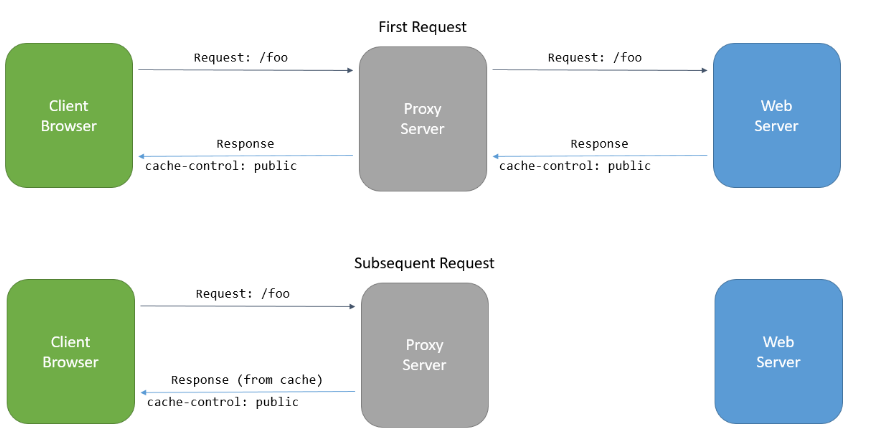
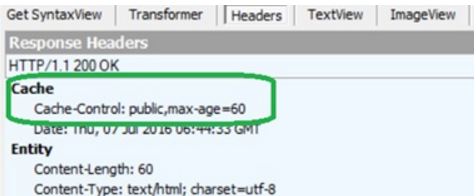
Caching

* Improves the Performance and Scalability of an app by reducing the work required to generate the

Content.

* Best to use when data changes infrequently
* It makes copy of data that can be returned much faster than the original source.
* It just adds the Cache-Control header in response now it’s the client choice to cache the response in the browser or some intermediate server. Which helps us to reduce the number of server roundtrip.





Type of Caches it Supports

1. IMemoryCache

* Cache Stored in the memory of web server.
* It can evict cache entries under memory pressure unless CachePriority is set to .NeverRemove.

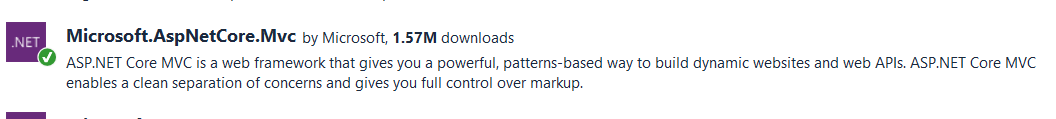
1. Distributed Cached

* It can support higher scale out than an in memory Cache
* Using Distributed cache offloads the cache memory to an external process.

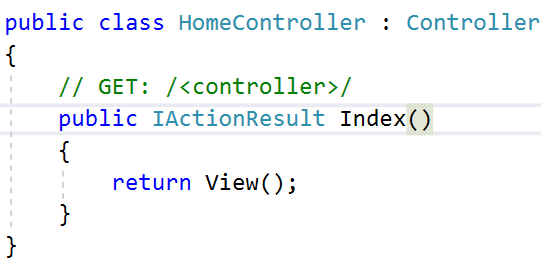
Demo 1 : Working with IMemoryCache

In memory Caching is a Service that is referenced from your app using DI we have to call AddMemoryCache in ConfigureServices.

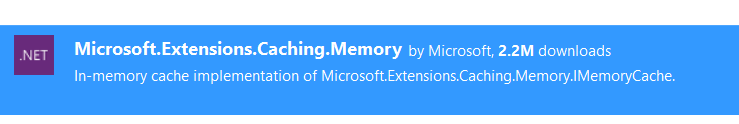
1. Create new empty CachingDemoApplication
2. Add Support for MVC thru nuggets



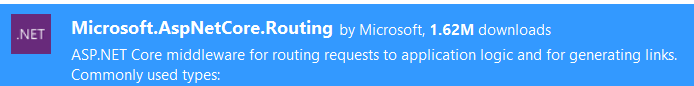
1. Add HomeController



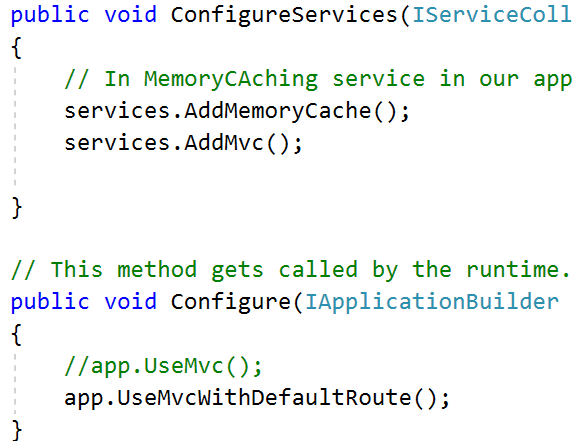
1. Add Support for Caching



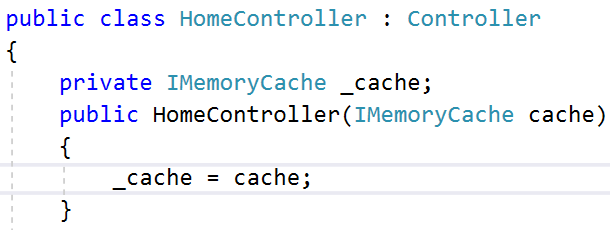
1. Add Support for Routing



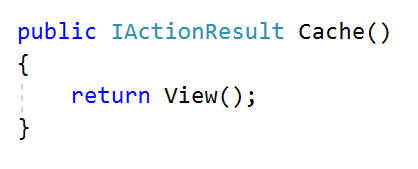
1. Add Support for Caching in ConfigureServices



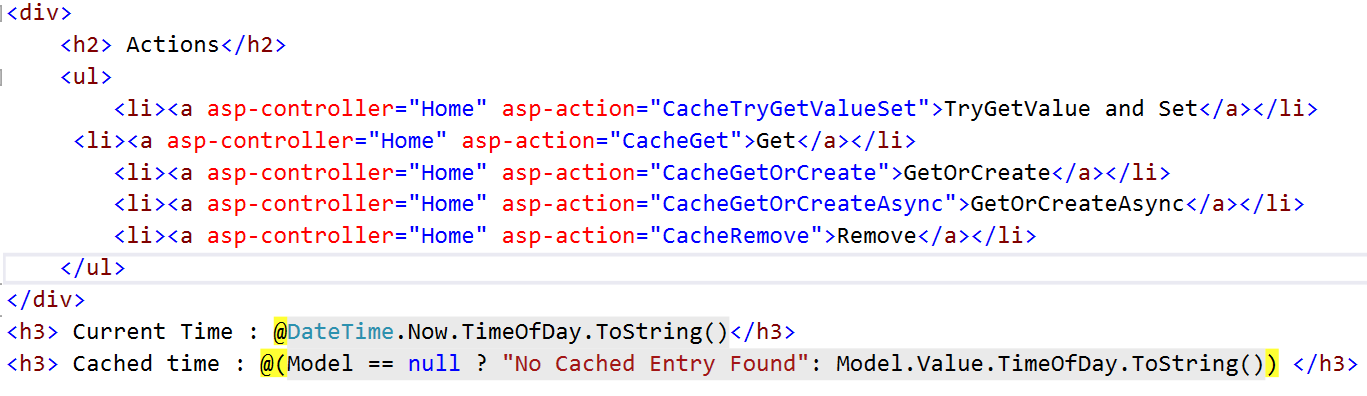
1. Request the IMemoryCache instance in HomeController



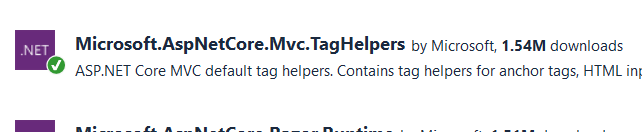
1. Add new CacheMethod



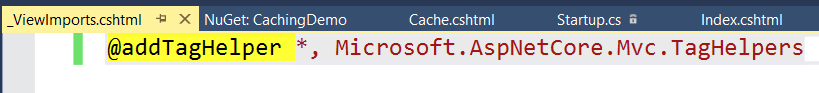
1. Add View for the same



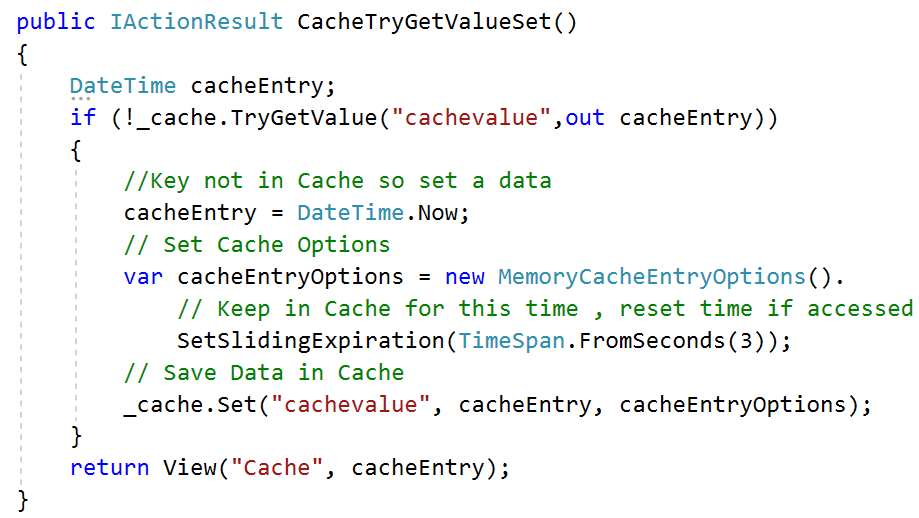
1. Add TagHelper Support



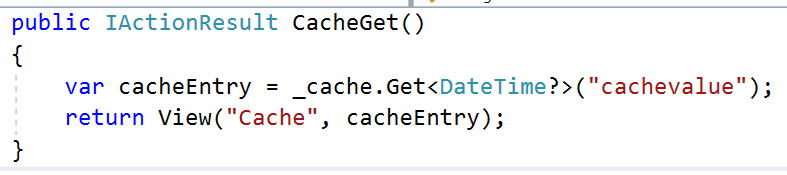
1. Inject this TagHelper in application for that add \_ViewImports.cshtml and add TagHelper



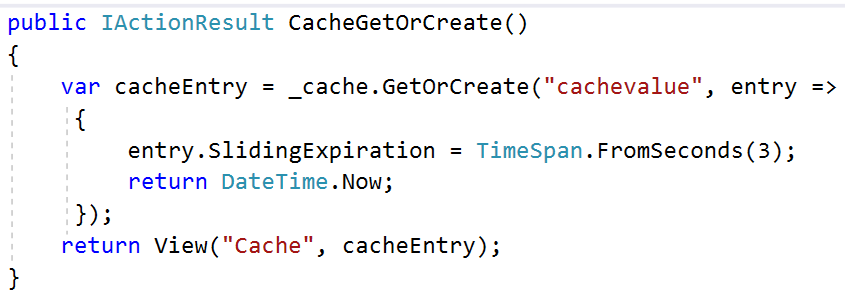
1. Try to Set CacheData



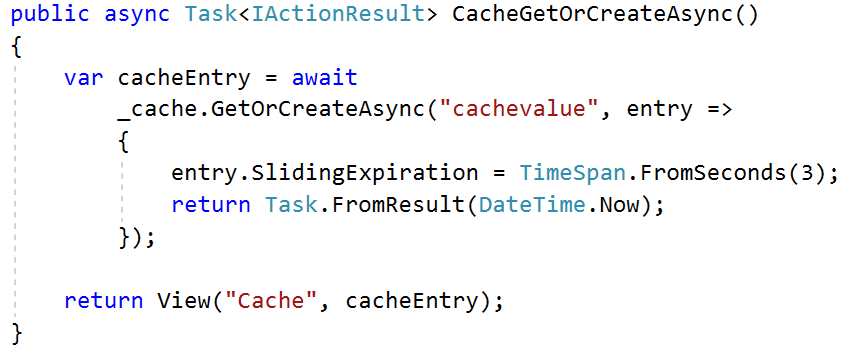
1. Add Get method and try to get the value from the Cache



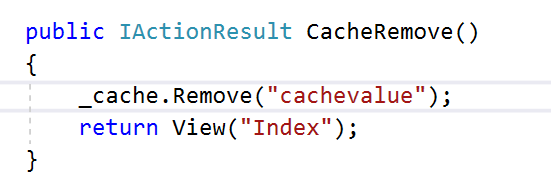
1. Add CacheGetOrCreate method



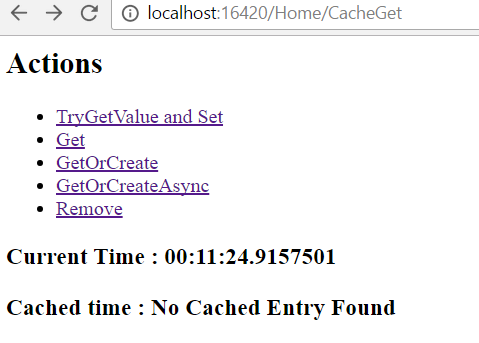
1. Add Async method for the same



1. Add Remove method to remove data from Cache



1. Check the Output



Response Caching

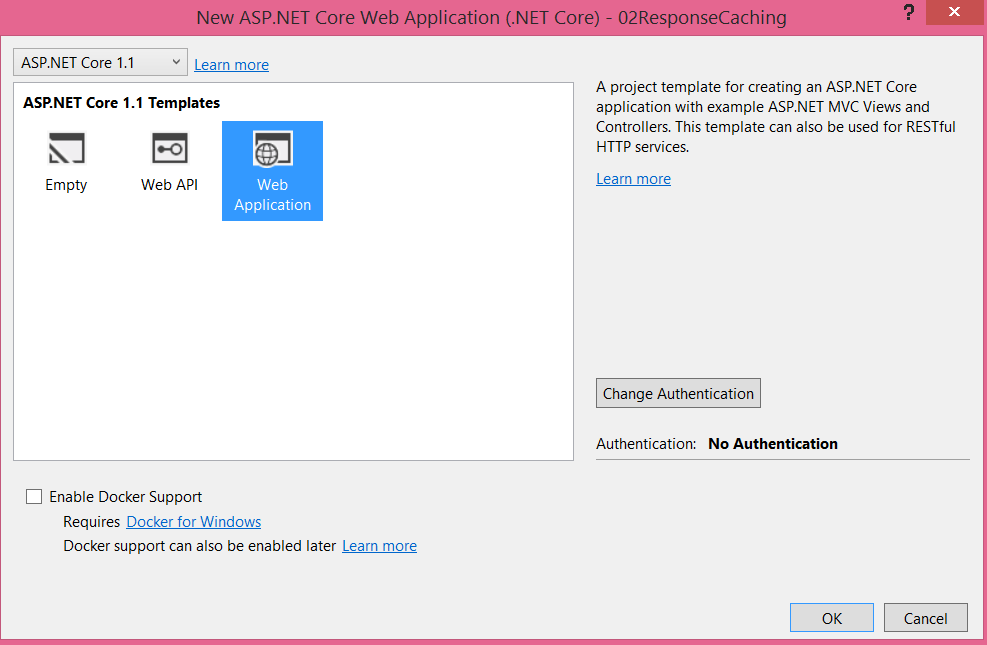
* It adds Cache Related headers to responses.
* It specifies how you want client, proxy and middleware to cache responses.
* Reduces the number of requests a client or proxy makes to web server.
* Response Caching can also reduce the amount of work the web server performs to generate the response.

ResponseCacheAttributes

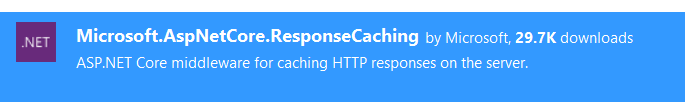
* It specifies the parameters necessary for setting appropriate headers in response Caching.

Demo : 1 How to do response Caching

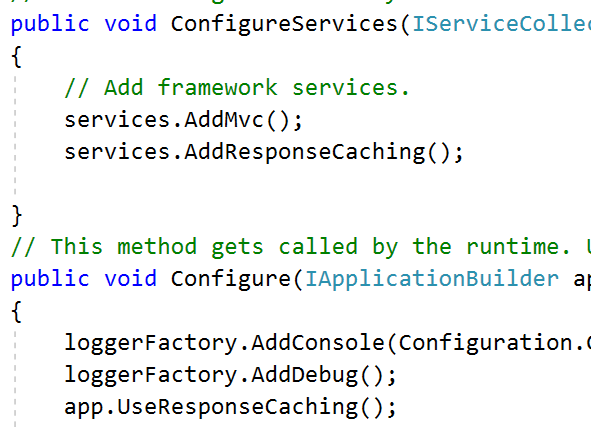
1. Create ASP.Net MVC Application Don’t Select empty now



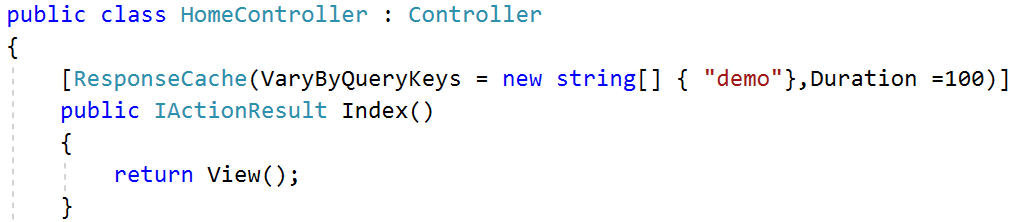
1. Add MiddleWare



1. Configure Caching Service in StartUp class



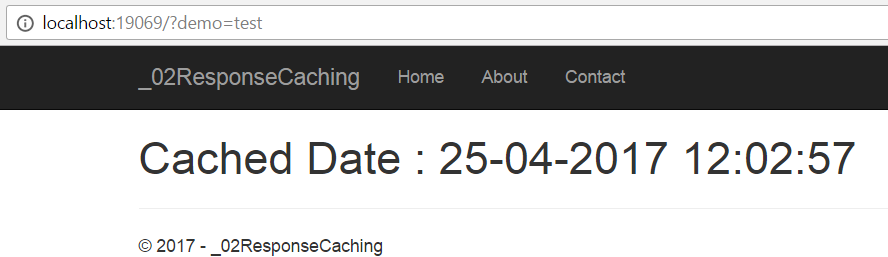
1. Go to Home controller and Add HomeController code

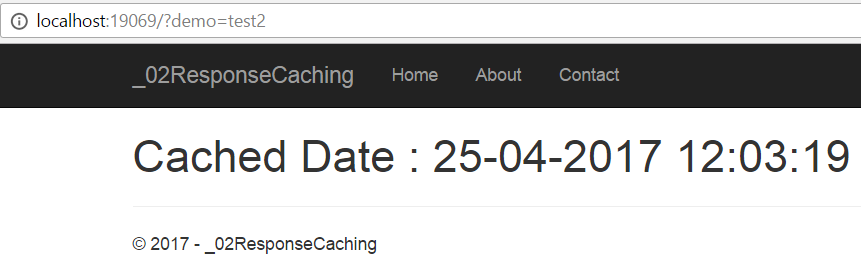


1. Change Index.cshtml to

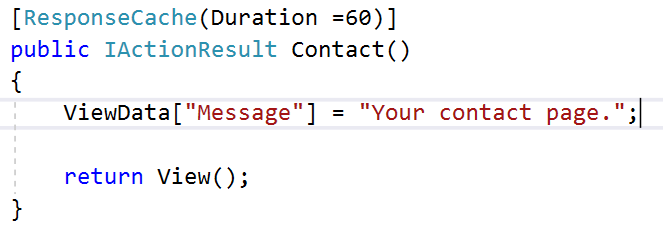


1. HomeController here cached by the parameter so if we pass different parameter for each parameter the response will Cached on Server





1. Add The Caching for a particular Action without having any Parameter sort of restriction. Change Contact method



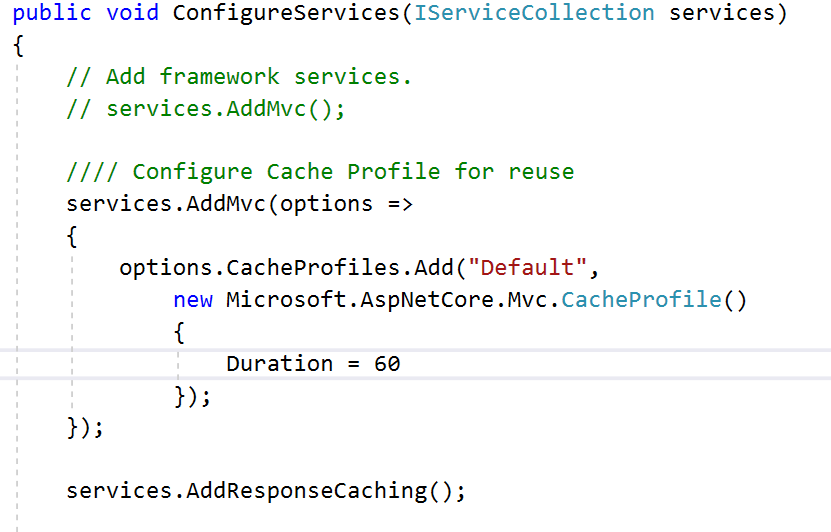
1. Change the contact.cshtml to same as Index.cshtml
2. Contact.cshtml will be cached for a specific amount of time.

Cache Profiles

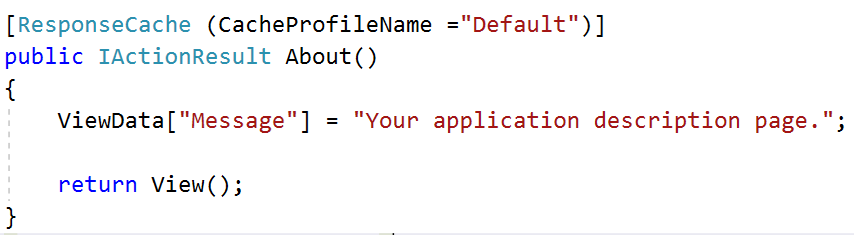
* Instead of duplicating ResponseCache settings on many controller action attributes cache profiles can be configured as options when setting up MVC in the configure services method in startup.
* Values in Cache profile will be used as the defaults by the response cache.

Demo : Continue

1. Change AddMvc Method from Startup class to define the Caching mechanism



1. Ref this in About action

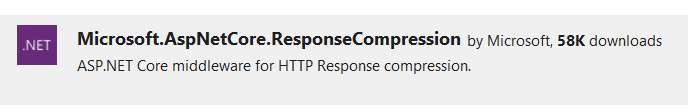


1. Check the Output of About method , its cached.

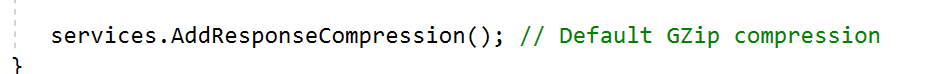
Response Compression Middleware

* Network bandwidth is a limited resource , if you can reduce response payload sizes and this ssend less data to clients it will increase the responsiveness of your application.
* One way to do this is to reduce payload sizes

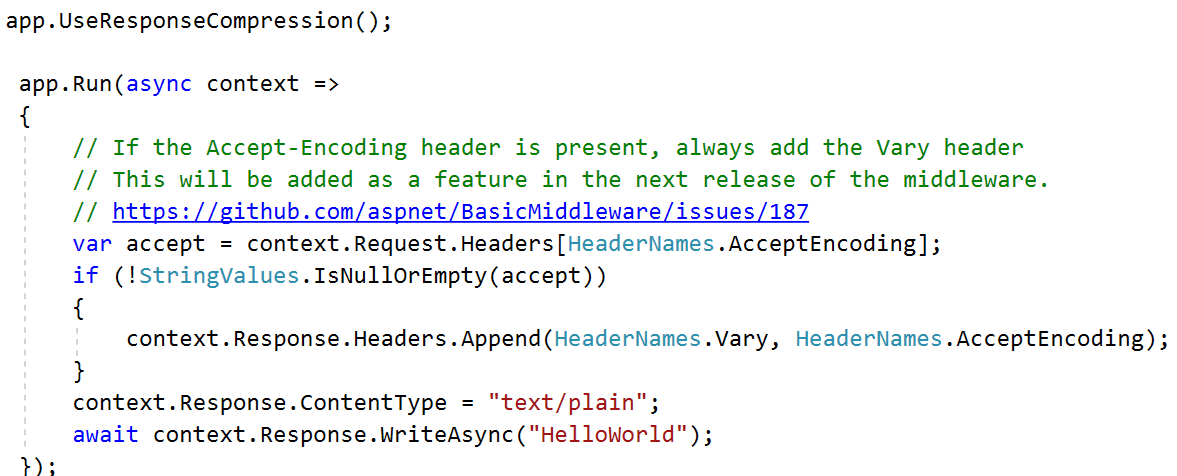
1. Add ResponseCompression MiddleWare



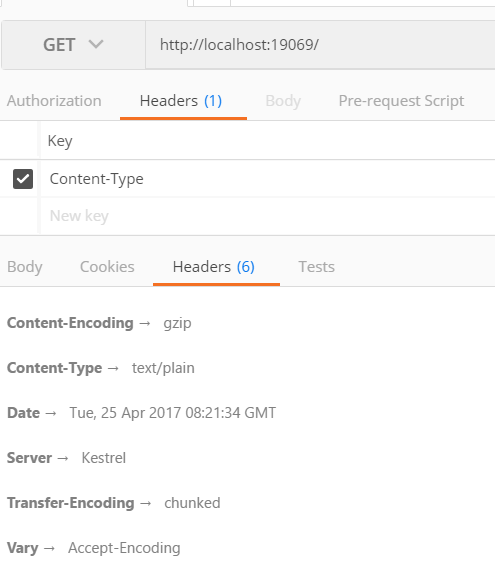
1. Add Service in the ConfigureService for Compression



1. Add Use Compression in Configure method



1. Check the Output using PostMan



Working with Distributed Cache

* Distributed caches can improve the performance and scalability of ASP.Net Core apps which is useful in a cloud or server farm environment.