Asp.NET Application Configuration and Deployment of Application

UNIT - 9

Overview of ASP.NET Configuration

- ASP.NET Configuration is made up in XMML file format because XML is platform independent so that it can applied on any type of machine, platform etc.
- ASP.NET Configuration allows you to define configuration settings at the time of creating your web application for the first time.
- You can also add configuration even after your web application is created.
- Advantages:
  - It can be stored and retrieved easily in any platform , hardware, OS, etc.,
  - You do not need to restart your web server even you have changed some configuration settings. ASP.NET configures your changes automatically and applies to your Web Application.
  - Configuration file is in XML format so that can be opened with any simple text editor.

Configuration Files

- There are two types of configuration files which are used to configure either all web applications stored in your machine or to configure individual web application.
  - Machine Config
  - Web Config

Web.Config

- Each and every web application has its own Web.Config file which can configure entire web application or a particular web page of your web application.
- Initially Web.Config file is generated from Machine.Config file.
- Initial Web.Config file is stored in folder your machine.
  - C:\<Windows>\Microsoft.NET\Framework\<version>\Config
- Then a copy of Web.Config is kept to your web application folder.
- If your web application has multiple folders, each folder can have own different set of Web.Config file, which are inherited from parent Web.Config file in root folder of your web application.
- Before using Web.Config you should have through knowledge of Web.Config, so that you can apply different settings in different sections of Web.Config file.
- Web.Config file is XML based file, All XML files can have only one root element. The root element of Web.Config file is <configuration> element. Your entire configuration settings and all other sub elements are inside <configuration> as follows:

Web.Config

- <?xml version="1.0">

  <configuration>

  all other configration sections wil be place here

  </configuration>

- <configuration> section consist of many other sub elements :
  - <configSections> : This allows you to configure different sections of web application like java script, web services, resources etc,
  - <appSettinfgs> : This allows you to add any application related variable or settings. You can <add>, <remove> or <clear> any application specific variable or settings.
  - <connectionStrings> : This allows you to create connectionString variables which can be used in any Web Form of your web application.
  - <system.web> : This is probably important element used to configure your web application. This section allows you to do most of configuration as follows:

    <compilation> <authentication>

    <caching> <customErrors>

    <deployment> <roleManager>

    <trace> <sessionState>

    <webServices>

Web.Config

- <system.codedom> : This section stores compiler specific information used for your web application under <compilers> element.
- <system.webServer> : This section stores information and configuration settings for web servers which will be used for your deploying and running your web application. This is used when you have created “File System” web application which uses ASP.NET Development Server.
- <runtime> : This section is used to specify some run time settings for your web application. Run Time settings can include no. of settings like connection strings, tracing , data providers, etc.

Common Configuration Settings

- Tracing Web Application
- Customizing Errors (Redirecting Errors to Customized Pages)
- Authentication And Authorization
- Enabling Role Manager
- Session Configuration
- Trust Levels
- Web service Configuration
- Caching

Tracing Web Application

- Tracing allows you to check for any type of bugs that your web application may have.
- Along with tracing your web application, you can also maintain log files which give you some additional information at alter stage.
- These log files can help you to find out some problems that occur during web application usage.

Customizing Errors

- While using web site, sometimes different errors occur.
- You are not aware that which error is going to come.
- Web site users feel uncomertable when they get some unexpected screen.
- Many web sites may generate error. But if you customizing errors, you can redirect of these errors, users wont feel uncomfortable.
- By customizing errors, you can redirect your error to specific page which can display error but in your design specific way.

Authentication and Authorization

- This is probably most used settings for any type of website.
- Authorized users are requirement for most websites.
- To allow authenticated users, you must authorize users in a such a way that any even any unauthorized users visit your website, should not be able to harm your information.
- This can be done with the help of combing authentication and authorization

Enabling Role Manager

- Enabling Role Manager is a feature which is used along with Authentication and authorization.
- You can use this feature to allow some specific type of users.
- For example,

  some of the web page your website are important ,.
- you want only “Administrator” level user can access those.
- “Guest” users should not be able to access these pages.
- You can do it with help of Role Manager.

Session Configuration

- You can configure session related settings under them.
- Under session configuration, you can specify session settings like Time out period, session mode, cookie based session or cookieless session , sqlcommandtimeout , networktimeout etc.,
- Many of these sessions based configuration can be done at different levels of session handling.

Trust Levels

- there can be many different types of website users.
- For example, web master can have access to all pages of website, web developer will have some limited access to web pages of website, administrator will have access to all the back end pages for website administration where at last level is end users who will have access to only those pages which are informative,
- So different types of trust levels can be defined using Trust Levels.

Web Service Configuration

- You have already run how to create and consume web service
- Web services can also be configured.
- You can co nfigure web servkces to lock some resources from being accessed by some specific users.

Caching

- Caching means keeping your output or data into cache so that it can be accessed in faster way instead of gathering output from web server or searching data from database
- Caching is the new concept introduced since ASP.NET which allows you to cache output or data depending on requirement.

Tracing

- To trace means to Monitor.
- Tracing is a way to monitor how your Web Application executes. When you execute your website or any particular page within the web sites, you cant detect the actual source if any problem.
- Sometimes, you cant catch from where the problem begins. You need to trace your website or webpage in some circumstances.
- With the help of tracing you can record some of the execution and your program flow so that it does not even affect output of your web page.
- Tracing us important to test your website in order to give perfect output as per developers requirement which fulfills users need.
- Tracing can be done at two levels :

  Page Level Tracing

  Application Level

Page Level Tracing

- Page level tracing is useful when you want to trace only selected web pages from your website.
- Page level tracing allows you to trace a particular selected page.
- You can enable or disable Page Level Tracing with the help of “Trace” attribute in <%@Page ... %>.
- Tracing can be enabled at page level by writing following line :
  - <%@page Trace=“true” TraceMode=“SortByCategory” Language=“C#” CodeFile=“default.cs” Inherits=“default” %>
- Trace = “true” : It will enable tracing at page level. By default trace is disabled for any web page.
- TraceMode=“SortByCategory | SortByTime” : It display information based on how trace information is written. You can write trace information using Trace.Write() or Trace.Warn() method.

Application Level Tracing

- Application Level Tracing automatically enables tracing for all the web pages within your web application.
- When you enable tracing for entire application, you do not need to enable individual page level tracing.
- To enable tracing for entire application in web.config, we need to specify inside <configuration><system.web> elements of web.config file as follows:

```
<configuration>
  <system.web>
        <trace enable="true" requestLimit="20" localonly="false"
traceMode="SortByTime" pageOutput="true"/>
  </system.web>
</configuration>
```

- Enable : "true" specifies that tracing is enabled .

Application Level Tracing

- When you enable tracing for entire application, ASP.NET gives you information about each and every request that occurs within web application.By setting requestLimit="20" , it will collect no. of request and willl be displayed on each page.
- By default tracing visible only when you are using Local Web Server .When you are working on remote based web server, you need to specify localonly="false". It means even though you are working on remote client , application level tracing information should be visible in client pc.
- pageOutput="true" specifies that along with page output you want to see tracing information. If you specify pageOutput="false", by default traced information is not visible on the screen. You need to go for "Trace Viewer" by writing "Trace.axd" file on the current url of browser.

Error Handling

- Error handling is one of the most important part of any web application.
- Each error has to be caught and suitable action to be taken to resolve that problem.
- Normally we try to trap errors using try catch finally block. But what happens in the case of an unhandled exception in application.
- ASP.NET provides a simple yet powerful way to deal with errors that occur in your web application.
- ASP.NET provides three main methods that allow us to trap and respond to errors when they occur.
  - Page_Error() event of individual .aspx file
  - Application_Error() event in Global.asax file
  - <customErrors> section of the application configuration files(web.config)

Custom Errors

- <customErrors> is used to deal with different types of standard errors.
- Web.cofig file gives you <customErrors> element, by configuring which you can define your own "Custom Error Message" for ASP.NET web application.
- **Showing your own customized error message** so that user's can understand it as it can be written in easy language.
- **Showing all the error messages in common format**, so user's don't feel that they are thrown out of the scope due to some error.
- In the <customErrors> section we can specify a redirect pafe as default error page or specify to particular page based on the HTTP error code that is raised.

Custom Errors

- <customErrors> section in web.config has two attributes that affect what error shown:
- **defaultRedirect** : the defaultRedirect attribute is optional. If provided , it specifies the URL of the custom error page and indicates that the custom error page should be shown instead of the Runtime Error.
- **mode**: the mode attributeis required and accepts one of three values : on, off or RemoteOnly.
  - On : this option specifies that custom errors are enabled. If no default redirect is specified , users see a usual error.
  - Off : this option specifies that custom errors are disabled. This allows the display of detailed errors.
  - RemoteOnly: this option specifies that custom errors are shown only to remote clients and ASP.NET errors are shown to local host. This is the default.