

**School of Information Technology**

Course : Diploma in Business Informatics

Subject : ITP282 - Enterprise Application Development & Project

AY / Sem : 2018 S2

**Lab 3a: Master Pages, CSS and Themes**

**OBJECTIVES:**

At the end of this practical, the student should be able to:

1. Utilise Master Pages to implement a consistent look and field across all their web pages.
2. Utilise CSS with Master Pages.
3. Apply Themes to their web pages.

# Introduction

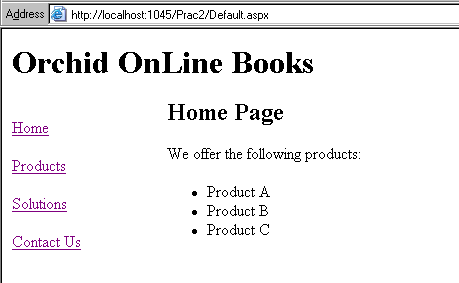
A professional looking web site will typically contain features such as a common header, menu system for the entire site, and a common footer providing copyright information. These elements are present on every page and should have a consistent layout, and look and feel. Instead of repeating the codes necessary to maintain this consistency across all pages of a website, it is better if we can use a template that all pages can reference to. This way, any changes made to the website design need to be made on the template only.

In ASP.NET, this template is called a Master Page.

A Master Page defines the layout used by all pages of the web site. Regions that are suppose to contain the contents of the different web pages are indicated by *ContentPlaceHolders* controls.

Each Web Form (.aspx pages) reference to a Master Page. This reference is coded into the *MasterPageFile* property in the Page Directive. Each Web Form has one or more *ASP.NET Content* controls that correspond to each of the *ContentPlaceHolders* controls in the Master Page. This is where contents such as text, HTML and other ASP.NET controls are added. **When the Web Form is requested, ASP.NET will map the contents of the *Content* controls into the respective *ContentPlaceHolder* controls of the Web Page to create a complete web page** and send that to the client as a complete web page.

|  |  |  |
| --- | --- | --- |
| **Master Page** |  | **Content Page** |



## Exercise 1: Master Pages

1. **Create a new *ASP.NET* *Empty Web Site*** *(File-> New-> Web Site)* and name it **Lab3\_MasterPage**.
2. **Add a *Master Page*** (by right-clicking on website -> Add new item, select Master Page) and:
   * Make sure it has *Code Behind*.
   * Do NOT have a *Master Page*. (From VS 2008 onwards, *Master Pages* can have *Master Pages*, but we do not need that in this exercise.)
   * Name it *Main.Master*.
3. Examine *Main.Master* in *Source View*. Note that:
   * It has the HTML <html>, <head> and <body> elements.
   * It has a <%@ Master … %> directive (instead of <%@ Page … %>). This will inform the compiler that this is a *Master Page*.
   * By default, **TWO** *ContentPlaceHolder* controls (denoted by <asp:ContentPlaceHolder … ></asp:ContentPlaceHolder> tags) are created. One of them is placed within the HTML <head> element and the other in the <body> element. Their IDs are 'head' and 'ContentPlaceHolder1' respectively.

The 'head' *ContentPlaceHolder* allow you to place content that needs to be in the <head> element of the web page, such as JavaScript, CSS, meta tags, etc. The 'ContentPlaceHolder1' *ContentPlaceHolder* allows you to place content that are to appear in the <body> element of the web page. (Before VS2008, *ContentPlaceHolder* controls were only allowed in the <body> element.)

* + You can add more *ContentPlaceHolder* controls if you so desire.

1. Edit *Main.Master* and add/edit the code shown in Figure 1 and Figure 2.



Figure 1

We will have these warnings (green squiggly lines) because we have not created these pages yet.



Figure 2

1. Switch to *Design View*. Click on the box with the purple outline. Note the ID in the tab above the box. This is the *ContentPlaceHolder* with the ID of 'ContentPlaceHolder1'. Remember its location. Later, when we put content in the Web Forms, this is the location they will appear.
2. Can you find the 'head' *ContentPlaceHolder* in the *Design View*? Why?
3. Add a new Web Form with:
   1. Code Behind.
   2. Master Page: *Main.Master*.
   3. Name: *Default.aspx*.
4. Examine *Default.aspx* in *Source View*. Note the following:
   * The page directive starts with <%@ Page … %>. How is this different from the page directive of *Main.Master*?
   * There are NO HTML HTML <html>, <head> and <body> elements.
   * The page directive has a *MasterPageFile* property. What is the name of the file?
   * It contains **TWO** *Content* controls. Both of them have their own ID property, and also a *ContentPlaceHolderID* property each. The values of the *ContentPlaceHolderID* properties are 'head' and 'ContentPlaceHolder1' respectively. What do you think the *ContentPlaceHolderID* property do?
5. Switch *Default.aspx* to *Design View*. Note the following:
   * The design created in Main.Master is visible, but NOT editable.
   * There is also purple outlined box in the page. The tab shows 'ContentPlaceHolder1' which is the value of the *ContentPlaceHolderID* property of the 'Content2' *Content* control, instead of the control type and ID like other ASP.NET controls.
6. Type 'Welcome to Orchid Online Books.' (without quotes) into the purple outlined box.
7. Switch back to *Source View*. Do you see the words you typed in the previous step? Note that they are within the opening and closing tags of the 'Content2' *Content* control.
8. Change the line to **<h2>**Welcome to Orchid Online Books**.</h2>**.
9. Now add the pages (with Code Behind and Main.Master as the Master Page) and add in the HTML as shown in Table 1 into their *Content* controls.

Table 1

|  |  |
| --- | --- |
| Page | HTML |
| Products.aspx | <h2>Our Products</h2> |
| Solutions.aspx | <h2>Our Solutions</h2> |
| ContactUs.aspx | <h2>Contact Us </h2> <p>Contact us at Tel:(65)6888888 / email:sales@orchid.com.sg</p> <p>Alternatively, leave us your contact information.<br /> Our sales representative will contact you shortly</p> <table> <tr> <td><asp:Label ID="lblName" runat="server" Text="Name"></asp:Label></td> <td><asp:TextBox ID="txtName" runat="server"></asp:TextBox></td> </tr> <tr> <td><asp:Label ID="lblEmail" runat="server" Text="Email" ></asp:Label></td> <td><asp:TextBox ID="txtEmail" runat="server"></asp:TextBox></td> </tr> <tr> <td></td> <td><asp:Button ID="btnSubmit" runat="server" Text="Submit" /></td> </tr> </table> |

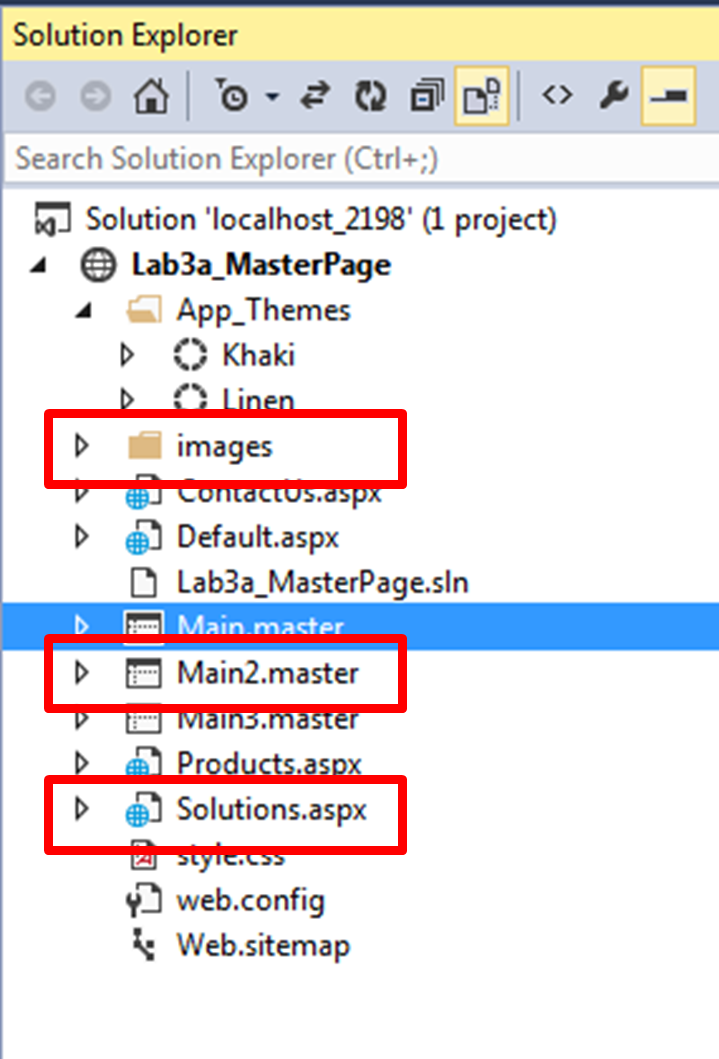
1. Test the web site. Navigate to the various pages using the menu on the left. You should see the various web pages *Default.aspx*, *Products.asp*, etc displaying their contents while maintaining a common look and feel as defined in the *Master Page*.

## Exercise 2: Cascading Style Sheets

CSS allows us to define a common style across all our web pages. A good place to add CSS is the Master Page as it will apply to all pages.

1. Download the **Lab3\_Resources.zip** file. The resource file contain *Main2.master*, *images folder*. Copy its contents into *Solution Explorer*.
2. Your Solution Explorer should look as shown in Figure 3. If not, drag the new files and folders to match.

Figure 3



1. Open *Main2.Master* and view it in *Design View*. The design is very plain.
2. Switch to *Source View* and remove only the comment tags around the <link … > element in the <head> element:

<%--

<link …> -- do not remove

--%>

1. Switch back to *Design View*. What does the page look like now? Why?

Answer:

1. Test the web site. Is it using the new Master Page?
2. Open all the .aspx pages in *Source View* and change the *MasterPageFile* property of their page directives to point to *Main2.Master*.
3. Test the web site again. This time, does the new design show like Figure 4? Take note the colour of the background is white.

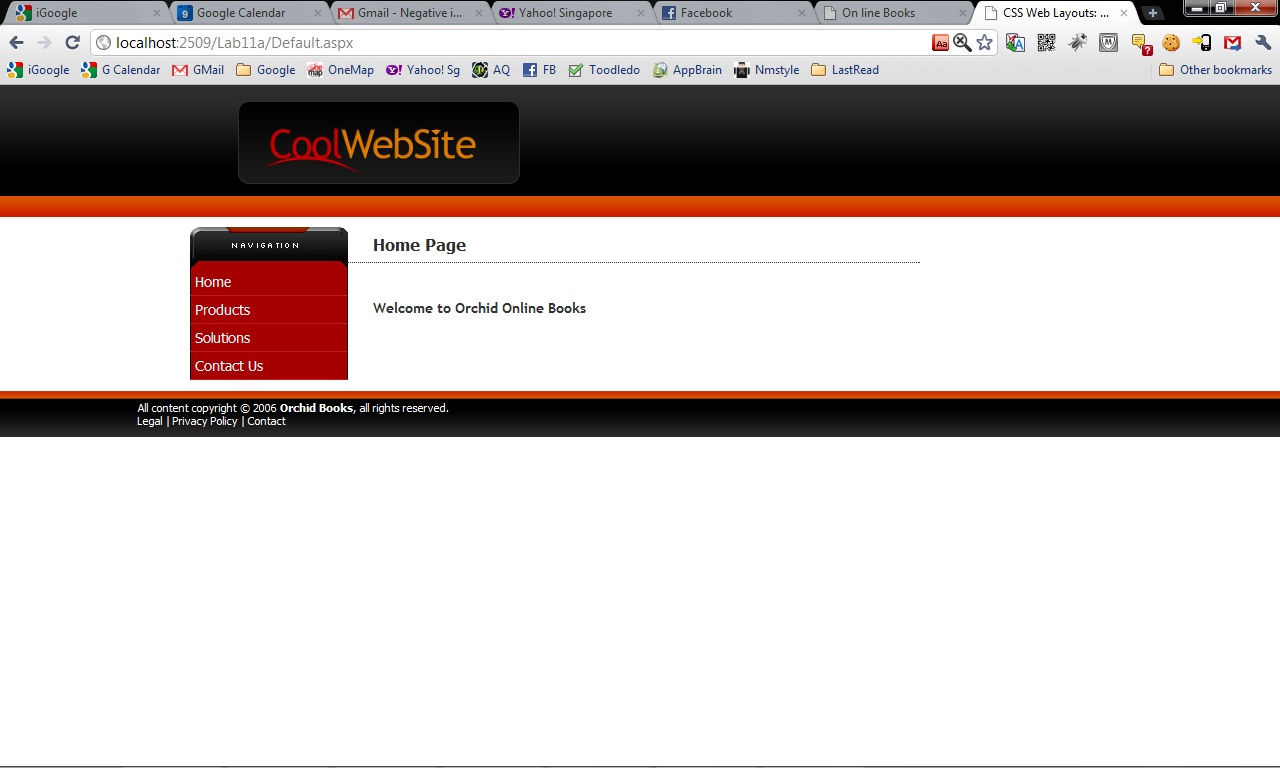


Figure 4

## Exercise 3: ASP.NET Themes

While CSS applies to formatting HTML elements, ASP.NET has Themes for formatting ASP.NET server controls in a way which is very similar to the way CSS applies formatting to HTML elements.

A **Theme** is *a group of files that define the look and feel of a site*. Generally, *themes contain style sheets, images and skin files*. These files contain the settings which are applied to the site's content. By keeping the design information in one place, when changes need to be made to the appearance of the site, you can make them without having to touch the actual content files.

1. Add a new Theme Folder by:
   1. Right-clicking the root of your project in the *Solution Explorer*.
   2. Select *Add* 🡺 *Add ASP.NET Folder 🡺 Theme* as shown in Figure 5.
   3. Name it *Khaki* as shown in Figure 6.

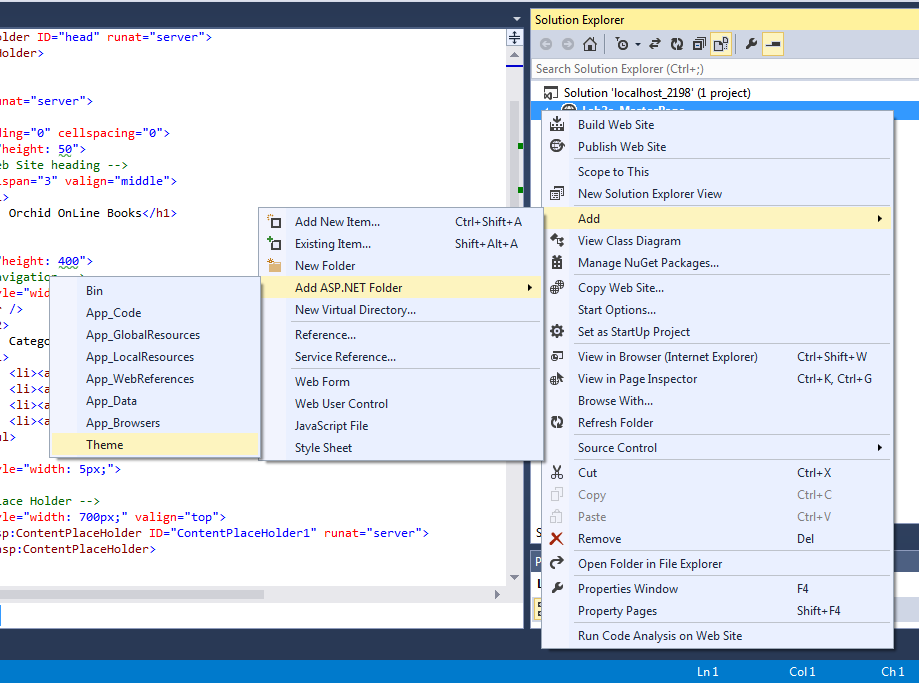


Figure 5

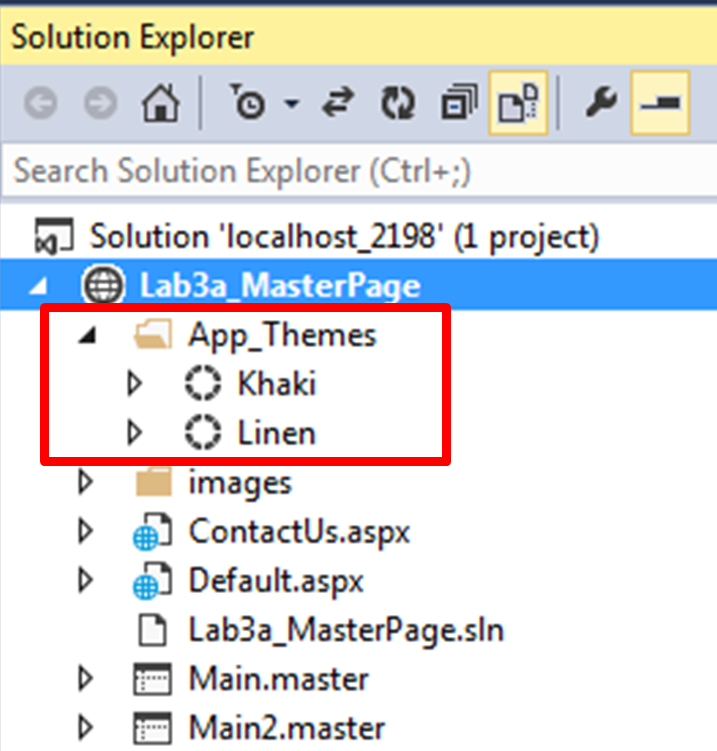


Figure 6

1. Add a second Theme folder in the same way, but name it *Linen*.
2. Add a new style sheet into the Khaki theme:
   1. Right-click the *Khaki* folder.
   2. Select *Add New Item...*.
   3. Select *Style Sheet* and name the files *style.css*.
   4. Click the *Add* button.
3. Add another *style.css* style sheet to the Linen folder. Your App\_Themes folder should look like Figure 7.

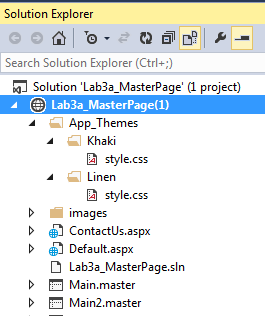


Figure 7

1. Edit both styles sheets to add the codes shown in Figure 8. Make sure you add the codes to the correct files.

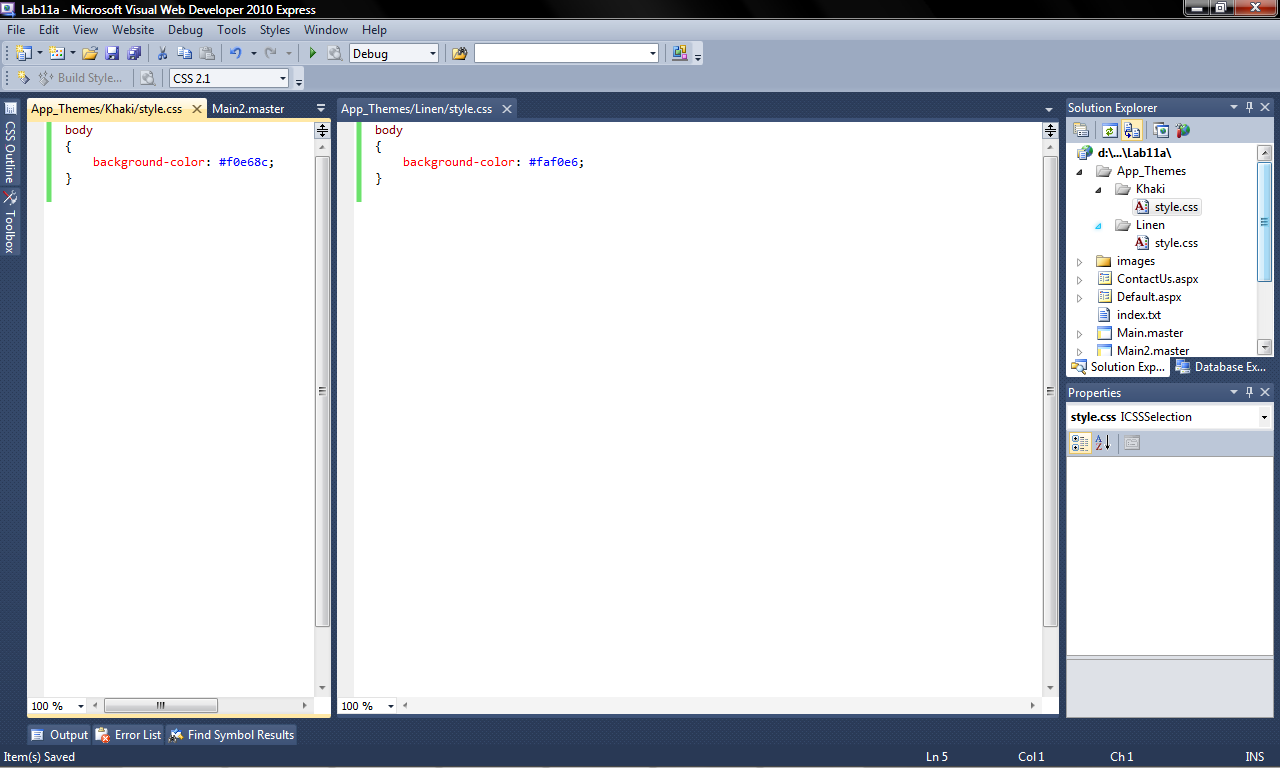


Figure 8

1. Now we need to apply the Themes to the web site. To do this, we have to add code to *web.config* file. Open *web.config*. (If the *web.config* file does not exist in your project, use *Add New Item…* to add a *Web Configuration File* and name it *web.config*).
2. Add the line shown in Figure 9. Make sure you add it in the correct location.

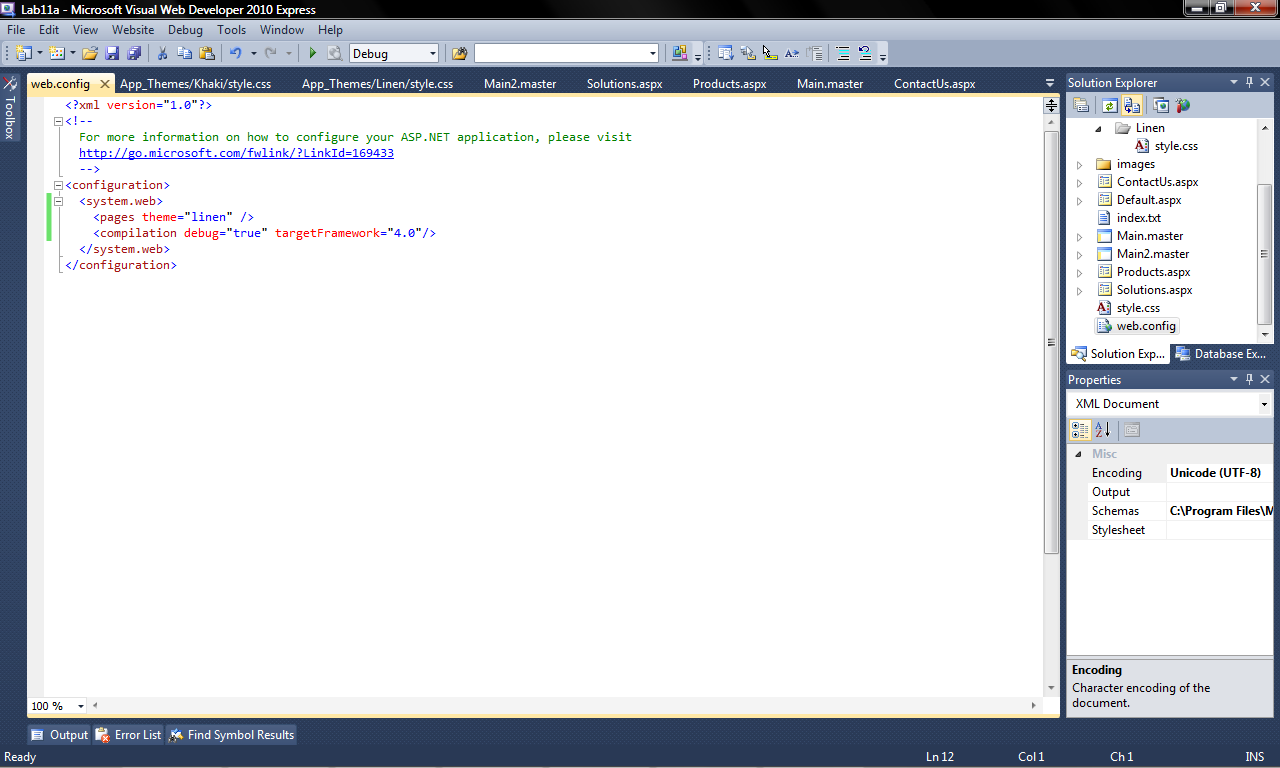


Figure 9

1. Test your web site again. Navigate to different pages. This time, what is the colour of the background?
2. Return to VS and edit the *web.config* file. Change *Linen* to *Khaki*.
3. Test the web site. What is the background colour this time?

## Exercise 4: Themes and Skins

*Skin Files* are simply files with a .skin extension that contain formatting information for different ASP.NET controls. They are similar in nature to CSS files, but while CSS is applied to HTML elements and is processed by the client, the formatting in skin files is applied to ASP.NET server controls and is processed by the server.

1. Using *Add New Item…* on the *Khaki* folder and add a *Skin File* named *SkinFile.skin*.
2. Add another *SkinFile.skin* to the *Linen* folder. Your *Solution Explorer* should display the files as shown in Figure 10.

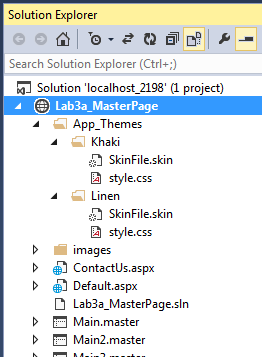


Figure 10

1. Open both SkinFile.skin and add the codes shown in Figure 11. Make sure you add the codes to the correct file. Notice that the code looks similar to the code we use to add these controls to a .aspx page. The difference is that the ID property is missing.

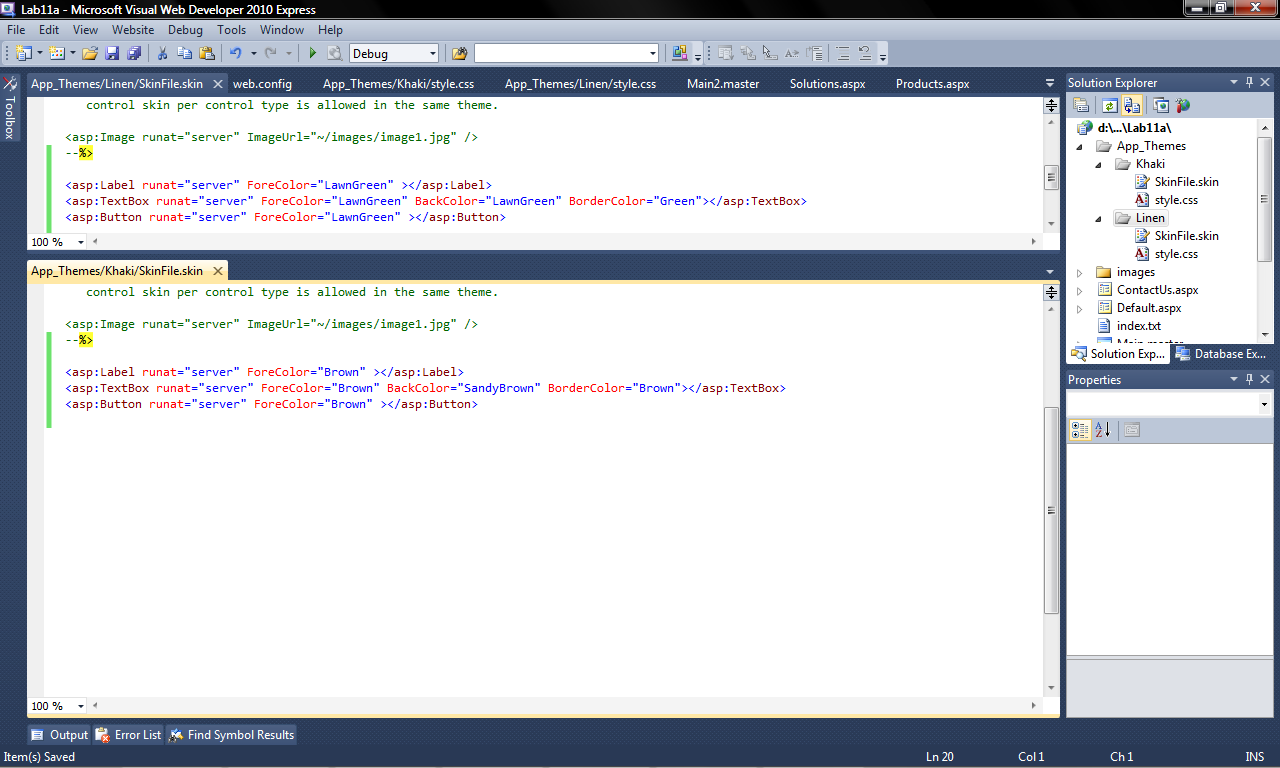


Figure 11

1. Test the web site again and navigate to the *Contact Us* page. Notice how the *TextBoxes*, *Labels* and *Button* have changed in appearance. Change the *Theme* by editing the *web.config* file and observe the differences.

***========== End ==========***