.NET Framework 4 - ASP.NET

**ASP.NET Configuration Overview**

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.

The following sections describe the features of the ASP.NET configuration system.

For information about configuring .NET Framework client applications, see [Configuring Applications](http://msdn.microsoft.com/en-us/library/kza1yk3a.aspx).

|  |
| --- |
| **Description: NoteNote** |
| The features of the ASP.NET configuration system only apply to ASP.NET resources. For example, Forms Authentication only restricts access to ASP.NET files, not to static files or ASP (classic) files unless those resources are mapped to ASP.NET file name extensions. Use the configuration features of Internet Information Services (IIS) to configure non-ASP.NET resources. For information, see [Working with the IIS Metabase](http://go.microsoft.com/fwlink/?LinkId=46796) and [IIS Metabase Property Reference](http://go.microsoft.com/fwlink/?LinkId=46795). |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifConfiguration Files

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.

For more information about how data is organized in ASP.NET configuration files, see [ASP.NET Configuration Files](http://msdn.microsoft.com/en-us/library/ms178684.aspx). Available configuration settings are described in the [ASP.NET Configuration Settings](http://msdn.microsoft.com/en-us/library/b5ysx397.aspx).

**Configuration File Hierarchy and Inheritance**

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](http://msdn.microsoft.com/en-us/library/b6x6shw7.aspx) 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.

|  |
| --- |
| **Description: NoteNote** |
| The root Web.config file inherits some basic configuration settings from the Machine.config file, which is located in the same directory. Some of these settings cannot be overridden in Web.config files. For more information, see [ASP.NET Configuration File Hierarchy and Inheritance](http://msdn.microsoft.com/en-us/library/ms178685.aspx). |

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](http://msdn.microsoft.com/en-us/library/7w2sway1.aspx) section has been changed.

For more information about how the ASP.NET configuration hierarchy works, see [ASP.NET Configuration File Hierarchy and Inheritance](http://msdn.microsoft.com/en-us/library/ms178685.aspx) and [ASP.NET Configuration Scenarios](http://msdn.microsoft.com/en-us/library/dtbwsx8s.aspx).

**Directly Editing Configuration Files**

You can use a text editor or an XML editor to edit the configuration files directly. For proper syntax, see the reference topics in the configuration sections in [ASP.NET Configuration Settings](http://msdn.microsoft.com/en-us/library/b5ysx397.aspx) and [General Configuration Settings (ASP.NET)](http://msdn.microsoft.com/en-us/library/ms228112.aspx). For more information, see [Editing ASP.NET Configuration Files](http://msdn.microsoft.com/en-us/library/ackhksh7.aspx).

**Additional Configuration Elements in the ASP.NET 3.5 Web.config file**

The Web.config file for an ASP.NET application that targets the .NET Framework version 3.5 contains configuration elements that are not found in the Web.config files for earlier versions. This expanded Web.config file is also created when you open an existing Web site in and upgrade the Web site to target the .NET Framework version 3.5. In that case, Visual Studio updates the application's Web.config file to include the additional configuration elements.

|  |
| --- |
| **Description: NoteNote** |
| For more information about how to configure the application pool for the .NET Framework version 3.0 or 3.5, see the entry [How to set an IIS Application or AppPool to use ASP.NET 3.5 rather than 2.0](http://go.microsoft.com/fwlink/?LinkId=186691) on Scott Hanselman's blog. |

The following table shows the new configuration elements and the changes to configuration elements from earlier versions of the .NET Framework.

|  |  |
| --- | --- |
| **Configuration element** | **Changes in version 3.5** |
| **system.codedom** | (New section) Specifies how the .NET Framework Code Document Object Model (CodeDOM) should compile the source code at run time. |
| **configSections** | (New section) Defines the **system.web.extensions** section that is used by ASP.NET AJAX to define how Web services are called from client script. For more information, see [Web Services in ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398785.aspx). |
| **assemblies** | (New section in the **compilation** element) Specifies the collection of assemblies that are referenced when ASP.NET pages are compiled. Assemblies that are new to ASP.NET version 3.5 are included in this section. |
| **namespaces** | (Updated section) This section specifies what namespaces are imported by default. The [System.Linq](http://msdn.microsoft.com/en-us/library/system.linq.aspx), [System.Xml.Linq](http://msdn.microsoft.com/en-us/library/system.xml.linq.aspx), and [System.Collections.Generic](http://msdn.microsoft.com/en-us/library/system.collections.generic.aspx) namespaces have been added. |
| **controls** | (Updated section) This section registers assemblies that contain controls, and provides a prefix to reference the control, similar to the way the [Directives for ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/t8syafc7.aspx) page directive registers controls in an individual page. By default this section registers controls in the System.Web.Extensions assembly. These include the ListView control and AJAX-related controls. |
| **system.webServer** | This section replaces the AJAX-related HTTP handlers and modules that are added in the **httpHandlers** and **httpModules** sections. This section makes the handlers and modules available to IIS 7.0 when it runs in Integrated mode. |
| **assemblyBinding** | (Updated section) This section directs the runtime to use the ASP.NET AJAX framework that is part of ASP.NET version 3.5 instead of using any earlier version of the ASP.NET AJAX framework. |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifConfiguration Tools

The ASP.NET configuration system provides tools that make application configuration easier than using a text editor because they include error detection.

**ASP.NET MMC Snap-in**

The Microsoft Management Console (MMC) snap-in for ASP.NET provides a convenient way to manipulate ASP.NET configuration settings at all levels on a local or remote Web server. The ASP.NET MMC snap-in uses the ASP.NET configuration API, but it simplifies the process of editing configuration settings by providing a graphical user interface (GUI). In addition, the tool supports the ASP.NET configuration API features that control whether settings can be inherited by Web applications, and manage the dependencies between levels of the configuration hierarchy.

|  |
| --- |
| **Description: NoteNote** |
| To use the ASP.NET MMC snap-in, you must log on to the computer with an account that has administrative privileges. |

The ASP.NET MMC snap-in appears as an **ASP.NET** tab on the property sheet of a virtual directory.

For more information, see [MMC Snap-In for ASP.NET](http://msdn.microsoft.com/en-us/library/ms186189.aspx).

**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. For information about how the tool works, see [ASP.NET Web Site Administration Tool](http://msdn.microsoft.com/en-us/library/ms228053.aspx).

**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. For more information, see [.NET Framework Tools](http://msdn.microsoft.com/en-us/library/d9kh6s92.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifASP.NET Configuration API

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.

For more information, see [ASP.NET Configuration API](http://msdn.microsoft.com/en-us/library/ms178688.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifConfiguration Security

The ASP.NET configuration system helps protect configuration files from access by unauthorized users. ASP.NET configures IIS to deny access to any browser that requests access to the Machine.config or Web.config files. HTTP access error 403 (Forbidden) is returned to any browser that attempts to request a configuration file directly.

Additionally, configuration files in one ASP.NET application are prevented from accessing configuration settings in other ASP.NET applications unless your configuration application is running in Full trust under an account that has permissions to read the configuration file in the other application.

For more information, see [Securing ASP.NET Configuration](http://msdn.microsoft.com/en-us/library/ms178699.aspx) and [Encrypting Configuration Information Using Protected Configuration](http://msdn.microsoft.com/en-us/library/53tyfkaw.aspx).