**IDistributedCache接口**

IDistributedCache表示分布式缓存接口，其接口如下：

public interface IDistributedCache

{

    // 获取键为key的缓存项

    byte[] Get(string key);

    Task<byte[]> GetAsync(string key, CancellationToken token = default);

    // 移除键为key的缓存项

    void Remove(string key);

Task RemoveAsync(string key, CancellationToken token = default);

    // 设置键为key的缓存项

    void Set(string key, byte[] value, DistributedCacheEntryOptions options);

    Task SetAsync(string key, byte[] value, DistributedCacheEntryOptions options, CancellationToken token = default);

}

**注册分布式缓存服务**

**内存分布式缓存**

services.AddDistributedMemoryCache();

注：内存分布式缓存使用的是本地内存，但其确实现的分布式缓存的接口

**Redis分布式缓存**

services.AddStackExchangeRedisCache (options => {

    // 配置 Redis 连接字符串

    // 127.0.0.1:6379 为 Redis 服务地址，123456 为 Redis 连接密码

    options.Configuration = "127.0.0.1:6379,password=123456";

    // 随便起的实例名，我们在 Redis 服务器保存的 key 均以该实例名开头

    options.InstanceName = "SampleInstance";

});

**示例**

public class ValuesController : ControllerBase {

    private readonly IDistributedCache \_cache;

    // 注入分布式缓存

    public ValuesController (IDistributedCache cache) {

        \_cache = cache;

}

    [HttpGet]

    public ActionResult<string> Get () {

        // 获取键为 mykey1 的缓存项

        string value = \_cache.GetString ("mykey1");

        if (string.IsNullOrEmpty (value)) {

            // 设置键为 mykey1 的缓存项

            \_cache.SetString ("mykey1", "abc");

            return "no value";

        }

        return value;

}

    [HttpGet]

    public ActionResult Remove () {

        // 从缓存中移除键为mykey1的缓存项

        \_cache.Remove ("mykey1");

        return Ok ();

    }

}