickrPicture>
GetPicturesByTag( string tagName )
{
    return
DBDataAccessLayer.getPicturesByTag(
    tagName);
}

  [WebMethod(Description = "Get all
Pictures")]
public List< SlickrPicture>
GetPictures()
{
    return
DBDataAccessLayer.getPictures();
}

}
```

Figure 6. Slickr Web Service and methods.

Web methods must return data about the pictures in a form that other web applications can use directly. For example, the url returned in a SlickrPicture must be absolute (i.e.,http://myserver.com/me/slickr/images/mypic.jpg) while the value stored in the database is simply the name of the image file.

Figure 7 shows the format of the sample data returned`in XML format when testing the GetPicturesByTag() web method.


SlickrWebService Web Service - Microsoft Internet Explo...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:4188/SlickrOO/SlickrWebService.asmx Go Links

SlickrWebService

Click here for a complete list of operations.

GetPicturesByTag

Get Pictures with a given Tag Name

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
tagName:	logo

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /SlickrOO/SlickrWebService.asmx HTTP/1.1
```

http://localhost:4188/SlickrOO/SlickrWebServic...
File Edit View Favorites Tools Help
Back Search
Address http://localhost:4188/SlickrOO/SlickrWebSe Go Links

```
  <?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfSlickrPicture
    xmlns:xsi="http://www.w3.org/2001/XMLSchem
    -instance"
    xmlns:xsd="http://www.w3.org/2001/XMLScher
    xmlns="http://tempuri.org/">
  - <SlickrPicture>
      <Description>Our National
        Logo</Description>
      <Title>National Logo</Title>

        <Url>http://it.bentley.edu/studentsf06/cs3
    - <TagArray>
      - <SlickrTag>
          <Tag>2007</Tag>
        </SlickrTag>
      - <SlickrTag>
          <Tag>national</Tag>
        </SlickrTag>
      - <SlickrTag>
          <Tag>logo</Tag>
        </SlickrTag>
      </TagArray>
    </SlickrPicture>
  - <SlickrPicture>
      <Description>Our Regional
        Logo</Description>
      <Title>Regional Logo</Title>

        <Url>http://it.bentley.edu/studentsf06/cs3
    - <TagArray>
      - <SlickrTag>
          <Tag>2007</Tag>
        </SlickrTag>
      - <SlickrTag>
          <Tag>logo</Tag>
        </SlickrTag>
      - <SlickrTag>
          <Tag>regional</Tag>
        </SlickrTag>
      </TagArray>
    </SlickrPicture>
  </ArrayOfSlickrPicture>
```

Figure 7. Testing a Web Method and examining the XML-formatted data it returns.

Phase 3: AJAX Enhancements

Students had to implement two AJAX enhancements in their Slickr projects. The first was to develop a slide show to display randomly chosen photos every two seconds. The second was to select at least one additional control from the ASP.NET AJAX toolkit to incorporate into the project. Their AJAX enhancements included:

- Using the Ratings control to rate photos, and display an average rating for each.
- Using the ResizableControlExtender for Dynamic resizing of photos.
- Using CascadingDropDown controls to display drill down queries from albums to pictures within a selected album, or to display tags and pictures by tag.

Appendix figures (a) and (e) show screen shots from two students' solutions that illustrate the first two of these features.

Figure 8 shows a code fragment from the SlideShow.aspx page. The page invokes the web service's GetPictures() method to retrieve the photos, and then uses the AJAX Timer control to select one at random, every two seconds, as specified by the Interval=2000 attribute. This was a good example of how only the portion of a page within the UpdatePanel control updates. Students appreciated that they were able to re-use code they had already written (and that was already working) as part of this exercise.

```
<asp:ScriptManager
id="ScriptManager1" runat="server"
EnablePartialRendering="true">
    </asp:ScriptManager>
    <br />
    <asp:UpdatePanel
id="UpdatePanel1" runat="server">
        <ContentTemplate>
 <asp:Image id="Image1"runat="server"
      ImageUrl="~/data/logo.jpg"
</asp:Image>
<br />
<asp:Button id="Button1"
onclick="Button1_Click"
runat="server" Text="Start">
</asp:Button>
<asp:Timer id="Timer1" runat="server"
 Interval="2000" OnTick="Timer1_Tick"
 enabled="false">
    </asp:Timer>
    </ContentTemplate>
</asp:UpdatePanel>
```

Figure 8. SlideShow.aspx

Figure 9 shows the code for handling the events to load the page and select a new photo every two seconds. The helper method GetRandomPicture() selects a random photograph from the list that the web service returns. The Timer's Tick method randomly selects a new photo. The Button's Click method turns the timer on or off.

```
public partial class Slideshow :
System.Web.UI.Page
{
    public SlickrWS.SlickrWebService
ws = new SlickrWS.SlickrWebService();
    public SlickrWS.SlickrPhoto[]
photos;

    protected void Page_Load(object
sender, EventArgs e)
    {
        photos = ws.GetPictures();
        Image1.ImageUrl =
GetRandomPicture ();
    }
    protected void Timer1_Tick(object
sender, EventArgs e)
    {
        Image1.ImageUrl = GetPhoto();
    }
    protected void
Button1_Click(object sender,
EventArgs e)
    {
        if(Timer1.Enabled)
        {
            Timer1.Enabled = false;
            Button1.Text = "Start";
        }
        else
        {
            Timer1.Enabled = true;
            Button1.Text = "Stop";
        }
    }
    private string GetRandomPicture()
    {
        Random r = new Random();
        int n =
r.Next(0,photos.Length - 1);
        return photos[n].Url;
    }
}
```

Figure 9. Event handling methods for the SlideShow page.

Figure (d) in the Appendix shows the slide show in operation.

It is worthwhile to note that this code for a web application resembles code for a similar desktop application that one might write with Visual Basic or C#. This is a simple enhancement of a web application that has the look and feel of a desktop application. Figure (d) in the Appendix shows screen shots of the SlideShow page in one student's Slickr project.

6. ACKNOWLEDGEMENTS

The author acknowledges Hunter Jones and Adam Sternberg, two CS 380 students whose Slickr projects are shown in the Appendix. In addition, Hunter's comments on object design aspects of the Slickr project were valuable during the preparation of this paper.

7. CONCLUSION

CS 380 is the only course that students in order take to develop individual competency in creating distributed web applications. Skills learned in this course are useful in a future applied project management course. They come to have a better understanding of distributed application architecture. Said one student:

> CS 380 projects required me to think not only about the coding aspect, but the design of web applications even more. Learning which and how things work together, and performance issues were all important. Using server controls and web services gave me a better understanding of how web development works. When I visit a site such as Amazon.com now, I know that millions of users are connecting to their Web Services, and pulling data out of their public databases and into their own applications. By completing this [Slickr] project and other homework projects in CS 380, I now have a sense of the some of the components and related issues that must be considered when building large-scale web-based business applications.

Teaching web services to students in a Multi-Tiered Application Development course provides insights into reusability of software modules, underscores the importance of APIs and good software design, and offers a concrete example of distributed system architectures. By creating their own and calling each other's web service methods, students also learn the importance of abstraction and proper object-oriented design. ASP.NET AJAX provides a new outlet for web services, as several of the pre-defined controls in its toolkit accept input from web service methods.

Teaching AJAX also introduces students to an asynchronous architecture found in many current web applications. Students also learn to create their own web applications with more involved user interfaces.

The Slickr sample assignment described in this paper integrates all of these concepts. Completing it requires students to apply several course concepts into a single web site on a small scale that mimics a real-world Web 2.0 application.

8. REFERENCES

Assuncao, L., & Osorio, A. L. (2006). Teaching web services using .NET platform. ACM SIGCSE Bulletin (p. 339). New York, NY: ACM Press.

Connolly, R. (2005). A Funny Thing Happened on the Way to the Form: Using Game Development and Web Services in an Emerging Technology Course. Information Systems Education Journal , 38 (3), 1-9.

Homer, A., & Sussman, D. (2006). ASP.NET 2.0 Illustrated. Boston, MA: Addison-Wesley.

Humphrey, M. (2004). Web Services as the Foundation for Learning Complex Software System Development. 35th SIGCSE Technical Symposium on Computer Science Education (pp. 457-461). Norfolk, VA: ACM Press.

LaMonica, M. (2005, October 4). Ajax gives software a fresh new look. Retrieved May 1, 2007, from ZDNet: http://news.zdnet.com/2100-3513_22-5886709.html

Lim, B. B., Jong, C., & Mahatanankoon, P. (2005). On integrating web services from the ground up into CS1/CS2. Proceedings of the 36th SIGCSE technical symposium on Computer science education (pp. 241-245). St. Louis, Missouri: ACM Press .

Microsoft Corporation. (n.d.). ASP.NET AJAX:. Retrieved May 1, 2007, from ASP.NET AJAX: The Official Microsoft ASP.NET AJAX Site: http://ajax.asp.net/

Neubauer, B. J. (2007). Introducing SOA and workflow modeling to non-technical students. Journal of Computing Sciences in Colleges , 101-107.

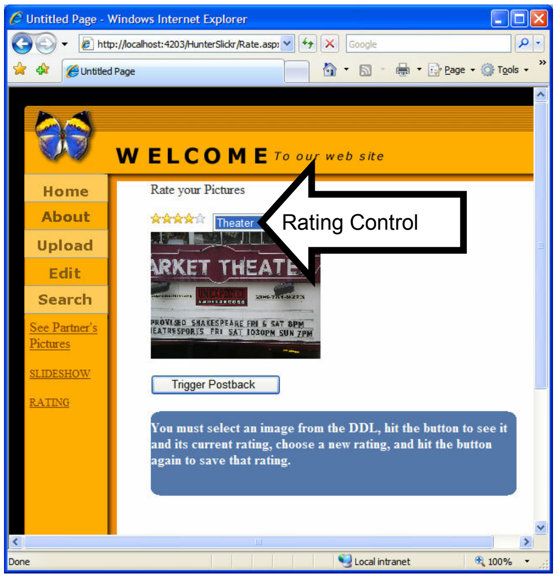
Noonan, R. E. (2007). A course in web programming. Journal of Computing Sciences in Colleges , 23-28.

O'Rielly, T. (2005, September 30). What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Retrieved May 1, 2007, from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Peslak, A. R. (2006). Web Services: Introduction and Travel Project Tutorials Using Visual Studio and ASP.NET. The Proceedings of ISECON 2006 (p. 15). Dallas: ISECON.

Appendix

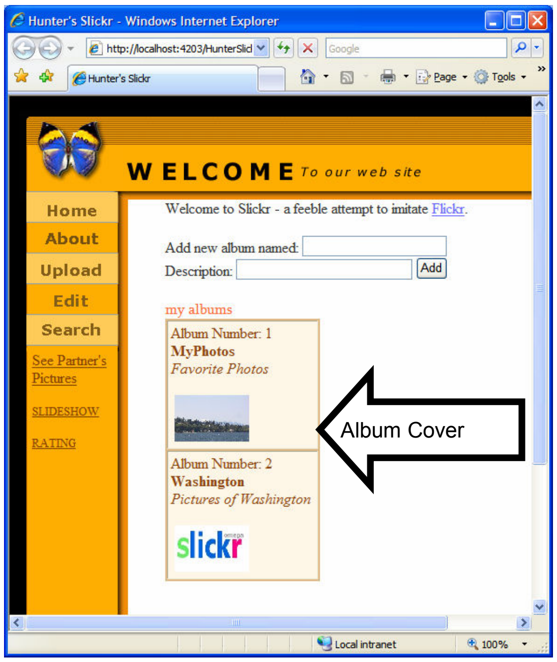
This Appendix includes several screen shots from two student implementations of the Slickr application.



Rate your Pictures

You must select an image from the DDL, hit the button to see it and its current rating, choose a new rating, and hit the button again to save that rating.

(a) Using the AJAX ratings control to rate photos. The user selects a photo from the dropdown list, and then assigns a rating to it.



Welcome to Slickr - a feeble attempt to imitate Flickr.

Add new album named:

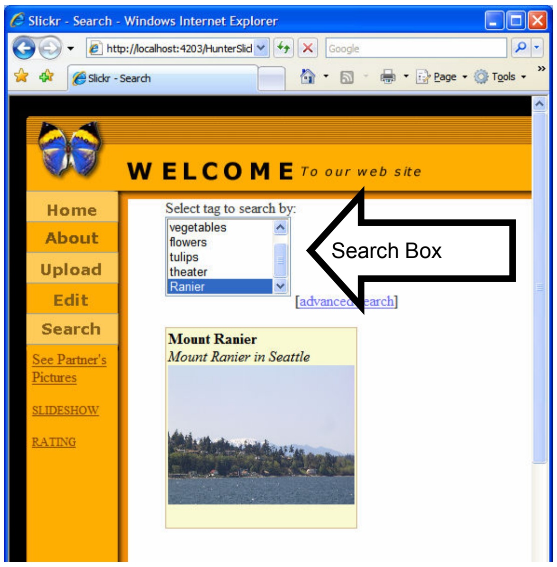
Description:

my albums

Album Number: 1
MyPhotos
Favorite Photos

Album Number: 2
Washington
Pictures of Washington

(b) Create new albums page. The first album has a photo selected for the album cover; the second does not.

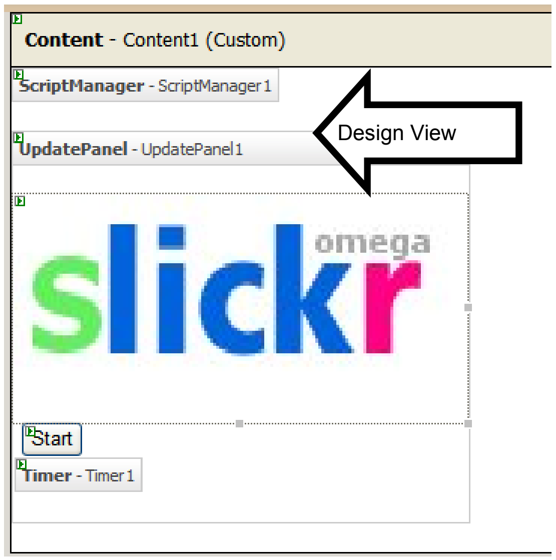


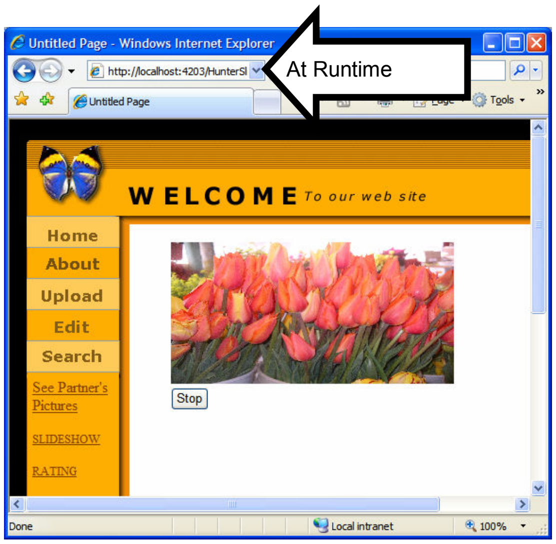
Select tag to search by:

vegetables
flowers
tulips
theater
Ranier

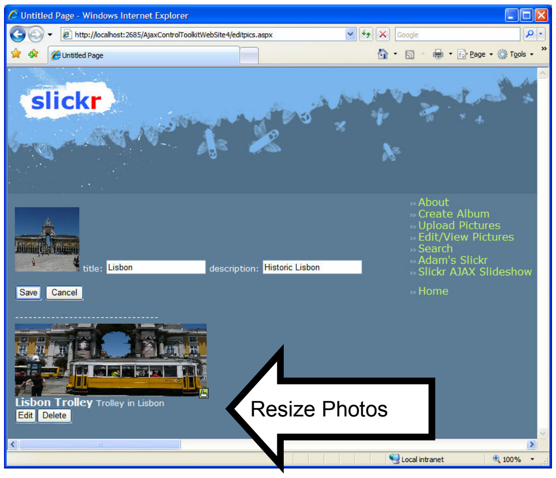
Mount Ranier
Mount Ranier in Seattle

(c) Search page. The user selects one or more tags from the list box, and sees all matching photos. Only the photos update.


Content - Content1 (Custom)
ScriptManager - ScriptManager1
UpdatePanel - UpdatePanel1
Design View
slickr omega
Start
Timer - Timer1



(d) SlideShow Page. In Design View above, the Content region shows the AJAX ASP.NET ScriptManager and UpdatePanel containing the image to be updated at each Timer interval. When the page runs, only the image updates every two seconds. The rest of the page does not refresh.



(e) Edit and Update Pictures Page. This page uses the AJAX ResizableControlExtender Control to dynamically resize pictures on the page.