# 依赖注入

这里具体学习的是core框架内置的依赖注入的实现。

依赖注入是实现对象跟依赖项解耦的技术，将实现以某种方式传递，而不是每次使用都初始化，或者静态调用。

Core已经内置了一个ioc容器,但不是要取代其他容器.下面是内置的依赖注入的使用方法.

例1,构造函数注入

新建一个接口

public interface IDateTime

{

DateTime Now { get; }

}

新建一个类并继承了上面的接口

public class SystemDateTime : IDateTime

{

public DateTime Now {

get { return DateTime.Now; }

}

}

在控制器添加下面代码

private readonly IDateTime \_datetime;

public HomeController(IDateTime datetime)

{

\_datetime = datetime;

}

public IActionResult Index()

{

ViewData["time"] = \_datetime.Now.ToString();

return View();

}

这时候如果运行会报错,因为我们还没在容器配置这个关系.

最后在Startup 类的ConfigureServices函数添加:

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc();//添加MVC服务

services.AddTransient<IDateTime, SystemDateTime>();//配置关系

}

例2,[FromServices]注入

public IActionResult Error([FromServices] IDateTime datetime)

{

ViewData["time"] = $"现在时间为:{datetime.Now}";

return View();

}

**---生命周期**

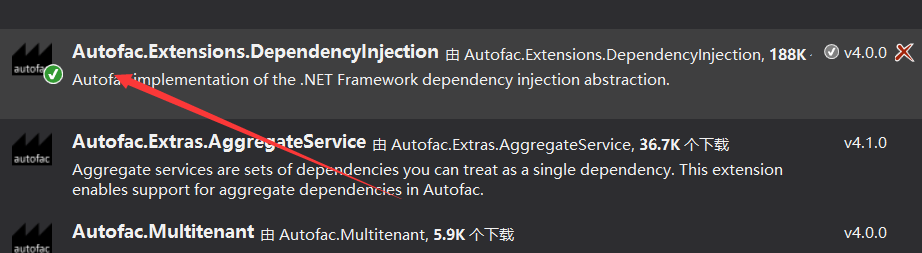
**Transient**:生命周期服务在它们每次请求时被创建。这一生命周期适合轻量级的，无状态的服务。

**Scoped**:生命周期服务在每次请求被创建一次。

**Singleton**:生命周期服务在它们第一次被请求时创建（或者如果你在 ConfigureServices  
运行时指定一个实例）并且每个后续请求将使用相同的实例。如果你的应用程序需要单例行为，建议让服务容器管理服务的生命周期而不是在自己的类中实现单例模式和管理对象的生命周期。

**---使用autofac**

1.添加打开NuGet包管理添加下面的引用.



2,添加一个接口

public interface IMsgService

{

string Html();

}

3,添加一个类,并且实现了上面的接口

public class MsgService : IMsgService

{

public string Html()

{

return $"<label>tag</label>";

}

}

1. 添加一个类BaseModule们用来配置关系

public class BaseModule:Module

{

protected override void Load(ContainerBuilder builder)

{

builder.RegisterType<MsgService>().As<IMsgService>();

}

}

5.把Startup类修改如下

//使用自定义IOC容器 要把返回类型void改为IServiceProvider

public IServiceProvider ConfigureServices(IServiceCollection services)

{

// Add framework services.

services.AddMvc();

// Add Autofac

var containerBuilder = new ContainerBuilder();

containerBuilder.RegisterModule<BaseModule>();

containerBuilder.Populate(services);

var container = containerBuilder.Build();

return container.Resolve<IServiceProvider>();

}

6.最后在控制器构造函数注入

private readonly IMsgService \_msgservice;

public HomeController(IMsgService msgservice)

{

\_msgservice = msgservice;

}

public IActionResult Index()

{

ViewData["msg"] = \_msgservice.Html();

return View();

}

视图页面调用

