.NET Framework 4 - ASP.NET

**ASP.NET Side-by-Side Execution Overview**

You can install multiple versions of the .NET Framework on the same computer. In ASP.NET, Web applications are said to be running side-by-side when they are installed on the same computer but use different versions of the .NET Framework. This topic describes side-by-side support in ASP.NET.

This topic contains the following sections:

* [Scenarios](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#scenarios)
* [Side-by-Side Features](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#sidebyside_features)
* [Installing Multiple Versions of the .NET Framework](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#installing_multiple_versions_of_the_net_framework)
* [Specifying the .NET Framework Version for a Web Application](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#specifying_the_net_framework_version_for_a_web_application)
* [Using Different Versions of the .NET Framework in Nested Applications](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#different_versions_of_the_net_framework_in_nested_applications)
* [Using Different Versions of the .NET Framework in a Single Application](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#different_versions_of_the_net_framework_in_a_single_application)
* [ASP.NET Performance Counters](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#aspnet_performance_counters)
* [ASP.NET Session State](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#aspnet_session_state)
* [Third-Party Redistribution of the .NET Framework](http://msdn.microsoft.com/en-us/library/a99txfy5.aspx#thirdparty_redistribution_of_the_net_framework)

http://i.msdn.microsoft.com/Global/Images/clear.gif Scenarios

You have to run multiple versions of the .NET Framework side-by-side when the following conditions are true:

* You want to upgrade to a new version of the .NET Framework but you cannot upgrade all existing Web applications at the same time. For example, you might want to test each application as it is upgraded and your test resources are limited.
* You do not want to maintain a separate server for each version of the .NET Framework that a Web application runs under. You might not have enough servers to support that scenario, or you might want to minimize the number of servers in order to reduce expenses.

http://i.msdn.microsoft.com/Global/Images/clear.gif Side-by-Side Features

ASP.NET side-by-side execution enables you to perform the following tasks:

* Deploy new Web applications that run under the latest version of the .NET Framework on the same server as existing applications that run under an older version.
* Consolidate existing Web applications that require different versions of the .NET Framework onto one server.
* Specify the version of the .NET Framework that each Web application will run under.

You can run .NET Framework 1.1 applications, .NET Framework 2.0 applications, .NET Framework 3.0 applications, .NET Framework 3.5 applications, and .NET Framework 4 applications all on the same server.

http://i.msdn.microsoft.com/Global/Images/clear.gif Installing Multiple Versions of the .NET Framework

To add a version of ASP.NET to a computer, run the .NET Framework redistributable package Setup program (Dotnetfx.exe) for the version that you want to install.

Each ASP.NET application is configured to run under a specific version of the common language runtime (CLR). Some versions of the .NET Framework use the CLR that was introduced with an earlier version of the .NET Framework. The .NET Framework 1.1, the .NET Framework 2.0, and the .NET Framework 4 include different versions of the CLR. Because .NET Framework 3.0 and the .NET Framework 3.5 use the .NET Framework 2.0 CLR, ASP.NET applications developed for these versions of the .NET Framework run under the .NET Framework 2.0 CLR.

When you install an additional version of the CLR on a server that already has an earlier version, existing ASP.NET applications are not automatically upgraded to use the newly installed version even if it is later.

|  |
| --- |
| **NoteNote** |
| You do not have to use the /noaspupgrade command-line option with the Dotnetfx.exe setup program when you install the .NET Framework 4. Existing applications will not be upgraded automatically to the .NET Framework 4. The /noaspupgrade option has an effect when you install the .NET Framework 4 only if the server has no existing version of the .NET Framework installed and uses IIS 6.0. In that case, new ASP.NET applications that you add later will not automatically use the .NET Framework 4. You will have to explicitly register scriptmaps for them. |

http://i.msdn.microsoft.com/Global/Images/clear.gif Specifying the .NET Framework Version for a Web Application

The procedure to specify a version of the .NET Framework for a Web application depends on whether you are using IIS 7.0 or IIS 6.0.

**IIS 7.0**

In IIS 7.0, the version of the CLR that an application runs under is determined by its application pool. An application pool is a group of one or more URLs that are served by a worker process or a set of worker processes. You associate an application pool with a version of the .NET Framework and you associate an application with an application pool. For more information about how to manage application pools, see [IIS 7.0: Managing Application Pools in IIS 7.0](http://go.microsoft.com/fwlink/?LinkId=148298).

You must make sure that all applications that are assigned to an application pool target the CLR that the application pool is associated with. If you assign an application to an application pool that is associated with a different CLR version than the application targets, run-time errors will result.

|  |
| --- |
| **NoteNote** |
| Application pools (or scriptmaps in IIS 6.0) determine which version of the CLR a Web application runs under. The [supportedRuntime](http://msdn.microsoft.com/en-us/library/w4atty68.aspx) and [requiredRuntime](http://msdn.microsoft.com/en-us/library/a5dzwzc9.aspx) elements of configuration files do not apply to ASP.NET applications. |

**IIS 6.0**

In IIS 6.0, the scriptmap that is registered for an application associates a file name extension (such as .aspx) and an HTTP verb (such as GET and POST) with the appropriate ISAPI extension (Aspnet\_isapi.dll). Each version of the CLR has its own version of the ISAPI extension. For information about application configuration in IIS 6.0, see [Setting Application Mappings in IIS 6.0 (IIS 6.0)](http://go.microsoft.com/fwlink/?LinkId=47456).

You must make sure that all applications that are allocated to an application pool use the same version of the CLR. If you assign two different applications to the same application pool, but the scriptmaps for the applications reference different CLR versions, an error might be returned to the browser indicating that the server is unavailable.

**The ASP.NET IIS Registration Tool**

Each installation of the .NET Framework has its own version of the ASP.NET IIS Registration tool (Aspnet\_regiis.exe). By default, this tool is installed in the following directory:

*%system root%*\Microsoft.NET\Framework\*versionNumber*

You can use the registration tool to map an ASP.NET application to the version of the CLR that matches the tool. In IIS 7.0, the .NET Framework 4 version of the tool can move an application to an application pool that is set to use the .NET Framework 4.

The .NET Framework 2.0 is included as part of Windows Vista, Windows Server 2008, and Windows 7. Therefore, the 2.0 version of the IIS Registration tool can be used to perform general tasks with IIS 7.0, such as registering or unregistering ASP.NET with IIS 7.0. The difference between the 2.0 and the 4 version of the IIS Registration tool is that the 2.0 version does not support the **-s** options that are used to move an application from a .NET Framework 2.0 application pool to a .NET Framework 4 application pool.

The tool can also be used to display the status of all installed versions of ASP.NET, to register the associated version of ASP.NET, to create client-script directories, and to perform other configuration operations.

For more information, see [ASP.NET IIS Registration Tool (Aspnet\_regiis.exe)](http://msdn.microsoft.com/en-us/library/k6h9cz8h.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Using Different Versions of the .NET Framework in Nested Applications

If an IIS Web application that targets the .NET Framework 3.5 or earlier is nested in an IIS application that targets the .NET Framework 4, the compiler might report errors when it compiles the nested application. This is because Web.config files inherit settings from files that are higher in the configuration file hierarchy. The .NET Framework 4 is backward compatible; therefore, a nested application that targets the .NET Framework 4 can inherit settings from Web.config files that are for earlier versions. But earlier versions of the .NET Framework are not forward compatible; therefore, they cannot inherit settings from a .NET Framework 4 Web.config file.

ASP.NET 4 applications that are configured as children of applications that run earlier versions of ASP.NET also might not run. For information about how to fix this, see the section "ASP.NET 4 Child Applications Fail to Start When Under ASP.NET 2.0 or ASP.NET 3.5 Applications" in the document [ASP.NET 4 Beta 2 Breaking Changes](http://go.microsoft.com/fwlink/?LinkId=182908) on the ASP.NET Web site.

http://i.msdn.microsoft.com/Global/Images/clear.gif Using Different Versions of the .NET Framework in a Single Application

An ASP.NET Web application can contain components that are built by using different versions of the .NET Framework. For example, suppose component A is built by using .NET Framework version 2.0, component B is built by using the .NET Framework version 4, and both components refer to the **System.Data** assembly.

By default, the version of the .NET Framework that a Web application is associated with in its application pool or scriptmaps determines which assembly versions that it uses. In this example, if the application is associated with the .NET Framework version 4, the version of the **System.Data** assembly that was installed with the .NET Framework version 4 is loaded and shared by all components in the application. The reference from component A to an earlier version of the **System.Data** assembly is promoted at run time to the later version.

Later versions of the .NET Framework are designed to be backward compatible with earlier versions. However, check the readme documentation of any new version of the .NET Framework to learn about breaking changes. The behavior of a component that was developed for an earlier version of the .NET Framework might have been intentionally changed in the newer version of the .NET Framework.

|  |
| --- |
| **NoteNote** |
| It is not possible to run multiple versions of the CLR in the same process (application pool). Although the .NET Framework 4 supports in-process side-by-side execution, ASP.NET does not use it or allow it. |

http://i.msdn.microsoft.com/Global/Images/clear.gif ASP.NET Performance Counters

Performance counters enable you to monitor the operation of ASP.NET from the Windows Performance console. Each version of the CLR has a separate set of performance counters that monitor ASP.NET applications and ASP.NET as a whole. These counters use the naming conventions "ASP.NET Apps v.*#.#.###.#*" and "ASP.NET v.#*.#.####.#*", where v*.#.#.####.#* denotes the ASP.NET version.

The latest version of ASP.NET that is installed on a server also provides performance counters that apply to all installed versions of ASP.NET. If the latest version of ASP.NET is uninstalled, these counters are automatically re-associated with the latest remaining version of ASP.NET. Applications configured to run under earlier versions of ASP.NET report information only for counters that already are in the earlier version. For example, if ASP.NET 4 provides performance counters A, B, and C, and if ASP.NET 2.0 provides counters A and B, ASP.NET 2.0 applications will not report performance data for counter C.

http://i.msdn.microsoft.com/Global/Images/clear.gif ASP.NET Session State

If ASP.NET state service is used to manage session state on a computer, all versions of ASP.NET that are installed on the same computer share the state service that is installed with the latest version of ASP.NET. This version is registered using the service name **aspnet\_state**. When this version of ASP.NET is uninstalled, the latest remaining version of the service on the computer is then registered and used in its place.

If SQL Server is used to manage session state, all versions of ASP.NET (of the .NET Framework) that are installed on the same computer can share the SQL state server that is installed with the latest version of ASP.NET. The schema for session state is the same in all versions of ASP.NET.

For more information about the ASP.NET state service and SQL state server, see [Session-State Modes](http://msdn.microsoft.com/en-us/library/ms178586.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Third-Party Redistribution of the .NET Framework

Third-party applications can package and install the .NET Framework as part of their setup process under a special redistribution license. This enables an application to install the version of the .NET Framework it needs. The .NET Framework redistributable package (Dotnetfx.exe) is provided for an application to deploy with the runtime.

The default installation script of Dotnetfx.exe might not suit the needs of your application. If so, you must use the options described here for installing the .NET Framework as part of an application's setup process.

For more information, see [Deploying the .NET Framework and Applications](http://msdn.microsoft.com/en-us/library/6hbb4k3e.aspx). For more information about the end-user license agreement (Microsoft Software License Terms) of the redistributable package, see [Installing the .NET Framework](http://msdn.microsoft.com/en-us/library/5a4x27ek.aspx).