# UnitOfWork和Repository

1 使用环境：

因为.net core 自带依赖注入，所以容器与控制器之间会产生依赖。

2 解决的问题：

当在依赖注入中注入dbcontext的时候，dbcontext与控制器产生依赖，但一个网站可能会在某些情况下从一种数据库切换带另外一种数据库，为了减少切换数据库时的工作量，所以使用UnitOfWork & Repository

提高代码重用

3 原理：

1在容器中注入<接口，实现类>服务之后，控制器接收注入进来的实现类：

private readonly IPostRepository repository;

private readonly IUnitOfWork unitOfWork;

public PostsController(IPostRepository Repository,IUnitOfWork unitOfWork)

{

repository = Repository;

this.unitOfWork = unitOfWork;

}

这样控制器与容器之间进行了解耦

2 接口类中写一些方法

public interface IPostRepository

{

Task<IEnumerable<Post>> GetPostsAllAsync();

void AddPost(Post post);

}

3 实现类中写实现（一般写一些层删改查的方法）

public class PostRepository : IPostRepository

{

private readonly MyContext myContext;

public PostRepository(MyContext myContext)

{

this.myContext = myContext;

}

public void AddPost(Post post)

{

myContext.Posts.Add(post);

}

public async Task<IEnumerable<Post>> GetPostsAllAsync()

{

return await myContext.Posts.ToListAsync();

}

}

4 至此实现下图的结构

DI容器

Repository接口

UnitOfWork接口

控制器

Repository实现类

UnitOfWor实现类

5 当要切换不同的数据库时：

修改dbcontext

修改注册服务

修改实现类的具体实现

完成

主要受益：代码复用，其次切换数据库时候修改量减少

注意：

一般来说Repository负责具体增删改查，UnitOfWork负责保存

因为一个http请求只有有一个context，可能同一时刻很多方法在调用，如果保存放在Repository中可能在异步情况下出现a完成b没完成，a执行保存导致b报错的情况，所以保存在一个方法的最后单独用UnitOfWork调用