**安装包**

Microsoft.Extensions.Caching.Memory

**在setup中注册服务**

public void ConfigureServices (IServiceCollection services)

{

    services.AddMemoryCache ();

    ...

}

**使用**

public class ValuesController : ControllerBase {

    private IMemoryCache \_cache;

    public ValuesController (IMemoryCache memoryCache) {

        \_cache = memoryCache;

}

    // GET api/values

    [HttpGet]

    public ActionResult<string> Get () {

        // 从缓存中获取键为 key1 的项，如果没有或过期，则调用第二个参数Fun生成

        return \_cache.GetOrCreate<string> ("key1", cacheEntity => {

            // 设置过期时间，过期后会重新调用该函数生成值

            // 设置过期时间

            // 如果5秒内没有访问该值，则过期

            // 否则，过期时间刷新为访问的时间 +5 秒

            cacheEntity.SlidingExpiration = TimeSpan.FromSeconds (5);

            // 设置绝对过期时间

            // 过期时间为15秒后，无论这15秒内是否有访问都会过期

            cacheEntity.AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds (15);

            // 设置该项优先级

            // NeverRemove 为不会从缓存中移除，除非过期

            // 优先级越低，内存不足时，将会移除该项

            cacheEntity.SetPriority (CacheItemPriority.NeverRemove);

            // 注册缓存项从缓存移除时的回调函数

            cacheEntity.RegisterPostEvictionCallback (

                (object key, object value, EvictionReason reason, object state) => { });

            return DateTime.Now.ToString ();

        });

}

    [HttpGet]

    public ActionResult Remove () {

        // 从缓存中移除键为key1的缓存项

        \_cache.Remove ("key1");

        return Ok ();

    }

}

**内存缓存与EFCore**

EFCore也使用内存缓存，如果我们使用依赖注入注入IMemoryCache ，则注入的可能是EFCore使用的缓存，所以最好的方法是我们重新注入我们的缓存（单例）

    public class IEMemoryCache : MemoryCache, IIEMemoryCache

    {

        public IEMemoryCache() : base(new MemoryCacheOptions {

            SizeLimit = null

        })

        {

        }

    }