

# Response caching in ASP.NET Core

11/04/2019 • 8 minutes to read •      +8

## In this article

[HTTP-based response caching](#)

[HTTP-based caching respects request Cache-Control directives](#)

[Other caching technology in ASP.NET Core](#)

[ResponseCache attribute](#)

[Additional resources](#)

By [John Luo](#) , [Rick Anderson](#) , and [Steve Smith](#)

[View or download sample code](#) (how to download)

Response caching reduces the number of requests a client or proxy makes to a web server. Response caching also reduces the amount of work the web server performs to generate a response. Response caching is controlled by headers that specify how you want client, proxy, and middleware to cache responses.

The [ResponseCache attribute](#) participates in setting response caching headers. Clients and intermediate proxies should honor the headers for caching responses under the [HTTP 1.1 Caching specification](#) .

For server-side caching that follows the HTTP 1.1 Caching specification, use [Response Caching Middleware](#). The middleware can use the [ResponseCacheAttribute](#) properties to influence server-side caching behavior.

## HTTP-based response caching

The [HTTP 1.1 Caching specification](#) describes how Internet caches should behave. The primary HTTP header used for caching is [Cache-Control](#) , which is used to specify cache *directives*. The directives control caching behavior as requests make their way from clients to servers and as responses make their way from servers back to clients. Requests and responses move through proxy servers, and proxy servers must also conform to the HTTP 1.1 Caching specification.

Common Cache-Control directives are shown in the following table.

Directive	Action
<a href="#">public</a>	A cache may store the response.
<a href="#">private</a>	The response must not be stored by a shared cache. A private cache may store and reuse the response.
<a href="#">max-age</a>	The client doesn't accept a response whose age is greater than the specified number of seconds. Examples: <code>max-age=60</code> (60 seconds), <code>max-age=2592000</code> (1 month)
<a href="#">no-cache</a>	<p><b>On requests:</b> A cache must not use a stored response to satisfy the request. The origin server regenerates the response for the client, and the middleware updates the stored response in its cache.</p> <p><b>On responses:</b> The response must not be used for a subsequent request without validation on the origin server.</p>
<a href="#">no-store</a>	<p><b>On requests:</b> A cache must not store the request.</p> <p><b>On responses:</b> A cache must not store any part of the response.</p>

Other cache headers that play a role in caching are shown in the following table.

Header	Function
<a href="#">Age</a>	An estimate of the amount of time in seconds since the response was generated or successfully validated at the origin server.
<a href="#">Expires</a>	The time after which the response is considered stale.
<a href="#">Pragma</a>	Exists for backwards compatibility with HTTP/1.0 caches for setting <code>no-cache</code> behavior. If the <code>Cache-Control</code> header is present, the <code>Pragma</code> header is ignored.
<a href="#">Vary</a>	Specifies that a cached response must not be sent unless all of the <code>Vary</code> header fields match in both the cached response's original request and the new request.

## HTTP-based caching respects request Cache-Control directives

The [HTTP 1.1 Caching specification for the Cache-Control header](#) requires a cache to honor a valid `Cache-Control` header sent by the client. A client can make requests with a

no-cache header value and force the server to generate a new response for every request.

Always honoring client Cache-Control request headers makes sense if you consider the goal of HTTP caching. Under the official specification, caching is meant to reduce the latency and network overhead of satisfying requests across a network of clients, proxies, and servers. It isn't necessarily a way to control the load on an origin server.

There's no developer control over this caching behavior when using the [Response Caching Middleware](#) because the middleware adheres to the official caching specification. [Planned enhancements to the middleware](#) are an opportunity to configure the middleware to ignore a request's Cache-Control header when deciding to serve a cached response. Planned enhancements provide an opportunity to better control server load.

## Other caching technology in ASP.NET Core

### In-memory caching

In-memory caching uses server memory to store cached data. This type of caching is suitable for a single server or multiple servers using *sticky sessions*. Sticky sessions means that the requests from a client are always routed to the same server for processing.

For more information, see [Cache in-memory in ASP.NET Core](#).

### Distributed Cache

Use a distributed cache to store data in memory when the app is hosted in a cloud or server farm. The cache is shared across the servers that process requests. A client can submit a request that's handled by any server in the group if cached data for the client is available. ASP.NET Core works with SQL Server, [Redis](#), and [NCache](#) distributed caches.

For more information, see [Distributed caching in ASP.NET Core](#).

### Cache Tag Helper

Cache the content from an MVC view or Razor Page with the Cache Tag Helper. The Cache Tag Helper uses in-memory caching to store data.

For more information, see [Cache Tag Helper in ASP.NET Core MVC](#).

# Distributed Cache Tag Helper

Cache the content from an MVC view or Razor Page in distributed cloud or web farm scenarios with the Distributed Cache Tag Helper. The Distributed Cache Tag Helper uses SQL Server, [Redis](#) , or [NCache](#) to store data.

For more information, see [Distributed Cache Tag Helper in ASP.NET Core](#).

## ResponseCache attribute

The [ResponseCacheAttribute](#) specifies the parameters necessary for setting appropriate headers in response caching.

### Warning

Disable caching for content that contains information for authenticated clients. Caching should only be enabled for content that doesn't change based on a user's identity or whether a user is signed in.

[VaryByQueryKeys](#) varies the stored response by the values of the given list of query keys. When a single value of \* is provided, the middleware varies responses by all request query string parameters.

[Response Caching Middleware](#) must be enabled to set the [VaryByQueryKeys](#) property. Otherwise, a runtime exception is thrown. There isn't a corresponding HTTP header for the [VaryByQueryKeys](#) property. The property is an HTTP feature handled by Response Caching Middleware. For the middleware to serve a cached response, the query string and query string value must match a previous request. For example, consider the sequence of requests and results shown in the following table.

Request	Result
http://example.com?key1=value1	Returned from the server.
http://example.com?key1=value1	Returned from middleware.
http://example.com?key1=value2	Returned from the server.

The first request is returned by the server and cached in middleware. The second request is returned by middleware because the query string matches the previous request. The third


request isn't in the middleware cache because the query string value doesn't match a previous request.

The [ResponseCacheAttribute](#) is used to configure and create (via [IFilterFactory](#)) a `Microsoft.AspNetCore.Mvc.Internal.ResponseCacheFilter`. The `ResponseCacheFilter` performs the work of updating the appropriate HTTP headers and features of the response. The filter:


- Removes any existing headers for `Vary`, `Cache-Control`, and `Pragma`.
- Writes out the appropriate headers based on the properties set in the [ResponseCacheAttribute](#).
- Updates the response caching HTTP feature if [VaryByQueryKeys](#) is set.

## Vary

This header is only written when the [VaryByHeader](#) property is set. The property set to the `Vary` property's value. The following sample uses the [VaryByHeader](#) property:

C#	 Copy
<pre>[ResponseCache(VaryByHeader = "User-Agent", Duration = 30)] public class Cache1Model : PageModel {</pre>	

Using the sample app, view the response headers with the browser's network tools. The following response headers are sent with the `Cache1` page response:

	 Copy
<pre>Cache-Control: public,max-age=30 Vary: User-Agent</pre>	


## NoStore and Location.None

[NoStore](#) overrides most of the other properties. When this property is set to `true`, the `Cache-Control` header is set to `no-store`. If [Location](#) is set to `None`:


- `Cache-Control` is set to `no-store,no-cache`.
- `Pragma` is set to `no-cache`.

If `NoStore` is `false` and `Location` is `None`, `Cache-Control`, and `Pragma` are set to `no-cache`.

`NoStore` is typically set to `true` for error pages. The `Cache2` page in the sample app produces response headers that instruct the client not to store the response.

C#	 Copy
<pre>[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)] public class Cache2Model : PageModel {</pre>	

The sample app returns the `Cache2` page with the following headers:

	 Copy
<pre>Cache-Control: no-store,no-cache Pragma: no-cache</pre>	

## Location and Duration

To enable caching, `Duration` must be set to a positive value and `Location` must be either `Any` (the default) or `Client`. The framework sets the `Cache-Control` header to the location value followed by the `max-age` of the response.


`Location`'s options of `Any` and `Client` translate into `Cache-Control` header values of `public` and `private`, respectively. As noted in the `NoStore and Location.None` section, setting `Location` to `None` sets both `Cache-Control` and `Pragma` headers to `no-cache`.

`Location.Any` (`Cache-Control` set to `public`) indicates that the *client or any intermediate proxy* may cache the value, including [Response Caching Middleware](#).


`Location.Client` (`Cache-Control` set to `private`) indicates that *only the client* may cache the value. No intermediate cache should cache the value, including [Response Caching Middleware](#).

Cache control headers merely provide guidance to clients and intermediary proxies when and how to cache responses. There's no guarantee that clients and proxies will honor the [HTTP 1.1 Caching specification](#). [Response Caching Middleware](#) always follows the caching rules laid out by the specification.

The following example shows the Cache3 page model from the sample app and the headers produced by setting [Duration](#) and leaving the default [Location](#) value:

C#	 Copy
<pre>[ResponseCache(Duration = 10, Location = ResponseCacheLocation.Any, NoStore = false)] public class Cache3Model : PageModel {</pre>	

The sample app returns the Cache3 page with the following header:

	 Copy
Cache-Control: public,max-age=10	

## Cache profiles

Instead of duplicating response cache settings on many controller action attributes, cache profiles can be configured as options when setting up MVC/Razor Pages in `Startup.ConfigureServices`. Values found in a referenced cache profile are used as the defaults by the [ResponseCacheAttribute](#) and are overridden by any properties specified on the attribute.

Set up a cache profile. The following example shows a 30 second cache profile in the sample app's `Startup.ConfigureServices`:

C#	 Copy
<pre>public void ConfigureServices(IServiceCollection services) {     services.AddMvc(options =&gt;     {         options.CacheProfiles.Add("Default30",             new CacheProfile()             {                 Duration = 30             });     }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2); }</pre>	

The sample app's Cache4 page model references the `Default30` cache profile:

C#

 Copy

```
[ResponseCache(CacheProfileName = "Default30")]  
public class Cache4Model : PageModel  
{
```

The [ResponseCacheAttribute](#) can be applied to:

- Razor Pages: Attributes can't be applied to handler methods.
- MVC controllers.
- MVC action methods: Method-level attributes override the settings specified in class-level attributes.

The resulting header applied to the Cache4 page response by the Default30 cache profile:

 Copy

```
Cache-Control: public,max-age=30
```

## Additional resources

- [Storing Responses in Caches](#)
- [Cache-Control](#)
- [Cache in-memory in ASP.NET Core](#)
- [Distributed caching in ASP.NET Core](#)
- [Detect changes with change tokens in ASP.NET Core](#)
- [Response Caching Middleware in ASP.NET Core](#)
- [Cache Tag Helper in ASP.NET Core MVC](#)
- [Distributed Cache Tag Helper in ASP.NET Core](#)

## Is this page helpful?

 Yes  No