ASP.NET Core 2.2

Version

3.0 Preview 2

2.2

2.1

2.0

1.1

1.0

Response caching in ASP.NET Core

01/06/2018 8 minutes to read Contributors all

In this article

By John Luo, Rick Anderson, Steve Smith, and Luke Latham

> **Note**
>
> Response caching in Razor Pages is available in ASP.NET Core 2.1 or later.

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, which clients may honor when caching responses. Response Caching Middleware can be used to cache responses on the server. The middleware can use `ResponseCache` attribute 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 reponses 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
public	A cache may store the response.
private	The response must not be stored by a shared cache. A private cache may store and reuse the response.
max-age	The client won't accept a response whose age is greater than the specified number of seconds. Examples: `max-age=60` (60 seconds), `max-age=2592000` (1 month)
no-cache	**On requests**: A cache must not use a stored response to satisfy the request. Note: The origin server re-

	generates the response for the client, and the middleware updates the stored response in its cache. **On responses**: The response must not be used for a subsequent request without validation on the origin server.
no-store	**On requests**: A cache must not store the request. **On responses**: A cache must not store any part of the response.

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

Header	Function
Age	An estimate of the amount of time in seconds since the response was generated or successfully validated at the origin server.
Expires	The date/time after which the response is considered stale.
Pragma	Exists for backwards compatibility with HTTP/1.0 caches for setting `no-cache` behavior. If the `Cache-Control` header is present, the `Pragma` header is ignored.
Vary	Specifies that a cached response must not be sent unless all of the `Vary` 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.