1.webapi的控制反转

用Web API的Decpendency Resolver来实例化容器，之后再用对应的容器来实现控制反转

2.webapi的后端调用

1).httpclient

httpclient的连接应用是基于tcp连接的，就算用using完也不能立即释放，而其根本也是类似一个连接池，当并发数多了连接池不够的话，那么就会提示资源不足。所以应该用单例的模式，共同的用那一个单例的连接即可

2).httpwebrequest

3.webapi的安全校验

1).直接传递账号密码，数据库后台校验

2).约定的密码进行校验

3).最常用的Basic Auth基本授权验证，类似aop的形式来经询批量的对类中的方法经询校验。

在HttpBasicAuthAttribute中自定义对应的校验。如何用特性的方法直接在方法中加上对应的特性。

也可以用config.filter.add("HttpBasicAuthAttribute"),来实现全局生效

4.webapi的filter的使用

> ## 5类Filter
>
> WebAPI为我们定义了5类Filter。分别如下：
>
> AuthenticationFilter：用于请求认证（IAuthenticationFilter）。
>
> AuthorizationFilter：用于请求授权（IAuthorizationFilter）。
>
> ActionFilter:ActionFilter：注册的操作会在Action执行的前后被调用（IActionFilter）。
>
> ExceptionFilter：当目标Action抛出异常是调用（IExceptionFilter）。
>
> OverrideFilter：用于屏蔽当前域之前的Filter（IOverrideFilter）。

5.ActionFiter的使用

在方法上面标记对应的[XXActionFilterAtttribute]

OnActionExecuting:在方法执行之前会调用

OnActionExecuted:在方法执行之后调用

6.WebApi的跨域

跨域的定于：除了协议，域名，端口相同的调用其他的都算跨域

浏览器本身不允许这样的操作，除非在服务器的响应头里面指定：Access-Control-Allow-Origin

1）jsonp的方式跨域,其实就是利用了浏览器的同源策略，用对应的src去加载连接。

2）CORS的方式，利用服务器的响应头里面指定：Access-Control-Allow-Origin来实现跨域

config.EnableCors(new EnableCorsAttribute("*","*",8""))