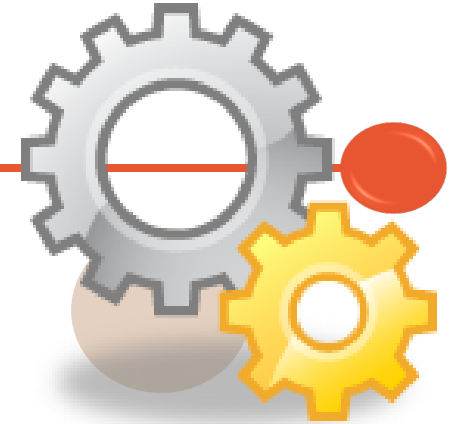




**TARUMT**  
TUNKU ABDUL RAHMAN  
UNIVERSITY OF  
MANAGEMENT AND TECHNOLOGY

**MDEC**<sup>TM</sup>  
Premier Digital  
Tech Institution



# Configuration and Optimization

## Chapter 10

# What Are You Going To Learn?

---

- At the end of this lesson, you will be able to:
  - Differentiate two types of configuration files (machine.config and web.config).
  - Explain how configuration files work.
  - Customize machine.config and web.config files.
    - Database connection (LocalSqlServer)
    - Session States
    - Error handling settings
    - Debugging routines (debug=false)
    - General Configuration

# Overview of Configuration

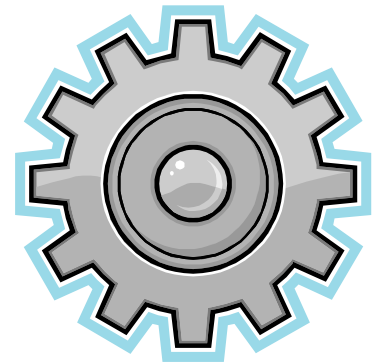
---

- ASP.NET has made configuration even more powerful and flexible by adopting XML-based configuration files.
- The config files are used to configure any component of ASP.NET, by writing your own code of explaining how you like ASP.NET to perform certain operation.

# Types of Configuration File

---

- machine.config
- web.config





# The configuration files

## machine.config

- Contains machine-specific settings that ASP.NET needs to function
- used to configure the application according to a particular machine

## web.config

- Stores configuration info for a specific Web application;
- Can override default functionality defined in the machine.config.
- Contains the subset of the settings in machine.config

# machine.config

## Location:

- For Visual Studio 2017 (.NET Framework 4.0 and above)
  - %SystemRoot%\ Microsoft.NET\Framework\v4.0.30319\Config\machine.config



Do not rename the original machine.config – ASP.NET relies on this file for its configuration

# How the Runtime Locates Configuration Files

How the CLR locates and binds to the assemblies that make up your application

Page is initialized

Information in machine.config is read

ASP.NET reads the individual web.config files stored in the application root directory

ASP.NET reads the individual web.config files stored in the child directories

Iteration continues until all web.config files in the tree have been processed

# What if...

---

- The subdirectory does not contain a web.config file?
  - The pages within this subdirectory will inherit their settings from the parent file node
- I have only ONE page that needs special settings to function?
  - Place this page in a subdirectory with its own web.config file
  - The changes will then just affect that page but not the whole application



# Configuration File Rules

---

1. They must have a root element `<configuration>`.
2. Elements must be enclosed between correspondent tag `<>`, that must be closed and they are case-sensitive.
3. Attributes must be enclosed in `"`, e.g `<add name="ConString" />`
4. Elements must be nested and not overlap.

# The Format

---

- The configuration files are structurally divided into 2 main areas:
  1. Declarations:
    - Individual classes are defined to manipulate information
    - This section is delimited by <configSections>
  - Settings:
    - Values are assigned to the classes declared in the first section
    - This section is delimited by <sectionGroup>

# Practical Config Example

## To change the mail server's SMTP

```
<system.net>  
  <mailSettings>  
    <smtp>  
      <network host="smtp.somehost.net" />  
    </smtp>  
  </mailSettings>  
</system.net>
```

**Can be used to change other ASP components settings.**

# Structure of machine.config

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
+ <configSections>
+ <configProtectedData
  defaultProvider="RsaProtectedConfigurationProvider">
  <runtime />
+ <connectionStrings>
+ <system.data>
+ <system.web>
+ <system.serviceModel>
</configuration>
```

# Structure of web.config

```
<?xml version="1.0" ?>  
+ <configuration>  
+ <configSections>  
  <appSettings />  
  <connectionStrings />  
+ <system.web>  
+ <system.codedom>  
+ <system.webServer>  
+ <runtime>  
  </configuration>
```

# The Settings of system.web

The most frequently used configurations:

- ✓ Database connection
- ✓ Session States
- ✓ Error handling settings
- ✓ General Configuration
- ✓ Debugging routines
  - ✓ Security (covered in Chapter 5)
    - Authentication Technique
    - Role Manager settings (on or off?)
    - Anonymous Identification settings (permitted or not)
- ✓ E-mail settings for the Simplified Mail Transfer Protocol (SMTP) (not in your syllabus)

# What can be stored in Web.config file: Database Connections

---

- To store **database connection string**.
  - ✓ any modifications to the database configurations can be maintained at a single location.
  - ✓ It can be read and used anywhere in the program.

# Example

```
<connectionStrings>  
  <add name="Northwind"  
    connectionString="Server=localhost;  
    Integrated Security=True;Database=Northwind"  
    providerName="System.Data.SqlClient" />  
</connectionStrings>
```

all the configuration information was stored in  
human-readable, clear-text format



# Encrypting the <connectionString> section

- Run this code if you host the website in IIS

```
aspnet_regiis -pe "connectionStrings" -app "/MyApplication"
```

- Run this code for the physical path where you store the web.config

```
aspnet_regiis -pef "connectionStrings" "path"
```

- E.g.

```
aspnet_regiis -pef "connectionStrings" "C:\Dev\SampleWebApp
```

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727>aspnet_regiis -pef connectionStrings E:\Prac10
Encrypting configuration section...
Succeeded!
```

# Encrypting the `<connectionString>` section

- Running this bit of script will encrypt the connection string in `web.config`
- the `-pe` command specifies the section in the `web.config` file to encrypt,
- the `-app` command specifies which application to actually work with in IIS.

**See Demo 2 to see the result**

# To Decrypt

- Run this code if you host the website in IIS

```
aspnet_regiis -pd "connectionStrings" -app "/MyApplication"
```

- Run this code for the physical path where you store the web.config

```
aspnet_regiis -pdf "connectionStrings" "path"
```

- E.g.

```
aspnet_regiis -pdf "connectionStrings" "C:\Dev\SampleWebApp"
```

# What can be stored in Web.config file:


## Error Handling

- Allows us to configure what to display when an error occurs in our application.
- To enable us to provide a friendly message (rather than the complete exception information), by redirecting the user to a particular custom error page based on different types of error.


# What can be stored in Web.config file: Error Handling (cont)

- Sample Code:

```
<customErrors mode="RemoteOnly"
  defaultRedirect="GenericErrorPage.htm">
  <error statusCode="403" redirect="NoAccess.htm" />
  <error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```



HTTP Status  
Code



Custom page  
that created by  
programmer

**See Demo 3 ErrorHandling**

# What can be stored in Web.config file: Error Handling (cont)

## Mode

- = “On”
  - Show custom error to everyone when error occurs. All callers receive filtered exception information.
- = “Off”
  - Never shows a custom error to anyone. All callers receive complete exception information.
- = “RemoteOnly” [default mode]
  - Shows your custom error to browsers that are not located on your server. Local callers receive complete exception information; remote callers receive filtered exception information.

# What can be stored in Web.config file:

## Debugging routines

- Debugging routine should be
  - turned on at the time of compilation.
    - To provide output to the page describing any problems that were found during the build of the page
  - Turned off prior to deployment

```
<!-- to turn on-->  
<system.web>  
  <compilation debug="true">  
  </compilation>
```

# Configuring ASP.NET Runtime Settings

`<configuration>`

`<system.web>`

`<httpRuntime`

`useFullyQualifiedRedirectUrl="false"`

`enable="true"`

`executionTimeout="110"`

`maxRequestLength="4096" />`

`</system.web>`

`</configuration>`

Some mobile devices require fully qualified URLs

Enabling and Disabling ASP.NET Applications

amount of time in seconds which a resource can execute

maximum file-size upload, in KB



# Notes

---

- Always backup the config files before modifying them.
- Do not change anything in machine.config. Changes to this file will affect all web applications on your computer. Do so only if you know what you are doing.

# Performance Optimization

---

- Cache (covered in Chapter 6)
  - Output Cache
    - when output caching is used, ASP.NET is doing less work to satisfy a request, thus allowing the response to be served faster.
  - Fragment Cache
    - When it is not practical to cache the entire page, but some portions of the page can be cache.
    - It is worthwhile to create user controls that do not change for a defined time period.



Next

# DEPLOYING ASP.NET APPLICATIONS