# Quadrant II – Transcript and Related Materials

**Programme: B.Sc.**

**Subject: Computer Science**

**Paper Code: CSS 105**

**Paper Title: Web Application Development Using ASP.NET**

**Unit: VI**

**Module Name: Site Navigation Controls: Treeview, Menu, SiteMapPath; Master Pages; Web Configuration File 'web.config', 'web.sitemap'**

**Module No: 20**

**Name of the Presenter: Mrs. Liana da Costa**

**Notes**

## Navigation Controls in ASP.NET

- TreeView
- Menu Control
- SiteMapPath

## TreeView

- display the data in hierarchical list manner.
- when TreeView is displayed for the first time, it displays all of its nodes.
- user can control it by setting the property called ExpandDepth.
- used for displaying the files and folders on the webpage.

## Properties of TreeView Control

- **DataSourceID:** specify the data source to be used using sitemap files data source.
- **ShowLines:** specify the lines to connect the individual item in the tree.
- **CssClass:** specify the CSS class attribute for the control.
- **ExpandDepth:** specify the level at which items in the tree are expanded.

## Menu

- used to display menu in web page.
- used in combination with SiteMapDataSource control for navigating the web Site.
- used to display the site data structure vertically and horizontally.
- displays two types of menu static menu and dynamic menu.

**Static menu** is always displayed in menu Control, by default only menu items at the root levels are displayed, used to display the parent menu items and their sub menu items on the page.

**Dynamic menu** is displayed only when the use moves the mouse pointer over the parent menu that contains a dynamic sub menu.

**Properties of Menu Control**

- **DataSourceID:** specify the data source to be used using sitemap file as data source.
- **CssClass:** specify the CSS class attribute for the control.
- **ImgeUrl:** specify the image that appear next to the menu item.
- **Orientation:** specify the alignment of menu control, can be horizontal or vertical.
- **Tooltip:** specify the tooltip of the menu item when you mouse over.
- **Text:** specify the text to display in the menu.
- **NavigateUrl:** specify the target location to send the user when menu item is clicked.
- **Target:** specify the target page location, can be in new window or same window.
- **Value:** specify the unique id to use in server side events.

**SiteMapPath**

**Site maps**

- are XML files which are mainly used to describe the logical structure of the web application.
- defines the layout of all pages in web application and how they relate to each other.

- Whenever you want you can add or remove pages to your site map there by managing navigation of website efficiently.
- Site map files are defined with .sitemap extension.
- <sitemap> element is the root node of the sitemap file.

  **Attributes:**

  **Title:** provides textual description of the link.

  **URL:** provides the location of the valid physical file.

  **Description:** used for tooltip of the link.

**SiteMapPath control**

- displays the navigation path of the current page.
- the path acts as click able links to previous page.
- uses the web.
- creates the navigation mechanism which is linear path defining where the user is currently located in navigation arrangement.
- display the current page's context within the entire structure of a website.
- helps end user to know his location in relation to the rest of the site.

**Properties of SiteMapPath Control:**

**PathSeparator:** get or set the out separator text.

**NodeStyle:** set the style of all nodes that will be displayed.

**RootNodeStyle:** set the style on the absolute root node.

**PathDirection:** set the direction of the links generated in the output.

**CurrentNodeStyle:** set the style on node that represent the current page.

**ShowToolTips:** set the tooltip for the control. Default value is true.

**PathSeparatorStyle:** set the style of path separator.

## Master Pages

A **master page** is an ASP.NET file with the extension **.master** (for example, MySite.master) with a predefined layout that can include static text, HTML elements, and server controls. The master page is identified by a special @ **Master** directive that replaces the @ **Page** directive that is used for ordinary .aspx pages.

- allow you to create a consistent look and behaviour for all the pages (or group of pages) in your web application.
- provides a template for other pages, with shared layout and functionality.
- defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page.
- every master page has one or more content pages in an application.

**Advantages of Master Pages**

- provide an object model allowing users to customize the master page from the individual content pages.
- allows user design the rendering of the controls in the placeholder.
- centralized with common functionality of all pages to makes updates in one place.
- code can be applied on one set of controls and the results to the set of pages in the application.  For example, you can use controls on the master page to create a menu that applies to all pages.
- Creating a set of controls that are common across all the web pages and attaching them to all the web pages.
- A centralized way to change the above created set of controls which will effectively change all the web pages.
- They give you fine-grained control over the layout of the final page by allowing you to control how the placeholder controls are rendered.
- Dynamically changing the common UI elements on master page from content pages based on user preferences.

## Master Pages - Terminology

- Masterpage: Gives us a way to create common set of UI elements that are required on multiple pages of our website.
- ContentPage: The ASP.NET web page that will use master page to have the common UI elements displayed on rendering itself.
- ContentPlaceHolder: A control that should be added on the MasterPage which will reserve the area for the content pages to render their contents.
- ContentControl: A control which will be added on content pages to tell these pages that the contents inside this control will be rendered where the MasterPage's ContentPlaceHolder is located.

## Run-time Behaviour of Master Pages

1. Users request a page by typing the URL of the content page.
2. When the page is fetched, the @ Page directive is read. If the directive references a master page, the master page is read as well. If this is the first time the pages have been requested, both pages are compiled.
3. The master page with the updated content is merged into the control tree of the content page.
4. The content of individual Content controls is merged into the corresponding ContentPlaceHolder control in the master page.
5. The resulting merged page is rendered to the browser.

**Web Configuration File ‘web.config’**

- A configuration file (web.config) is used to manage various settings that define a website, which deal with a single application
- The settings are stored in XML files that are separate from your application code.
- Generally a website contains a single Web.config file stored inside the application root directory. However there can be many configuration files that manage settings at various levels within an application.
- Benefits of XML-based Configuration files
- ASP.NET Configuration system is extensible and application specific information can be stored and retrieved easily. It is human readable.
- You need not restart the web server when the settings are changed in configuration file. ASP.NET automatically detects the changes and applies them to the running ASP.NET application.
- You can use any standard text editor or XML parser to create and edit ASP.NET configuration files.

**What web.config file contains?**

- Database connections
- Caching settings
- Session States
- Error Handling
- Security

<configuration>

In a Web.config file, all the configuration information for an ASP.NET application must reside between the **<configuration> and </configuration> tags**. This is the root node, which contains the declaration of all other sections of the Web.config file.

<appSettings>

This section of the Web.config file provides a way to define custom application settings for an application. The section can have multiple <add> subelements.

```
<appSettings>
    <add key="connectionstring"
        value="localhost;uid=readonly;pwd=user"/>
</appSettings>
```

<compilation>

This element sets several compilation settings for the application. Some of the settings involve setting a default language and debug option for the application setting. Debug pages are larger and execute more slowly, so you should use them only for testing purposes. This section also provides support for the **<compilers>, <assemblies> , and <namespaces> subelements**.

<namespaces>

This subelement is used to add or remove namespace references for assemblies that must be made available when compiling Web pages.

<authentication>

The <authentication> element sets the authentication policy for the application. Possible modes are "Windows," "Forms," "Passport," and "None."

<pages>

The <pages> element allows configuration of page application specific settings.

<customErrors>

The <customErrors> element provides a means for defining custom error messages for an ASP.NET application. This is generally used to point users to a friendlier message than the default error messages. The <customErrors> element section supports multiple <error> subelements that are used to define custom errors.

**Contents of web.config file**

**Example 1:**

```
<configuration>
    <appSettings>
      <add key="Application Name" value="MyApplication" />
    </appSettings>
    <connectionStrings>
```

```
        <add name="myCon"

connectionString="server=MyServer;database=gcq;uid=gcqadmin;pwd=
gcqpwd" />
      </connectionStrings>
   </configuration/>
```

**Example 1:**

```
<?xml version="1.0" encoding="utf-8" ?>

   <configuration>
       <system.web>
       </system.web>
  </configuration>
```

**Processing of a web.config file**

- When you initially run your web application, the runtime builds a cache of the configuration settings for your web application by flattening the layer of configuration files as below,
- The Machine.config file settings are retrieved.
- The settings from the root Web.config files are added to the caches, overwriting any conflicting settings that were earlier while reading the Machine.config file.
- If there is a Web.config file at the root of the website, this file is read into the cache, all overwriting any existing settings. The resulting cache contains the setting for this website.
- If you have subdirectories in your web application, each subdirectory can have a Web.config file that includes settings that are specific to the files and folders that are contained within the subdirectory. To calculate the effective setting for the folders, the website settings are read (step 1-4) and then this Web.config file is

read into cache for this folder, overwriting (and thereby overriding) any existing settings.

**Web Configuration File ‘web.sitemap’**

- By default, ASP.NET site navigation works with an XML file that is named Web.sitemap that describes the hierarchy of the Web site. However, you might want to use more than one site-map file or site-map provider to describe the navigation structure of a complete Web site.
- sitemap, or XML sitemap, is a list of different pages on a website.
- sitemap will tell search engines the location of a page on your website, when it was updated, the updating frequency, and the importance of the page as it’s related to other pages on your site.

A sitemap is a file where you provide information about the pages, videos, and other files on your site, and the relationships between them. Search engines like Google read this file to crawl your site more efficiently. A sitemap tells Google which pages and files you think are important in your site, and also provides valuable information about these files. For example, when the page was last updated and any alternate language versions of the page.

A sitemap video entry can specify the video running time, category, and age-appropriateness rating.

A sitemap image entry can include the image subject matter, type, and license.

A sitemap news entry can include the article title and publication date.

```
<siteMap defaultProvider="MyXmlSiteMapProvider"
enabled="true">
  <providers>
    <add name="MyXmlSiteMapProvider"
      description="The site map provider that reads in the .sitemap
XML files."
      type="System.Web.XmlSiteMapProvider, System.Web,
Version=2.0.3600.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
      siteMapFile="my.sitemap" />
  </providers>
</siteMap>
```

**You might need a sitemap if:**

- site is really large.
- site has a large archive of content pages that are isolated or not well linked to each other.
- site is new and has few external links to it.
- site has a lot of rich media content (video, images) or is shown in Google News.

**You might not need a sitemap if:**

- site is "small" (e.g. about 500 pages or fewer)
- site is comprehensively linked internally.
- don't have many media files (video, image) or news pages that you want to show in search results.