基础易错笔记

**1. Cast<>()函数与OfType<TResult>()函数，将 IEnumerable 的元素强制转换为指定的类型。**

功能类似，但是也有不同之处。使用Cast如果无法转换为指定类型则会报错。OfType不会报错只是不会把这个值加到新的返回值里。

**2.MVC控制器向视图传递数据的方式**

A．ViewData["key"] = value， value可以为任意类型的数据

B. 具有动态类型的对象：ViewBag.key = value，无须进行类型转换

C. TempData["key"] = value，它可以在Action之间共享数据

**3.常见的Action**

A. ViewResult 转向视图

B. RedirectResult：Action之间的跳转

C. JsonResult: 传递Json数据

D. RedirectToRoute: 传递路由数据（匿名对象 new{}）

E. FilePathResult

F. ContentResult：传递纯字符串

Action之间传递的数据只能是简单类型（int、string、double....）

无法传递复杂的类型（对象等）

**4. Session.Abandon 与Session.Clear**

Abandon 方法删除所有存储在 Session 对象中的对象并释放这些对象的源。如果您未明确地调用 Abandon 方法，一旦会话超时，服务器将删除这些对象 。

Session.Clear()就是把Session对象中的所有项目都删除了，Session对象里面啥都没有。但是Session对象还保留 。Session.Abandon()就是把当前Session对象删除了，下一次就是新的Session了。

主要的不同之处在于:  
    当使用Session.Abandon时，会调用Session\_End方法（InProc模式下）。当下一个请求到来时将激发Session\_Start方法;而Session.Clear只是清除Session 中的所有数据并不会中止该Session，因此也不会调用那些方法。

5. ASP.NET Core体系中重要的几个对象和概念

HttpContext：请求上下文，对应Http请求的整个生命周期

IHostingEnvironment：应用程序的运行的宿主环境配置信息

ILogger：日志接口

IConfiguration： 用于应用程序的配置

IServiceCollection：Aspnet Core原生的依赖注入容器，负责服务注册

IServiceProvider：负责解析注册的实例

IApplicationBuilder：用于构建请求管道

Middleware：中间件，管道的组成者

WebHost：应用宿主

6. UseAuthentication和UseAuthorization的区别

UseAuthentication(认证)：常用的认证方式有用户密码认证

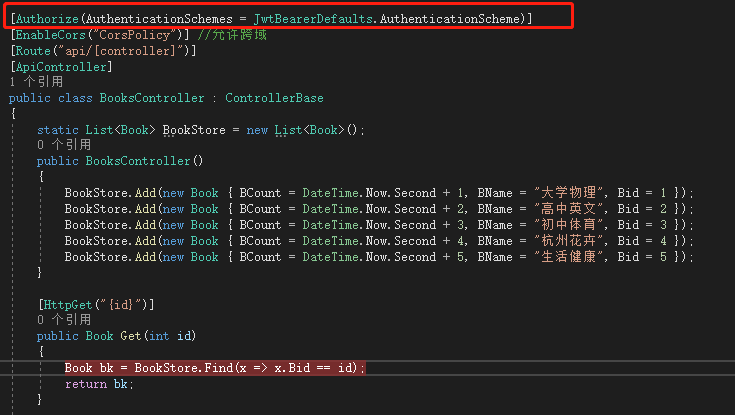
UseAuthorization(授权)：明确你是否有某个权限。当用户使用某个功能的时候，系统需要校验用户是否需要这个功能的权限。

所以这两个单词是不同的概念，不同层次的东西。UseAuthorization在asp.net core 2.0中是没有的。在3.0之后微软明确的把授权功能提取到了Authorization中间件里，所以我们需要在UseAuthentication之后再次UseAuthorization。否则，当你使用授权功能比如使用[Authorize]属性的时候系统就会报错。

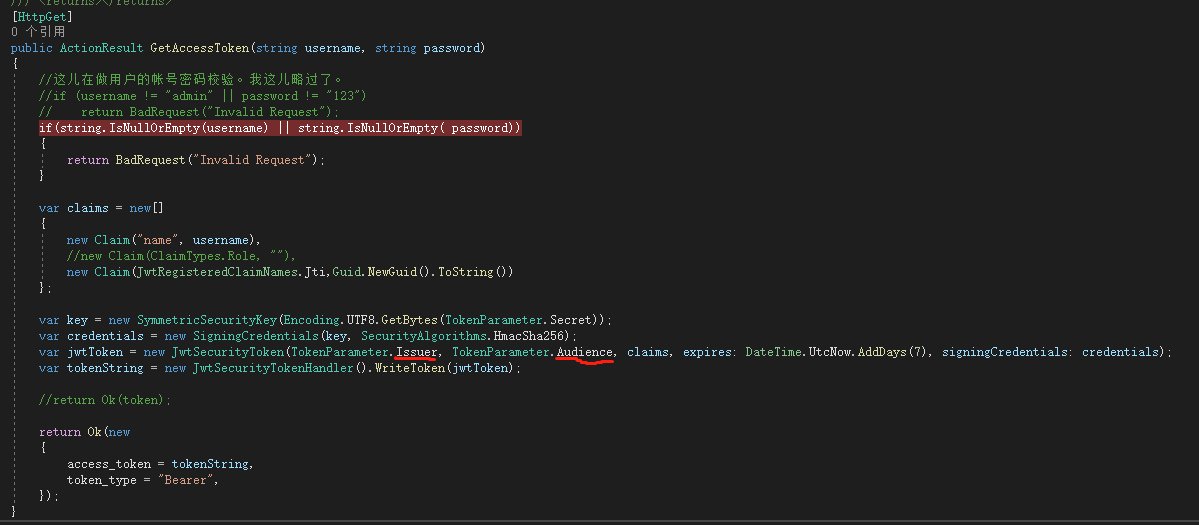
需要打开验证和鉴权时，则在Configure里面打开，如下图所示：



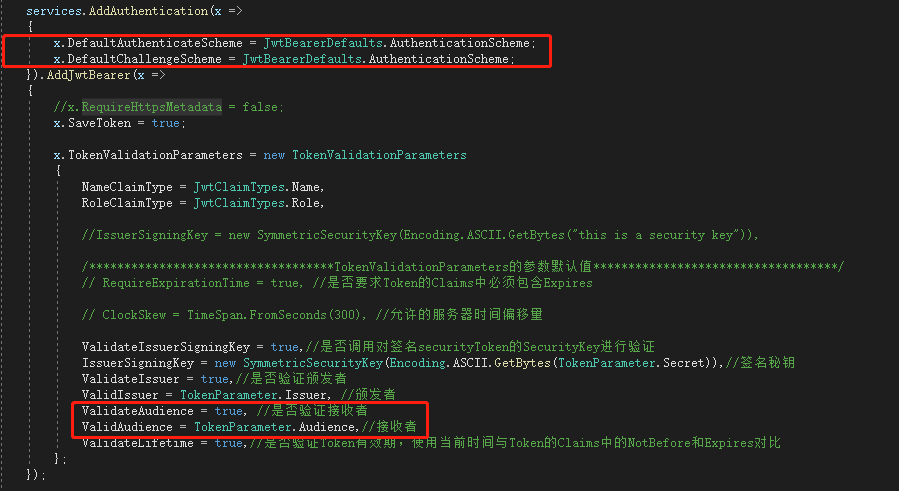
在使用Jwt进行Bearer验证时，则控制器头部或者方法属性上需要指定验证方式，不然是使用默认的认证验证方式，如图所示：

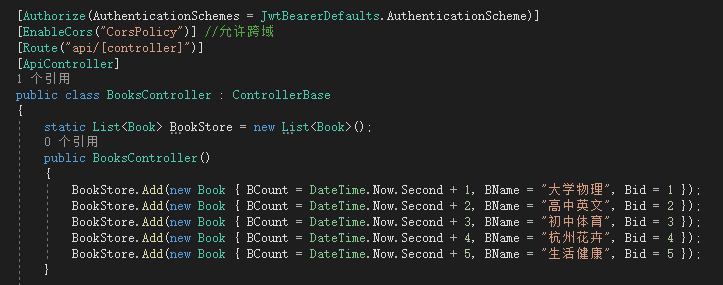


7.使用JWT验证时，如果设置了模板和开启了ValidIssuer，ValidateAudience这个验证，那加密的Token也是需要把这二项数据加密进去，不然验证出错，其它同理。



Jwt定义了默认认证方式，则在Controller也需要设置，以及在Startup上需要把生成的Token中的相关数据一致(如ValidIssuer和ValidAudience)。





8.跨域中间件设置需要在UseRouting和UseEndpoints 之间设置

app.UseRouting();

app.UseCors();//跨域 app.UseCors() must appear between app.UseRouting() and app.UseEndpoints

app.UseEndpoints

9. application builder的use和run方法有什么区别

这两个方法都在start up class的configure方法里面调用。都是用来向应用请求管道里面添加中间件的。Use方法可以调用下一个中间件的添加，而run不会。

10. 描述一下依赖注入后的服务生命周期

asp.net core主要提供了三种依赖注入的方式

其中AddTransient与AddSingleton比较好区别

AddTransient瞬时模式：每次都获取一个新的实例；

AddSingleton单例模式：每次都获取同一个实例；

而AddTransient与AddScoped的区别更不容易区别一点，但是我们可以继续分细一点，虽然不同的请求得到的结果不同，但是我们可以在同一次请求中去获取多次实例测试。

**小总结:**

AddTransient瞬时模式：每次请求，都获取一个新的实例。即使同一个请求获取多次也会是不同的实例；

AddScoped：每次请求，都获取一个新的实例。同一个请求获取多次会得到相同的实例；

AddSingleton单例模式：每次都获取同一个实例；

11. 值类型和引用类型的区别

所有继承自System.ValueType 的类型是值类型，而其他类型都是引用类型。值类型的赋值会产生一个新的数据副本，所以每个值类型都拥有一个数据副本，而引用类型的赋值则是赋值引用。值类型的对象分配在堆栈上，而引用类型的对象分配在堆上。当比较两个值类型时，进行的是内容比较，而比较两个引用类型时，进行的是引用比较。

12. 请解释委托的基本原理

委托是一类继承自System.Delegate 的类型，每个委托对象至少包含了一个指向某个方法的指针，该方法可以是实例方法，也可以是静态方法。委托实现了回调方法的机制，能够帮助程序员设计更加简洁优美的面向对象程序。

13. 什么是链式委托

链式委托是指一个由委托串成的链表，当链表上的一个委托被回调时，所有链表上该委托的后续委托将会被顺序执行。

14. 什么是内存映射文件

内存映射文件用于将文件内容映射到应用程序的逻辑地址。它使你能够在同一台计算机上运行多个进程以彼此共享数据。要获得一个内存映射文件对象, 可以使用MemoryMappedFile.CreateFromFiles()方法。它表示磁盘上文件中的持久性内存映射文件。

15. Mutex和lock有什么不同？⼀般⽤哪⼀种⽐较好

Mutex是⼀个基于内核模式的互斥锁，支持锁的递归调用，而Lock是⼀个混合锁，⼀般建议使用Lock更好，因为lock的性能更好。

16. MVC中的TempData\ViewBag\ViewData区别

TempData 保存在Session中，Controller每次执⾏请求的时候，会从Session中先获取 TempData，而后清除Session，获取完TempData数据，虽然保存在内部字典对象中，但是其集合中的每个条⽬访问⼀次后就从字典表中删 除，ViewData存的是Key/Value字典，使⽤时需要类型转换。

ViewBag和ViewData只在当前Action中有效，等同于View，ViewBag⽐ViewData慢，ViewBag存dynamic类型数据，使⽤时不需要类型转换

ViewData和ViewBag 中的值可以互相访问，因为ViewBag的实现中包含了ViewData，ViewData存的是Key/Value字典，使⽤时需要类型转换。

17. 什么是依赖注⼊,依赖注⼊有哪⼏种⽅式

依赖注入是⼀个过程，就是当⼀个类需要调用另⼀个类来完成某项任务的时候，在调用类里面不要去new被调⽤的类的对象，而是通过注入的方式来获取这样⼀个对象。具体的实现就是在调⽤类里面有⼀个被调用类的接口，然后通过调用接口的函数来完成任务。比如A调用B，而B实现了接口C，那么在A里面用C定义⼀个变量D，这个变量的实例不在A里面创建，而是通过A的上下文来获取。这样做的好处就是将类A和B分开了，他们之间靠接口C来联系，从而实现对接口编程。

依赖注入有以下常见方式：

A．属性注入（setter注⼊）

就是在类A⾥⾯定义⼀个C接⼝的属性D，在A的上下⽂通过B实例化⼀个对象，然后将这个对象赋值给属性D。主要就是set 与 get

B．构造函数注⼊

就是在创建A的对象的时候，通过参数将B的对象传⼊到A中。

还有常⽤的注入方式就是⼯厂模式的应用了，这些都可以将B的实例化放到A外面，从而让A和B没有关系。

还有⼀个接口注入，就是在客户类（A）的接⼝中有⼀个服务类(B)的属性。在实例化了这个接口的子类后，对这个属性赋值，这和setter注入一样。

C．接口注入

相比构造函数注入和属性注入，接口注入显得有些复杂，使用也不常见。具体思路是先定义⼀个接口，包含⼀个设置依赖的方法。然后依赖类，继承并实现这个接口。

18. 什么是中间件(常见的中间件有哪些)

中间件是组装到应⽤程序管道中以处理请求和响应的软件。 每个组件：选择是否将请求传递给管道中的下⼀个组件。可以在调⽤管道中的下⼀个组件之前和之后执⾏⼯作。请求委托（Request delegates）⽤于构建请求管道，处理每个HTTP请求。

常见中间件：

A．开发人员异常⻚中间件 (UseDeveloperExceptionPage) 报告应⽤运⾏时错误。 数据库错误⻚中间件报告数据库运⾏时错误。 当应⽤在⽣产环境中运⾏时： 异常处理程序中间件 (UseExceptionHandler) 捕获以下中间件中引发的异常。 HTTP 严格传输安全协议(HSTS) 中间件 (UseHsts) 添加 Strict-Transport-Security 标头。

B．HTTPS 重定向中间件 (UseHttpsRedirection) 将 HTTP 请求重定向到 HTTPS。

C．静态⽂件中间件 (UseStaticFiles) 返回静态⽂件，并简化进⼀步请求处理。

D．Cookie 策略中间件 (UseCookiePolicy) 使应⽤符合欧盟⼀般数据保护条例 (GDPR) 规定。

E．身份验证中间件 (UseAuthentication) 尝试对⽤户进⾏身份验证，然后才会允许⽤户访问安全资源。

F．⽤于授权⽤户访问安全资源的授权中间件 (UseAuthorization)。

19.