**Enable Caching in .NET Core 2.0**

**Response Caching**

You can enable the Response Caching by adding the ResponseCache attribute. You can add this attribute on top of the GET action.

// GET api/catalog

[HttpGet]

[ResponseCache(VaryByHeader="User-Agent", Duration=60)]

public IEnumerable<Product> Get()

{

return products;

}

**In-memory Cache**

1. You can configure in memory cache in Configure Services method

public void ConfigureServices(IServiceCollection services)

{

services.AddMemoryCache();

services.AddMvc();

}

1. Request the IMemoryCache instance in the constructor of controller. Import the “**Microsoft.Extensions.Caching.Memory**” package .

private IMemoryCache \_cache;

public CatalogController(IMemoryCache memoryCache)

{

\_cache = memoryCache;

}

1. Add the following action method in controller to use the in-momory cache

//GET api/catalog/all

[HttpGet]

public IEnumerable<Product> GetAll()

{

IEnumerable<Product> items;

if(!\_cache.TryGetValue("products", out items))

{

items=products;

MemoryCacheEntryOptions options=new MemoryCacheEntryOptions()

.SetSlidingExpiration(TimeSpan.FromSeconds(10));

\_cache.Set("products", items, options);

}

return items;

}

**Distributed Cache using Redis Cache**

You can use Azure Redis cache service or local redis service in your application.

1. Configure the Distributed Redis Cahe in the Configure Services method

services.AddDistributedRedisCache(options =>

{

options.Configuration = "localhost";

options.InstanceName = "CatalogInstance";

});

1. Open the controller and add the following code to work with Distributed cache. Import the “Microsoft.Extensions.Caching.Distributed” package to the controller class.

private IDistributedCache \_dcache;

public CatalogController(IMemoryCache cache, IDistributedCache dcache)

{

this.\_cache=cache;

this.\_dcache=dcache;

}

//GET api/catalog/products

[HttpGet]

public async Task<IEnumerable<Product>> GetProducts()

{

IEnumerable<Product> items;

var data=await \_dcache.GetStringAsync("products");

if(data!=null){

items=JsonConvert.DeserializeObject<IEnumerable<Product>>(data);

}else{

var obj=JsonConvert.SerializeObject(products);

await \_dcache.SetStringAsync("products",obj);

items=products;

}

return items;

}