**ASP.NET Caching Overview**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178597(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178597(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178597(d=printer,v=vs.80).aspx)

An application can often increase performance by storing data in memory that is accessed frequently and that requires significant processing time to create. For example, if your application processes large amounts of data using complex logic and then returns the data as a report accessed frequently by users, it is efficient to avoid re-creating the report every time that a user requests it. Similarly, if your application includes a page that processes complex data but that is updated only infrequently, it is inefficient for the server to re-create that page on every request.

To help you increase application performance in these situations, ASP.NET provides caching using two basic caching mechanisms. The first is application caching, which allows you to cache data you generate, such as a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) object or a custom report business object. The second is page output caching, which saves the output of page processing and reuses the output instead of re-processing the page when a user requests the page again.

[Application Cache](javascript:void(0))

The application cache provides a programmatic way for you to store arbitrary data in memory using key/value pairs. Using the application cache is like using application state. However, unlike application state, the data in the application cache is volatile. This means it is not stored in memory for the life of the application. The advantage of using the application cache is that ASP.NET manages the cache and removes items when they expire or become invalidated, or when memory runs low. You can also configure application caching to notify your application when an item is removed. For more information, see [Caching Application Data](http://msdn.microsoft.com/en-us/library/6hbbsfk6.aspx).

The pattern when using the application cache is to determine whether an item exists in the cache any time you access an item, and if it does, to use it. If the item does not exist, you can re-create the item and then place it back in the cache. This pattern ensures that you always have the latest data in the cache.

For more information, see [Caching in .NET Framework Applications](http://msdn.microsoft.com/en-us/library/dd997357.aspx) and [How to: Retrieve Values of Cached Items](http://msdn.microsoft.com/en-us/library/xhy3h9f9.aspx).

[Page Output Cache](javascript:void(0))

A typical kind of caching for server applications is output caching. Output caching enables you to store rendered HTML. The stored HTML is served in response to subsequent requests for the same page. You can use output caching to cache a whole Web page or just the output of an ASP.NET control. Output caching enables you to do the following:

* Configure ASP.NET to cache a particular output cache entry for a specific period.
* Cache a different version of the content based on the browser type or user-language preferences of the clients visiting your application.
* Cache a mobile version of a page that differs from a version that is optimized for a desktop browser.
* Configure ASP.NET to evict a cache entries based on an external event.

Output caching is extensible. You can use a custom output cache provider that can store data on any data storage device.

The page output cache stores the contents of a processed ASP.NET page in memory. This lets ASP.NET send a page response to a client without going through the page processing life cycle again. Page output caching is especially useful for pages that do not change often but that require significant processing to create. For example, if you are creating a high-traffic Web page to display data that is not frequently updated, page output caching can dramatically increase the performance of that page. Page caching can be configured individually for each page, or you can create cache profiles in the Web.config file, which allow you to define caching settings once and then use those settings with multiple pages.

Page output caching provides two models for page caching: full-page caching and partial-page caching. Full-page caching persists the complete contents of a page and uses the cached page content to fulfill client requests. Partial-page caching persists specified portions of a page and lets other portions of the page be created dynamically. For more information, see [Caching Portions of an ASP.NET Page](http://msdn.microsoft.com/en-us/library/h30h475z.aspx).

Partial-page caching can work in two ways: control caching and post-cache substitution. Control caching, also sometimes referred to as fragment caching, lets you cache parts of the page output by including the information in a user control and then marking the user control as cacheable. This enables specific content throughout a page to be cached, while the overall page is not cached and is therefore re-created on each request. For example, if you create a page that displays largely dynamic content, such as stock information, but has sections that are static, such as weekly summaries, you can place the static sections in user controls and specify that they are cached.

Post-cache substitution is the opposite. The page as a whole is cached, but fragments within the page are dynamic. For example, if you create a page that is static for set periods of time, you can set the entire page to be cached. If you added a [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label.aspx) control to the page that displayed the user's name, the [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label.aspx) would stay the same for each page refresh and each user, showing the name of the user who requested that page before it was cached. However, with post-cache substitution, you can configure the page to be cached, but mark individual sections of the page as not cacheable. In this case, you could add the [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label.aspx) controls to a non-cacheable section and they would be dynamically created for each user and page request. For more information, see [Caching Portions of an ASP.NET Page](http://msdn.microsoft.com/en-us/library/h30h475z.aspx).

**Caching Pages Based on Request Parameters**

In addition to caching a single version of a page, ASP.NET page output caching provides features to create multiple versions of the page that vary by different request parameters. For more information, see [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).

[Extensible Output Caching](javascript:void(0))

ASP.NET adds extensibility to output caching that enables you to configure one or more custom output-cache providers. Output-cache providers can use any storage mechanism to persist HTML content. These storage options can include local or remote disks, cloud storage, and distributed cache engines.

Output-cache provider extensibility in ASP.NET lets you design more aggressive and more intelligent output-caching strategies for Web sites. For example, you can create an output-cache provider that caches the "Top 10" pages of a site in memory, while caching pages that get lower traffic on disk. Alternatively, you can cache every vary-by combination for a rendered page, but use a distributed cache so that the memory consumption is offloaded from front-end Web servers.

You create a custom output-cache provider as a class that derives from the [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) type. You can then configure the provider in the Web.config file by using the new providers subsection of the outputCache element, as shown in the following example:

<caching>

<outputCache defaultProvider="AspNetInternalProvider">

<providers>

<add name="DiskCache"

type="Test.OutputCacheEx.DiskOutputCacheProvider, DiskCacheProvider"/>

</providers>

</outputCache>

</caching>

For more information and for examples that show how to configure the output cache, see [outputCache Element for caching (ASP.NET Settings Schema)](http://msdn.microsoft.com/en-us/library/ms228124.aspx). For more information about the classes that support caching, see the documentation for the [OutputCache](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcache.aspx) and [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) classes.

By default, all HTTP responses, rendered pages, and controls use the in-memory output cache that is illustrated in the previous example (where the defaultProvider attribute is set to AspNetInternalProvider). You can change the default output-cache provider that is used for a Web application by specifying a different provider name for defaultProvider.

You can also select different output-cache providers for individual control and for individual requests. The easiest way to choose a different output-cache provider for different Web user controls is to do so by declaratively using the providerName attribute in an @ Page or @ Control directive, as shown in the following example:

<%@ OutputCache Duration="60" VaryByParam="None"

providerName="DiskCache" %>

To specify a different output cache provider for an HTTP request, you override the new [GetOutputCacheProviderName](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getoutputcacheprovidername.aspx) method in the Global.asax file to programmatically specify which provider to use for a specific request. For more information, see [GetOutputCacheProviderName](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getoutputcacheprovidername.aspx).

[Automatic Data Removal](javascript:void(0))

ASP.NET can remove data from the cache for one of these reasons:

* Because memory on the server is low, a process known as scavenging.
* Because the item in the cache has expired.
* Because the item's dependency changes.

To help you manage cached items, ASP.NET can notify your application when items are removed from the cache.

**Scavenging**

Scavenging is the process of deleting items from the cache when memory is scarce. Items are removed when they have not been accessed in some time or when items are marked as low priority when they are added to the cache. ASP.NET uses the [CacheItemPriority](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitempriority.aspx) object to determine which items to scavenge first. For more information, see [How to: Add Items to the Cache](http://msdn.microsoft.com/en-us/library/18c1wd61.aspx).

**Expiration**

In addition to scavenging, ASP.NET automatically removes items from the cache when they expire. When adding an item to the cache, you can set it to expire as described in the following table.

|  |  |
| --- | --- |
| **Expiration Type** | **Description** |
| Sliding expiration | Specifies how long after an item was last accessed that it expires. For example, you can set an item to expire 20 minutes after it was last accessed in the cache. |
| Absolute expiration | Specifies that an item expires at a set time, regardless of how often it is accessed. For example, you can set an item to expire at 6:00 PM or after four hours. |

**Dependencies**

You can configure an item's lifetime in the cache to be dependent on other application elements such as files or databases. When the element that a cache item depends on changes, ASP.NET removes the item from the cache. For example, if your Web site displays a report that the application creates from an XML file, you can place the report in the cache and configure it to have a dependency on the XML file. When the XML file changes, ASP.NET removes the report from the cache. When your code requests the report, the code first determines whether the report is in the cache, and if not, your code can re-create it. Therefore, an up-to-date version of the report is always available.

ASP.NET caching supports the dependencies described in the following table.

|  |  |
| --- | --- |
| **Dependency** | **Description** |
| Key dependency | Items in the application cache are stored in key/value pairs. Key dependency allows an item to be dependent on the key of another item in the application cache. When the original item is removed, the item that has the key dependency is also removed. For example, you could add a cache item named ReportsValid, and then cache several reports that are dependent on the ReportsValid key. When the ReportsValid item is removed, all the dependent cached reports are similarly removed from the cache. |
| File dependency | An item in the cache is dependent on an external file. If the file is modified or deleted, the cached item is removed. |
| SQL dependency | An item in the cache is dependent on changes in a table in a Microsoft SQL Server 2005, SQL Server 2000, or SQL Server 7.0 database. For SQL Server 2005, an item can be dependent on a row in a table. For more information, see [Caching in ASP.NET with the SqlCacheDependency Class](http://msdn.microsoft.com/en-us/library/ms178604.aspx). |
| Aggregate dependency | An item in the cache is dependent on multiple elements through the use of the [AggregateCacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.aggregatecachedependency.aspx) class. If any of the dependencies change, the item is removed from the cache. |
| Custom dependency | An item in the cache is configured with a dependency that you create in your own code. For example, you can create a custom Web service cache dependency that removes data from the cache when a call to a Web service results in a particular value. |

[Application Cache Item Removal Notification](javascript:void(0))

You can be notified when an item is removed from the application cache. For example, if you have an item that takes considerable amount of processing time to create, you can be notified when it is removed from the cache so that you can replace it immediately. As a result, the next time that the item is requested, the user does not have to wait for it to be processed. For more information, see [How to: Notify an Application When an Item Is Removed from the Cache](http://msdn.microsoft.com/en-us/library/7kxdx246.aspx).

**Caching in .NET Framework Applications**

**.NET Framework 4.5**

[Other Versions](javascript:;)



* [.NET Framework 4](http://msdn.microsoft.com/en-us/library/dd997357(d=printer,v=vs.100).aspx)

Caching enables you to store data in memory for rapid access. When the data is accessed again, applications can get the data from the cache instead of retrieving it from the original source. This can improve performance and scalability. In addition, caching makes data available when the data source is temporarily unavailable.

The .NET Framework provides caching functionality that you can use to improve the performance and scalability of both Windows client and server applications, including ASP.NET.

|  |
| --- |
| **NoteNote** |
| In the .NET Framework 3.5 and earlier versions, ASP.NET provided an in-memory cache implementation in the [System.Web.Caching](http://msdn.microsoft.com/en-us/library/system.web.caching.aspx) namespace. In previous versions of the .NET Framework, caching was available only in the [System.Web](http://msdn.microsoft.com/en-us/library/system.web.aspx) namespace and therefore required a dependency on ASP.NET classes. In the .NET Framework 4, the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace contains APIs that are designed for both Web and non-Web applications. |

[Caching Data](javascript:void(0))

You can cache information by using classes in the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace. The caching classes in this namespace provide the following features:

* Abstract types that provide the foundation for creating custom cache implementations.
* A concrete in-memory object cache implementation.

The abstract base caching class ([ObjectCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.objectcache.aspx)) defines the following caching tasks:

* Creating and managing cache entries.
* Specifying expiration and eviction information.
* Triggering events that are raised in response to changes in cache entries.

The [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class is an in-memory object cache implementation of the [ObjectCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.objectcache.aspx) class. You can use the [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class for most caching tasks.

|  |
| --- |
| **NoteNote** |
| The [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class is modeled on the ASP.NET cache object that is defined in the [System.Web.Caching](http://msdn.microsoft.com/en-us/library/system.web.caching.aspx) namespace. Therefore, the internal caching logic similar to the logic that was provided in earlier versions of ASP.NET. |

For an example of how to use to caching in a WPF application, see [Walkthrough: Caching Application Data in a WPF Application](http://msdn.microsoft.com/en-us/library/dd997362.aspx).

[Caching in ASP.NET Applications](javascript:void(0))

The caching classes in the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace provide functionality for caching data in ASP.NET.

|  |
| --- |
| **NoteNote** |
| If your application targets the .NET Framework 3.5 or earlier, you must use the caching classes that are defined in the [System.Web.Caching](http://msdn.microsoft.com/en-us/library/system.web.caching.aspx) namespace. For more information, see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx). |

|  |
| --- |
| **NoteNote** |
| When you develop new applications, we recommend that you use the [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class. The API that is provided in the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace is like the API that is provided in the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) namespace. Therefore, the API will be familiar if you used caching in earlier versions of ASP.NET. For an example of how to use caching in ASP.NET applications, see [Walkthrough: Caching Application Data in ASP.NET](http://msdn.microsoft.com/en-us/library/ff477235.aspx). |

[Output Caching](javascript:void(0))

To manually cache application data, you can use the [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class in ASP.NET. ASP.NET also supports output caching, which stores the generated output of pages, controls, and HTTP responses in memory. You can configure output caching declaratively in an ASP.NET Web page or by using settings in the Web.config file. For more information, see [outputCache Element for caching (ASP.NET Settings Schema)](http://msdn.microsoft.com/en-us/library/ms228124.aspx).

ASP.NET lets you extend output caching by creating custom output-cache providers. By using custom providers, you can store cached content using other storage devices such as disks, cloud storage, and distributed cache engines. To create a custom output cache provider, you create a class that derives from the [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) class and configure the application to use the custom output cache provider.

[Caching in WCF REST Services](javascript:void(0))

For WCF REST services, the .NET Framework enables you to take advantage of the declarative output caching that is available in ASP.NET. This enables you to cache responses from your WCF REST service operations. When a user sends an HTTP GET request to a service that is configured for caching, ASP.NET sends back the cached response, and the service method is not called. After the cache expires, the next time that a user sends an HTTP GET request, your service method is called and the response is again cached.

The .NET Framework also enables you to implement conditional HTTP GET caching. In REST scenarios, a conditional HTTP GET request is often used by services to implement intelligent HTTP caching as described in the [HTTP Specification](http://go.microsoft.com/fwlink/?LinkId=165800). For more information, see [Caching Support for WCF Web HTTP Services](http://go.microsoft.com/fwlink/?LinkId=184598).

[Extending Caching in the .NET Framework](javascript:void(0))

Caching in the .NET Framework is designed to be extensible. The [ObjectCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.objectcache.aspx) class enables you to create a custom cache implementation. This class provides members that are available to all managed applications, including Windows Forms, Windows Presentation Foundation (WPF), and Windows Communications Foundation (WCF). You might do this in order to create a cache class that uses a different storage mechanism, or if you want granular control over cache operations.

To extend caching you can do the following:

* Create a custom class that derives from the [ObjectCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.objectcache.aspx) class and then provide a custom cache implementation in the derived class.
* Create a class that derives from [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class and customize or extend the derived class. For an example of how to do this, see [Caching Application Data by Using Multiple Cache Objects in an ASP.NET Application](http://blogs.msdn.com/aspnetue/archive/2010/03/22/caching-application-data-by-using-multiple-cache-objects-in-an-asp-net-application.aspx).
* Create a class that derives from the [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) class and configure the application to use the custom output cache provider.

For more information, see the entry [Extensible Output Caching with ASP.NET 4 (VS 2010 and .NET 4.0 Series)](http://go.microsoft.com/fwlink/?LinkId=185772) on Scott Guthrie's blog.

**How to: Retrieve Values of Cached Items**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/xhy3h9f9(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/xhy3h9f9(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/xhy3h9f9(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/xhy3h9f9(d=printer,v=vs.71).aspx)

To retrieve data from the cache, you specify the key that the cached item was stored under. However, because information stored in the cache is volatile—that is, it might be removed by ASP.NET—the recommended development pattern is to determine first whether the item is in the cache. If it is not, you add it back to the cache and then retrieve the item.

**To retrieve the value of a cached item**

* Check to see if the item is not null (Nothing in Visual Basic), in the [Cache](http://msdn.microsoft.com/en-us/library/system.web.ui.page.cache.aspx) object. If it exists, assign it to your variable. Otherwise, recreate the item, add it to the cache, and then access it.

The following code example shows how to retrieve the item named CacheItem from the cache. The code assigns the contents of the item to the variable named cachedString. If the item is not in the cache, the code adds the item to the cache and then assigns the item to cachedString.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1d0a2bf4-a3ac-4484-be4a-b0ef7c3c72f9');)

string cachedString;

cachedString = (string)Cache["CacheItem"];

if (cachedString == null)

{

cachedString = "Hello, World.";

Cache.Insert("CacheItem", cachedString);

}

VB

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_869a68bb-0a29-489c-865b-d8b2bd0c217b');)

Dim cachedString As String

cachedString = CStr(Cache("CacheItem"))

If cachedString Is Nothing Then

cachedString = "Hello, World."

Cache.Insert("CacheItem", cachedString)

End If

**How to: Add Items to the Cache**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/18c1wd61(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/18c1wd61(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/18c1wd61(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/18c1wd61(d=printer,v=vs.71).aspx)

You can access items in the application cache using the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object. You can add an item to the application cache using the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object's [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method. The method adds an item to the cache and has several overloads that enable you to add the item with different options for setting dependencies, expiration, and removal notification. If you use the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method to add an item to the cache and an item with the same name already exists, the existing item in the cache is replaced.

You can also add items to the cache using the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) method. This method enables you to set all the same options as the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method; however, [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) method returns the object you added to the cache. Additionally, if you use the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) method and an item with the same name already exists in the cache, the method will not replace the item and will not raise an exception.

The procedures in this topic illustrate the following ways to add items to the application cache:

* Adding an item to the cache by directly setting the item via key and value.
* Adding items to the cache using the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method.
* Adding an item to the cache and adding a dependency so that the item is removed from the cache when the dependency changes. You can set dependencies based on other cache items, on files, and on multiple objects.
* Adding an item to the cache with expiration policies. In addition to being able to set an item's dependency, you can set the item to expire after a period of time (a sliding expiration) or at a specific time (an absolute expiration). You can define either an absolute expiration or a sliding expiration, but not both.
* Adding an item to the cache and defining the relative priority of the cached item. Relative priorities help the .NET Framework determine what cache items to remove; lower priority items are removed from the cache before higher priority items.
* Adding an item by calling the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) method.

In addition to the dependencies shown here, you can create a dependency on a SQL Server table or based on a custom dependency. For more information, see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx) and [Caching in ASP.NET with the SqlCacheDependency Class](http://msdn.microsoft.com/en-us/library/ms178604.aspx).

You can also have the application cache notify your application when the item is removed from the cache, using the [CacheItemRemovedCallback](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitemremovedcallback.aspx) delegate. For a full example, see [How to: Notify an Application When an Item Is Removed from the Cache](http://msdn.microsoft.com/en-us/library/7kxdx246.aspx).

**To add an item to the cache by directly setting the item via key and value**

* Add items to the cache as you would add items to a dictionary by specifying the item's key and value.

The following code example adds an item named CacheItem1 to the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object:

C#

Cache["CacheItem1"] = "Cached Item 1";

VB

Cache("CacheItem1") = "Cached Item 1"

**To add items to the cache by using the Insert method**

* Call the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method, passing the key and value of the item to add.

The following code example adds a string under the name CacheItem2:

C#

Cache.Insert("CacheItem2", "Cached Item 2");

VB

Cache.Insert("CacheItem2", "Cached Item 2")

**To add an item to the cache by specifying a dependency**

* Call the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method, passing it an instance of the [CacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.cachedependency.aspx) object

The following code example adds an item named CacheItem3 that is dependent on another item in the cache named CacheItem2:

C#

string[] dependencies = { "CacheItem2" };

Cache.Insert("CacheItem3", "Cached Item 3",

new System.Web.Caching.CacheDependency(null, dependencies));

VB

Dim dependencies As String() = {"CacheItem2"}

Cache.Insert("CacheItem3", "Cached Item 3", \_

New System.Web.Caching.CacheDependency( \_

Nothing, dependencies))

The following code example shows an item named CacheItem4 added to the cache and having a file dependency set on the file named XMLFile.xml:

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_cf40cd2d-2329-4a4b-a75d-a76ca7763cae');)

Cache.Insert("CacheItem4", "Cached Item 4",

new System.Web.Caching.CacheDependency(

Server.MapPath("XMLFile.xml")));

VB

Cache.Insert("CacheItem4", "Cached Item 4", \_

New System.Web.Caching.CacheDependency( \_

Server.MapPath("XMLFile.xml")))

The following code example shows how to create multiple dependencies. It adds a key dependency on another item in the cache named CacheItem1 and a file dependency on the file named XMLFile.xml.

C#

System.Web.Caching.CacheDependency dep1 =

new System.Web.Caching.CacheDependency(Server.MapPath("XMLFile.xml"));

string[] keyDependencies2 = { "CacheItem1" };

System.Web.Caching.CacheDependency dep2 =

new System.Web.Caching.CacheDependency(null, keyDependencies2);

System.Web.Caching.AggregateCacheDependency aggDep =

new System.Web.Caching.AggregateCacheDependency();

aggDep.Add(dep1);

aggDep.Add(dep2);

Cache.Insert("CacheItem5", "Cached Item 5", aggDep);

VB

Dim dep1 As CacheDependency = \_

New CacheDependency(Server.MapPath("XMLFile.xml"))

Dim keyDependencies2 As String() = {"CacheItem1"}

Dim dep2 As CacheDependency = \_

New System.Web.Caching.CacheDependency(Nothing, \_

keyDependencies2)

Dim aggDep As AggregateCacheDependency = \_

New System.Web.Caching.AggregateCacheDependency()

aggDep.Add(dep1)

aggDep.Add(dep2)

Cache.Insert("CacheItem5", "Cached Item 5", aggDep)

**The add an item to the cache with expiration policies**

* Call the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method, passing it an absolute or sliding expiration time.

The following code example adds an item to the cache with an absolute expiration of one minute:

C#

Cache.Insert("CacheItem6", "Cached Item 6",

null, DateTime.Now.AddMinutes(1d),

System.Web.Caching.Cache.NoSlidingExpiration);

VB

Cache.Insert("CacheItem6", "Cached Item 6", \_

Nothing, DateTime.Now.AddMinutes(1.0), \_

TimeSpan.Zero)

The following code example adds an item to the cache with a sliding expiration time of 10 minutes:

C#

Cache.Insert("CacheItem7", "Cached Item 7",

null, System.Web.Caching.Cache.NoAbsoluteExpiration,

new TimeSpan(0, 10, 0));

VB

Cache.Insert("CacheItem7", "Cached Item 7", \_

Nothing, System.Web.Caching.Cache.NoAbsoluteExpiration, \_

New TimeSpan(0, 10, 0))

**To add an item to the Cache with priority settings**

* Call the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method, specifying a value from the [CacheItemPriority](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitempriority.aspx) enumeration.

The following code example adds an item to the cache with a priority value of [High](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitempriority.aspx):

C#

Cache.Insert("CacheItem8", "Cached Item 8",

null, System.Web.Caching.Cache.NoAbsoluteExpiration,

System.Web.Caching.Cache.NoSlidingExpiration,

System.Web.Caching.CacheItemPriority.High, null);

VB

Cache.Insert("CacheItem8", "Cached Item 8", \_

Nothing, System.Web.Caching.Cache.NoAbsoluteExpiration, \_

System.Web.Caching.Cache.NoSlidingExpiration, \_

System.Web.Caching.CacheItemPriority.High, \_

Nothing)

**To add an item to the cache using the Add method**

* Call the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) method, which returns an object representing the item.

The following code example adds an item to the cache named CacheItem9 and sets the value of the variable CachedItem9 to be the item that was added.

C#

string CachedItem9 = (string)Cache.Add("CacheItem9",

"Cached Item 9", null,

System.Web.Caching.Cache.NoAbsoluteExpiration,

System.Web.Caching.Cache.NoSlidingExpiration,

System.Web.Caching.CacheItemPriority.Default,

null);

VB

Dim CachedItem9 As String = CStr(Cache.Add("CacheItem9", \_

"Cached Item 9", Nothing, \_

System.Web.Caching.Cache.NoAbsoluteExpiration, \_

System.Web.Caching.Cache.NoSlidingExpiration, \_

System.Web.Caching.CacheItemPriority.Default, \_

Nothing))

**How to: Delete Items from the Cache in ASP.NET**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/edfcywt6(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/edfcywt6(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/edfcywt6(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/edfcywt6(d=printer,v=vs.71).aspx)

Data in the ASP.NET cache is volatile — that is, it is not permanently stored. It might be automatically removed from the cache for one of these reasons:

* Because the cache is full.
* Because the item has expired.
* Because an item it is dependent on changes.

For more information see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx).

The specific method for removing items from the cache is determined in the code used to add the item to the cache. For more information, see [How to: Add Items to the Cache](http://msdn.microsoft.com/en-us/library/18c1wd61.aspx). You can be notified when an item is removed from the cache. For more information see [How to: Notify an Application When an Item Is Removed from the Cache](http://msdn.microsoft.com/en-us/library/7kxdx246.aspx).

In addition to allowing items to be removed from the cache automatically, you can explicitly remove them.

|  |
| --- |
| **NoteNote** |
| If you call the [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) method and add an item to the cache with the same name as an existing item, the old item will be deleted from the cache. |

**To delete an item from the cache explicitly**

* Call the [Remove](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.remove.aspx) method, passing the key of the item you want to remove.

The following example shows how to remove an item with the key MyData1.

VB

Cache.Remove("MyData1")

C#

Cache.Remove("MyData1");

**How to: Notify an Application When an Item Is Removed from the Cache**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer,v=vs.71).aspx)

In most cache scenarios, when an item is removed from the cache, you do not have to be notified when it has been removed. The typical development pattern is to always check the cache for the item before using it. If the item is in the cache, you use it. If it is not in the cache, you retrieve the item again and add it back to the cache.

However, in some cases it is useful for your application to be notified when an item is removed from the cache. For example, you might want to track when and why items are removed from the cache in order to tune cache settings.

To enable notification of items being removed from the cache, ASP.NET provides the [CacheItemRemovedCallback](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitemremovedcallback.aspx) delegate. The delegate defines the signature for an event handler to call when an item is removed from the cache. Typically, you implement the callback by creating a handler in a business object that manages the cache data.

|  |
| --- |
| **NoteNote** |
| This topic explains how to handle a notification after an item has been removed from the cache. You can also be notified before an item has been removed. You can then prevent the item from being removed instead of re-creating the object. This might be more efficient for items that require a significant amount of processing time to re-create. For more information, see [CacheItemUpdateCallback](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitemupdatecallback.aspx). |

**To notify an application after an item is removed from the cache**

1. In a business class (not in a page or user control class), create a method that handles the callback when a cache item is removed. The method must have the same signature as the [CacheItemRemovedCallback](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitemremovedcallback.aspx) delegate.

You must make sure that this method is available when the cache item is deleted. Using a static class is one way to accomplish this. Note that in a static class, all static methods must be thread-safe.

Because the methods that add items to the cache and get items from the cache do not have to be available when the cache item is removed, it is common to put them in a separate class from the one that contains the callback handler.

|  |
| --- |
| **Caution noteCaution** |
| Do not implement the method invoked by [CacheItemRemovedCallback](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitemremovedcallback.aspx) in a page, user control, or any other class that is repeatedly loaded and disposed, because the method might not be available when it is needed. In addition, pointing the callback to a method of an object can prevent the memory that is used by the object from being reclaimed by garbage collection. This happens because the callback contains a reference to the object and the garbage collector will not remove an item from memory if the item has any references. During periods of application load, this could cause memory to be used up very quickly. |

1. In the callback method, add logic that will run when the item is removed from the cache.

[Example](javascript:void(0))

The following example shows a class named ReportManager. The GetReport method of this class creates a report that consists of the string "Report Text". The method saves this report in the cache, and on subsequent calls it retrieves the report from the cache.

|  |
| --- |
| **NoteNote** |
| To simplify the example, the methods to manage the cache and the callback method are all in the same class. In a production environment, you typically separate these methods into separate classes. |

If more than 15 seconds elapses between calls to GetReport, ASP.NET removes the report from the cache. When that event occurs, the ReportRemovedCallback method of the ReportManager class is called. This method sets a private member variable to "Re-created [date and time]", where [date and time] is the current date and time. The next time that GetReport is called after the cache item has expired, the method re-creates the report and appends the value of the variable that was set by the ReportRemovedCallback method to the report. The ShowReport.aspx page displays the report string that GetReport returns, which includes the date and time that the report was last re-created.

To see this behavior, load the page, wait more than 15 seconds, and then reload the page in the browser. You will see the date and time added to the report text.

C#

[VB](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

using System;

using System.Text;

using System.Web;

using System.Web.Caching;

public static class ReportManager

{

private static string \_lastRemoved = "";

public static String GetReport()

{

string report = HttpRuntime.Cache["MyReport"] as string;

if (report == null)

{

report = GenerateAndCacheReport();

}

return report;

}

private static string GenerateAndCacheReport()

{

string report = "Report Text. " + \_lastRemoved.ToString();

HttpRuntime.Cache.Insert(

"MyReport",

report,

null,

Cache.NoAbsoluteExpiration,

new TimeSpan(0, 0, 15),

CacheItemPriority.Default,

new CacheItemRemovedCallback(ReportRemovedCallback));

return report;

}

public static void ReportRemovedCallback(String key, object value,

CacheItemRemovedReason removedReason)

{

\_lastRemoved = "Re-created " + DateTime.Now.ToString();

}

}

C#

[VB](http://msdn.microsoft.com/en-us/library/7kxdx246(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

<%@ Page Language="C#" AutoEventWireup="true" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<%=ReportManager.GetReport() %>

</div>

</form>

</body>

</html>

**Caching Portions of an ASP.NET Page**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer,v=vs.71).aspx)

Sometimes it is impractical to cache an entire page because portions of the page might need to change on each request. In those cases, you can cache just a portion of a page. There are two options to do this: control caching and post-cache substitution.

In control caching, also known as fragment caching, you can cache parts of the page output by creating user controls to contain the cached content and then marking the user controls as cacheable. This allows specific content within a page to be cached, while the overall page is recreated each time. For example, if you create a page that displays largely dynamic content, such as stock information, but also has sections that are static, such as weekly summaries, you can create the static sections in user controls and configure the user controls to be cached.

Post-cache substitution is the opposite. The page is cached, but fragments within the page are dynamic and should therefore not be cached. For example, if you create a page that is entirely static for set periods of time such as a page of news stories, you can set the entire page to be cached. If you added rotating ad banners to the cached page, they would not change between page requests. However, with post-cache substitution, the page can be cached, but you can mark specific parts as non-cacheable. In the example, you would mark your ad banners as non-cacheable. They would then be dynamically created for each page request and added to the cached page output. For more information on post-cache substitution, see [Dynamically Updating Portions of a Cached Page](http://msdn.microsoft.com/en-us/library/ms227429.aspx).

[Control Caching](javascript:void(0))

Creating user controls to cache content allows you to separate portions of a page that take valuable processor time to create, such as database queries, from other parts of the page. The parts of the page that require fewer server resources can be generated dynamically for each request.

Once you identify the portions of the page that you want to cache and create the user controls that contain each of those portions, you must determine the caching policies for the user controls. You can set these policies declaratively using the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, or by using the [PartialCachingAttribute](http://msdn.microsoft.com/en-us/library/system.web.ui.partialcachingattribute.aspx) class in the code for the user control.

For example, if you include the following directive at the top of a user control file (.ascx file), a version of the control is stored in the output cache for 120 seconds.

<%@ OutputCache Duration="120" VaryByParam="None" %>

If you want to set caching parameters in code, you can use an attribute in the user control's class declaration. For example, if you include the following attribute in the metadata of your class declaration, a version of the content in the output cache is stored for 120 seconds:

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

[PartialCaching(120)]

public partial class CachedControl : System.Web.UI.UserControl

{

// Class Code

}

For more information on the attributes that you can set on page output, see the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) topic. For more information on how to develop user controls, see [ASP.NET Web Server Controls Overview](http://msdn.microsoft.com/en-us/library/zsyt68f1.aspx).

|  |
| --- |
| **NoteNote** |
| Because you can nest user controls on a page, you can also nest user controls that have been placed in the output cache. You can specify different cache settings for the page and for nested user controls. |

**Referencing Cached User Controls Programmatically**

When creating a cacheable user control declaratively, you can include an [ID](http://msdn.microsoft.com/en-us/library/system.web.ui.control.id.aspx) attribute in order to programmatically reference that instance of the user control. However, before referencing the user control in code, you must verify the existence of the user control in the output cache. A user control that is cached is dynamically generated only for the first request; any subsequent requests are satisfied from the output cache until the specified time expires. After you determine that the user control has been instantiated, you can programmatically manipulate the user control from the containing page. For example, if you declaratively assign a user control an [ID](http://msdn.microsoft.com/en-us/library/system.web.ui.control.id.aspx) of SampleUserControl, you can check for its existence with the following code:

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

protected void Page\_Load(object sender, EventArgs e)

{

if (SampleUserControl != null)

// Place code manipulating SampleUserControl here.

}

[Caching the Page and User Controls for Different Durations](javascript:void(0))

You can set different output cache duration values for a page and for user controls on the page. If the page output cache duration is longer than the user control output cache duration, the page output cache duration takes precedence. For example, if page output caching is set to 100 seconds and the user control output caching is set to 50 seconds, the entire page, including the user control, is stored in the output cache for 100 seconds, regardless of the shorter setting for the user control.

The following code example shows the effect of a longer cache duration in the page than in a user control. The page is configured to be cached for 100 seconds.

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

<%@ Page language="C#" %>

<%@ Register tagprefix="SampleControl" tagname="Time" src="uc01.ascx" %>

<%@ OutputCache duration="100" varybyparam="none" %>

<SampleControl:Time runat="server" /><br /> <br /> <br />

This page was most recently generated at:<p>

<% string t = DateTime.Now.ToString();

Response.Write(t); %>

The following code example shows the user control included in the page. The cache duration for the control is set to 50 seconds.

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

<% @Control language="C#" %>

<% @OutputCache duration="50" varybyparam="none" %>

This user control was most recently generated at:<p>

<% string t = DateTime.Now.ToString();

Response.Write(t); %>

If the page output cache duration is less than that of a user control, the user control will be cached until its duration has expired, even after the remainder of the page is regenerated for a request. For example, if page output caching is set to 50 seconds and the user control's output caching is set to 100 seconds, the user control expires once for every two times the rest of the page expires.

The following code shows the markup for a page that contains the user control with a longer cache duration than that of the page. The page is configured to be cached for 50 seconds.

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

<%@ Page language="C#" %>

<%@ Register tagprefix="SampleControl" tagname="Time" src="uc2.ascx" %>

<%@ OutputCache duration="50" varybyparam="none" %>

<SampleControl:Time runat="server" /><br /> <br /> <br />

This page was most recently generated at:<p>

<% string t = DateTime.Now.ToString();

Response.Write(t); %>

The following code shows the user control included in the page. The cache duration for the control is set to 100 seconds.

C#

[VB](http://msdn.microsoft.com/en-us/library/h30h475z(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

<% @Control language="C#" %>

<% @OutputCache duration="100" varybyparam="none" %>

This user control was most recently generated at:<p>

<% string t = DateTime.Now.ToString();

Response.Write(t); %>

**Dynamically Updating Portions of a Cached Page**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms227429(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms227429(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms227429(d=printer,v=vs.80).aspx)

Caching a page can dramatically increase the performance of a Web application. However, in some cases you need most of the page to be cached and some fragments within the page to be dynamic. For example, if you create a page of news stories that is entirely static for set periods of time, you can set the entire page to be cached. If you wanted to include a rotating ad banner that changed on every page request, then the part of the page containing the advertisement needs to be dynamic.

To allow you to cache a page but substitute some content dynamically, you can use ASP.NET post-cache substitution. With post-cache substitution, the entire page is output cached with specific parts marked as exempt from caching. In the example of the ad banners, the [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx) control allows you to take advantage of post-cache substitution so that ads dynamically created for each user and for each page refresh.

There are three ways to implement post-cache substitution:

* Declaratively, using the [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control.
* Programmatically, using the [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control API.
* Implicitly, using the [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx) control.

[Substitution Control](javascript:void(0))

The ASP.NET [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control specifies a section of a cached page that is created dynamically rather than cached. You place a [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control at the location on the page where you want the dynamic content to appear.

At run time, the [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control calls a method that you specify with the [MethodName](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.methodname.aspx) property. The method must return a string, which then replaces the content of the [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control. The method must be a static method on the containing [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) or [UserControl](http://msdn.microsoft.com/en-us/library/system.web.ui.usercontrol.aspx) control.

Using the substitution control causes client-side cacheability to be changed to server cacheability, so that the page will not be cached on the client. This ensures that future requests to the page call the method again to generate dynamic content.

[Substitution API](javascript:void(0))

To create dynamic content for a cached page programmatically, you can call the [WriteSubstitution](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.writesubstitution.aspx) method in your page code, passing it the name of a method as a parameter. The method that handles the creation of the dynamic content takes a single [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx) parameter and returns a string. The return string is the content that will be substituted at the given location. An advantage of calling the [WriteSubstitution](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.writesubstitution.aspx) method instead of using the [Substitution](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.substitution.aspx) control declaratively is that you can call a method of any arbitrary object, rather than calling a static method of the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) or the [UserControl](http://msdn.microsoft.com/en-us/library/system.web.ui.usercontrol.aspx) object.

Calling the [WriteSubstitution](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.writesubstitution.aspx) method causes client-side cacheability to be changed to server cacheability, so that the page will not be cached on the client. This ensures that future requests to the page call the method again to generate dynamic content.

[AdRotator Control](javascript:void(0))

The [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx) server control implements support for post-cache substitution internally. If you place an [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx) control on your page, it will render unique advertisements on each request, regardless of whether the parent page is cached. As a result, a page that includes an [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx) control is only cached server-side.

**Caching Multiple Versions of a Page**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/xadzbzd6(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/xadzbzd6(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/xadzbzd6(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/xadzbzd6(d=printer,v=vs.71).aspx)

At times you might have a page that you want to cache, but for which different versions are created based on the request. For example, the page might have different output depending on the values passed in the query string.ASP.NET allows you to cache multiple versions of the same page in the output cache. You can vary the output cache by the following:

* The query string in an initial request (HTTP GET).
* Control values passed on postback (HTTP POST values).
* The HTTP headers passed with a request.
* The major version number of the browser making the request.
* A custom string in the page. In that case, you create custom code in the Global.asax file to specify the page's caching behavior.

You can cache multiple versions of a page output declaratively using attributes of the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, and programmatically using the properties and methods of the [HttpCachePolicy](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.aspx) class.

The [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive includes four attributes that enable you to cache multiple versions of page output:

* The [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute allows you to vary the cached output depending on the query string.
* The [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) attribute allows you to vary the cached output depending on a control value.
* The [VaryByHeader](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyheader.aspx) attribute allows you to vary the cached output depending on the request's HTTP header.
* The [VaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycustom.aspx) attribute allows you to vary the cached output by browser type or by a custom string that you define.

|  |
| --- |
| **NoteNote** |
| You must include either the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute or the [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) attribute in any [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive. However, if you do not need to vary your cached output by control or parameters, you can define [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) with its value to None. |

The [HttpCachePolicy](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.aspx) class provides two properties and a method that allow you to programmatically specify the same cache configuration that you can set declaratively. The [VaryByParams](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyparams.aspx) and [VaryByHeaders](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyheaders.aspx) properties allow you to specify the query string parameter and header names, respectively, that you want to vary the cache policy by. The [SetVaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setvarybycustom.aspx) method allows you to define custom strings by which to vary the output cache.

**How to: Set the Cacheability of an ASP.NET Page Declaratively**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/zd1ysf1y(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/zd1ysf1y(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/zd1ysf1y(d=printer,v=vs.80).aspx)

The cacheability of a page or user control refers to whether or not a page can be cached on a device during its response life cycle. These devices include the client (browser) making the request, the Web server responding to the request, and any cache-capable devices, such as proxy servers, that are in the request or response stream.

If you know at design time what cacheability setting you need for a page, you can set cacheability declaratively. The page will then use the same cacheability settings for all requests. For more information, see [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx).

**To set a page's cacheability declaratively**

1. Include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive in the page and define [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) and [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attributes.
2. Include a [Location](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.location.aspx) attribute in the @ OutputCache directive and define its value as one of the following values in the [OutputCacheLocation](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx) enumeration: [Any](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx), [Client](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx), [Downstream](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx), [Server](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx), [ServerAndClient](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx), or [None](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx).

The following code shows how to set the page's cacheability to 60 seconds:

<%@ OutputCache Duration="60" VaryByParam="None"%>

|  |
| --- |
| **NoteNote** |
| The default setting is [Any](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx). If you do not define a [Location](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.location.aspx) attribute, the page output can be cached on all cache-capable network devices that are involved in the response. These include the requesting client, the origin server, and any proxy servers through which the response passes. |

**To set a page's cacheability declaratively using a cache profile**

1. Define a cache profile in your application's Web.config file and in the profile, include duration and varyByParam settings.

The following [<caching>](http://msdn.microsoft.com/en-us/library/ms164606.aspx) configuration element defines a cache profile named Cache30Seconds, which will cache the page on the server for 30 seconds.

<caching>

<outputCacheSettings>

<outputCacheProfiles>

<add name="Cache30Seconds" duration="30"

varyByParam="none" />

</outputCacheProfiles>

</outputCacheSettings>

</caching>

1. Include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx)directive in each ASP.NET page that uses the profile, and set the [CacheProfile](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.cacheprofile.aspx) attribute to the name of the cache profile defined in your Web.config file.

The following code specifies that the page should use the cache profile named Cache30Seconds:

<%@ OutputCache CacheProfile="Cache30Seconds" %>

**Setting the Cacheability of a Page**

**.NET Framework 4**

The cacheability of a page or user control refers to whether a page can be cached on a device during the page's response life cycle. Devices that can cache a page include the browser making the request, the Web server responding to the request, and any other cache-capable devices, such as proxy servers, that are in the request or response stream.

When a Web server sends a response to the requesting browser, the server includes in the response a Cache-Control field in the HTTP header that defines the devices on which the page can be cached. Depending on the needs of your application, you can define which devices should or should not cache individual ASP.NET pages. For example, you might want the cacheability settings for a user logon page to be different from those for a page that displays a selection of products from a catalog. In the case of the logon page, for security reasons you might want to cache the page only on the server, while the catalog page can be cached on any device.

For ASP.NET pages, you can set cacheability by using values in the [HttpCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) enumeration. The enumeration has the following values. The first three map directly to Cache-Control HTTP header settings, and the last three are special values.

* [NoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Specifies that the device making the request should get the response from the Web server each time.
* [Public](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Allows the response to be cached by clients and shared (proxy) caches.
* [Private](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Specifies that the response is cacheable only on the client and not by shared (proxy server) caches.
* [Server](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Specifies that the response is cached only at the origin server.
* [ServerAndNoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Applies the settings of both Server and NoCache to indicate that the content is cached at the server but all others are explicitly denied the ability to cache the response.
* [ServerAndPrivate](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx)   Specifies that a response should be cached only on the origin server and on the requesting client; proxy servers are not allowed to cache the response.

You can set a page's cacheability declaratively by including a [Location](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.location.aspx) attribute in the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive and specifying one of the [OutputCacheLocation](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation.aspx) enumeration values. You can also set a page's cacheability programmatically using the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) method to specify an [HttpCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) value for the page. The method is accessible through the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) class.

|  |
| --- |
| **NoteNote** |
| If you use the @ OutputCache directive to set your page's cacheability, you must declare the [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute and either the [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) attribute or the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute along with the [Location](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.location.aspx) attribute. The [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute must be set to a value larger than zero. You can set the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute to "None" if you do not want to use the functionality of the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) or [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) parameters. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx) and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx). |

As an alternative to setting a page's cacheability using the @ OutputCache directive, you can create a cache profile in your application's Web.config file and then reference the profile in your page. For more information, see [Cache Configuration in ASP.NET](http://msdn.microsoft.com/en-us/library/ms178606.aspx).

**How to: Set a Page's Cacheability Programmatically**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/z852zf6b(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/z852zf6b(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/z852zf6b(d=printer,v=vs.80).aspx)

The cacheability of a page or user control refers to whether a page can be cached on a device during the page's response life cycle. Devices that can cache a page include the browser making the request, the Web server responding to the request, and any cache-capable devices, such as proxy servers, that are in the request or response stream.

You can set cacheability programmatically if your application will determine cacheability based on run-time conditions, such as reading the request header. For more information, see [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx).

**To set a page's cacheability programmatically**

* In the page's code, call the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) method on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) object.

The following code sets the Cache-Control HTTP header to [Public](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx).

C#

[VB](http://msdn.microsoft.com/en-us/library/z852zf6b(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

Response.Cache.SetCacheability(HttpCacheability.Public);

If you pass either [NoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) or [ServerAndNoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) to the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) method to prevent a requesting browser from caching a page in its History folder, any time a user clicks a back or forward button, a new version of the response will be requested. You can override this behavior conditionally by calling the [SetAllowResponseInBrowserHistory](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setallowresponseinbrowserhistory.aspx) method on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.ui.page.cache.aspx) property and passing true for the allow parameter.

If you set cacheability to any value other than [NoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) or [ServerAndNoCache](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx), ASP.NET ignores the value set by the [SetAllowResponseInBrowserHistory](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setallowresponseinbrowserhistory.aspx) method.

**How to: Cache Versions of a Page Using Requesting Browser**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/77a0146y(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/77a0146y(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/77a0146y(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/77a0146y(d=printer,v=vs.71).aspx)

If your Web application contains a page that creates different output based on the type of the requesting browser, you can cache versions of page's output by the major version of the browser that requests the page. For example, when an Internet Explorer 6 browser requests a page, one version of the page is cached. When a Netscape Navigator browser, another version of Internet Explorer, or any other browser requests the page, another version of the page is added to the output cache.

|  |
| --- |
| **NoteNote** |
| Major version and browser type information is passed through the [MajorVersion](http://msdn.microsoft.com/en-us/library/system.web.configuration.httpcapabilitiesbase.majorversion.aspx) property of the [HttpBrowserCapabilities](http://msdn.microsoft.com/en-us/library/system.web.httpbrowsercapabilities.aspx) object in the current request. For more information, see [How to: Detect Browser Types and Browser Capabilities in ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/3yekbd5b.aspx). |

**To cache multiple versions of a page declaratively based on browser type**

1. In the ASP.NET page, include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive with the required [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) and [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) or [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) attributes. The [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute must be set to an integer greater than zero. If you want to cache only by browser type, set the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute to "None".
2. In the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, include the [VaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycustom.aspx) attribute and set it to "browser".

The following example will cause the page to be cached for 10 seconds. The output will vary by the browser type.

<%@ OutputCache Duration="10" VaryByParam="None" VaryByCustom="browser" %>

**To cache multiple versions of a page programmatically based on browser type.**

1. In the page code, call the [SetExpires](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setexpires.aspx) and [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) methods on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the page's [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) property.
2. Call the [SetVaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setvarybycustom.aspx) method, passing the value "browser" in the custom parameter.

The following code example shows how to cache multiple versions of a page for one minute. The output will vary by the type of browser that made the request.

C#

protected void Page\_Load(object sender, EventArgs e)

{

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1d));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

Response.Cache.SetVaryByCustom("browser");

}

VB

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

Response.Cache.SetVaryByCustom("browser")

End Sub

**How to: Cache Versions of a Page Using Parameters**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/y96218s9(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/y96218s9(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/y96218s9(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/y96218s9(d=printer,v=vs.71).aspx)

There are times when you want to cache a page, but the page might generate different output based on the value of a query string parameter or values sent with the page on postback. For example, if you have a page that shows locations of stores based on a user's state, the page might accept a "state" query string value that generates a different version of the page for each state.

You can cache multiple versions of page responses based on the parameters sent as query string values or form post values.

|  |
| --- |
| **NoteNote** |
| ASP.NET treats query string values or form post values with identical key/value pairs as identical for caching purposes, regardless of the order in which the parameters are passed. However, for caching purposes, parameter names are case-sensitive and ASP.NET will cache different versions of a page for uppercase and lowercase parameter names and values. |

**To cache multiple versions of page output declaratively using parameters**

1. In the ASP.NET page, include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive with a Duration attribute. The Duration attribute is required and must be set to an integer greater than zero.
2. In the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, include a VaryByParam attribute and set its value to the name of the query string or form post parameter that you want to vary the page by.

The following code example caches the page for 60 seconds and specifies that different versions of the page output will be cached based on the City query string value or form post parameter.

<%@ OutputCache Duration="60" VaryByParam="City" %>

|  |
| --- |
| **NoteNote** |
| If you want to vary the output cache by multiple parameters, include a list of parameter names separated by semicolons (;). If you want to vary the cache by all parameter values, set the VaryByParam attribute to an asterisk (**\***). The following code example shows how to vary the page output by City and ZipCode parameters. |

<%@ OutputCache Duration="60" VaryByParam="City;ZipCode" %>

**To cache multiple versions of page output programmatically using parameters**

1. In the page's Page\_Load event, call the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) and [SetExpires](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setexpires.aspx) methods on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) object.
2. Specify the parameter name as the argument for the [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) object's [VaryByParams](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyparams.aspx) property, and set the property to true.

The following code example shows how to cache multiple versions of a page when requests arrive at the server with different values for the Zip parameter.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_120e4adf-3a19-497b-b419-849425f92d1b');)

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

Response.Cache.VaryByParams["Zip"] = true;

VB

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

Response.Cache.VaryByParams("Zip") = True

|  |
| --- |
| **NoteNote** |
| If you want to vary the cached content by multiple parameters, set the [VaryByParams](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyparams.aspx) property multiple times. If you want to vary the cached content by all header values, set the [VaryByHeader](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyheader.aspx) attribute to an asterisk (\*). The following code example shows how to vary the page output by City and Zip parameters. |

C#

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

Response.Cache.VaryByParams["City"] = true;

Response.Cache.VaryByParams["Zip"] = true;

VB

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

Response.Cache.VaryByParams("City") = true

Response.Cache.VaryByParams("Zip") = true

**How to: Cache Versions of a Page Using HTTP Headers**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/a6s4ks5w(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/a6s4ks5w(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/a6s4ks5w(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/a6s4ks5w(d=printer,v=vs.71).aspx)

ASP.NET allows you to cache multiple versions of a page depending on the value of an HTTP header that you specify. You can specify caching by a single header, multiple headers, or all headers passed to your application when the page is requested.

**To cache versions of a page declaratively based on HTTP header values**

1. In the ASP.NET page, include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive with the required [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) and [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) or [VaryByControl](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycontrol.aspx) attributes. The [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute must be set to an integer greater than zero. If you want to cache only by HTTP header values, you must set the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute to "None".
2. In the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, include the [VaryByHeader](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyheader.aspx) attribute and set its value to the name of the HTTP header that you want to vary the cache content by.

The following example caches the page for 60 seconds and sets versions of a page to be cached based on the value passed with the Accept-Language HTTP header:

<%@ OutputCache Duration="60" VaryByParam="None" VaryByHeader="Accept-Language" %>

|  |
| --- |
| **NoteNote** |
| If you want to vary the cached content by multiple headers, include a list of header names separated by semicolons (;). If you want to vary the cached content by all header values, set the [VaryByHeader](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyheader.aspx) attribute to an asterisk (\*). |

**To cache versions of a page programmatically based on an HTTP header value**

1. In the page's Page\_Load method, call the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) and [SetExpires](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setexpires.aspx) methods on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the page's [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) object.
2. Set the value for the HTTP header in the [VaryByHeaders](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyheaders.aspx) property to true.

The following code example shows how to cache multiple versions of a page for one minute for requests with different values for the Accept-Language HTTP header.

C#

protected void Page\_Load(object sender, EventArgs e)

{

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1d));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

Response.Cache.VaryByHeaders["Accept-Language"] = true;

}

VB

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Response.Cache.SetExpires(DateTime.Now.AddMinutes(1.0))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

Response.Cache.VaryByHeaders("Accept-Language") = true

End Sub

|  |
| --- |
| **NoteNote** |
| If you want to vary the cached content by multiple headers, you need to set multiple values in the [VaryByHeaders](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyheaders.aspx) property. If you want to vary by all headers, set [VaryByHeaders](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.varybyheaders.aspx)["[VaryByUnspecifiedParameters](http://msdn.microsoft.com/en-us/library/system.web.httpcachevarybyheaders.varybyunspecifiedparameters.aspx)"] to true. |

**How to: Cache Versions of a Page Using Custom Strings**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/5ecf4420(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/5ecf4420(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/5ecf4420(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/5ecf4420(d=printer,v=vs.71).aspx)

In addition to varying the output cache by browser type and parameters, you can cache multiple versions of page output based on different strings returned by a method that you define.

When you cache pages based on custom strings, you first specify an identifier for the custom string to use. You then create a method in the application's Global.asax file that accepts the identifier and returns a value to vary the output cache by.

**To cache multiple versions of page output based on custom strings**

1. In the ASP.NET page, include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive with the required [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) and [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attributes. The [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute must be set to an integer greater than zero. If you do not want to use the functionality provided by the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute, you must set its value to "None".
2. To set the custom string declaratively, in the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, include the [VaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycustom.aspx) attribute that is set to the string that you want to vary the output cache by.

The following directive varies the page output by the custom string "minorversion".

<%@ OutputCache Duration="10" VaryByParam="None" VaryByCustom="minorversion" %>

1. To set the custom string programmatically, call the [SetVaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setvarybycustom.aspx) method and pass it the custom string to use.

The following code example shows how to set the custom string to "minorversion".

C#

Response.Cache.SetVaryByCustom("minorversion");

VB

Response.Cache.SetVaryByCustom("minorversion")

|  |
| --- |
| **NoteNote** |
| If you attempt to set the custom string both programmatically and declaratively, you will get an [InvalidOperationException](http://msdn.microsoft.com/en-us/library/system.invalidoperationexception.aspx). You need to choose one approach or the other. |

1. In the application's Global.asax file, override the [GetVaryByCustomString](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getvarybycustomstring.aspx) method to specify the behavior of the output cache for the custom string.

As its arg parameter, the overridden method accepts the string that you set in the [VaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycustom.aspx) attribute or in the [SetVaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setvarybycustom.aspx) method. For example, you might have pages that are cached by the minor version of the requesting browser. For these pages you can set the [VaryByCustom](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybycustom.aspx) attribute to "minorversion". Then, in the overridden [GetVaryByCustomString](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getvarybycustomstring.aspx) method, you can check the arg parameter and return different strings depending on whether the value of the arg parameter is "minorversion".

The following code example shows a Global.asax file with an override of the [GetVaryByCustomString](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getvarybycustomstring.aspx) method.

C#

<%@ Application language="C#" %>

<script runat="server">

public override string GetVaryByCustomString(HttpContext context,

string arg)

{

if(arg == "minorversion")

{

return "Version=" +

context.Request.Browser.MinorVersion.ToString();

}

return base.GetVaryByCustomString(context, arg);

}

</script>

VB

<script runat="server">

Public Overrides Function GetVaryByCustomString(context \_

As HttpContext, arg As String) As String

If (arg = "minorversion") Then

Return "Version=" & \_

context.Request.Browser.MinorVersion.ToString()

return base.GetVaryByCustomString(context, arg);

End Function

</script>

**How to: Set Expiration Values for ASP.NET Page Caching**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/y18he7cw(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/y18he7cw(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/y18he7cw(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/y18he7cw(d=printer,v=vs.71).aspx)

To cause a page to be added to the output cache, you establish an expiration policy for that page. You can do this declaratively or programmatically.

**To set output-cache expirations for a page declaratively**

* Include an [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive in the ASP.NET page (.aspx file) whose response you want to cache. Set the [Duration](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.duration.aspx) attribute to a positive numeric value, and set the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute to a value.

|  |
| --- |
| **NoteNote** |
| The [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive sets the Cache-Control header to Any by default. |

For example, the following @ OutputCache directive sets the page's expiration to 60 seconds:

<%@ OutputCache Duration="60" VaryByParam="None" %>

|  |
| --- |
| **NoteNote** |
| You must include a [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute when using the @ OutputCache directive or a parser error will occur. If you do not want to use the functionality offered by the [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute, set its value to "None". For more information, see [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx). |

**To set output-cache expirations for a page programmatically**

* In the page's code, set the expiration policy for the page on the [Cache](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cache.aspx) property of the [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) object.

|  |
| --- |
| **NoteNote** |
| If you set expirations for a page programmatically, you must set the Cache-Control header for the cached page as well. To do so, call the [SetCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.setcacheability.aspx) method and pass it the [HttpCacheability](http://msdn.microsoft.com/en-us/library/system.web.httpcacheability.aspx) enumeration value Public. |

The following code example sets the same cache policy as the @ OutputCache directive does in the preceding procedure.

C#

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

VB

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

When the cached page expires, the subsequent request for the page causes a dynamically generated response. This response page is cached for the specified duration.

**How to: Check the Validity of a Cached Page**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/a5e5hdyz(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/a5e5hdyz(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/a5e5hdyz(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/a5e5hdyz(d=printer,v=vs.71).aspx)

When a cached page is requested by a user, ASP.NET determines whether the cached output is still valid based on the cache policy you have defined in the page. If the output is valid, the cached output is sent to the client and the page is not re-processed. However, ASP.NET provides you with the ability to run code during this validation check using a validation callback, so that you can write custom logic to check whether the page is valid. The validation callback allows you to invalidate cached pages outside of the normal process of using cache dependencies.

**To programmatically check the validity of a cached page**

1. Define an event handler of type [HttpCacheValidateHandler](http://msdn.microsoft.com/en-us/library/system.web.httpcachevalidatehandler.aspx) and include code that checks the validity of the cached page response.

The validation handler must return one of the following [HttpValidationStatus](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx) values:

* + [Invalid](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx)   Indicates that the cached page is invalid, the page is evicted from the cache, and the request is handled as a cache miss.
  + [IgnoreThisRequest](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx)   Causes the request to be treated as a cache miss. The page is therefore processed again, but the cached page is not invalidated.
  + [Valid](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx)   Indicates that the cached page is valid.

The following code example illustrates a validation handler named ValidateCacheOutput that determines whether the query string variable status contains the values "invalid" or "ignore". If the status value is "invalid", the method returns [Invalid](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx) and the page is invalidated in the cache. If the status value is "ignore", the method returns [IgnoreThisRequest](http://msdn.microsoft.com/en-us/library/system.web.httpvalidationstatus.aspx) and the page is left in the cache but a new response is generated for this request.

C#

public static void ValidateCacheOutput(HttpContext context, Object data,

ref HttpValidationStatus status)

{

if (context.Request.QueryString["Status"] != null)

{

string pageStatus = context.Request.QueryString["Status"];

if (pageStatus == "invalid")

status = HttpValidationStatus.Invalid;

else if (pageStatus == "ignore")

status = HttpValidationStatus.IgnoreThisRequest;

else

status = HttpValidationStatus.Valid;

}

else

status = HttpValidationStatus.Valid;

}

VB

Public Shared Sub ValidatePage(ByVal context As HttpContext, \_

ByVal data As [Object], ByRef status As HttpValidationStatus)

If Not (context.Request.QueryString("Status") Is Nothing) Then

Dim pageStatus As String = context.Request.QueryString("Status")

If pageStatus = "invalid" Then

status = HttpValidationStatus.Invalid

ElseIf pageStatus = "ignore" Then

status = HttpValidationStatus.IgnoreThisRequest

Else

status = HttpValidationStatus.Valid

End If

Else

status = HttpValidationStatus.Valid

End If

End Sub

1. From one of the page life-cycle events (such as the page's [Load](http://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event), call the [AddValidationCallback](http://msdn.microsoft.com/en-us/library/system.web.httpcachepolicy.addvalidationcallback.aspx) method, passing as the first argument the event handler you defined in step 1.

The following code example sets the ValidateCacheOutput method to be the validation handler.

C#

protected void Page\_Load(object sender, EventArgs e)

{

Response.Cache.AddValidationCallback(

new HttpCacheValidateHandler(ValidateCacheOutput),

null);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, \_

ByVal e As System.EventArgs) Handles Me.Load

Response.Cache.AddValidationCallback( \_

New HttpCacheValidateHandler(AddressOf ValidatePage), Nothing)

End Sub

**How to: Cache Page Output with File Dependencies**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.71).aspx)

At times you might want to remove a page from the output cache when a file changes. For example, you might have a page that gets its contents from a process-intensive report that produces an XML file as output. The page needs to be reprocessed only if the XML file changes. To limit reprocessing to just those times when it is necessary, you can use the make the page's cache policy dependent on a single file. If required, you can make the cached page dependent on more than one file.

|  |
| --- |
| **NoteNote** |
| You can explicitly remove any page from the output cache by calling the [RemoveOutputCacheItem](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.removeoutputcacheitem.aspx) method. You can do this from the Global.asax file, from a custom ASP.NET server control, or from a page, depending on the needs of your application. |

**To make cached page output dependent upon a file**

1. Specify the settings for caching page output either declaratively or programmatically. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx), [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx), and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).
2. In the page code, call the [AddFileDependency](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addfiledependency.aspx) method. As the method's filename parameter, pass the path of the file on which you are creating a dependency.

The following code example sets a file dependency on the TextFile1.txt file. When the file changes, the page output will be removed from the cache.

C#

protected void Page\_Load(object sender, EventArgs e)

{

string fileDependencyPath = Server.MapPath("TextFile1.txt");

Response.AddFileDependency(fileDependencyPath);

// Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, \_

ByVal e As EventArgs) Handles Me.Load

Dim fileDependencyPath As String = \_

Server.MapPath("TextFile1.txt")

Response.AddFileDependency(fileDependencyPath)

' Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

End Sub

|  |
| --- |
| **NoteNote** |
| You cannot use these methods from an ASP.NET user control. However, in any user control that specifies the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive you can create a file dependency and assign it to the [Dependency](http://msdn.microsoft.com/en-us/library/system.web.ui.basepartialcachingcontrol.dependency.aspx) property. |

**To make cached page output dependent on a group of files**

1. Specify the settings for caching page output either declaratively or programmatically. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx), [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx), and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).
2. In the page code, create a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx) array or an [ArrayList](http://msdn.microsoft.com/en-us/library/system.collections.arraylist.aspx) that contains the paths of the files to make the page dependent on.
3. Call the [AddFileDependencies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addfiledependencies.aspx) method and as the filenames parameter, pass the array.

The following code example creates a string array of the file paths for the TextFile1.txt and XMLFile1.xml files and makes the page output dependent on the two files. If either one of the files is modified, the page output will be removed from the cache.

C#

protected void Page\_Load(object sender, EventArgs e)

{

string[] fileDependencies;

string fileDependency1 = Server.MapPath("TextFile1.txt");

string fileDependency2 = Server.MapPath("XMLFile1.xml");

fileDependencies = new String[] { fileDependency1,

fileDependency2 };

Response.AddFileDependencies(fileDependencies);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim fileDependencies() As String

Dim fileDependency1 As String = Server.MapPath("TextFile1.txt")

Dim fileDependency2 As String = Server.MapPath("XMLFile1.xml")

fileDependencies = New String() {fileDependency1, \_

fileDependency2}

Response.AddFileDependencies(fileDependencies)

End Sub

**How to: Cache Page Output with Cache Key Dependencies**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/t1d120ks(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/t1d120ks(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/t1d120ks(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/t1d120ks(d=printer,v=vs.71).aspx)

At times, you might want to remove a page from the output cache when another item in the cache is removed. For example, you might have a page that displays a process-intensive report that is placed in the application cache and used by multiple pages. When the report is changed or is removed from cache, you want the page output to be removed from the cache also because the report is no longer valid. To do this you can make cached page output dependent on other cached items.

|  |
| --- |
| **NoteNote** |
| You can explicitly remove any page from the output cache by calling the [RemoveOutputCacheItem](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.removeoutputcacheitem.aspx) method. You can do this from the Global.asax file, from a custom ASP.NET server control, or from a page, depending on the needs of your application. |

**To make cached page output dependent upon another cache item**

1. In a page, specify cache settings either declaratively or programmatically. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx), [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx), and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).
2. In page code, call the [AddCacheItemDependency](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addcacheitemdependency.aspx) method. As the cacheKey parameter, pass the name of the cache item on which to create a dependency.

The following code example shows how to set a dependency on the item named ProcessIntensiveReport. When this item is modified or removed, the page output will be removed from the cache.

C#

protected void Page\_Load(object sender, EventArgs e)

{

Response.AddCacheItemDependency("ProcessIntensiveReport");

// Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Response.AddCacheItemDependency("ProcessIntensiveReport")

' Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

End Sub

|  |
| --- |
| **NoteNote** |
| You cannot call the [AddCacheItemDependency](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addcacheitemdependency.aspx) method in an ASP.NET user control. However, in any user control that specifies the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive, you can create a [CacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.cachedependency.aspx) object that describes the cache key dependency and assign it to the [Dependency](http://msdn.microsoft.com/en-us/library/system.web.ui.basepartialcachingcontrol.dependency.aspx) property of the [UserControl](http://msdn.microsoft.com/en-us/library/system.web.ui.usercontrol.aspx) object. |

**Caching in ASP.NET with the SqlCacheDependency Class**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178604(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178604(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178604(d=printer,v=vs.80).aspx)

ASP.NET allows you to use the [SqlCacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependency.aspx) class to create a cache item dependency on a table or row in a database. When a change occurs in the table or in a specific row, the item that has a dependency is invalidated and removed from the cache. You can set a dependency on a table in Microsoft SQL Server 7.0, SQL Server 2000, and SQL Server 2005. If you are using SQL Server 2005 you can also set a dependency on a specific record.

Using caching with a SQL dependency can dramatically increase application performance in certain scenarios. For example, imagine you are building an e-commerce application that displays product information from a database. Without caching, your application must request the data from the database each time a user wants to view a product. You could cache the product information for a day at a time, ensuring fast response times because the product information is already in memory. However, if product information changes, the cached product information might then be out of sync with the data for up to a day.

Using SQL cache dependency, you could cache your product information and create a dependency on a database table or row change. When the data changes—and only then—the cache items based on that data are invalidated and removed from the cache. The next time you request that item from the cache, if it is not in the cache, you can re-add the updated version to the cache and be assured that you have the latest data.

SQL cache dependency is also available for the page output cache. For example, you could create a page named ViewProduct.aspx that shows information about a particular product. You could set that page's cache policy to be a SQL dependency as you would for an item you added manually to the cache. The page would then be stored in the cache until the table or row that it was dependent on changed. When the data changed, the page would be recreated and stored in the output cache again.

For more information see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx).

[Features](javascript:void(0))

ASP.NET SQL cache dependency offers the following features:

* You can use SQL cache dependency for both the application cache and the page output cache.
* You can use SQL cache dependency with SQL Server 7.0 and later versions.
* You can use SQL cache dependency in a Web garden (multiple processors on one server) or a Web farm (multiple servers running the same application).
* The database operations associated with SQL cache dependency are simple and therefore do not incur a heavy processing cost on the server.
* You do not need extensive SQL knowledge to configure SQL cache dependency in your application and in SQL Server. ASP.NET includes tools that automate the configuration. Additionally, you can use the [SqlCacheDependencyAdmin](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependencyadmin.aspx) class to programmatically configure SQL cache dependency.

[SQL Server 7.0 and SQL Server 2000 Implementation](javascript:void(0))

ASP.NET implements a poll model for SQL Server 7.0 and SQL Server 2000 cache dependency. A thread within the ASP.NET process polls the SQL Server database at a specified time interval to determine whether the data has changed. If so, dependent cache items are invalidated and removed from the cache. You can specify the poll interval in your application declaratively in the Web.config file or programmatically using the [SqlCacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependency.aspx) class.

SQL cache dependency is limited to data changes at the table level for SQL Server 7.0 and SQL Server 2000. You can configure ASP.NET to poll the database for changes in a table, but not in a specific row.

**Enabling SQL Caching**

In order to use SQL cache dependency in SQL Server 7.0 and SQL Server 2000, you must configure SQL Server to support it. ASP.NET provides utilities to configure SQL caching on SQL Server, including a tool named Aspnet\_regsql.exe and the [SqlCacheDependencyAdmin](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependencyadmin.aspx) class. For more information on enabling SQL cache dependency with SQL Server see [How to: Cache Page Output with Cache Key Dependencies](http://msdn.microsoft.com/en-us/library/t1d120ks.aspx).

[SQL Server 2005 Implementation](javascript:void(0))

SQL Server 2005 implements a different model for cache dependency than SQL Server 7.0 and SQL Server 2000. You do not need to go through any special configuration steps to enable SQL cache dependency on SQL Server 2005. Additionally, SQL Server 2005 implements a change notification model where notifications are sent to subscribing application servers, rather than relying on the polling model required in earlier versions of SQL Server.

SQL Server 2005 cache dependency is more flexible in the types of changes that receive notification. SQL Server 2005 monitors changes to the result set of a particular SQL command. If a change occurs in the database that would modify the results set of that command, the dependency causes the cached item to be invalidated. This allows SQL Server 2005 to provide row-level notification.

There are some requirements for the query used to test for changes. You must provide fully qualified table names, including the owner name (for example, dbo.authors). In general, SQL 2005 notification supports Select queries as well as stored procedures, and supports multiple and nested queries, but does not support aggregate operations such as COUNT(\*). For more information about what queries are supported and the rules for notification in SQL Server 2005, see the SQL Books Online topic named "Creating a Query for Notification."

[Configuring SQL Caching in ASP.NET Application](javascript:void(0))

Once you have configured SQL Server 7.0 or SQL Server 2000 for cache dependencies, or have created the appropriate command dependency in SQL Server 2005, you can configure your application to use SQL cache dependency just as you would configure any other cache dependency. For example, you can create a cache profile in the Web.config file and then reference that cache profile on each page that should use the SQL cache dependency. You could also use SQL cache dependency by enabling it programmatically using the [SqlCacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependency.aspx) class. For more information, see [How to: Cache Page Output with Cache Key Dependencies](http://msdn.microsoft.com/en-us/library/t1d120ks.aspx).

**How to: Cache Page Output with File Dependencies**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/67z4z916(d=printer,v=vs.71).aspx)

At times you might want to remove a page from the output cache when a file changes. For example, you might have a page that gets its contents from a process-intensive report that produces an XML file as output. The page needs to be reprocessed only if the XML file changes. To limit reprocessing to just those times when it is necessary, you can use the make the page's cache policy dependent on a single file. If required, you can make the cached page dependent on more than one file.

|  |
| --- |
| **NoteNote** |
| You can explicitly remove any page from the output cache by calling the [RemoveOutputCacheItem](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.removeoutputcacheitem.aspx) method. You can do this from the Global.asax file, from a custom ASP.NET server control, or from a page, depending on the needs of your application. |

**To make cached page output dependent upon a file**

1. Specify the settings for caching page output either declaratively or programmatically. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx), [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx), and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).
2. In the page code, call the [AddFileDependency](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addfiledependency.aspx) method. As the method's filename parameter, pass the path of the file on which you are creating a dependency.

The following code example sets a file dependency on the TextFile1.txt file. When the file changes, the page output will be removed from the cache.

C#

protected void Page\_Load(object sender, EventArgs e)

{

string fileDependencyPath = Server.MapPath("TextFile1.txt");

Response.AddFileDependency(fileDependencyPath);

// Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, \_

ByVal e As EventArgs) Handles Me.Load

Dim fileDependencyPath As String = \_

Server.MapPath("TextFile1.txt")

Response.AddFileDependency(fileDependencyPath)

' Set additional properties to enable caching.

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))

Response.Cache.SetCacheability(HttpCacheability.Public)

Response.Cache.SetValidUntilExpires(True)

End Sub

|  |
| --- |
| **NoteNote** |
| You cannot use these methods from an ASP.NET user control. However, in any user control that specifies the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive you can create a file dependency and assign it to the [Dependency](http://msdn.microsoft.com/en-us/library/system.web.ui.basepartialcachingcontrol.dependency.aspx) property. |

**To make cached page output dependent on a group of files**

1. Specify the settings for caching page output either declaratively or programmatically. For more information, see [How to: Set Expiration Values for ASP.NET Page Caching](http://msdn.microsoft.com/en-us/library/y18he7cw.aspx), [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx), and [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.aspx).
2. In the page code, create a [String](http://msdn.microsoft.com/en-us/library/system.string.aspx) array or an [ArrayList](http://msdn.microsoft.com/en-us/library/system.collections.arraylist.aspx) that contains the paths of the files to make the page dependent on.
3. Call the [AddFileDependencies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.addfiledependencies.aspx) method and as the filenames parameter, pass the array.

The following code example creates a string array of the file paths for the TextFile1.txt and XMLFile1.xml files and makes the page output dependent on the two files. If either one of the files is modified, the page output will be removed from the cache.

C#

protected void Page\_Load(object sender, EventArgs e)

{

string[] fileDependencies;

string fileDependency1 = Server.MapPath("TextFile1.txt");

string fileDependency2 = Server.MapPath("XMLFile1.xml");

fileDependencies = new String[] { fileDependency1,

fileDependency2 };

Response.AddFileDependencies(fileDependencies);

}

VB

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim fileDependencies() As String

Dim fileDependency1 As String = Server.MapPath("TextFile1.txt")

Dim fileDependency2 As String = Server.MapPath("XMLFile1.xml")

fileDependencies = New String() {fileDependency1, \_

fileDependency2}

Response.AddFileDependencies(fileDependencies)

End Sub

**Cache Configuration in ASP.NET**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178606(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178606(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178606(d=printer,v=vs.80).aspx)

ASP.NET provides many options for configuring page output caching and the cache API. You use the page output cache to cache page responses after they have been processed. You use the cache API to programmatically cache application data. For more information, see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx).

[Page Output Cache Configuration](javascript:void(0))

You can configure page output caching in these places:

* Configuration files   You can configure page output cache settings in any configuration file in the application configuration hierarchy, including the Machine.config file (to make settings for all Web applications on the computer) and your application-specific Web.config file (to make settings for a single application).
* Individual pages   You can set caching options in individual pages either declaratively or programmatically. You can also apply cache profiles created in the configuration file to individual pages.
* User controls   You can set caching in individual user controls either declaratively or programmatically. This is an easy way to cache content within a page that is otherwise not cached.

**Web.config Cache Configuration Settings**

There are two top-level configuration sections for the page output cache in the Web.config file: the [OutputCacheSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcachesection.aspx) and the [OutputCacheSettingsSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcachesettingssection.aspx).

The [OutputCacheSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcachesection.aspx) section is used to configure application-scope settings, such as whether page output caching is enabled or disabled. For example, you can disable page output caching for the entire application by adding enableOutputCache="false" to the [OutputCacheSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcachesection.aspx) in your Web.config file. Settings in the configuration file take precedence over cache settings in individual pages, so the example setting means that output cache will not be used.

The [OutputCacheSettingsSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcachesettingssection.aspx) is used to configure profiles and dependencies that can be used by individual pages. For example, the following code creates an [OutputCacheProfile](http://msdn.microsoft.com/en-us/library/system.web.configuration.outputcacheprofile.aspx) named CacheProfile1 that will cache the implementing page for 60 seconds:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_c766bc39-ff68-422e-9edb-bdd4f251d74e');)

<outputCacheSettings>

<outputCacheProfiles>

<add name="CacheProfile1" duration="60" />

</outputCacheProfiles>

</outputCacheSettings>

**Machine.config Cache Configuration Settings**

The configuration sections for the Machine.config file are the same as for the Web.config file, except that you can lock configuration settings in the Machine.config file so that they cannot be overridden by individual applications at any level. This might be necessary in a shared hosting scenario in which the hoster does not want individual applications modifying the cache configuration. For more information see [How to: Lock ASP.NET Configuration Settings](http://msdn.microsoft.com/en-us/library/ms178693.aspx).

**Page Cache Configuration Settings**

You can configure caching in individual pages by applying cache profiles that have been defined in a configuration file. Alternatively, you can configure individual cache properties in the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive or by setting attributes in the page's class definition. For more information see [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) and [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx).

**User Control Cache Configuration Settings**

You can configure user control caching by setting the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive in the user control file or by setting the [PartialCachingAttribute](http://msdn.microsoft.com/en-us/library/system.web.ui.partialcachingattribute.aspx) attribute in the control's class definition. For more information, see [Caching Portions of an ASP.NET Page](http://msdn.microsoft.com/en-us/library/h30h475z.aspx).

[Cache API Configuration Settings](javascript:void(0))

You can configure the application's cache API in your Web.config file. As with the page output cache, application hosters can set configuration properties in the Machine.config file and lock cache configuration settings for all applications. The application cache API is configured in the [CacheSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.cachesection.aspx). For example, you can disable item expiration with the following configuration element:

<cache disableExpiration="true" />

You can also specify other application cache API configuration settings by assigning values to attributes such as [DisableExpiration](http://msdn.microsoft.com/en-us/library/system.web.configuration.cachesection.disableexpiration.aspx) and [DisableMemoryCollection](http://msdn.microsoft.com/en-us/library/system.web.configuration.cachesection.disablememorycollection.aspx) within the configuration file's [CacheSection](http://msdn.microsoft.com/en-us/library/system.web.configuration.cachesection.aspx).