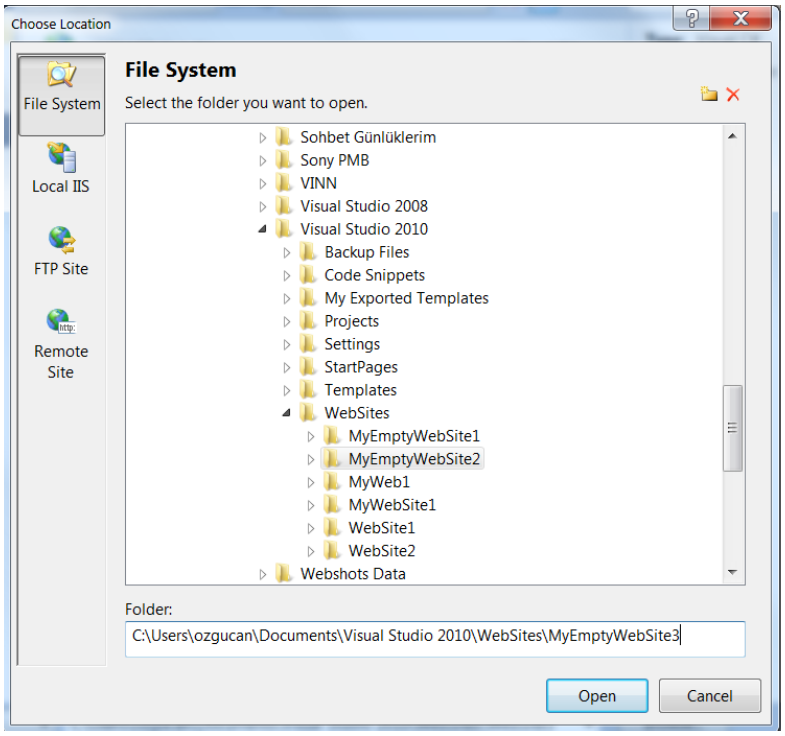
Navigation

Investigating the Behavior of the Virtual Path Property

1. Choose File ➪ New Web Site in VS to create a new web site.
2. Choose ASP.NET `Empty Web Site` as the template and set the Web Location to File System. Note that VS offers a path that ends with `WebSite1`. If you've created other web sites before without giving them an explicit name, you may have a path that ends with `WebSite2`, `WebSite3`, and so on. Click OK to create the site.



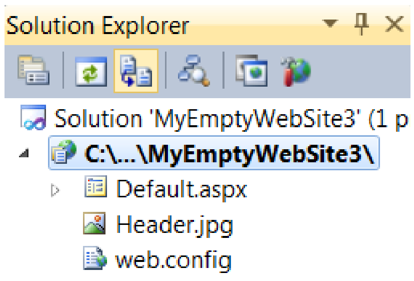
File System

Select the folder you want to open.

```
Sohbet Günlüklerim
Sony PMB
VINN
Visual Studio 2008
Visual Studio 2010
    Backup Files
    Code Snippets
    My Exported Templates
    Projects
    Settings
    StartPages
    Templates
    WebSites
        MyEmptyWebSite1
        MyEmptyWebSite2
        MyWeb1
        MyWebSite1
        WebSite1
        WebSite2
Webshots Data
```

Folder: C:\Users\ozgucan\Documents\Visual Studio 2010\WebSites\MyEmptyWebSite3

3. Add a new Web Form called `Default.aspx` to the site (use the default Web Form option and not your custom template) and then add an image to the root of the site called `Header.jpg`. You can drop one of the images from the previous examples in the root of the site, or you can use an existing image and add that to the site. If you don't rename the image to `Header.jpg`, make sure you adjust the code in the next step.



4. In `Default.aspx`, add the following code to the Source View that inserts three ASP.NET Image controls using different ways to address the image. The images are separated by a line break:

```
<asp:Image ID="Image1" runat="server" ImageUrl="Header.jpg" /><br />
<asp:Image ID="Image2" runat="server" ImageUrl="/Header.jpg" /><br />
<asp:Image ID="Image3" runat="server" ImageUrl="~/Header.jpg" /><br />
```

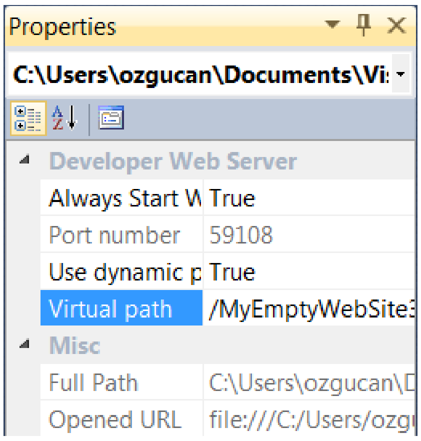
5. Press Ctrl+F5 to open the page in the browser. Note that the address bar of the browser reads something like http://localhost:59108/MyEmptyWebSite3/Default.aspx. Your port number and application name may be slightly different, but what's important to notice is that the web site is located in a separate folder under the web server called localhost. You'll also find that the second image shows up broken. That's because the leading slash refers to the root of the web server, so the image is looked for at http://localhost:59108/Header.jpg , which doesn't exist because the image is located in the `MyEmptyWebSite3` subfolder.

   Open up the source of the page in the browser and look at the three `<img>` elements:

```
<img id="Image1" src="Header.jpg" /><br />
<img id="Image2" src="/Header.jpg" /><br />
<img id="Image3" src="Header.jpg" /><br />
```

   The first two URLs are identical to what you added to the ASPX page. However, the third one has been modified to refer to an image in the same folder as the page that references the image.

6. Close your browser and go back to VS, click the root of the web site in the Solution Explorer, and press F4 to open up the web site's Properties Grid. Set the `Virtual Path` from `/MyEmptyWebSite3` to `/`.


Properties
C:\Users\ozgucan\Documents\Vi:

Developer Web Server	
Always Start W	True
Port number	59108
Use dynamic p	True
Virtual path	/MyEmptyWebSite3
Misc	
Full Path	C:\Users\ozgucan\D
Opened URL	file:///C:/Users/ozg

7. Press Ctrl+F5 again to reopen `Default.aspx` in the browser. The address bar now reads something like http://localhost:59108/Default.aspx . As you can see, the page `Default.aspx` is now located at the root of the server. Therefore, all three images show up correctly. If you look at the HTML source, you'll see this:

```
<img id="Image1" src="Header.jpg" /><br />
<img id="Image2" src="/Header.jpg" /><br />
<img id="Image3" src="Header.jpg" /><br />
```

8. Go back to VS and create a folder called `Test`. Drag the `Default.aspx` file from the root of the site into this new folder and then request the page in the browser. The address bar now reads something like http://localhost:59108/Test/Default.aspx .This time the first image will be broken. If you look at the HTML source, you'll see this:

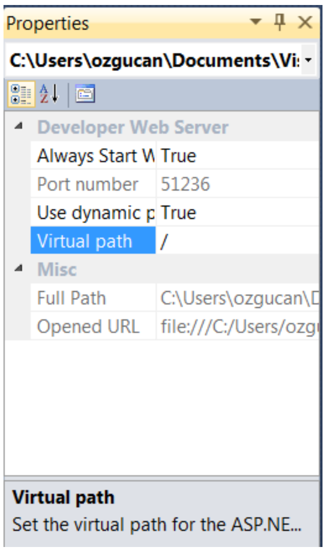
```
<img id="Image1" src="Header.jpg" /><br />
<img id="Image2" src="/Header.jpg" /><br />
<img id="Image3" src="../Header.jpg" /><br />
```

The first `<img>` element tries to find an image relative to the current document. Because the current document lives in the `Test` folder and the image is located at the root of the site, this results in a broken image. The other two `src` attributes point to the correct image in the root of the site.

9. You can close the test project in VS now, and delete it from disk because it is no longer needed.

Creating a Web.sitemap File

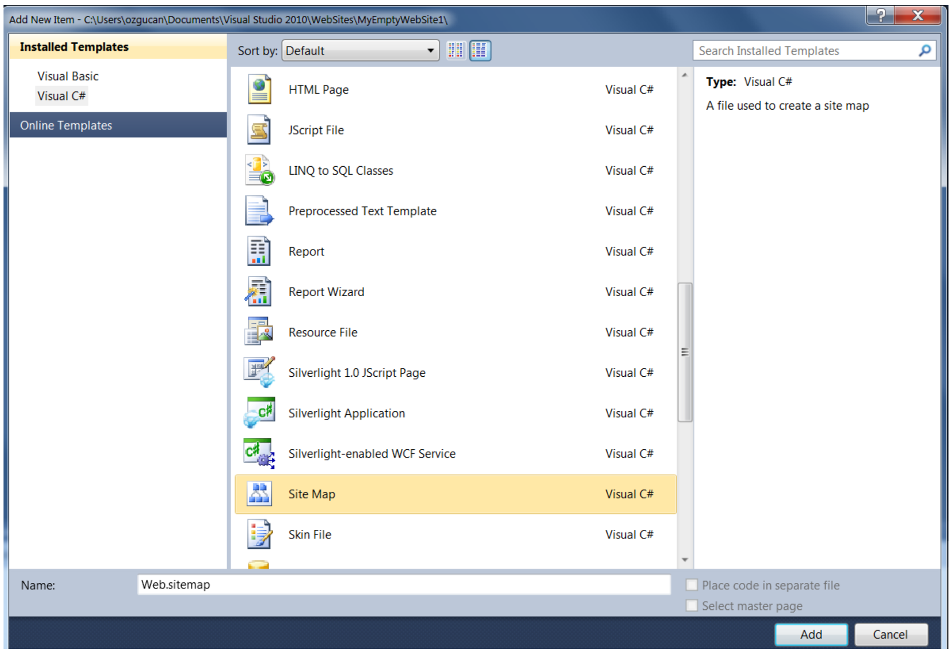
1. Open the project in VS and click the web site in the Solution Explorer to select it. Press F4 to open the Properties Grid and then set the `Virtual Path` property to a forward slash (`/`). From now on, it's assumed that you always run the web site with this root-based URL.



Properties	
C:\Users\ozgucan\Documents\Vi:	
Developer Web Server	
Always Start W	True
Port number	51236
Use dynamic p	True
Virtual path	/
Misc	
Full Path	C:\Users\ozgucan\D
Opened URL	file:///C:/Users/ozg

Virtual path
Set the virtual path for the ASP.NE...

2. Right-click the web site in the Solution Explorer, choose `Add New Item`, and click `Site Map`. Leave the default name set to `Web.sitemap` and click Add. You end up with one root element containing two child nodes in the `Web.sitemap` file.


Add New Item - C:\Users\ozgucan\Documents\Visual Studio 2010\WebSites\MyEmptyWebSite1\
Installed Templates
Visual Basic
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
HTML Page Visual C#
JScript File Visual C#
LINQ to SQL Classes Visual C#
Preprocessed Text Template Visual C#
Report Visual C#
Report Wizard Visual C#
Resource File Visual C#
Silverlight 1.0 JScript Page Visual C#
Silverlight Application Visual C#
Silverlight-enabled WCF Service Visual C#
Site Map Visual C#
Skin File Visual C#
Place code in separate file
Select master page
Add
Cancel

Type: Visual C#

A file used to create a site map

Name: Web.sitemap

3. Modify the `Web.sitemap` so it contains this code:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/" title="Home"  description="Home" >
        <siteMapNode url="~/Default.aspx" title="Home"  description="Go to the homepage" />
        <siteMapNode url="~/Reviews/Default.aspx" title="Reviews"  description="Reviews published on this site" >
          <siteMapNode url="~/Reviews/AllByGenre.aspx" title="By Genre"  description="All reviews grouped by genre" />
          <siteMapNode url="~/Reviews/All.aspx" title="All Reviews"  description="All reviews published on this site" />
        </siteMapNode>

        <siteMapNode url="~/About/Default.aspx" title="About"  description="About my site" >
          <siteMapNode url="~/About/Contact.aspx" title="Contact"  description="Contact me" />
          <siteMapNode url="~/About/AboutMe.aspx" title="About Me"  description="About myself" />
        </siteMapNode>

        <siteMapNode url="~/Login.aspx" title="Login"  description="Log in to the web site" />
     </siteMapNode>
</siteMap>
```

4. Save the file; you're done with it for now.

Adding a Menu to the Site

In this exercise, you see how to add a simple `Menu` control to the master page that uses the `Web.sitemap` file to build up the menu. The `Menu` is added to the `MenuWrapper` area in the master page and presents the menu items horizontally. Because of this orientation, this `Menu` is suitable only for the `Monochrome` theme.

1. Open the master page in Source View and locate the `<div>` called `MenuWrapper`. Remove the placeholder text `Menu Goes Here`. If you added some default text

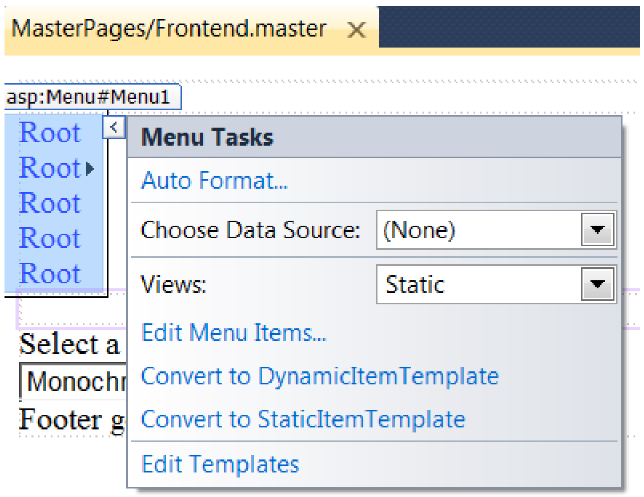
between the `ContentPlaceHolder` tags in the `MainContent <div>` earlier, now is a good place to remove that text again.

```
<div id="MenuWrapper"></div>
<div id="MainContent">
    <asp:ContentPlaceHolder id="cpMainContent" runat="server">

    </asp:ContentPlaceHolder>

</div>
```

2. From the `Navigation` category of the Toolbox, drag a `Menu` and drop it between the `MenuWrapper div` tags. Set the `CssClass` of the `Menu` control to `MainMenu`:

```
<div id="MenuWrapper">
    <asp:Menu ID="Menu1" runat="server" CssClass="MainMenu">
    </asp:Menu>
</div>
```

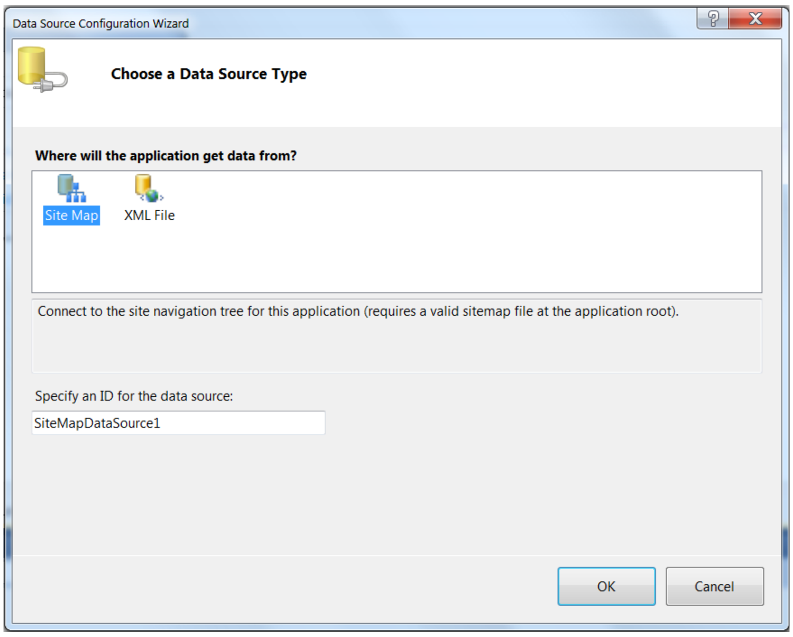
3. Switch to Design View. You may notice that the Design View doesn't look like the final page anymore. That's because you may have removed the `styleSheetTheme` attribute from the `<pages>` element in `web.config`. You can leave it like this for now. With much of the styling already done, this isn't so important. You can still see how the content inside the `cpMainContent` placeholder is going to end up in the browser.
4. Click the `Menu` control's grey arrow to open the Smart Tasks panel.



Choose Data Source: (None)

Views: Static

5. From the `Choose Data Source` drop-down list select `<New data source>`. In the dialog box that appears click the `Site Map` icon.



Data Source Configuration Wizard

Choose a Data Source Type

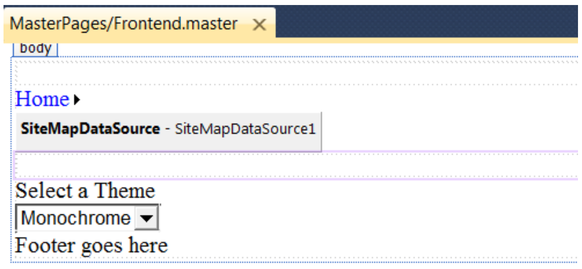
Where will the application get data from?

Connect to the site navigation tree for this application (requires a valid sitemap file at the application root).

Specify an ID for the data source:

SiteMapDataSource1

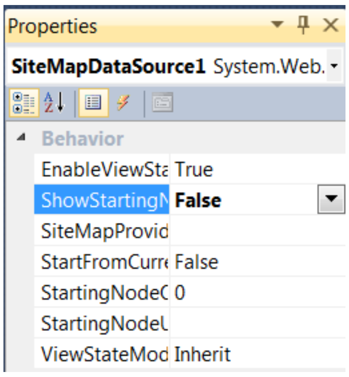
6. Click OK to close the dialog box.
7. When you return to the page, the `Menu` control now shows the top-level element, `Home`.



MasterPages/Frontend.master
body
Home
SiteMapDataSource - SiteMapDataSource1
Select a Theme
Monochrome
Footer goes here

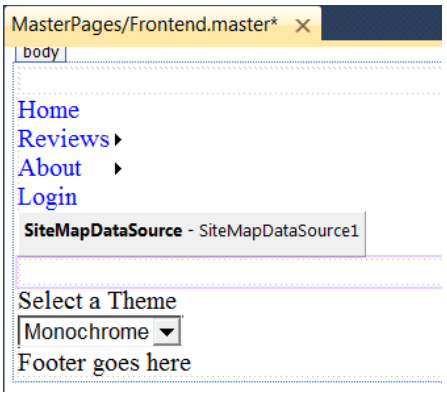
NOTE If your Design View doesn't look like this, but looks much closer to the final page, open the `web.config` file and remove the `styleSheetTheme` attribute from the `<pages>` element.

8. Click the `SiteMapDataSource` once and then press F4 to open or activate the Properties Grid. Change the `ShowStartingNode` property from `True` to `False`.


Properties
SiteMapDataSource1 System.Web.
Behavior

EnableViewSta	True
ShowStartingN	False
SiteMapProvid	
StartFromCurr	False
StartingNodeC	0
StartingNodeU	
ViewStateMod	Inherit

Note that as soon as you do this, the `Menu` control in the designer is updated and shows all direct child menus under the root element: `Home`, `Reviews`, `About`, and `Login`.


MasterPages/Frontend.master*
body
Home
Reviews
About
Login
SiteMapDataSource - SiteMapDataSource1
Select a Theme
Monochrome
Footer goes here

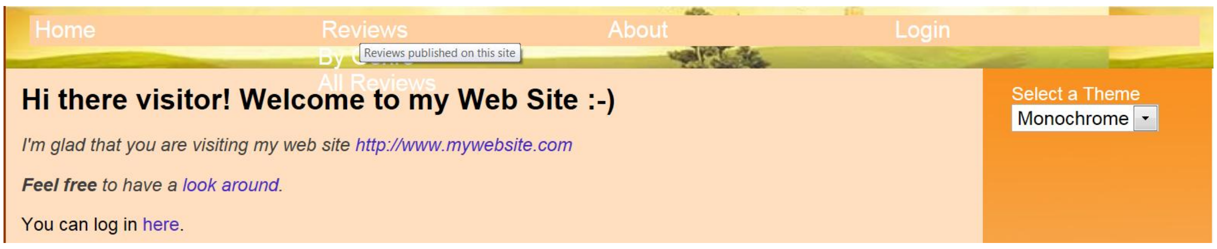
9. Click the `Menu` control once to select it and then make the following changes to the properties of the control using the Properties Grid. Because the `Menu` control has so many properties, you may find it easier to find them if you alphabetically sort the list of properties in the Properties Grid. You can do that by clicking the second button on the toolbar with an `A`, a `Z`, and an arrow on it.

PROPERTY	VALUE
`StaticEnableDefaultPopOutImage`	`False`
`Orientation`	`Horizontal`

When you’re ready, the code for your `Menu` should look like this:

```
<div id="MenuWrapper">
    <asp:Menu ID="Menu1" runat="server" CssClass="MainMenu"
        DataSourceID="SiteMapDataSource1" Orientation="Horizontal"
        StaticEnableDefaultPopOutImage="False">
    </asp:Menu>
    <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server"
        ShowStartingNode="False" />
</div>
```

10. Save the changes to the master page and then request `Default.aspx` in your browser. If necessary, use the `Theme` drop-down list to make `Monochrome` the active theme. You should now see the menu in the horizontal menu area. Hover your mouse over the items, and you'll see sub items appear.



Hi there visitor! Welcome to my Web Site :-)

I'm glad that you are visiting my web site http://www.mywebsite.com

***Feel free** to have a look around.*

You can log in here.

Note that the text on the sub items is hard to read. That's because the CSS from the `Monochrome` theme has changed the text of all anchors in the menu area to `white` and no explicit background color has been set. After you've seen how the `Menu` control works, you get a chance to fix its styling.

Styling the Menu Control

In this exercise you add some CSS rules to the `Monochrome.css` file to influence the way the `Menu` control is styled. By default, the `Menu` control adds CSS classes to the menu items such as `level1` and `level2`, which makes it easy to apply styling at various levels in the menu.

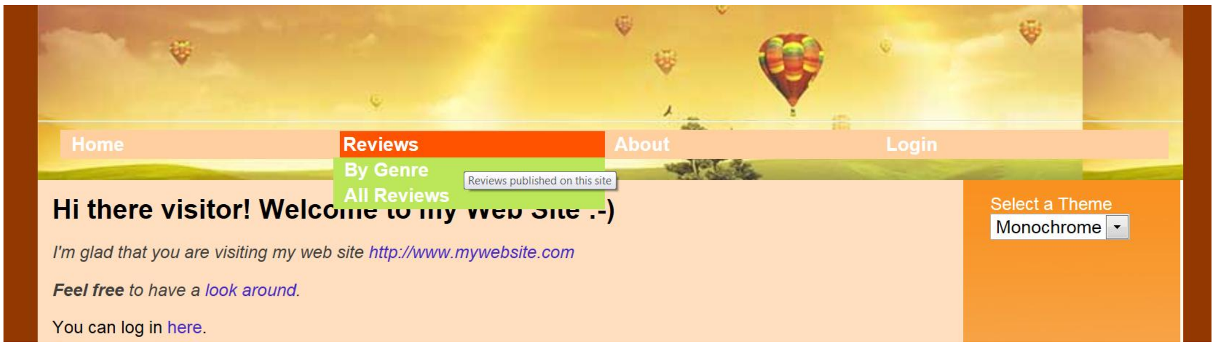
1. Open `Monochrome.css` from the `Monochrome` theme folder and add the following CSS rules. You can leave out the comments placed between `/*` and `*/`, because they only serve to describe the purpose of the selectors. Remember, CSS is **case sensitive**, so type the selectors exactly as shown here:

```
ul.level1
{
    /* Defines the appearance of main menu items*/
    font-size: 14px;
    font-weight: bold;
    height: 19px;
    line-height: 19px;
}

ul.level1.selected
{
   /* Defines  the appearance of active menu items */
    background-color: #FF5300;
}

a.level1
{
    /* Adds some white space to the left of the main menu item text */
    margin-left: 5px;
}

a.level2
{
    /* Defines the appearance of the sub menu items */
    background-color: #BCE75A;
    padding-left: 8px;
}

a.level1:hover, a.level2:hover
{
    /* Defines the hover style for the main and sub items*/
    background-color: #FF5300;
}
```

2. Save and close the file.
3. Next, create the following folders and Web Forms that you'll use. Use the `MyBasePage` template to create the new files. Also, in Source View, give each page a meaningful `Title` to avoid errors later.

FOLDER	FILE NAME	TITLE
/Reviews	Default.aspx	My Favorite Reviews
/Reviews	All.aspx	All Reviews
/Reviews	AllByGenre.aspx	Reviews Grouped by Genre
/About	Default.aspx	About this Site
/About	Contact.aspx	Contact Us
/About	AboutUs.aspx	About Us

4. Save all changes and open the `Default.aspx` page from the root in your browser. Your site menu now looks a lot better and more in line with the rest of the `Monochrome` theme. When you hover the mouse over a main menu, the submenus appear, showing the text on a `green` background. When you hover over a submenu, its background color changes again.



Hi there visitor! Welcome to my Web Site :-)

I'm glad that you are visiting my web site http://www.mywebsite.com

Feel free to have a look around.

You can log in here.

```
ul.level1
{
    /* Defines the appearance of main menu items*/
    font-size: 14px;
    font-weight: bold;
    height: 19px;
    line-height: 19px;
}
```

Building a Navigation System with the TreeView Control

In this exercise, you add a `TreeView` control to the `MenuWrapper <div>` tag, right below the `Menu` you created earlier. The `TreeView` is then bound to the same data source as the `Menu`. Next, you write some code that shows either the `Menu` or the `TreeView`, depending on the active theme.

1. Open the master page in Source View and just below the `Menu` control, add a `TreeView` control by dragging it from the Toolbox.

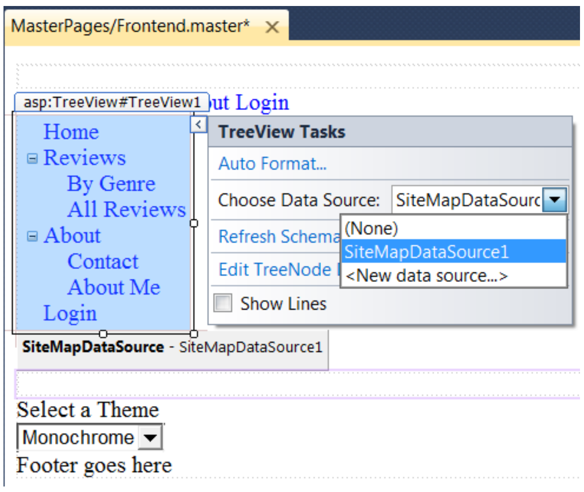
```
<div id="MenuWrapper">
    <asp:Menu ID="Menu1" runat="server" CssClass="MainMenu"
        DataSourceID="SiteMapDataSource1" Orientation="Horizontal"
        StaticEnableDefaultPopOutImage="False">
    </asp:Menu>
    <asp:TreeView ID="TreeView1" runat="server">
    </asp:TreeView>
    <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server"
        ShowStartingNode="False" />
</div>
```

2. Within the opening and closing tags of the control, add the following `<LevelStyles>` element:

```
<asp:TreeView ID="TreeView1" runat="server">
    <LevelStyles>
        <asp:TreeNodeStyle CssClass="FirstLevelMenuItems" />
    </LevelStyles>
</asp:TreeView>
```

The `FirstLevelMenuItems` class selector is defined in `DarkGrey.css` and is used to create some room above each tree item at the first level.

3. Switch to Design View, click the `TreeView` once, and click the small arrow to open the Smart Tasks panel. From the `Choose Data Source` drop-down, select `SiteMapDataSource1`, the data source control you created for the `Menu` control.



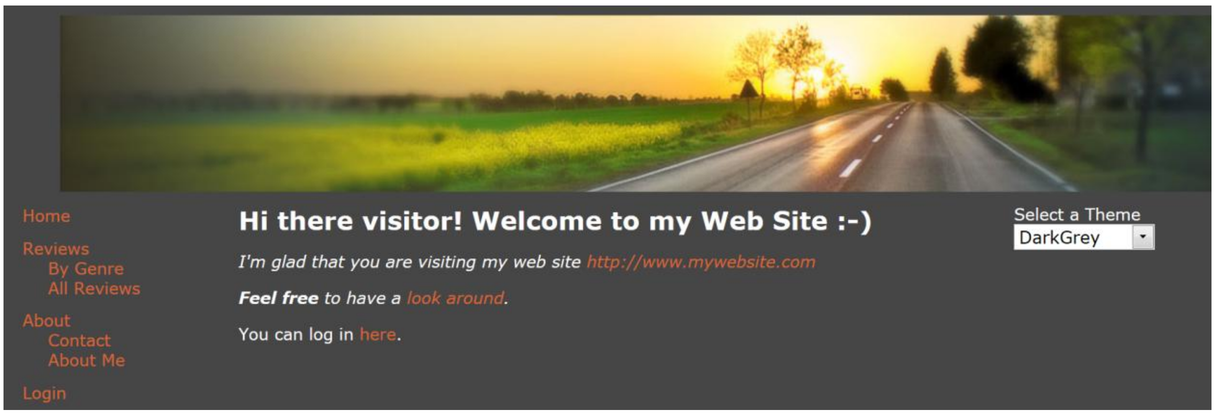
Choose Data Source: SiteMapDataSourc

As soon as you select the data source, the `TreeView` is updated in Design View; it now shows the correct menu items from the site map file.

4. Open the Properties Grid for the `TreeView` control and set the `ShowExpandCollapse` property to `False`.
5. Click somewhere in the document to put the focus on it and then press F7 to open the Code Behind of the master page file and locate the `Page_Load` event that you used earlier to preselect the theme in the `Theme` list. Right below that code, and before the end of the method, add the following code that shows or hides the `TreeView` and `Menu` controls based on the currently active theme:

```
    if(!string.IsNullOrEmpty(selectedTheme) && ThemeList.Items.FindByValue(selectedTheme)!=null)
    {
        ThemeList.Items.FindByValue(selectedTheme).Selected = true;
    }
}

switch (Page.Theme.ToLower())
{
    case "darkgrey":
        Menu1.Visible = false;
        TreeView1.Visible = true;
        break;
    default:
        Menu1.Visible = true;
        TreeView1.Visible = false;
        break;
}
```

6. Save all changes and open `Default.aspx` in the browser. Depending on your currently active theme, you should see either the `Menu` or the `TreeView` control. Select a different theme from the list and the page will reload, now showing the other control as the navigation system of the web site.


Home
Reviews
    By Genre
    All Reviews
About
    Contact
    About Me
Login

Hi there visitor! Welcome to my Web Site :-)

I'm glad that you are visiting my web site http://www.mywebsite.com

***Feel free** to have a look around.*

You can log in here.

Select a Theme: DarkGrey

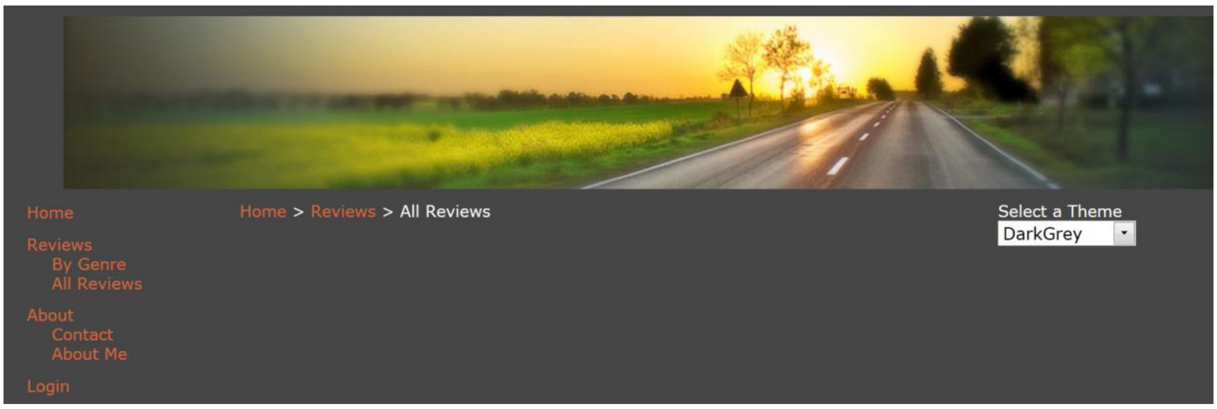
Creating a Breadcrumb with the SiteMapPath Control

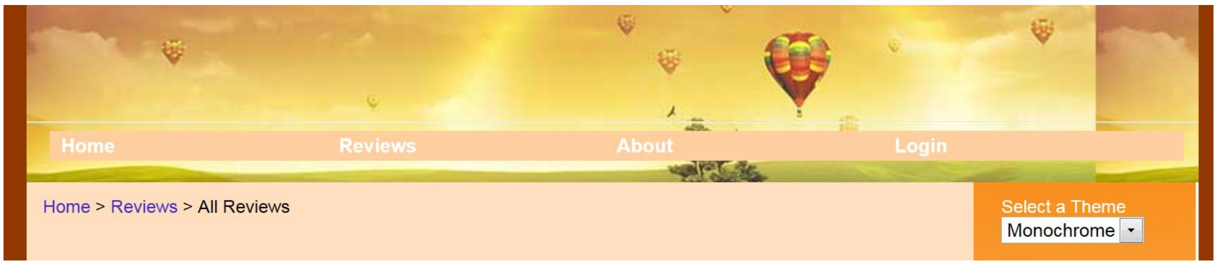
A good location for the `SiteMapPath` is in the global master page of the site. That way it becomes visible in all your pages automatically.

1. Open the master page in Source View and locate the opening tag of the `MainContent` element. Right after that tag, and before the `<asp:ContentPlaceHolder>` tag, drag a `SiteMapPath` from the Toolbox. Right after the `SiteMapPath` add two line breaks (`<br />`). You should end up with code like this:

```
<div id="MainContent">
    <asp:SiteMapPath ID="SiteMapPath1" runat="server">
    </asp:SiteMapPath><br /> <br />
    <asp:ContentPlaceHolder id="cpMainContent" runat="server">

    </asp:ContentPlaceHolder>

</div>
```

2. Save the changes and then request `Default.aspx` in the browser. Note that the page now shows the path from the root of the site (identified by the `Home` link) to the current page. Click a few of the items in the `Menu` or `TreeView` controls to navigate around the site and you'll see the breadcrumb change for each page. Figure shows the breadcrumb for the `All Reviews` page in Internet Explorer. The `All Reviews` page is a subelement of `Reviews`, which in turn falls under the `Home` root element.





When you navigate to one of the subpages, you can click the elements of the path to go up one or more levels. Clicking Reviews in the page takes you back to the main Reviews page, and clicking Home takes you back to the root of the site.

3. Using the Theme selector, switch to the other theme. Note that the SiteMapPath looks pretty much the same, except for the color of the links, which are defined in each of the themes' CSS file.

Redirecting the User to Another Page

This exercise shows you how to create a page that redirects from one page to another using Response.Redirect. The example uses a temporary redirect (the initial page remains accessible after the redirect), so the code uses Response.Redirect instead of Response.RedirectPermanent.

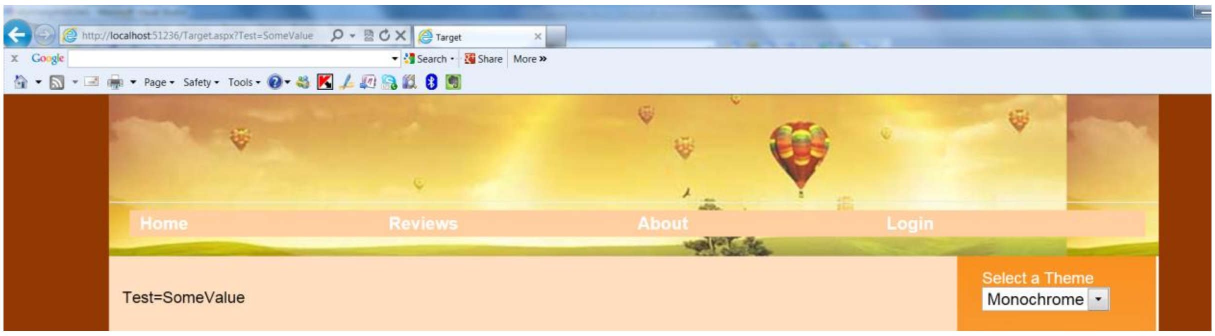
1. In the Demos folder, create two new Web Forms based on your custom template. Call them Source.aspx and Target.aspx. Set their Title to Source and Target, respectively.
2. Open Source.aspx in Design View and double-click somewhere in the grey, read-only area of the page outside the ContentPlaceHolder to set up a Page_Load handler. Inside this handler write the following code that redirects the user to the Target.aspx page. To show you how to pass additional data through the query string and how to read that information in the target page, the code passes a query string field called Test with SomeValue as the value:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Redirect("Target.aspx?Test=SomeValue");
}
```

3. Open `Target.aspx`, switch to Design View, and add a `Label` control to the `cpMainContent` placeholder. Leave its `ID` set to `Label1`. Set up a `Page_Load` handler similar to the one you created in the previous step by double-clicking the grey, read-only area of the page and then add the following code:

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = Request.QueryString.ToString();
}
```

4. Save all your changes, go back to `Source.aspx` and press Ctrl+F5 to open it in the browser. Instead of seeing `Source.aspx`, you now see the page depicted in the figure:



Test=SomeValue

Note that the address bar now reads `Target.aspx?Test=SomeValue`, the page you redirected to in the `Page_Load` event handler of the source page. The `Label` in the target page shows the query string that is passed to this page. Notice that `QueryString.ToString()` only contains `Test=SomeValue`. The address or even the question mark is not a part of the query string for the page.

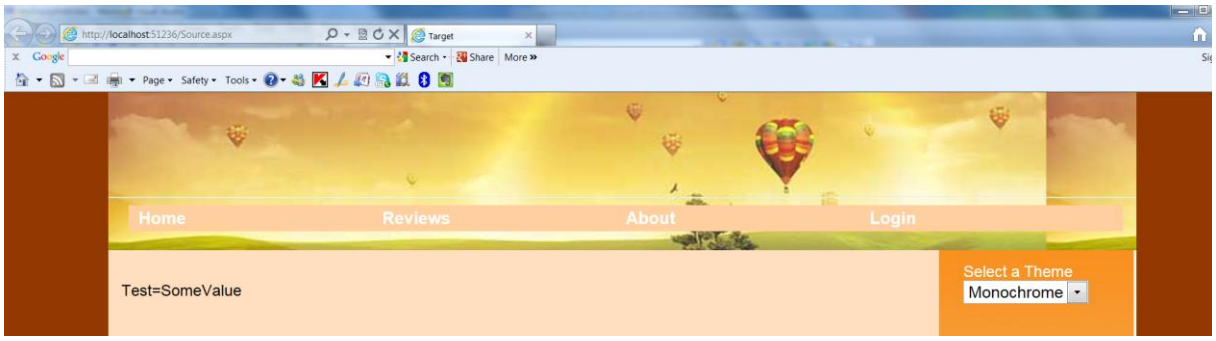
Server-Side Redirecting

It's easy to change the redirect code so it transfers the user to another page. All you need to do is replace `Response.Redirect` with `Server.Transfer` as demonstrated in this exercise.

1. Open the Code Behind of `Source.aspx` and replace the line with `Response.Redirect` with the following line:

```
protected void Page_Load(object sender, EventArgs e)
{
    Server.Transfer("Target.aspx?Test=SomeValue");
}
```

2. Save the changes and then press Ctrl+F5 to open `Source.aspx` in the browser.



Test=SomeValue

The `Label` control displays the query string values that were sent from `Source.aspx` to `Target.aspx`, demonstrating the fact that you are really viewing the output of the `Target.aspx` page. However, the browser's address bar is left unmodified and still shows `Source.aspx`, hiding the new page name and query string values from the user.