.NET Framework 4 - ASP.NET

**ASP.NET Caching Overview**

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.

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

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).

http://i.msdn.microsoft.com/Global/Images/clear.gif Page Output Cache

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).

http://i.msdn.microsoft.com/Global/Images/clear.gif Extensible Output Caching

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:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl37_ctl00_ctl01_code');" \o "Copy Code)

<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:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl37_ctl00_ctl05_code');" \o "Copy Code)

<%@ 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).

http://i.msdn.microsoft.com/Global/Images/clear.gif Automatic Data Removal

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. |

http://i.msdn.microsoft.com/Global/Images/clear.gif Application Cache Item Removal Notification

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).

.NET Framework 4 - ASP.NET

**Caching ASP.NET Pages**

ASP.NET allows you to cache some or all of the response generated by an ASP.NET page, referred to in ASP.NET as output caching. You can cache the page at the browser making the request, at the Web server responding to the request, and at any other cache-capable devices, such as proxy servers, that are in the request or response stream. Caching provides a powerful way for you to increase the performance of your Web applications. Caching allows subsequent requests for a page to be satisfied from the cache so the code that initially creates the page does not have to be run again. Caching your site's most frequently accessed pages can substantially increase your Web server's throughput, commonly measured in requests per second.

You can specify cache settings declaratively in a page or configuration file, or programmatically using a cache API. For more information, see [Setting the Cacheability of a Page](http://msdn.microsoft.com/en-us/library/w9s3a17d.aspx).

You can cache pages based on the values of query string parameters or form variables (control values). Caching based on these types of values must be explicitly enabled by using the [@ OutputCache](http://msdn.microsoft.com/en-us/library/hdxfb6cy.aspx) directive's [VaryByParam](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcacheparameters.varybyparam.aspx) attribute. For more information, see [Caching Multiple Versions of a Page](http://msdn.microsoft.com/en-us/library/xadzbzd6.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 for the page. If the output is valid, the cached output is sent to the client and the page is not re-processed. ASP.NET allows you to run code during this validation check so that you can write custom logic to check whether the page is valid. For more information see [How to: Check the Validity of a Cached Page](http://msdn.microsoft.com/en-us/library/a5e5hdyz.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 a portion of a page. ASP.NET provides functionality to cache only portions of an ASP.NET page. For more information see [Caching Portions of an ASP.NET Page](http://msdn.microsoft.com/en-us/library/h30h475z.aspx).

.NET Framework 4 - ASP.NET

**Caching Application Data**

ASP.NET has a powerful, easy-to-use caching mechanism that allows you to store objects in memory that require extensive server resources to create. Caching these types of resources can significantly improve the performance of your application.

Caching is implemented by the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) class, with cache instances private to each application. The cache lifetime is tied to that of the application; when the application is restarted, the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object is recreated.

The [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) class has been designed for ease of use. You can place items in the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) and later retrieve them using simple key/value pairs. For examples of how to do this, see [How to: Add Items to the Cache](http://msdn.microsoft.com/en-us/library/18c1wd61.aspx) and [How to: Retrieve Values of Cached Items](http://msdn.microsoft.com/en-us/library/xhy3h9f9.aspx).

The [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) class offers powerful features that allow you to customize how items are cached and how long they are cached. For example, when system memory becomes scarce, the cache automatically removes seldom-used or low-priority items to free memory. This technique is referred to as scavenging, and is one of the ways that the cache ensures that out-of-date data does not consume valuable server resources.

You can instruct the [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object to give certain items priority over other items when it performs scavenging. To indicate an item's importance, you can specify one of the [CacheItemPriority](http://msdn.microsoft.com/en-us/library/system.web.caching.cacheitempriority.aspx) enumeration values when you add an item using the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) or [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) methods.

You can also establish an expiration policy for an item when you add it to the cache using the [Add](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.add.aspx) or [Insert](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.insert.aspx) methods. You can define the lifetime for an item by specifying an exact time the item will expire (an absolute expiration) using a [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx) value. Alternatively, you can specify a sliding expiration using a [TimeSpan](http://msdn.microsoft.com/en-us/library/system.timespan.aspx) value, which allows you to specify the elapsed time before the item expires based on the time it was last accessed. Once an item expires, it is removed from the cache. Attempts to retrieve its value will return **null** (**Nothing** in Visual Basic) unless the item is added to the cache again.

For volatile items that are stored in the cache, such as those that have regular data refreshes or those that are valid for only a set amount of time, you typically set an expiration policy that keeps those items in the cache as long as their data remains current. For example, if you are writing an application that tracks sports scores by obtaining the data from another Web site, you can cache the scores for a game as long as those scores do not change on the source Web site. In this case, you can set an expiration policy that is based on how often the other Web site updates the scores. You can write code that determines if an up-to-date score is in the cache. If the score is not up to date, the code can read the score from the source Web site and cache the new value.

Finally, ASP.NET allows you to define the validity of a cached item based on an external file or directory (a file dependency) or on another cached item (a key dependency). If the item with the associated dependency changes, the cached item is invalidated and removed from the cache. You can use this technique to remove items from the cache when their data source changes. For example, if you write an application that processes financial data from an XML file, you can insert the data from the file in the cache and maintain a dependency on that XML file. When the file is updated, the item is removed from the cache, your application rereads the XML file and puts the refreshed data into the cache.

|  |
| --- |
| **NoteNote** |
| The [Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx) object has no information about the content of the items it contains. It merely holds a reference to those objects. It also provides methods to track their dependencies and set expiration policies. |

For more information on how to use these features, see [How to: Add Items to the Cache](http://msdn.microsoft.com/en-us/library/18c1wd61.aspx).

.NET Framework 4 - ASP.NET

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

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_code');" \o "Copy Code)

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

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl17_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl19_code');" \o "Copy Code)

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

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl23_code');" \o "Copy Code)

string[] dependencies = { "CacheItem2" };

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

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

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl24_code');" \o "Copy Code)

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 Code](javascript:CopyCode('ctl00_MTCS_main_ctl25_code');" \o "Copy Code)

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

new System.Web.Caching.CacheDependency(

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

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl27_code');" \o "Copy Code)

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);

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl28_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_code');" \o "Copy Code)

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

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

System.Web.Caching.Cache.NoSlidingExpiration);

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl32_code');" \o "Copy Code)

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

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

new TimeSpan(0, 10, 0));

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl33_code');" \o "Copy Code)

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:

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl37_code');" \o "Copy Code)

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

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

System.Web.Caching.Cache.NoSlidingExpiration,

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

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl38_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl40_code');" \o "Copy Code)

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);

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl41_code');" \o "Copy Code)

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))

.NET Framework 4 - ASP.NET

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

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 Code](javascript:CopyCode('ctl00_MTCS_main_ctl02_code');" \o "Copy Code)

string cachedString;

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

if (cachedString == null)

{

cachedString = "Hello, World.";

Cache.Insert("CacheItem", cachedString);

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl03_code');" \o "Copy Code)

Dim cachedString As String

cachedString = CStr(Cache("CacheItem"))

If cachedString Is Nothing Then

cachedString = "Hello, World."

Cache.Insert("CacheItem", cachedString)

End If

.NET Framework 4 - ASP.NET

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

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.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

Cache.Remove("MyData1")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

Cache.Remove("MyData1");

.NET Framework 4 - ASP.NET

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

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

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.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl01_code');" \o "Copy Code)

Imports System

Imports System.Text

Imports System.Web

Imports System.Web.Caching

Public Class ReportManager

Private Shared \_lastRemoved As String = ""

Public Shared Function GetReport() As String

Dim report As String = CStr(HttpRuntime.Cache("MyReport"))

If report Is Nothing Then

report = GenerateAndCacheReport()

End If

Return report

End Function

Private Shared Function GenerateAndCacheReport() As String

Dim report As String = "Report Text. " & \_lastRemoved

HttpRuntime.Cache.Insert( \_

"MyReport", \_

report, \_

Nothing, \_

Cache.NoAbsoluteExpiration, \_

New TimeSpan(0, 0, 15), \_

CacheItemPriority.Default, \_

New CacheItemRemovedCallback(AddressOf ReportRemovedCallback))

Return report

End Function

Public Shared Sub ReportRemovedCallback(ByVal key As String, \_

ByVal value As Object, ByVal removedReason \_

As CacheItemRemovedReason)

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

End Sub

End Class

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl02_code');" \o "Copy Code)

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();

}

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl03_code');" \o "Copy Code)

<%@ Page Language="VB" AutoEventWireup="false" %>

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

<html >

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<%=ReportManager.GetReport()%>

</div>

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl04_code');" \o "Copy Code)

<%@ 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 >

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<%=ReportManager.GetReport() %>

</div>

</form>

</body>

</html>

.NET Framework 4 - ASP.NET

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

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

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

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

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

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).

.NET Framework 4 - ASP.NET

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

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, Client, Downstream, Server, ServerAndClient, or None.

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

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl13_code');" \o "Copy Code)

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

|  |
| --- |
| **NoteNote** |
| The default setting is Any. 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.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl18_code');" \o "Copy Code)

<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:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl21_code');" \o "Copy Code)

<%@ OutputCache CacheProfile="Cache30Seconds" %>

.NET Framework 4 - ASP.NET

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

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.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl06_code');" \o "Copy Code)

Response.Cache.SetCacheability(HttpCacheability.Public);

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

Response.Cache.SetCacheability(HttpCacheability.Public)

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

.NET Framework 4 - ASP.NET

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

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:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl06_code');" \o "Copy Code)

<%@ 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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_code');" \o "Copy Code)

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

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(true);

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl17_code');" \o "Copy Code)

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.

.NET Framework 4 - ASP.NET

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

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   Indicates that the cached page is invalid, the page is evicted from the cache, and the request is handled as a cache miss.
  + IgnoreThisRequest   Causes the request to be treated as a cache miss. The page is therefore processed again, but the cached page is not invalidated.
  + Valid   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 and the page is invalidated in the cache. If the status value is "ignore", the method returns IgnoreThisRequest and the page is left in the cache but a new response is generated for this request.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

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;

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl09_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

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

{

Response.Cache.AddValidationCallback(

new HttpCacheValidateHandler(ValidateCacheOutput),

null);

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl13_code');" \o "Copy Code)

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

.NET Framework 4 - ASP.NET

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

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

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);

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl18_code');" \o "Copy Code)

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);

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl19_code');" \o "Copy Code)

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

.NET Framework 4 - ASP.NET

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

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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

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);

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

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. |

.NET Framework 4 - ASP.NET

**Setting the Cacheability of a Page**

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   Specifies that the device making the request should get the response from the Web server each time.
* Public   Allows the response to be cached by clients and shared (proxy) caches.
* Private   Specifies that the response is cacheable only on the client and not by shared (proxy server) caches.
* Server   Specifies that the response is cached only at the origin server.
* ServerAndNoCache   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   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).