**ICacheManager**

缓存的主要接口是ICacheManager。我们可以注入它并用它获取一个缓存，如：

public class TestAppService : ApplicationService

{

private readonly ICacheManager \_cacheManager;

public TestAppService(ICacheManager cacheManager)

{

\_cacheManager = cacheManager;

}

public Item GetItem(int id)

{

return \_cacheManager

.GetCache("MyCache")

.Get(id.ToString(), () => GetFromDatabase(id)) as Item;

}

}

在此例里，我们注入了ICacheManager，并获得一个名为MyCache的缓存。

警告：GetCache方法

如果你的类不是单例，不要在你构造器里使用GetCache，否则可能会销毁你的缓存。

**ICache**

ICacheManager.GetCache方法返回一个ICache。在示例代码里，我们看到了ICache.get方法的简单使用。它有两个参数：

key：字符串，必需，一个缓存项的键。

factory：一个action（行为），在找不到指定key的缓存项时调用，Factory方法应该创建并返回切实的项，如果指定key的缓存已存在，就不调用。

**ITypedCache**

我们可用泛型的GetCache扩展方法，获取一个ITypedCache：

ITypedCache<int, Item> myCache = \_cacheManager.GetCache<int, Item>("MyCache");

**配置**

默认缓存超时是60分钟，它可以改。如果你超过60分钟没有使用缓存中的项，会从缓存中自动移除。你可以配置指定的缓存或是全部的缓存。

// 配置所有缓存

Configuration.Caching.ConfigureAll(cache =>

{

cache.DefaultSlidingExpireTime = TimeSpan.FromHours(2);

});

// 配置指定缓存

Configuration.Caching.Configure("MyCache", cache =>

{

cache.DefaultSlidingExpireTime = TimeSpan.FromHours(8);

});

这段代码应该写在你模块的PreInitialize方法里

**实体缓存**

虽然ABP缓存系统出于普通的目的，但有一个EntityCache基类，可帮你缓存实体。如果我们通过它们的Id获取的实体，我们可以用这个基类缓存它们，就不用再频繁地从数据库查询。

假设我们有如下所示的一个person实体：

public class Person : Entity

{

public string Name { get; set; }

public int Age { get; set; }

}

首先，我们需要创建一个类来缓存项：

**[AutoMapFrom(typeof(Person))]**

public class PersonCacheItem

{

public string Name { get; set; }

}

我们不应该直接在缓存里存储实体，由于缓存可能需要序列化缓存对象，而实体不一定能序列化（尤其有导航属性的实体）。

添加AutoMapFrom特性，它可以自动地把Person转换成PersonCacheItem对象。

最后，我们可以为实体创建缓存类：

public class PersonCache : **EntityCache<Person, PersonCacheItem>**, **ITransientDependency**

{

public PersonCache(ICacheManager cacheManager, IRepository<Person> repository) : base(cacheManager, repository)

{

}

}

现在，任何需要Person的Name时，我们可以通过person的Id从缓存获取，使用Person缓存的示例如下：

public class MyPersonService : ITransientDependency

{

private readonly IPersonCache \_personCache;

public MyPersonService(**IPersonCache personCache**)

{

\_personCache = personCache;

}

public string GetPersonNameById(int id)

{

**return \_personCache[id].Name;** //alternative: \_personCache.Get(id).Name;

}

}

我们简单地注入IPersonCache，获取缓存项和获取Name属性。

**Redis 缓存集成**

默认缓存管理使用的是内存缓存。所以，如果你有多个并发的Web服务器使用同个应用，可能会成为一个问题，在这种情况下，你需要一个分布/集中缓存服务，你就可以简单的使用Redis做为你的缓存服务器。

首先，你要在你的应用里，安装Abp.RedisCache的Nuget包（例如，你可在你的Web项目里安装）。接着为AbpRedisCacheModule添加DependsOn特性，然后在你模块预初始化方法里调用useRedis扩展方法。如下所示：

//...other namespaces

**using Abp.Runtime.Caching.Redis;**

namespace MyProject.AbpZeroTemplate.Web

{

[DependsOn(**typeof(AbpRedisCacheModule))]**

public class MyProjectWebModule : AbpModule

{

public override void PreInitialize()

{

**Configuration.Caching.UseRedis();**

}

}

}

Abp.RedisCache包使用”localhost“作为默认连接字符串，你可以在配置文件里添加连接字符串重写它，例如：

<add name="**Abp.Redis.Cache**" connectionString="localhost"/>

同样，你可以向appSettings里添加Redis的数据库Id，例如：

<add key="**Abp.Redis.Cache.DatabaseId**" value="2"/>

不同的数据库Id，在同一服务器上，帮助创建不同的键空间（独立缓存）。