Web Applications With Java Server Pages and Microsoft .NET Web Forms

Ryan Tillotson

Abstract

As the web continues to develop, web applications will become more prominent. Because of this, it is important to determine the similarities and differences between the most common web application frameworks. To this effect, the syntactic and architectural differences between Java Server Pages (JSP's) and ASP.NET Web Forms were analyzed. It was determined that although the two technologies bear many similarities, their current implementations offer different strengths and weaknesses. Which technology is best depends on the specific circumstances surrounding the project. These findings can help companies determine which technology would work best with existing infrastructure or assist developers in switching from one technology to the other.

CONTENTS

LIST OF FIGURES

```
<!DOCTYPE html>
<html>
  <body>
    John is 50 years old
  </body>
</html>
```

Fig. 1: *Basic HTML that will result in a webpage that says "John is 50 years old"*

I. Introduction

The growth of computing over the last fifty years has brought many changes into how companies conduct their business. Originally, either the development of entire desktop applications or the purchase of existing software was needed to perform business tasks. This put many businesses in a difficult position, because creating an entire desktop-based application system is expensive. Further, there is a significant chance that pre-created systems will not be well-suited for a company's needs. A more cost-effective method of development was needed. One of the most time-consuming parts of development was the creation of a user interface However, by modifying existing web technologies, it becomes much easier to build applications. Although static web content was not suitable for business purposes, web content that was dynamic would work well. It was this that sparked the growth of web application frameworks such as ASP.NET, Ruby on Rails, Java Server Pages, the Oracle Application Development Framework, Spring, and many more. Now, ten years after this movement began, application development frameworks have managed to significantly reduce the cost and increase the effectiveness of enterprise-level applications.

II. Survey

A. Static vs Dynamic Webpages

Hypertext Markup Language (HTML) has been used to create webpages since the creation of the internet. However, because it is static, a new page must be created and rendered for new content to be delivered. Thus, HTML alone is not suitable for use in business processes. The HTML code in Figure 1 shows basic syntax and prints out "John is 50 years old". Although this language has become a fundamental part of webpage creation, it relies heavily on other languages to provide content.

Out of the need for dynamic pages, client-side languages such as JavaScript were born. Client-side languages allow the modification of the display through client-side events and user inputs. These languages act entirely on the client's side of the browser and have no server interaction. The example in Figure 2 shows JavaScript that has been embedded in HTML. This code will again print out "John is 50 years old", but it is now printed through JavaScript that modifies the HTML of the page based on variables. This takes place after the page has been loaded. Now, if text boxes and a submit button were added to this page, the data could be bound between those text boxes and the *firstname, lastname, age, and eyecolor* attributes of the person class. JavaScript functions like this can modify how page elements are displayed, rewrite HTML, respond to user input, and overall create a dynamic web experience. While client-side languages were a major improvement over static pages, websites still needed the capability to interact with a database.

```
<%@ Language=VBScript %>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <%
      FirstVar = "Hello world!"
    %>
    The time is: <%=time%> <BR>
  </body>
</html>
```

Fig. 3: *Classic ASP Program* [1]

The need for database interaction was a driving force behind the creation of technologies like Active Server Pages, now known as Classic ASP. This platform for creating dynamic webpages was first released in 1996 by Microsoft Corporation [6]. These pages employ the languages of VBScript and JScript (Microsoft's version of JavaScript) for server-side and client-side functions, respectively. Code in VBScript and JScript is implanted into HTML markup and is then executed. In Figure 3, an example of classic ASP is shown. The VBScript on this page performs the time function on the server and displays the result on the page. Server-side functions can be custom-defined and the page can be updated to display requested data for a user. In contrast to static webpages and those that only employ client-side languages, ASP pages can also be built to store and retrieve information from databases. This is a major advantage over standard HTML and JavaScript-based pages and led Microsoft to the integration of ASP into their .NET Framework.

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      person={firstname:"John",lastname:"Doe",
                       age:50,eyecolor:"blue"}

      document.write(person.firstname + " is "
                       + person.age + " years old");
    </script>
  </body>
</html>
```

Fig. 2: *JavaScript embedded in HTML* [2]

B. *Incorporating Classic ASP into the .NET Framework*

The .NET Framework relies on the Common Language Runtime (CLR) for its increased performance and reliability. The CLR is a common-compilation language created by Microsoft. This means that multiple programming languages can be used to provide information to a program or application. In the world of .NET, compiling a program in C#, VB.NET, C++, F#, or any other .NET language results in a CLR-based program. This program is then processed using advanced compilation methods that result in increased execution speed and decreased resource usage [7]. One of the primary complaints about ASP was its commonly slow performance. Unless the programmer of the ASP page specifically optimized his or her page, it would suffer from inefficiencies. Out of this was born one of the modern web application development frameworks: ASP.NET.

This framework, which was introduced in 2004, is described by Microsoft as "a unified web development model that includes the services necessary for you to build enterprise-class web applications with a minimum of coding" [8]. ASP.NET has all of the features of Classic ASP, but few of the weaknesses for which Classic ASP became famous. This, coupled with the .NET classes now available to the ASP.NET pages, results in a development framework that is efficient and well-integrated with current Microsoft Software. This application framework has been continuously updated since its creation and the current version is .NET 4.5.1 [8].

III. Technical Analysis

A. *ASP.NET Web Forms*

1) Introduction to Web Forms: Although both of these platforms are robust and far-reaching, they each still revolve around a core set of technology. For Microsoft's .NET platform, this technology is their ASP.NET Web Forms. These pages are commonly seen on web sites with *Active Server Pages* – those that have the ".aspx" extension [8]. The benefit of these Web Forms is that they use common tags with numerous properties and settings that can be tweaked to meet the needs of the developer. These forms are then rendered to the page in HTML, CSS, and Javascript. This rendering process leaves much of the "heavy lifting" with the server and allows the developer to focus more closely on the functionality of the program.

According to Microsoft [9], the strengths of ASP.NET Web Forms are that:

- The current browser can be detected and the proper markup can be sent out to the user.
- Data can be bound to objects on the server-side resulting in faster development times.
- Multiple server-side languages are supported.
- Programs can be precompiled resulting in faster execution time.

2) Application Creation: In addition to adding significant features to Web Forms over the past eight years, Microsoft has also developed programs to make it easier for Web Forms to be learned. For example, Microsoft's Integrated Development Environment, Visual Studio, is a powerful tool that comes in several tiers: Express, Professional, and Ultimate. The Express Edition is free for all while Visual Studio Professional and Ultimate

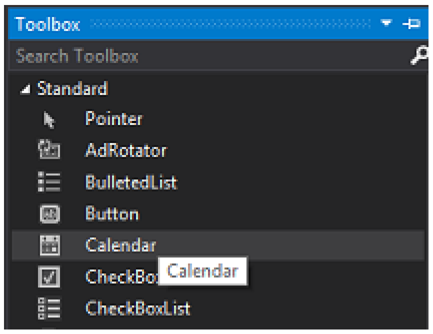


Fig. 4: *The "Standard" section of the Toolbox in Visual Studio Professional 2013*

both generally require a licensing fee [6]. However, Visual Studio Professional is available to most students via Microsoft's Dreamspark program [10]. This is a great way for new developers to get hands-on experience with Microsoft products free of charge. Each version of Visual Studio allows the creation of ASP.NET Web Forms both through source code and a visual representation. The option of creating an application page visually, in conjunction with the free copies of Visual Studio that are available, allows new developers to easily begin creating applications.

Figure 4 shows a portion of the Visual Studio toolbox. By selecting the "Calendar" and dragging it into an ASP.NET Web Forms page's "Design" section, the Calendar element is placed into the corresponding area of the ASPX source code. From here, different settings can be modified on the element and events handlers can be implemented to perform different tasks. For example, if a developer needs to initialize this calendar with conditional settings or data from another source, they can handle the "Init" method shown below in Figure 5. Further, a developer can also change properties by using right-clicking on the object in design mode and selecting "Properties" from the menu. A portion of this menu can be seen below in Figure 6.

This visual method of program creation, coupled with the tutorials available on Microsoft's website, increases the speed with which new developers can create useful applications. Although the complete version of ASP.NET Web Forms does cost money, it will also save money in the long run with lower development times and lowered barriers to entry for new developers. As Microsoft progresses with this technology, it will continue to offer strong security and increased development efficiency for companies around the world [11].

B. JSP's and Servlets

1) Introduction to JSP's and Servlets: According to Oracle Corporation, JSP's enable programmers to develop "information rich, dynamic Web pages that leverage existing business systems" Further, "JSP technology enables rapid development of Web-based applications that are platform independent" [12]. Although it is difficult to directly compare ASP.NET Web Forms with JSP's and Servlets, they still bear many similarities [12] in their implementation:

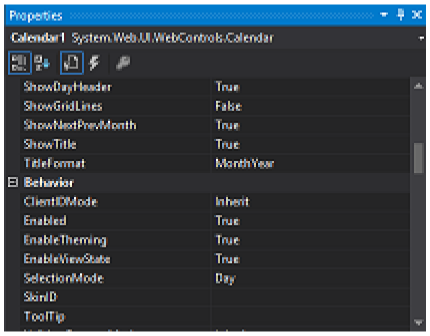
- Applications are still created using a combination of dynamic and static HTML that is combined to make the final HTML passed to a user's browser.
- Javascript (and its libraries) can still be used to modify and make sites more interactive on the client-side.
- Servlets and JSP's can perform both server-side methods and actions.

2) Application Creation: Unlike ASP.NET, programming JSP's has a steep learning curve and there are not as many visual tools like those in Visual Studio. Because of this, a developer must learn the technology primarily through the use of external resources and examples. Another reason JSP's and Servlets maintain such a high learning curve is because of the amount of manual implementation required during development. In ASP.NET, much of the behind-the-scenes programming is handled automatically. This yields an upfront bonus in development efficiency, but may result in less efficient applications. Overall, JSP's are more difficult to learn, but they leave more areas open for program customization. The net result is that JSP's and Servlets are more customizable and can be optimized for individual applications, but require more time to setup [13].

Now, As an example of simple JSP syntax, Figure 7 shows a Java expression embedded in JSP tags. The HTML that is generated by the server is shown in Figure 8. This example is basic, but it illustrates the ability of declaration operators and Java expressions within a page to print data. By offsetting Java code in special tags, the expressions will be evaluated and the results will be output by the application server in static HTML. ASP.NET Web Forms and JSP with Servlets both output static HTML for a browser to render, but to properly compare these two technologies, a deeper understanding of how each of them processes and displays this information is required.

```
protected void cal_OnInit(object sender, EventArgs e){
    //Make the calendar invisible if there are 31 days in this month
    if (DateTime.DaysInMonth(DateTime.Now.Year, DateTime.Now.Month) == 31)
        Calendar1.Visible = false;
}
```

Fig. 5: *The "init" method can be handled in the ASPX page's associated code file. This method can then be specified in the properties of the calendar and will be called as it is initialized*



Calendar1 System.Web.UI.WebControls.Calendar

ShowDayHeader	True
ShowGridLines	False
ShowNextPrevMonth	True
ShowTitle	True
TitleFormat	MonthYear
⊟ Behavior	
ClientIDMode	Inherit
Enabled	True
EnableTheming	True
EnableViewState	True
SelectionMode	Day
SkinID	
ToolTip	

Fig. 6: *A selection of properties available for the Calendar element in Visual Studio Profession 2013. These can be edited and modified by the user. The ASP.NET compiler will then render HTML, CSS, and Javascript to the browser based on these settings.*

C. A Syntactical Comparison of ASP.NET, JSP, and Servlets

To demonstrate the difference between ASP.NET Web Forms and A similar application is shown with ASP.NET Web Forms in Figure 9. This program prints out similar statistics about the current ASP.NET session to the HTML. This is printed prior to the HTML output because it is added to the page during the Page_Load Event. In comparison with the example in Figure 7, the syntax is entirely different. Although both pages have similar results, they are achieved through different means. The ASP.NET code-behind page offers a distinct split between server-side and client-side controls on the Web Forms while the JSP example embeds the server-side code within the static HTML. This is the primary syntactic difference between JSP and ASP.NET Web Forms.

The closer syntactic comparison to an ASP.NET code-behind page is the Servlet. A Servlet comes into existence either through direct creation in code or when a JSP page requires processing. Essentially, upon execution, a server translates a JSP page into a HTTPServlet which then prints out the required HTML using Java. A good rule of thumb is that a Servlet is "HTML inside Java" and a JSP is "Java inside HTML" [14]. Although JSP pages are powerful, Servlets serve other functions that are necessary to the proper execution of an application. In many ways using JSP with Servlets is similar to ASP.NET Web Forms, but when it comes to the execution and browser response processing, they differ heavily.

IV. Application Server Design

A. Internet Information Server

1) Introduction to IIS: Internet Information Server (IIS) is a Windows service that is available in all modern Windows installations as well as in Microsoft Windows Server. This robust application is capable of hosting web applications either locally or on the internet. It is the only "official" .NET application server for use with ASP.NET Web Forms and other portions of the .NET Framework [15]. As a result, Microsoft fully supports IIS and offers excellent integration between IIS and Visual Studio.

2) IIS Processing Architecture: In terms of architecture, the HTTP Request path followed by IIS maintains some similarities with the JSP HTTP Request path, but is ultimately different. The biggest difference is that the request makes two passes through the activation service – one to retrieve the configuration and another to place the request in the IIS processing pipeline. In the JSP application server, a request is

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html;">
                <title>Test of JSP!</title>
        </head>
        <body>
                It's a JSP. <%= new java.util.Date() %>
                <br/>Java Version:   <%= System.getProperty("java.version") %>
                <br/>Operating System:   <%= System.getProperty("os.name") %>
                <br/>User Name:   <%= System.getProperty("user.name") %>
        </body>
</html>
```

Fig. 7: *This JSP page will generate the HTML that prints out the current date and time on the system, the version of java running, the name of the operating system and the name of the current user.*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html;">
                <title>Test of JSP!</title>
        </head>
        <body>
                It's a JSP. Thu Apr 03 21:05:41 CDT 2014
                <br/>Java Version:   1.7.0_51
                <br/>Operating System:   Windows 8
                <br/>User Name:   Ryan
        </body>
</html>
```

Fig. 8: *This is the HTML generated by the request sent to the server by the JSP page. It displays the current date and time on the system, java version, name of the operating system, and the name of the current user.*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){
        Response.Write("Time: " + DateTime.Now + "<br/>");
        Response.Write("User: " + Page.User.Identity.Name.ToString() + "<br/>");
        Response.Write("OS: " + Request.UserAgent.ToString() + "<br/>");
    }
}
```

Fig. 9: *This function is called when the ASPX page (Default.aspx) that this page is attached to is opened. At this time, the application server writes the specified information to the page.*

```
Time: 4/3/2014 11:09:41 PM<br/>
User: <br/>
OS: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0<br/>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Welcome to Speak Salty!</title>
</head>
<body>
    <form method="post" action="Default.aspx" id="form1">
                <div class="aspNetHidden">
                        <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
                            value="WNOWQ0RZ5XzIsaaudVESgbgihFIhOnh+
                            7bSiKtIT3wFGYElIMqP+K0sCUta3E4y
                            xXAkaPqY41QMW6I5Wn+/tqeG1vwNe3Jg1ydPIGpI+GhQ=" />
                </div>
    </form>
</body>
</html>
```

Fig. 10: *This HTML is generated when the .NET page is instantiated and the Page_Load() function is called.*

made and the configuration information is passed along with the request. This allows the JSP server to progress to compilation without iterating back through any activation services.

The IIS Request Processing breaks down into the following steps [3]:

1) Client requests are intercepted.
2) Initial contact is made with Windows Activation Service(WAS) which manages application lifetimes.
3) WAS requests configuration information from the respective XML file.
4) Configuration information is sent to the WWW service which then publishes it.
5) Configured HTTP contacts WAS again.
6) WAS starts a worker process for this application.
7) Worker process goes through IIS 7.0 Processing Pipeline where various code modules are used to process the request.
8) Response is returned to the client via the HTTP protocol stack.

The steps path a standard HTTP request takes when it enters IIS are shown in Figure 12.

B. JSP Application Servers

1) Introduction to JSP Application Servers: Unlike Microsoft's .NET there are many different application servers capable of handling JSP's and Servlets. Several of the most popular are Jetty, Apache Tomcat, Glassfish, Oracle Weblogic, and IBM Websphere. In addition, *most* of these application servers can run on much less computing power than IIS. This is a major benefit to companies that have older hardware. In addition, since most of the software necessary to create JSP applications with Servlets is free, companies can pool their capital to hire superior developers.

2) JSP Processing Architecture: The following steps occur when a JSP is accessed by a browser [4]:

1) Browser sends HTTP request to a web server.
2) Web server acknowledges that the request is for a JSP page and sends it to the proper application server.
3) JSP page is loaded from the server disk and it is converted into a Servlet.
4) Application server compiles the Servlet into a class and forwards this to the engine that executes these classes.
5) Servlet class is executed and the result is an HTML file. This is passed to the web server in an HTTP response.
6) Web server forwards this response to the browser in a static HTML page.
7) Browser handles this page like a standard HTML page.

```
// doGet is called when a browser performs an HTTP "GET".
protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");

        //Print out the information to the output writer.
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("Time: " + new java.util.Date() + "<br/>");
        out.println("Version: " + System.getProperty("java.version")+ "<br/>");
        out.println("OS: " + System.getProperty("os.name")+ "<br/>");
        out.println("User: " + System.getProperty("user.name")+ "<br/>");
        out.println("</html>");
}
```

Fig. 11: *This servlet prints the same information as the JSP in Figure 7. However, it does so by directly writing the HTML directly. The original JSP was translated to a servlet and then printed by the web server into static HTML.*

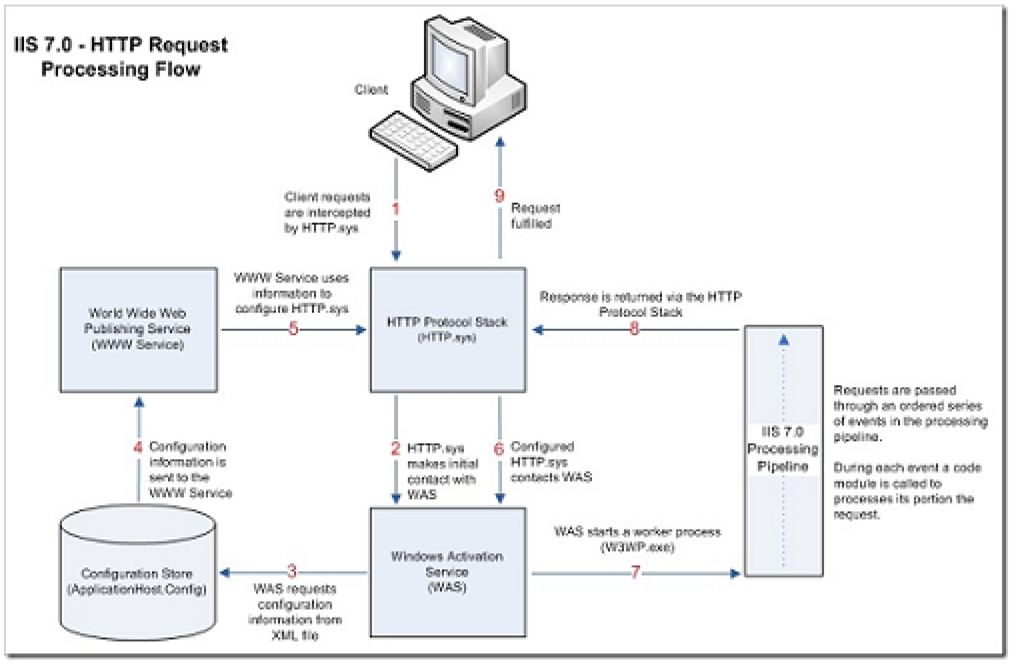

IIS 7.0 - HTTP Request Processing Flow
Client
Client requests are intercepted by HTTP.sys
1
9
Request fulfilled
World Wide Web Publishing Service (WWW Service)
WWW Service uses information to configure HTTP.sys
5
HTTP Protocol Stack (HTTP.sys)
Response is returned via the HTTP Protocol Stack
8
IIS 7.0 Processing Pipeline
Requests are passed through an ordered series of events in the processing pipeline.
During each event a code module is called to processes its portion the request.
4 Configuration information is sent to the WWW Service
2 HTTP.sys makes initial contact with WAS
6 Configured HTTP.sys contacts WAS
Configuration Store (ApplicationHost.Config)
3
WAS requests configuration information from XML file
Windows Activation Service (WAS)
WAS starts a worker process (W3WP.exe)
7

Fig. 12: *A diagram depicting the path that a .NET application follows when an HTTP request is initiated [3].*

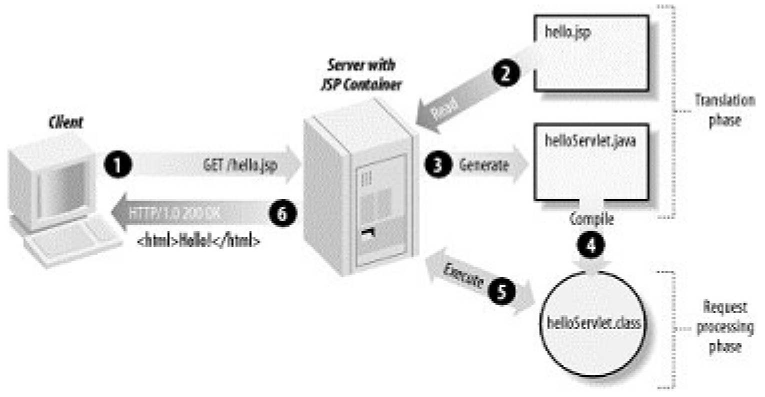


Fig. 13: *A diagram depicting a standard JSP response path [4].*

These steps allow for the handling of various events and the addition of new Servlets into a JSP page's output. Further, it should be noted that these steps are invisible to a user. A visualization of this process can be seen in Figure 13.

V. Speak Salty Blog – The Prototype

Please see Appendix B for the full code listing.

A. Introduction to Speak Salty

In order to demonstrate the effectiveness of modern web application technologies, I created a blog that is constructed from the database level upward using Microsoft SQL Server and ASP.NET Web Forms. The original intention of this blog was to outline the travel and cooking habits of the writer, but it could be customized to discuss anything. On it, each post is made up of a body, title, and date along with any number of corresponding images that are uploaded by the user. When a page is loaded, the posts are compiled in reverse chronological order for the requester with the images on top. This information will be pulled from the database, compiled into HTML by the server, and then returned to the user's browser to be rendered.

B. Database

One of the most important parts of this project was designing the database structure to compliment the code that would be calling it. To this effect, I created a database named "SS_BLOG" that holds the two tables that comprise the data structure for this project.

1) Database Tables:

1) *SS_POSTS* - This table holds all of the non-image information contained in a post. Because a post can only contain at most one post title, post date, and post body, this table is suitable for holding these records. There will be a number of posts equal to the number of rows in this table. This table has the columns:
   - POST_ID - Primary Key, INT, NOT NULL.
   - POST_TITLE - VARCHAR(500), NULL. Title of the post.
   - POST_DATE - DATETIME, NULL. Date of the post.
   - POST_BODY - VARCHAR(MAX), NULL. Body of the post.
2) *SS_IMAGES* - This table holds all of the image-based information contained ina post. It does this by specifying the post id of a given post. This allows a 1 to many relationship which allows for the easy incorporation of multiple pictures into a single post.
   - IMAGE_ID - Primary Key, Int, NOT NULL.
   - IMAGE_POSTID - Int, NOT NULL. Post id this image belongs to.
   - IMAGE_PATH - VARCHAR(500), NOT NULL. Path / file name of the image.

2) Database Stored Procedures: Many of the queries that retrieve data for this project are written in the code, but two more complicated procedures for inserting blog entries and images are included as stored procedures inside the database. These can be seen in Figures 14 and 15.

```
@postTitle AS VARCHAR(500),
@postBody AS VARCHAR(MAX)

AS
BEGIN

INSERT INTO dbo.ss_posts
(post_title, post_date, post_body)
VALUES
(@postTitle, GETDATE(), @postBody)

SELECT SCOPE_IDENTITY() AS 'postID'

END
```

Fig. 14: *This stored procedure – dbo.insert_blog_entry inserts the title, body, and current date into a new SS_POSTS entry. It is called by the new entry page in the application.*

```
@postID AS INTEGER,
@imagePath AS VARCHAR(500)

AS
BEGIN

INSERT INTO dbo.ss_images
(image_postid, image_path)
VALUES
(@postID, @imagePath)

END
```

Fig. 15: *This stored procedure – dbo.insert_blog_entry_image inserts the image path and post id for an image into the SS_IMAGES table. There can be many image entries for a single post.*

C. *Creating a New Entry*

A user is able to create a new post by using the new entry screen as seen in Figure 16. This page allows the user to input the title, body, and any pictures they would like on the blog post. To do this, the filenames of the images and the files are uploaded to the server. A new BlogEntry object is created using the title, body, and list of images, and then the InsertBlogEntry() function shown in Figure 17. This function calls the stored procedure shown in Figure 14 which returns the POST_ID of that image. This primary key value is then used when the stored procedure in Figure 15 is called for each image in the blog entry. After this, the user is redirected to the default page of the blog where the newly added post will appear at the top. Creating the BlogEntry class will reduce the amount of code necessary in the "code behind" vb file increasing its readability.

D. *Generating the Default.aspx Page*

When a user requests the default blog page, the Page Load event will be called. As shown in Figure 19, this event will retrieve all entries from the SS_POSTS table in the database, put these into BlogEntry objects, compile the HTML, and send it back to the user. Printing the HTML dynamically allows the page to display as many images as the writer selected without sacrificing the time needed for a developer to physically create the image links in HTML. An example blog entry shown in Figure 18. A future extension of this would be to reduce the number of BlogEntries per page. This would enable the page to load faster and ultimately increase usability.

Overall, this prototype took about 7 hours of programming. If a developer has even more experience with ASP.NET, it would have taken less. Because of the heavy lifting performed by .NET Framework functions, a smaller amount of code needs to be written when developing these applications. ASP.NET provides a way for developers to create useful applications with minimal programming.

VI. Future Trends

A. *The Future of ASP.NET*

ASP.NET was originally developed by Microsoft as a replacement for the original Active Server Pages technology. Since then, the development world has seen Microsoft incorporate many programming and scripting languages as well as create an extensive class library that provides access to databases, security algorithms, networking functions, and application development tools. It is through this that .NET has gained its popularity and it is through extensions of these frameworks that Microsoft's technology will evolve. One of the most recent additions to the repertoire of Microsoft is that of Windows Azure – an "open and flexible cloud platform that enables you to quickly build, deploy, and manage applications across a global network of Microsoft-managed datacenters" [16]. This is an extension of the current .NET Framework and over the next few years it will become more common. Overall, it will allow companies to more rapidly develop and deploy web applications throughout the world and provide load-balancing services as well as a pay-as-you-go expense model.

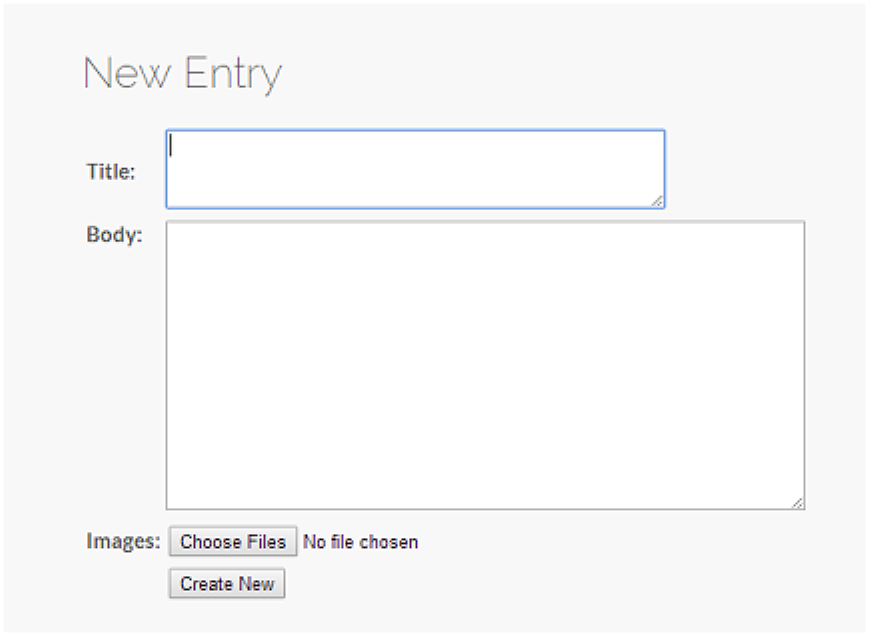
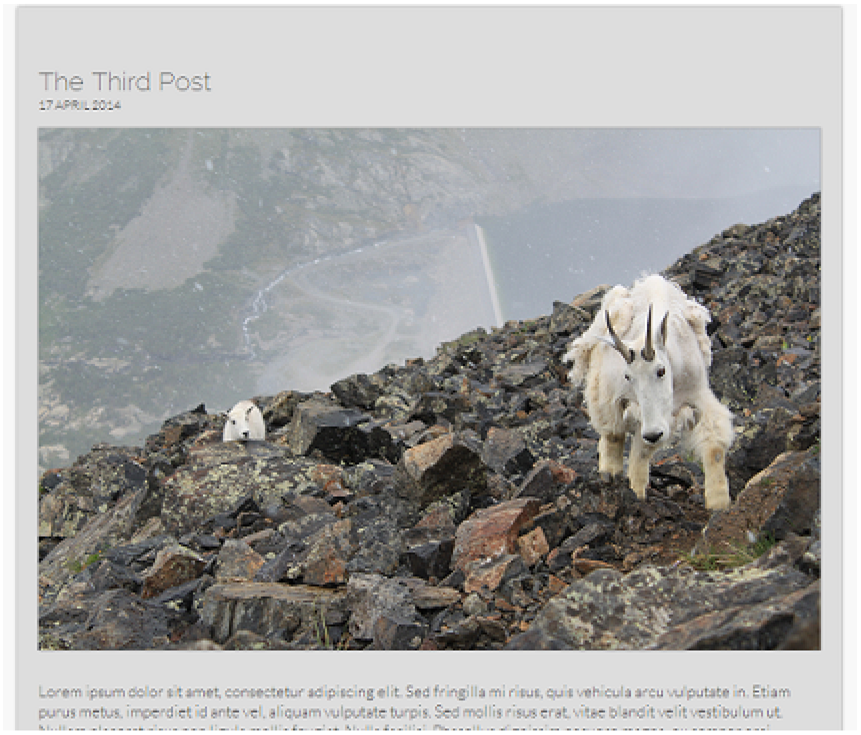


Fig. 16: *This figure shows the entry screen as seen by the blog administrator. This allows the user to enter a title, body, and select images to be uploaded to the server.*

```
Public Sub InsertBlogEntry()
    Dim cmd As New SqlCommand("EXEC dbo.insert_blog_entry @postTitle, _
                                                @postBody", ssBlog)
    Dim postID As Integer = 0
    cmd.Parameters.AddWithValue("@postTitle", postTitle)
    cmd.Parameters.AddWithValue("@postBody", postBody)

    ssBlog.Open()
    postID = cmd.ExecuteScalar()
    ssBlog.Close()

    cmd.Parameters.Clear()
    cmd.CommandText = "EXEC dbo.insert_blog_entry_image @postID, @imagePath"
    cmd.Parameters.AddWithValue("@postID", postID)

    For Each path As String In imageList
        cmd.Parameters.AddWithValue("@imagePath", "site_images\" + path)

        ssBlog.Open()
        cmd.ExecuteScalar()
        ssBlog.Close()

        cmd.Parameters.RemoveAt("@imagePath")
    Next
End Sub
```

Fig. 17: *This function will insert a new blog entry and all images in the imagelist into the database. It does this by storing the new POST_ID returned from the first stored procedure and using it to relate the images to the blog entry.*



The Third Post

17 APRIL 2014

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed fringilla mi risus, quis vehicula arcu vulputate in. Etiam purus metus, imperdiet id ante vel, aliquam vulputate turpis. Sed mollis risus erat, vitae blandit velit vestibulum ut.

Fig. 18: *This is an example blog entry. The text, date, and placeholder text are shown along with the only image associated with this post.*

In addition to advances in the deployment and architecture of ASP.NET applications, Microsoft will continue to contribute heavily to development programs in colleges. Their Dreamspark campaign gives all students at eligible schools access to Microsoft's development tools – SQL Server, Visual Studio, and more. Because this has been successful in driving the interest of students to learn Microsoft technologies, they will continue to extend the offerings of this service. As can be seen in Figure 20, .NET technologies have seen a large spike in job postings. By increasing the number of developers that enjoy the .NET Framework, Microsoft will be able to affect the status quo of development technologies in their favor.

In recent years, Microsoft has begun open-sourcing many portions of the .NET Framework. Although this decision is very late in the evolution of .NET, it will allow more developers to contribute to the projects. Ultimately, since Microsoft has started this initiative along with those that offer rapid development processes, they will maintain relevancy and potentially begin to dominate the market. Historically, Microsoft has lagged behind when developing products for new portions of the technology market (ie Bing, Surface, etc.). When their technologies reach maturity, however, they've consistently produced strong products. As .NET grows, Microsoft will likely continue to increase community involvement and incorporate these features into the standard software.

B. The Future of JSP's

Although JSP's and Servlets have played a critical role in the evolution of web applications, it is mostly through their community-driven extensions that this growth has taken place. Thus, it is through improvements of these extensions that JSP's and Servlets will continue to improve. However, after the recent Heartbleed vulnerability in the OpenSSL technology, many open-source projects are facing heavy criticism. Because of this, some of the open-source projects such as Spring and Struts could face global scrutiny. So, in the coming years, it is likely that closed-source projects such as Oracle's Application Development Framework (ADF) will gain popularity. The misconceptions surrounding open-source technology could damage the foothold JSP's and Servlets have in the market.

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient
Imports System.Web.Configuration
Imports System.Data
Imports System.Collections.Generic

Partial Class _Default
    Inherits System.Web.UI.Page

    Dim ssBlog As New SqlConnection(WebConfigurationManager. _
                ConnectionStrings("ssBlog").ToString)

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load

        Dim cmdGetEntries As New SqlCommand("SELECT * FROM dbo.ss_posts " _ +
                "ORDER BY post_date DESC", ssBlog)
        Dim dt As New DataTable
        Dim adapt As New SqlDataAdapter
        adapt.SelectCommand = cmdGetEntries

        ssBlog.Open()
        adapt.Fill(dt)
        ssBlog.Close()

        Dim blogEntry As BlogEntry
        Dim entryList As New List(Of BlogEntry)

        'Insert a new entry for each data row retrieved
        For Each dr As DataRow In dt.Rows
            blogEntry = New BlogEntry(dr("post_id"), dr("post_title"), _
                        dr("post_date"), dr("post_body"))
            entryList.Add(blogEntry)
        Next

        For Each be As BlogEntry In entryList
            contentContainer.InnerHtml += be.GetEntryHTML()
        Next
    End Sub

End Class
```

Fig. 19: *This function will pull all entries from the SS_POSTS table, put them into BlogEntry objects, and subsequently display them in HTML for the user.*

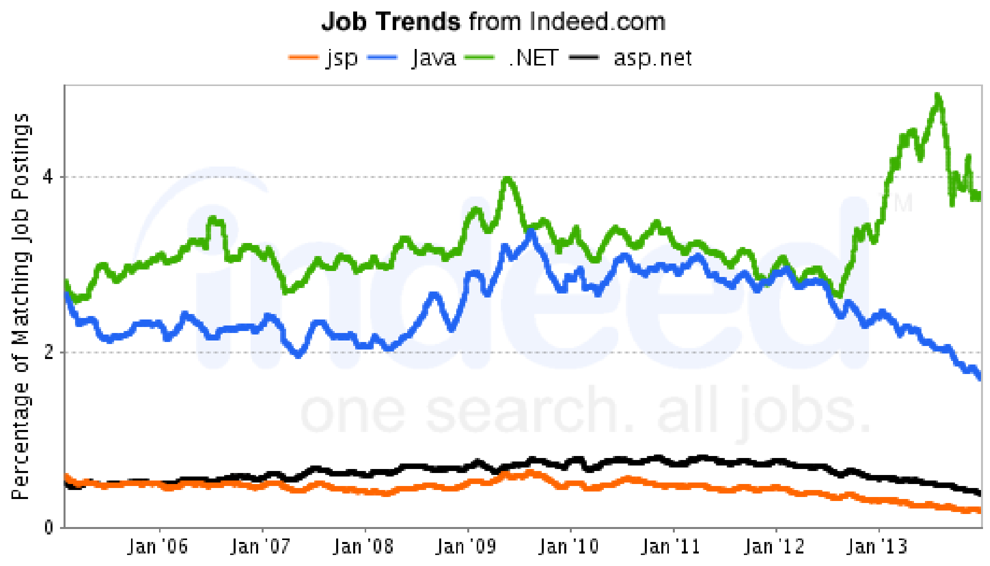


Fig. 20: *Presence of key technology terms on Indeed - a job search engine. [5]*

Also, unlike ASP.NET, there are fewer official resources available for learning these frameworks. Often times training new employees to develop with these technologies is time consuming. This increased training time for new employees is one of the factors that has led to the overall decline in both Java and JSP job postings as shown in Figure 20. However, many new methods for learning these technologies are surfacing, but improvement is still needed. In the future, it is likely that as different JSP frameworks become popular, the documentation will improve and the learning curve will decrease. These frameworks have the potential to keep pace with .NET's growth, but more resources are needed for developers.

VII. Conclusion

As web and cloud-based technology becomes more prominent, the demand for web applications will increase. Because of this, companies will look for technology stacks that are easily expandable, cost-effective, and quickly developed. Through this prototype, it has been demonstrated that ASP.NET is easy to develop with and also very expandable. However, this convenience costs both money and the ability to customize the programs. Microsoft is attempting to combat this inflexibility by pushing their projects to open-source, but for the time being the page life cycle is not entirely open to developers. JSP's, on the other hand, allow developers access to the whole life cycle, are much less expensive to implement, and have been open-source for over a decade. The result of this is community-built frameworks based on JSP's that can assist in the creation of applications. Now, when choosing which technology to use, it depends on what technologies it will need to exist with, the skill set of the developers, and the amount of money available to spend. Both of these offerings are equally effective in creating large-scale web applications.

References

[1] Microsoft Corporation. ASP Tutorial Write Run ASP Pages. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms972337.aspx#asptutorial_writerun
[2] W3Schools. Try it yourself javascript. [Online]. Available: http://www.w3schools.com/js/tryit.asp?filename=tryjs_create_object1
[3] Tom Woolum. IIS 7.0 HTTP Request Processing. [Online]. Available: http://blogs.iis.net/tomwoolums/archive/2008/12/16/iis-7-0-http-request-processing.aspx
[4] Tutorials Point. JSP - Architecture. [Online]. Available: http://www.tutorialspoint.com/jsp/jsp_architecture.htm
[5] Job trends from indeed.com. [Online]. Available: http://www.indeed.com/trendgraph/jobgraph.png?q=jsp%2C+Java%2C+.NET%2C+asp.net
[6] Microsoft Corporation, "Active Server Pages." [Online]. Available: http://msdn.microsoft.com/en-us/library/aa286483.aspx
[7] ——. Common Language Runtime. [Online]. Available: http://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspxn
[8] ——. ASP.NET Overview. [Online]. Available: http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx
[9] Paul Sheriff. Introduction to ASP.NET and Web Forms. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms973868.aspx#introwebforms_topic1
[10] Microsoft Corporation. Microsoft Dreamspark. [Online]. Available: https://www.dreamspark.com/
[11] ——. Introduction to ASP.NET Web Forms. [Online]. Available: http://www.asp.net/web-forms/what-is-web-forms
[12] Oracle Corporation. Java Server Pages Overview. [Online]. Available: http://www.oracle.com/technetwork/java/overview-138580.html
[13] Tutorials Point. JSP - Overview. [Online]. Available: http://www.tutorialspoint.com/jsp/jsp_overview.htm
[14] Chua Hock-Chuan. Getting started with JSP by Examples. [Online]. Available: http://www3.ntu.edu.sg/home/ehchua/programming/java/JSPByExample.html
[15] Microsoft Corporation. ASP.NET Integration with IIS 7. [Online]. Available: http://www.iis.net/learn/application-frameworks/building-and-running-aspnet-applications/aspnet-integration-with-iis
[16] Microsoft azure overview. [Online]. Available: http://www.azure.microsoft.com/en-us/overview/what-is-azure/

Appendix

A. *Reflection of Learning*

Throughout my many years of college, I've learned one thing: with enough time and effort, I can conquer anything. When I started out at Saint John's, quite frankly, I struggled. My accounting classes were boring and I didn't feel challenged. As a result, I nearly failed out of school. Since then, I've learned the importance of hard work and forced myself to keep my head up as I finish classes that I do not care for. At the end of the day it was my job at Nahan Printing, Inc. that allowed me to apply the skills I learned in my education. While some of the classes I attended were more helpful than others, in the end I took something away from each of them.

1) The Impact of Previous Experiences: This project was not directly impacted by my previous coursework; it was through these courses that I attained a strong base skill in programming. It is because of this skill that I have been able to pick up this material so quickly. Two of the specific classes that helped me are Computer Science II and Systems Development at Saint Cloud State University. These were two of the most challenging programming classes I've ever taken. It was from here that my knowledge base started and it was at my job as an Application Developer at Nahan Printing that these skills improved. These classes taught me about modular program design and object-oriented design concepts. For example, in my project I implemented an object (BlogEntry.vb) that allowed my prototype to function smoothly with minimal code in the files. However, this project also showed me that I could have extended this design further and made my code even more readable.

At Nahan Printing, we do a lot of work with ASP.NET Web Forms. This gave me a very strong base for this project. However, I learned a lot about these topics that I didn't know before. For example, printing out HTML text directly into a "div" is something I had never done before. This allowed me to generate a page entirely on the server-side. Another thing I learned is how to set up a database server. I had used databases, created new tables, and made numerous lengthy queries prior to this, but I had never installed/created my own database. This is good experience for me because it provides me with infrastructure and architecture experience. Connecting to this database was challenging, but the experience it provided me with was worth it.

2) Impact on My Future: In addition to Computer Science 373, I am also taking Computer Science 230. I implemented a JSP application that allows a user to view, compare, and save universities to find the one that is right for them. Unfortunately, I was met with significant chaos on my school computing account when attempting to create this project, so I ended up hosting my own database and tomcat instance on my virtual private server. During this process, I was forced to troubleshoot many of the same problems I experienced connecting to a database with ASP.NET. I was also able to gain an appreciation for everything that ASP.NET implements for the developer by default. However, I did like the flexibility I had with the JSP's. I think that being able to directly compare and contrast these two types of programs during development gave me significant insight into their differences. This will help me in my next job where I will be doing Oracle Software Consulting for a company in Des Moines, Iowa. These programs have definitely shaped the way I view my applications and made me want to become a more effective application developer. This project directly influenced this opinion by offering me the chance to examine both technologies and implement an ASP.NET program.

An additional note: learning LaTeX was one of the most helpful things about this class. It will fundamentally change the way in which I write official documentation in the future. I was able to apply my skills learned to both work and other courses. It is a very effective tool and I am glad to have learned about it.

B. Full Code Listing

1) NewEntry.aspx:

```
<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="NewEntry.aspx.vb" Inherits="NewEntry" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">

<div id="formTitle">
New Entry
</div>
<div id="formContainer" runat="server">
<br />
<table>
   <tr>
       <td>
           <asp:Label ID="lblTitle" runat="server" Text="Title: "
               Font-Bold="true" CssClass="postBody">
           </asp:Label>
       </td>
       <td>
           <asp:TextBox ID="txtTitle" runat="server" Width="350px" Height="50px"
               MaxLength="500" TextMode="Multiline">
           </asp:TextBox>
       </td>
   </tr>
   <tr>
       <td style="vertical-align: top;">
           <asp:Label ID="lblBody" runat="server" Text="Body: " Font-Bold="true"
            CssClass="postBody">
           </asp:Label>
       </td>
       <td>
           <asp:TextBox ID="txtBody" runat="server" Width="450px" Height="200px"
           MaxLength="5000" TextMode="MultiLine"></asp:TextBox>
       </td>
   </tr>
   <tr>
       <td>
           <asp:Label ID="lblPictures" runat="server" Text="Images: "
           Font-Bold="true" CssClass="postBody">
           </asp:Label>
       </td>
       <td>
           <asp:FileUpload ID="upControl" runat="server" AllowMultiple="True" />
       </td>
   </tr>
```

```
  <tr>
      <td></td>
      <td>
            <asp:Button ID="addNew" runat="server" Text="Create New"
            UseSubmitBehavior="false" />
      </td>
  </tr>
</table>



</div>
</asp:Content>
```

2) *NewEntry.aspx.vb:*

```
Partial Class NewEntry
    Inherits System.Web.UI.Page

    Protected Sub addNew_Click(sender As Object, e As EventArgs) Handles addNew.Click

        Dim imageList As New List(Of String)

        'Save Files into Data Folder & make list of filenames
        If upControl.HasFiles() Then
            For Each file As HttpPostedFile In upControl.PostedFiles
                imageList.Add(file.FileName)
                file.SaveAs("D:\My Documents\Visual Studio 2013\WebSites\ "_ +
                "Salty\site_images\" + file.FileName)
            Next
        End If
        Dim blogEntry As New BlogEntry(txtTitle.Text, txtBody.Text, imageList)
        blogEntry.InsertBlogEntry()

        Response.Redirect("~/Default.aspx")
    End Sub
End Class
```

3) *Default.aspx:*

```
<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">




    <div id="contentContainer" runat="server">

    </div>

    <asp:SqlDataSource ID="SqlDataSource1" runat="server"></asp:SqlDataSource>
</asp:Content>
```

4) Default.aspx.vb:

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient
Imports System.Web.Configuration
Imports System.Data
Imports System.Collections.Generic

Partial Class _Default
    Inherits System.Web.UI.Page

    Dim ssBlog As New SqlConnection(WebConfigurationManager. _
                                        ConnectionStrings("ssBlog").ToString)

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load

        Dim cmdGetEntries As New SqlCommand("SELECT * FROM dbo.ss_posts " _ +
                "ORDER BY post_date DESC", ssBlog)
        Dim dt As New DataTable
        Dim adapt As New SqlDataAdapter
        adapt.SelectCommand = cmdGetEntries

        ssBlog.Open()
        adapt.Fill(dt)
        ssBlog.Close()

        Dim blogEntry As BlogEntry
        Dim entryList As New List(Of BlogEntry)

        'Insert a new entry for each data row retrieved
        For Each dr As DataRow In dt.Rows
            blogEntry = New BlogEntry(dr("post_id"), dr("post_title"), _
                            dr("post_date"), dr("post_body"))
            entryList.Add(blogEntry)
        Next

        For Each be As BlogEntry In entryList
            contentContainer.InnerHtml += be.GetEntryHTML()
        Next


    End Sub

End Class
```

5) BlogEntry.vb:

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient
Imports System.Web.Configuration
Imports System.Data
Imports System.Collections.Generic

Public Class BlogEntry
```

```
#Region "Blog Variables"
    Private ssBlog As New SqlConnection(WebConfigurationManager. _
                                    ConnectionStrings("ssBlog").ToString)
    Public postID As Integer
    Public postTitle As String
    Public postDate As Date
    Public postBody As String
    Public imageList As List(Of String)
#End Region


    Public Sub New()
        Me.postID = -1
        Me.postTitle = Nothing
        Me.postDate = Nothing
        Me.postBody = Nothing
    End Sub

    'constructor that creates the image list for the entry
    Public Sub New(ByVal postID As Integer, ByVal postTitle As String, _
                   ByVal postDate As Date, ByVal postBody As String)
        Me.postID = postID
        Me.postTitle = postTitle
        Me.postDate = postDate
        Me.postBody = postBody
        imageList = New List(Of String)
        GetEntryImages()
        ChangeNulltoEmpty()
    End Sub

    Public Sub New(ByVal postTitle As String, ByVal postBody As String, _
                   ByVal imageList As List(Of String))
        Me.postID = -1
        Me.postTitle = postTitle
        Me.postBody = postBody
        Me.imageList = imageList
    End Sub

    'Change all db null values to empty
    Private Sub ChangeNulltoEmpty()
        postTitle = CheckNull(postTitle)
        postDate = CheckNull(postDate)
        postBody = CheckNull(postBody)
    End Sub

    'Check for null database values
    Private Function CheckNull(ByVal e As String) As String
        If (IsDBNull(e)) Then
            Return Nothing
        Else
            Return e
        End If
    End Function

    Public Sub GetEntryImages()
        Dim cmdGetImages As New SqlCommand("SELECT image_path  " _
                                        + "FROM dbo.ss_images " _
                                        + "WHERE image_postid = " _
                                        + postID.ToString, ssBlog)
        Dim dt As New DataTable
        Dim adapt As New SqlDataAdapter(cmdGetImages)

        ssBlog.Open()
        adapt.Fill(dt)
        ssBlog.Close()


        If Not IsNothing(dt) Then
            For Each dr As DataRow In dt.Rows
                imageList.Add(CheckNull(dr("image_path")))
            Next
        End If
    End Sub

    Public Function GetEntryHTML() As String
        Dim html As String = ""

        html = "<div class='postContainer'>" _
            + "<div class='postTitle'>" _
            + "<h1>" + postTitle + "</h1>" _
            + "<h2>" + Day(postDate).ToString + "   " _
            + MonthName((Month(postDate)).ToString).ToUpper _
            + "   " + Year(postDate).ToString + "</h2>" _
            + "</div>"

        'add images to html
        For Each s As String In imageList
            html += "<img src='" + s + "' />"
        Next

        'Replace all carriage returns with a line break
        html += "<div class='postBody'>" _
                    + postBody.Replace(vbCr, "<br/>") _
                    + "</div>"
        html += "</div>"
        Return html
    End Function

    Public Sub InsertBlogEntry()
        Dim cmd As New SqlCommand("EXEC dbo.insert_blog_entry " _
                                            +"@postTitle, @postBody", ssBlog)
        Dim postID As Integer = 0
        cmd.Parameters.AddWithValue("@postTitle", postTitle)
        cmd.Parameters.AddWithValue("@postBody", postBody)

        ssBlog.Open()
        postID = cmd.ExecuteScalar()
        ssBlog.Close()

        cmd.Parameters.Clear()
        cmd.CommandText = "EXEC dbo.insert_blog_entry_image @postID, @imagePath"
        cmd.Parameters.AddWithValue("@postID", postID)

        For Each path As String In imageList
            cmd.Parameters.AddWithValue("@imagePath", "site_images\" + path)

            ssBlog.Open()
            cmd.ExecuteScalar()
            ssBlog.Close()

            cmd.Parameters.RemoveAt("@imagePath")
        Next
    End Sub
End Class
```