**Introduction to ASP.NET Web Configuration**





[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

Contents

[Introduction 3](#_Toc223527570)

[Machine.config 3](#_Toc223527571)

[Web.config 4](#_Toc223527572)

[Configuration Schema 6](#_Toc223527573)

[Browser Capabilities 9](#_Toc223527574)

## Introduction

The ASP.NET configuration system features an extensible infrastructure that enables you to define configuration settings at the time your ASP.NET applications are first deployed so that you can add or revise configuration settings at any time with minimal impact on operational Web applications and servers.

The ASP.NET configuration system provides the following benefits:

* Configuration information is stored in XML-based text files. You can use any standard text editor or XML parser to create and edit ASP.NET configuration files.
* Multiple configuration files, all named Web.config, can appear in multiple directories on an ASP.NET Web application server. Each Web.config file applies configuration settings to its own directory and all child directories below it. Configuration files in child directories can supply configuration information in addition to that inherited from parent directories, and the child directory configuration settings can override or modify settings defined in parent directories. The root configuration file named *systemroot*\Microsoft.NET\Framework\*versionNumber*\CONFIG\Machine.config provides ASP.NET configuration settings for the entire Web server.
* At run time, ASP.NET uses the configuration information provided by the Web.config files in a hierarchical virtual directory structure to compute a collection of configuration settings for each unique URL resource. The resulting configuration settings are then cached for all subsequent requests to a resource. Note that inheritance is defined by the incoming request path (the URL), not the file system paths to the resources on disk (the physical paths).
* ASP.NET detects changes to configuration files and automatically applies new configuration settings to Web resources affected by the changes. The server does not have to be rebooted for the changes to take effect. Hierarchical configuration settings are automatically recalculated and reached whenever a configuration file in the hierarchy is changed. The <processModel> section is an exception.
* The ASP.NET configuration system is extensible. You can define new configuration parameters and write configuration section handlers to process them.
* ASP.NET help protect configuration files from outside access by configuring Internet Information Services (IIS) to prevent direct browser access to configuration files. HTTP access error 403 (forbidden) is returned to any browser attempting to request a configuration file directly.

### Machine.config

The machine configuration file, Machine.config, contains settings that apply to an entire computer. This file is located in the %runtime install path%\Config directory. Machine.config contains configuration settings for machine-wide assembly binding and ASP.NET.

The configuration system first looks in the machine configuration file for the <appSettings> element and other configuration sections that a developer might define. It then looks in the application configuration file. To keep the machine configuration file manageable, it is best to put these settings in the application configuration file. However, putting the settings in the machine configuration file can make your system more maintainable. For example, if you have a third-party component that both your client and server application uses, it is easier to put the settings for that component in one place. In this case, the machine configuration file is the appropriate place for the settings, so you don't have the same settings in two different files.

### Web.config

Using the features of the ASP.NET configuration system, you can configure all of the ASP.NET applications on an entire server, a single ASP.NET application, or individual pages or application subdirectories. You can configure features, such as authentication modes, page caching, compiler options, custom errors, debug and trace options, and much more.

ASP.NET configuration data is stored in XML text files that are each named Web.config. Web.config files can appear in multiple directories in ASP.NET applications. These files allow you to easily edit configuration data before, during, or after applications are deployed on the server. You can create and edit ASP.NET configuration files by using standard text editors, the ASP.NET MMC snap-in, the Web Site Administration Tool, or the ASP.NET configuration API.

ASP.NET configuration files keep application configuration settings separate from application code. Keeping configuration data separate from code makes it easy for you to associate settings with applications, change settings as needed after deploying an application, and extend the configuration schema.

Each Web.config file applies configuration settings to the directory that it is in and to all of the child directories below it. Settings in child directories can optionally override or modify settings that are specified in parent directories. Configuration settings in a Web.config file can optionally be applied to individual files or subdirectories by specifying a path in a location element.

The root of the ASP.NET configuration hierarchy is the systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Web.config file, which includes settings that apply to all ASP.NET applications that run a specific version of the .NET Framework. Because each ASP.NET application inherits default configuration settings from the root Web.config file, you need to create Web.config files only for settings that override the default settings.

At run time, ASP.NET uses the Web.config files to hierarchically compute a unique collection of configuration settings for each incoming URL request. These settings are calculated only once and then cached on the server. ASP.NET detects any changes to the configuration files and then automatically applies those changes to the affected applications, restarting the applications in most cases. Hierarchical configuration settings are automatically calculated and cached again whenever a configuration file in the hierarchy is changed. The IIS server does not have to be restarted for the changes to take effect unless the processModel section has been changed.

##### Web Site Administration Tool

The Web Site Administration Tool allows anyone with administrative privileges for the Web site to manage the configuration settings for that Web site. The Web Site Administration Tool is designed to provide a user-friendly, graphical editing tool for the configuration settings that are most commonly used in individual Web sites. Because the tool uses a browser-based interface, it allows you to change Web site settings remotely, which is useful for administering a site that is already deployed to a production Web server, such as a hosted Web site.

The Web Site Administration Tool differs from the ASP.NET MMC snap-in in several ways. For instance, the ASP.NET MMC snap-in is ideal for administrator-level configuration because it provides access to the entire hierarchy of configuration files on the Web server instead of the configuration settings for a single Web site. Also, you must be an administrator to use the ASP.NET MMC snap-in, whereas the Web Site Administration Tool only allows individual Web site owners to configure the Web.config file in the root directory of sites to which they have administrative privileges. Finally, you cannot use the ASP.NET MMC snap-in to administer IIS remotely, but the browser interface for the Web Site Administration Tool allows remote configuration for versions of IIS beginning with IIS 6.0.

The Web Site Administration Tool includes a tabbed interface that groups related configuration settings on the following tabs:

* A Security tab, which contains settings to help secure Web-application resources and to manage user accounts and roles.
* A Profile tab, which contains settings to manage how visitor information is gathered by the Web site.
* An Application tab, which contains settings to manage configuration elements that affect ASP.NET applications.
* A Provider tab, which contains settings to add, edit, delete, test, or assign application providers.

The Web Site Administration Tool is automatically installed with the .NET Framework version 2.0 and later.

##### Command-line Tools

The .NET Framework includes command-line tools that perform specific configuration operations. For example, the Aspnet\_regiis.exe tool allows you to specify which version of the .NET Framework applies to your ASP.NET application.

The ASP.NET configuration system provides a complete managed interface for programmatically configuring ASP.NET applications without directly editing the XML configuration files. In addition, the ASP.NET configuration API does the following:

* Simplifies administrative tasks by providing an integrated view of data from all levels of the configuration hierarchy.
* Supports deployment tasks, including creating configurations and configuring multiple computers with one script.
* Provides a single programming interface for developers who build ASP.NET applications, console applications and scripts, Web-based management tools, and MMC snap-ins.
* Prevents developers and administrators from making invalid configuration settings.
* Allows you to extend the configuration schema. You can define new configuration parameters and write configuration section handlers to process them.
* Provides static methods for obtaining configuration information from the application that is currently running, and non-static methods for obtaining configuration information from a separate application. Using static methods allows your application to run faster, but these methods can only be used from within the application about which you are obtaining configuration data.

###  Configuration Schema

Configuration information for ASP.NET resources is contained in a collection of configuration files, each named Web.config. Each configuration file contains a nested hierarchy of XML tags and subtags with attributes that specify the configuration settings. Because the tags must be well-formed XML, the tags, subtags, and attributes are case-sensitive. Tag names and attribute names are camel-cased, which means that the first character of a tag name is lowercase and the first letter of any subsequent concatenated words is uppercase. Attribute values are Pascal-case, which means that the first character is uppercase and the first letter of any subsequent concatenated words is uppercase. Exceptions are **true** and **false**, which are always lowercase.

All configuration information resides between the **<configuration>** and **</configuration>** root XML tags. Configuration information between the tags is grouped into two main areas: the configuration section handler declaration area and the configuration section settings area.

Configuration section handler declarations appear at the top of the configuration file between **<configSections>** and **</configSections>** tags. Each declaration contained in a **<section>** tag specifies the name of a section that provides a specific set of configuration data and the name of the .NET Framework class that processes configuration data in that section.

The configuration section settings area follows the **<configSections>** area andcontains the actual configuration settings. There is one configuration section for each declaration in the **<configSections>** area. Each configuration section contains subtags with attributes that contain the settings for that section.

The following Web.config file example declares two configuration **<section>** handlers. One manages application settings and the other manages session state.

<configuration>

 <configSections>

 <section name="appSettings"

 type="System.Configuration.NameValueFileSectionHandler,

 System, Version=1.0.3300.0,

 Culture=neutral, PublicKeyToken=b77a5c561934e089"/>

 <section name="sessionState"

 type="System.Web.SessionState.SessionStateSectionHandler,

 System.Web, Version=1.0.3300.0, Culture=neutral,

 PublicKeyToken=b03f5f7f11d50a3a"

 allowDefinition="MachineToApplication"/>

 </configSections>

 <appSettings>

 <add key="dsn" value="localhost;uid=MyUserName;pwd=;"/>

 <add key="msmqserver" value="server\myqueue"/>

 </appSettings>

 <sessionState cookieless="true" timeout="10"/>

</configuration>

You need to declare a configuration section handler only once. You can place it in the server's root Machine.config file or in a Web.config file in the virtual directory containing the Web application files. Configuration files in subdirectories automatically inherit configuration handlers declared in parent directories.

Configuration settings are often nested together under section grouping tags. These top-level section tags typically represent the namespace to which the configuration settings apply. For example, the top-level **<system.net>** tag represents the settings for network classes, and the **<system.web>** tag represents the settings for the ASP.NET classes.

The following example shows tag nesting.

<configuration>

 <configSections>

 <sectionGroup name="system.net">

 <section name="authenticationModules"

 type="System.Net.Configuration.NetAuthenticationModuleHandler,

 System, Version=1.0.3300.0, Culture=neutral,

 PublicKeyToken=b77a5c561934e089"/>

 <section name="webRequestModules"

 type="System.Net.Configuration.WebRequestModuleHandler,

 System, Version=1.0.3300.0, Culture=neutral,

 PublicKeyToken=b77a5c561934e089"/>

 </sectionGroup>

 <sectionGroup name="system.web">

 <section name="authorization"

 type="System.Web.Configuration.AuthorizationConfigHandler,

 System.Web, Version=1.0.3300.0, Culture=neutral,

 PublicKeyToken=b03f5f7f11d50a3a"/>

 <section name="sessionState"

 type="System.Web.SessionState.SessionStateSectionHandler,

 System.Web, Version=1.0.3300.0, Culture=neutral,

 PublicKeyToken=b03f5f7f11d50a3a"

 allowDefinition="MachineToApplication"/>

 </sectionGroup>

 </configSections>

 <system.net>

 <! — Net Class Settings would go here. -->

 </system.net>

 <system.web>

 <authorization>

 <allow users="\*"/> <!-- Allow all users -->

 <!-- Allow or deny specific users.

 allow users="[comma separated list of users]"

 roles="[comma separated list of roles]"/>

 <deny users="[comma separated list of users]"

 roles="[comma separated list of roles]"/>

 -->

 </authorization>

 <sessionState

 sqlConnectionString="data source=localhost;

 Integrated Security=SSPI;

 Initial Catalog=northwind"

 cookieless="false"

 timeout="10"/>

 </system.web>

</configuration>

The ASP.NET configuration infrastructure makes no assumptions about the types of configuration data that the infrastructure supports. Configuration section handler classes process all Web.config data. You can use the predefined configuration section handlers that are supplied with the .NET Framework or you can create your own handlers to process your own custom configuration data.

###  Browser Capabilities

Different browsers and different versions of the same browsers support different features. In your application, you might need to determine what type of browser a user is viewing pages with, and perhaps determine if the browser supports certain features.

Query the Browser property, which contains an HttpBrowserCapabilities object. This object gets information from the browser or client device during an HTTP request, telling your application the type and level of support the browser or client device offers. The object in turn exposes information about browser capabilities using strongly typed properties and a generic name-value dictionary.

The following code example shows how you can displays browser information the information in a text box on the page.

C#

private void Button1\_Click(object sender, System.EventArgs e)

{

 System.Web.HttpBrowserCapabilities browser = Request.Browser;

 string s = "Browser Capabilities\n"

 + "Type = " + browser.Type + "\n"

 + "Name = " + browser.Browser + "\n"

 + "Version = " + browser.Version + "\n"

 + "Major Version = " + browser.MajorVersion + "\n"

 + "Minor Version = " + browser.MinorVersion + "\n"

 + "Platform = " + browser.Platform + "\n"

 + "Is Beta = " + browser.Beta + "\n"

 + "Is Crawler = " + browser.Crawler + "\n"

 + "Is AOL = " + browser.AOL + "\n"

 + "Is Win16 = " + browser.Win16 + "\n"

 + "Is Win32 = " + browser.Win32 + "\n"

 + "Supports Frames = " + browser.Frames + "\n"

 + "Supports Tables = " + browser.Tables + "\n"

 + "Supports Cookies = " + browser.Cookies + "\n"

 + "Supports VBScript = " + browser.VBScript + "\n"

 + "Supports JavaScript = " +

 browser.EcmaScriptVersion.ToString() + "\n"

 + "Supports Java Applets = " + browser.JavaApplets + "\n"

 + "Supports ActiveX Controls = " + browser.ActiveXControls

 + "\n";

 TextBox1.Text = s;

}