https://doi.org/10.48009/1_iis_2014_209-216

Issues in Information Systems
Volume 15, Issue I, pp. 209-216, 2014

USING STANDARDS AND BEST PRACTICES TO HELP STUDENTS WITH LIMITED TECHNICAL SKILLS CREATE FULL-FEATURED WEB SITES

Thom Luce, Ohio University, luce@ohio.edu

ABSTRACT

This paper outlines a pedagogical process for teaching relatively non-technical MIS students how to create full-function online sales oriented web sites using HTML and CSS standards based pages with .NET features and one shareware component. The process involves selecting a suitable web site template from the thousands of HTML/CSS standard templates available and then modifying it to create a .Net compatible master page. Once the master page is created individual content pages are designed to implement a simple content management system using FreeTextBox from freetextbox.com. Product images are uploaded to the web site and relevant information is stored in a database. Customers are allowed to register with the system and once registered and logged in can select items from product pages and add them to a shopping cart. Finally, registered users can edit their shopping carts and order the selected items.

Keywords: Systems development, web development, pedagogy, standards-based development

BACKGROUND

The MIS program at Ohio University is in an AACSB accredited College of Business. Because we have a large common core of courses required of all business majors the MIS major itself is limited to six courses designed in a fixed sequence: MIS2200 & MIS2800 (Systems Analysis and Design and Business Intelligence and Information Systems) followed by MIS3200 and MIS3800 (Systems Development and Enterprise Systems Implementation) and finally MIS4200 and MIS4800 (Information Systems Consulting Project and MIS Capstone). The final two courses are both considered capstone courses, with MIS4200 essentially reviewing the "build" side of the systems development "buy/build" decision and the other MIS4800 focusing more on the "buy" side along with the strategic importance of MIS in current business practice.

Like many capstone development courses, a major portion of MIS4200 is a live-client based development project with the goal of producing dynamic web applications that can be easily maintained by our clients. Typical web-based solutions involve some or all of the following: content management (both text and images), end-user registration functions with access to some pages or portions of a page based on how a user is logged in, calendaring/event functions, file upload/download, and possibly shopping carts and payment functions. All of this should be done following best practices [1, 2, 4, 5, 6] and design principles learned in an earlier MIS course taken by all students in the college.

MIS4200 is preceded by one programming class, MIS3200 (Systems Development), that introduces C# programming using Microsoft Visual Web Developer Express (VWD) as the interactive development environment. Before taking MIS3200 students are introduced to database concepts in a prerequisite class (MIS2800) that focuses on data and knowledge management including databases using SQL Server Express. Since most of our students are never going into hard-core programming shops the author's department chose to make MIS3200 a web-based application development class. MIS3200 is the first and only programming course in our major and most of our students take the class with no programming background. Because of this the course moves slowly and it is a stretch to get them to the point where we can introduce database programming using .NET controls and code.

THE PROBLEM

Because of their background mentioned above students entering MIS4200 generally have a weak programming background and few options when it comes to the development side of client problem solutions. The author tried using the default project template available in Visual Web Developer, which includes a simple menu structure and uses cascading style sheets (CSS) for formatting. Unfortunately the default site wasn't very appealing and the CSS

was quite simple. Using this approach generally violated one of Chen's [5, pg 2] best practices that says "settling for a cheap and amateurish site will devalue your business and can do more harm to your professional image and reputation than not having a web site at all."

The author also tried using Expression Studio for several years because it had better, although still somewhat limited templates (Note: Expression Studio was a standards based replacement for Microsoft's earlier proprietary product, FrontPage). Unfortunately Expression Studio used a different type of master page, called dynamic web templates, than that found in VWD and these dynamic web templates made it difficult or impossible to work on an application in both Expression Studio and VWD at the same time. As a result students created the initial look-and-feel for their site in Expression Studio and then convert the dynamic web template into a .NET master page and do all remaining work in VWD. To compound these problems Microsoft recently dropped support for Expression Studio [9] and it isn't possible to get additional product keys, even under Microsoft Dream Spark [10].

In addition to using basic web site templates, the class introduced students to a series of tools and techniques using standard features of VWD such as data bound controls, the calendar control, the FileUpload control, membership controls (CreateUser, Login, ChangePassword and LoginView) and one shareware product called FreeTextBox. These will be discussed under the current solution.

Because of the limitations of both straight VWD and Expression Studio, student teams often looked to cloud-based solutions for their client projects and ended up using products such as Wordpress, wix.com, weebly.com, webs.com, and others. While the cloud-based solutions provided numerous templates and allowed the teams to create a variety of good-looking web sites, they also required time for an additional learning curve on the part of the student teams and often didn't support all the functionality the client wanted without the purchase of add-ons or having team members learning a new development environment. Since the course develops client solutions for free and most of our clients aren't willing to pay for anything, cloud-based solutions didn't always provide the full functionality the client wanted. In addition, since most of our clients are small organizations or small businesses without internal technical support, student teams often found it necessary to develop extensive user manuals explaining how to use the developer interface to the cloud-based solution to modify text, upload new images, change links, etc.

THE CURRENT SOLUTION

In an attempt to come up with better client solutions and to help our students develop a skill set they could add to their web development toolkit and hopefully apply to future projects the author switched the course from using predefined templates to using free, standards-compliant templates available from a number of web sites including: All-free-download [11], Open Source Web Design [12], free CSS templates.org [13], Open Designs [14] and others.

Prior to beginning the client project, students now work through a series of exercises that lead to the creation of a well-designed web site to sell legal products or services of their choice. These exercises include:

1. Locating a template they like
2. Make changes to the template to customize it to their needs
3. Converting the template into an ASP.NET master page
4. Converting navigational links to work in the .NET environment
5. Modifying the menu bar to fit their needs
6. Creating content pages for each menu item
7. Adding content management to each page
8. Uploading project/service images
9. Allowing users to register at the site
10. Allowing users to place items in a stored shopping cart
11. Simulating the sale of the ordered items

The first seven steps will be discussed in the following sections.

Step One – Find a template

Students are first instructed to locate sources of free web site templates and browse through to find a design they like and that looks like it would be appropriate for what they are selling. Most of the free sites have clickable thumbnail images of each template that open into a full-page view of the page. Many of the sites also include user comments and reviews. Figure 1 shows the thumbnail and the full-page view of a template called "Business Design 2007" from www.oswd.org.

Step Two – Customize to the current requirements

Students are encouraged to study the design they have selected for both general look-and-feel issues but also conformance with design best-practices[6]. Do they like the placement of images and text? Are there different designs for different kinds of users [1, pg. 34]? Do the location, number and style of menus work and follow standards [1, pg. 32, 33][4][7]? What menu items do they need? Do they want one or two menu bars? Do they want the menu at the top or on the side? Which side? Does the color scheme work for them? Are the fonts okay? Do they want a one-column format or do they need two columns like Figure 1? What other changes are required to meet their needs?

Based on answers to questions like these students learn how cascading style sheets work, how to link external style sheets to a web page [2] and how to edit them to achieve the desired look and feel. Students learn how different style selectors work [2][7] and how to create and set styles that affect all the desired parts of the web page but nothing else. They learn how to use the style editing capabilities of VWD and how to edit style sheets directly.

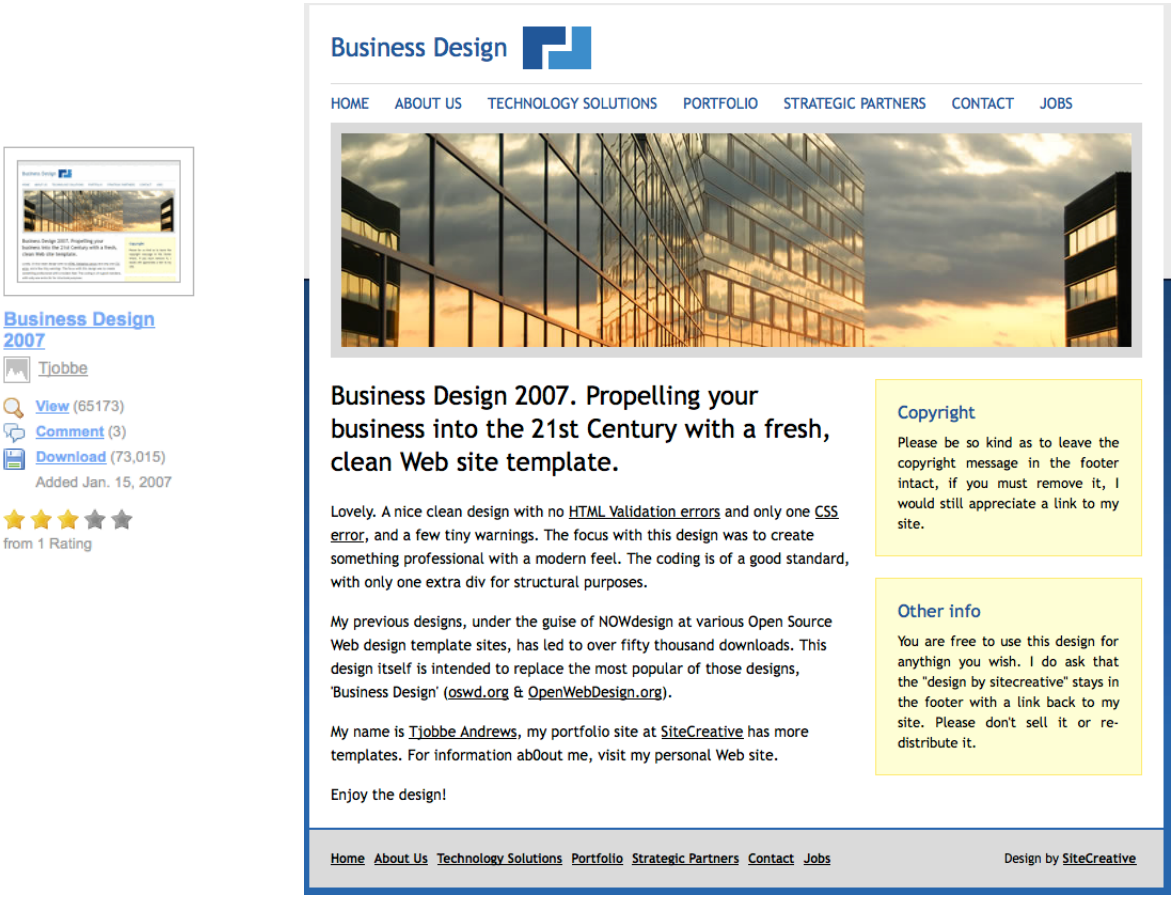

Figure 1 – Iconic and full-page view of template.

Step Three - Convert the design into a .NET master page

After customizing the template, students study the overall design and decide what parts of the page should be on the master page and what parts should be on content pages. Considering the template shown in Figure 1, it is likely that the two menu bars should be on every page and thus belong on the master page. The image may, or may not belong on every page so moving it to the content pages allows for some flexibility and the two columns of text under the image are also things that can change from page to page and as such don't belong on the master page. Figure 2 shows one way to look at this. Areas A and E go to the master page while B, C and D belong to content pages.

Students are shown two different ways to change the template into a master page: 1) edit the template adding the appropriate source code to convert it into a master page and 2) create a new master page in VWD and then copy most of the template file into the master page with appropriate edits.

Approach 1: Master pages begin with a master page directive, <@ Master Language …> which is added to the top of the page. Inside the <body> tag they require a <form> tag with the runat="server" attribute and one or more ContentPlaceHolders to individual page content. Students are shown how to make these changes and then generate content pages using the new master.

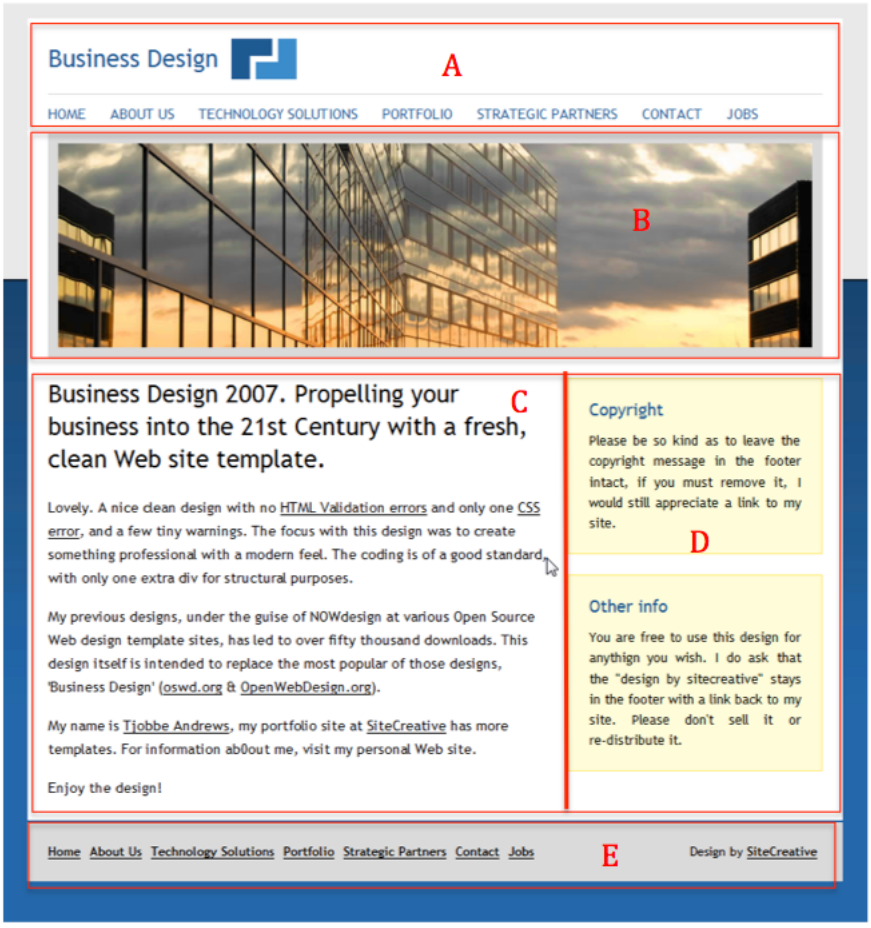

Figure 2 – Identification of different areas on the web page.

Approach 2: Students create a master page in VWD. By default VWD creates two ContentPlaceHolders on a master page, one in the page header and one in the body. Content from the template's header is copied into the first placeholder while content from the body of the template is copied into the second placeholder (VWD automatically places the second ContentPlaceHolder inside a <form> tag so nothing extra has to be done for that).

In either case the students need to understand how CSS designers format web pages by applying styles to tags. For example, section C in Figure 2 is defined by a with the ID of "content" while section D is defined by a with ID of "news". The CSS that selects these IDs not only define text and colors for each section but also causes the content to float into the two-column format (best practice – don't use tables to format web pages [2]). In order to preserve this styling the new ContentPlaceHolders must be added inside the tags.

If a design decision is made to show the image in section B on some pages, show different images on other pages or possibly not show any image at all on some pages, then another ContentPlaceHolder should be added inside the defining this part of the page.

Step Four - Convert navigational links to work in the .NET environment

Normal HTML/CSS web pages follow best practices and create menus from unordered lists of anchor tags[6]. These tags use traditional MS-DOS navigation conventions to refer to locations such as the current directory, the parent directory or the root directory of the current drive. ASP.Net web sites use the concept of a Web Application and have a special symbol, ~, to represent the root of the application. Because of this an anchor tag such as Home which refers to home.HTML in the current directory, or Home which refers to home.HTML in the parent of the current directory, must be converted to an ASP hyperlink such as <ASP:HyperLink ID="hlHome" runat="server" NavigateUrl= "~home.aspx">Home</ASP:HyperLink> that reference the home.aspx page in the application root. The benefit of the ASP hyperlink is that it always refers to the same place, regardless of where the current page is located.

Step Five - Modify the menu bar to fit the application's needs

As mentioned in step four, most HTML/CSS web sites format unordered lists of anchor tags into menu items. Students may need to delete some menu items provided by the template designer, change the names of other items or add new items to suit the needs of their application. Figure 3 shows a set of anchor tags from the Business Design 2007 template converted to a set of ASP hyperlinks.

Step Six - Create content pages for each menu item

Once the template has been converted to a master page and saved the student can generate content pages using the new master page. If everything was done correctly they should see a page like Figure 4 with four empty content areas (content areas replace content place holders on the master page)

```
<div id="navigation">
    <ul>
        <li><a href="#">HOME</a></li>
        <li><a href="#">ABOUT US</a></li>
        <li><a href="#">TECHNOLOGY SOLUTIONS</a></li>
        <li><a href="#">PORTFOLIO</a></li>
        <li><a href="#">STRATEGIC PARTNERS</a></li>
        <li><a href="#">CONTACT</a></li>
        <li><a href="#">JOBS</a></li>
    </ul>
</div>
```

```
<div id="navigation">
    <ul>
        <li><asp:HyperLink ID="hlHome" runat="server" NavigateUrl="~/Default.aspx">Home</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlAbout" runat="server" NavigateUrl="~/About.aspx">About Us</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlSolution" runat="server" NavigateUrl="~/Solutions.aspx">Technology Solutions</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlPortfolio" runat="server" NavigateUrl="~/Portfolio.aspx">Portfolio</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlPartners" runat="server" NavigateUrl="~/partners.aspx">Strategic Partners</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlContact" runat="server" NavigateUrl="~/Contact.aspx">Contact</asp:HyperLink></li>
        <li><asp:HyperLink ID="hlRegistration" runat="server" NavigateUrl="~/Registration.aspx">Registration</asp:HyperLink></li>
    </ul>
</div>
```

Figure 3 – Conversion of HTML anchor tags to ASP hyperlinks.

Figure 4 – Empty content page created using the new master page.

The student can then populate the image area or the right and left columns with content appropriate for the page.

Step Seven – Add content management to each page

Rather than hardcoding content on pages, students are encouraged to follow a best practice [4] using a content management system (CMS) for most, if not all, of their page content. The CMS introduced to handle this requirement is a simple, do-it-yourself system using a database to hold page content, a data bound control to display the content and an elegant piece of shareware software to simplify data entry.

The database requirements are very simple: one table with at least three fields. The suggested names for the fields are "Page", "Section" and "Content". Page and Section are used to associate Content with a specific page (Home, About Us, etc.) and portion of the page (RightSide, LeftSide, etc.).

The data bound control most frequently used in this CMS is a FormView because it is designed to display a single record at a time and the format of the display is completely up to the designer. Stripped down to its simplest form a FormView needs only a single label to display the page content and a text box for data entry and edit. There are, however, several problems with this simple configuration: 1) A text box only accepts unformatted text and 2) even if you use a multiline text box, the new-line codes created by the Enter key aren't recognized when the text is displayed in the Label. Labels do, however, recognize and respond to HTML tags but it is probably not a good idea to allow users to enter HTML tags directly into a text box, even if you assume they are able to enter syntactically correct tags!

This is where the shareware tool mentioned above comes in. FreeTextBox is a rich text editor that offers Microsoft Word like editing capabilities on a web page and, as its name implies, it is free. To use FreeTextBox you must download a dll file from freetextbox.com (there are different versions for each version of the .Net framework) and copy it into a bin directory in the web application root. The dll is then registered with the following statement on each page that uses it:
<%@ Register TagPrefix="FTB" Namespace="FreeTextBoxControls" Assembly="FreeTextBox" %>

Finally, you replace the text boxes in the FormView's EditTemplate and InsertTemplate with the following:

<FTB:FreeTextBox ID="mainFTB" runat="server" Text='<%# Bind("SectionContent") %>' ToolbarLayout="ParagraphMenu,FontFacesMenu,FontSizesMenu,FontForeColorsMenu,FontForeColorPicker,FontBackColorsMenu,FontBackColorPicker|Bold,Italic,Underline,Strikethrough,Superscript,Subscript|JustifyLeft,JustifyRight,JustifyCenter,JustifyFull;BulletedList,NumberedList,Indent,Outdent;CreateLink,Unlink,InsertImage|Cut,Copy,Paste,Delete|SymbolsMenu|InsertRule,InsertDate,InsertTime" Height="200px" Width="100%" />

The important part of this that is not mentioned at freetextbox.com is the Text='<%# Bind("SectionContent") %>' part that allows FreeTextBox to work with the database. The ToolbarLayout attribute lists all the tool bars that are available with FreeTextBox and you can delete the ones you don't want and rearrange the others to suit your needs.

Figure 5 shows a portion of a web page formatted using FreeTextBox and what it looks like in the FreeTextBox editor.

The Update and Cancel link buttons shown below the FreeTextBox are on the FormView's EditTemplate and its InsertTemplate. There are corresponding Insert, Edit and Delete link buttons on the FormView's ItemTemplate but they are not visible in Figure 5 because they were moved inside a LoginView and restricted to Admin users (creating users and adding them to roles is discussed in the second paper in this series).

One final consideration: FreeTextBox formats text by inserting HTML tags in the text. Since imbedding script tags in text is fundamental to "insertion attacks" on web sites, .NET and Internet Information Server (ISS) don't normally allow text containing tags to be saved in a database. To let the system know that it is okay to save this text it is necessary to add 'ValidateRequest="false"' to the @Page directive of each page using FreeTextBox. On systems running the .Net 4.0 Framework or above it is also necessary to add <httpRuntime requestValidationMode="2.0"/> to the <system.web> section of the web.config file.

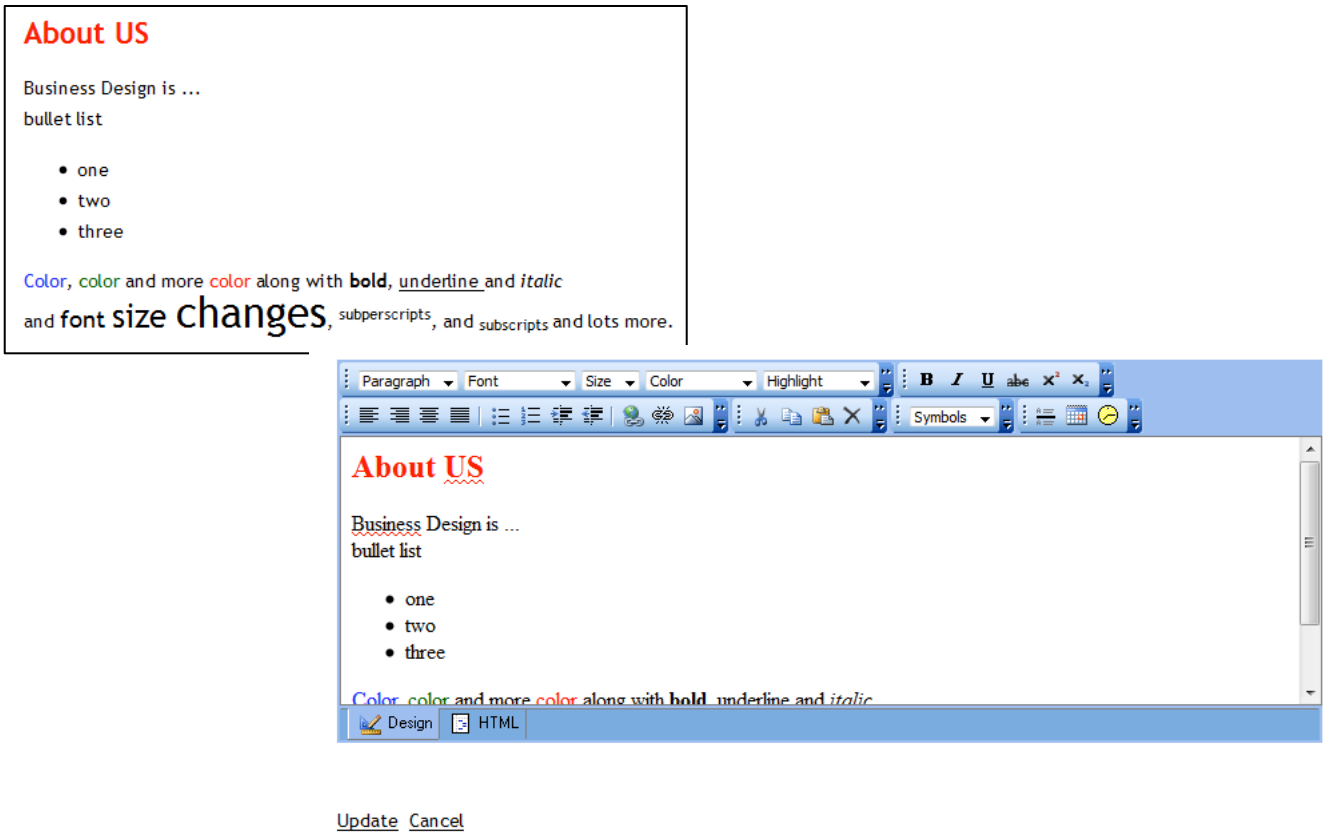

Figure 5 – Formatted text on the web page and inside FreeTextBox.

Steps Eight through Eleven – Finishing the web site

The remaining steps in this process used to help students develop full-featured web sites are a bit more conventional than the previous steps and are only summarized here.

Step Eight – Upload images of products or services: Use the FileUpload control to upload images to a folder in the web site and store the image path in the database. Then data bind the image URL to image controls on a products page.

Step Nine – Add a registration function: Create a registration form to capture and store customer data in the database. At the time the data is stored use the ASP Membership CreateUser method to register the customer with the system. Once this is done the LoginView control can be used to differentiate anonymous users from authenticated users.

Step Ten – Allow users to place items in a stored shopping cart: Using the LoginView, only allow logged-in users to add items to the shopping cart. Also allow users to edit and delete items from their shopping cart.

Step Eleven – Simulate the sale of ordered items: Create an invoice header from customer information stored in Step nine. Create detailed line items, including extended prices, using product and quantity information stored in the shopping cart. Calculate and display the total due for the order. Finally, delete the user's items from the shopping cart.

CONCLUSIONS

This paper outlines a pedagogical process for teaching relatively non-technical MIS students how to create full-function online sales oriented web sites using HTML and CSS standards based pages with .NET features and one shareware component. The process involves selecting a suitable web site template from the thousands of HTML/CSS standard templates available and then modifying it to create a .Net compatible master page. Once the master page is created individual content pages are designed to implement a simple content management system using FreeTextBox from freetextbox.com. Product images are uploaded to the web site and relevant information is stored in a database. Customers are allowed to register with the system and once registered and logged in can select items from product pages and add them to a shopping cart. Finally, registered users can edit their shopping carts and order the selected items.

REFERENCES

(1) Fowler, Susan and Victor Stanwick, *Web Application Design Handbook: Best Practices or Web-Based Software*, Morgan Kaufmann, 2008.
(2) Irish, Paul, Nick Cooley, Adam McIntyre, Rob Larsen, Joel Oliviera, Jared Williams, Annette Arabasz and Rob Cherny, "Front-end Code Standards & Best Practices", http://isobar-idev.github.io/code-standards/, last accessed 1 May 2014.
(3) Jacobs, Denise, "CSS Architectures: New Best Practices", Sitepoint, http://www.sitepoint.com/CSS-architectures-new-best-practices/, last accessed 1 May 2014.
(4) FrozenFire, "Best Practices in Website Design", http://frozenfire.com/best-practices-in-website-design/, last accessed 1 May 2014.
(5) Chin, Paul, "Best Practices for Developing a Web Site", http://www.cbv.ns.ca/Departments/FinancialServices/modules/mastop_publish/files/files_4e68d44fb0d77.pdf, last accessed 1 May 2014.
(6) Morris, Terry Ann, "Web Design Best Practices Checklist, http://terrymorris.net/bestpractices/, last accessed 1 May 2014.
(7) Web Design Works, "Web Development Best Practice Tips", http://www.webdesignworks.com/tips-advice/web-development-best-practice-tips/, last accessed 1 May 2014.
(8) Cats who Code, "Top 10 best practices for front-end web developers", http://www.catswhocode.com/blog/top-10-best-practices-for-front-end-web-developers, last accessed 1 May 2014.
(9) Microsoft Expression Changes, http://www.microsoft.com/expression/eng/, last accessed 1 May 2014
(10) Microsoft DreamSpark, https://www.dreamspark.com, last accessed 1 May, 2014.
(11) All-free-download http://all-free-download.com/free-website-templates/,
(12) Open Source Web Design http://www.oswd.org/designs/browse/page/1/
(13) Free CSS templates.org http://www.freeCSStemplates.org/CSS-templates/
(14) Open Designs http://www.freeCSStemplates.org/CSS-templates/