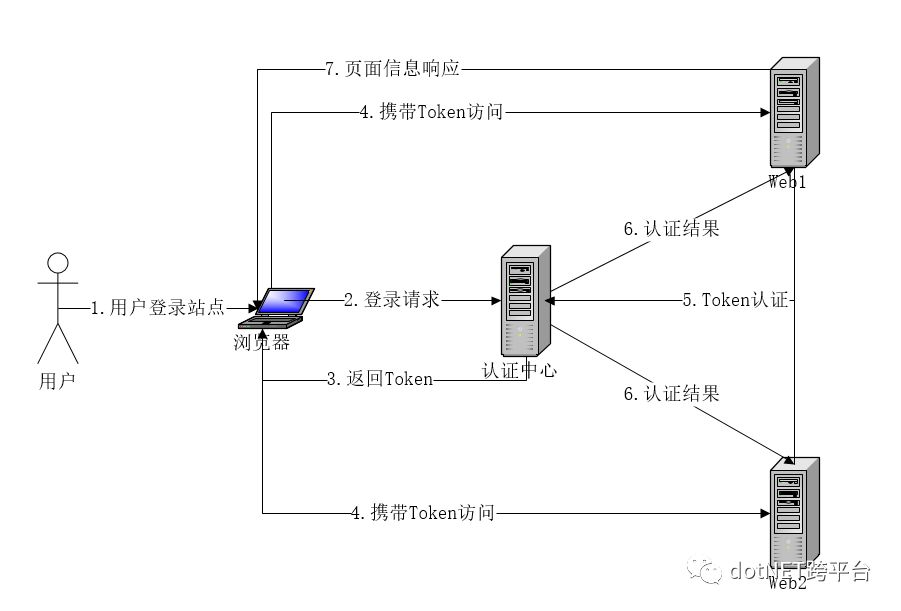
源码地址：<https://github.com/SkyChenSky/Core.SSO>

# **SSO简介**

单点登录，全称为Single Sign On，在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。



# **流程描述**

未登录状态访问业务Web应用会引导到认证中心。

用户在认证中心输入账号信息通过登录后，认证中心会根据用户信息生成一个具有安全性的token，将以任何方式持久化在浏览器。

此后访问其他Web应用的时候，必须携带此token进行访问，业务Web应用会通过本地认证或者转发认证而对token进行校验。

## **Token的生成**

方式有多种：

可以通过Web框架对用户信息加密成Token。

Token编码方式也可以为JSON WEB TOKEN（JWT）

## 

## **Token的共享**

浏览器存储有三种方式：

Cookie

容量4KB限制

过期时间

localStorage

容量5MB限制

生命周期永久

sessionStorage

容量5MB限制

生命周期当前会话，关闭浏览器则失效

无法与服务端交互

Cookie的使用是可以在同域共享的，因此在实现SSO的时候复杂度又分为****同域****与****跨域****。

同域的共享比较简单，在应用设置Cookie的Domain属性进行设置，就可以完美的解决。

## 

## **Token校验**

校验分两种情况：

转发给认证中心认证

业务应用自身认证

　不需要转发，那就意味着业务应用认证规则与认证中心的认证规则必须是一致的。

# **设计要点**

****只要统一Token的产生和校验方式，无论授权与认证的在哪（认证系统或业务系统），也无论用户信息存储在哪（浏览器、服务器），其实都可以实现单点登录的效果。****

此次使用****.NET Core MVC****框架，以****Cookie认证****通过****业务应用自身认证的方式****进行同父域的SSO实现。

## **为什么要使用Cookie认证方式?**

1.会话状态分布在客户浏览器，避免大量用户同时在线对服务端内存容量的压力。

2.横向扩展良好性，可按需增减节点。

## **统一应用授权认证**

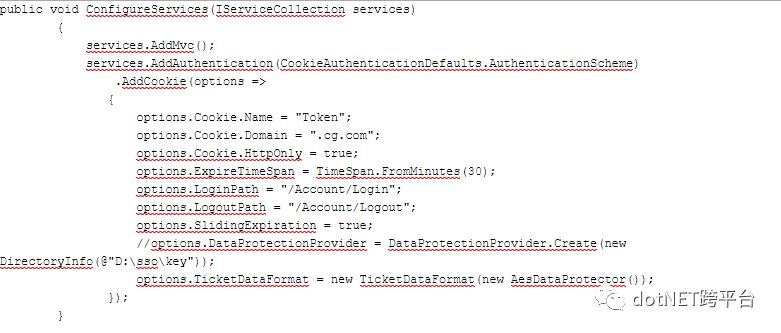
将以Core的Cookie认证进行实现，那么意味着每个应用对用户信息的加解密方式需要一致。

因此对AddCookie的设置属性DataProtectionProvider或者TicketDataFormat的加密方式进行重写实现。

# **.NET Core的SSO实现**

## **Cookie认证**

### **认证中心AddCookie的设置**



### **业务应用AddCookie的设置**



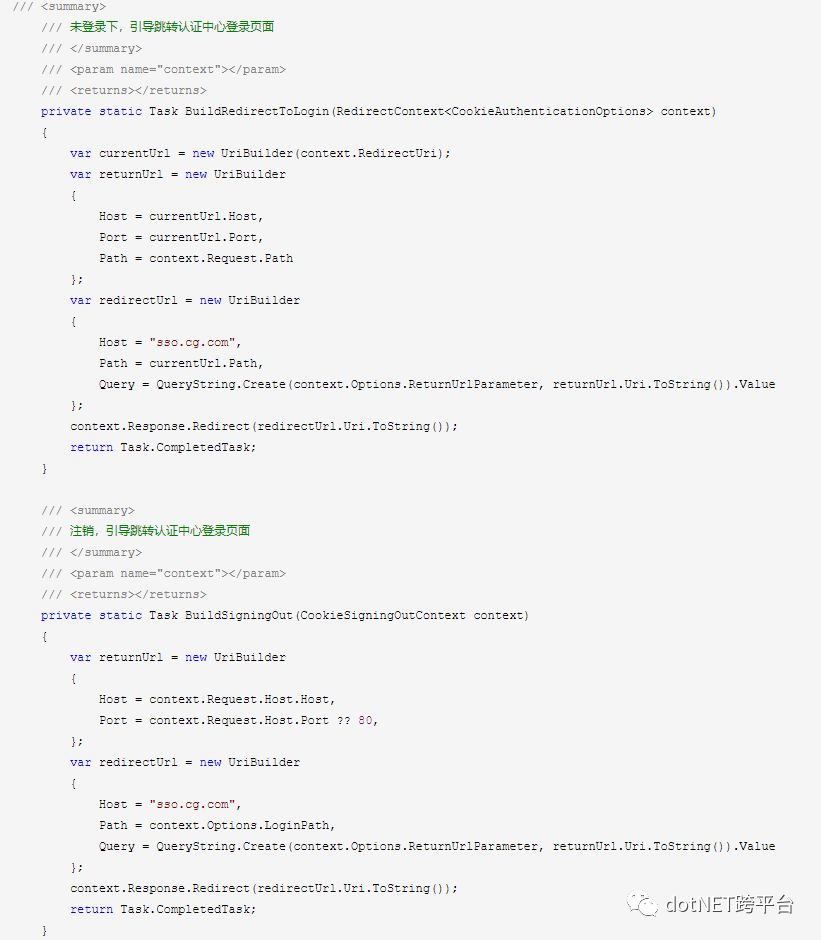
两者的区别不大，ticket的加密方式统一使用了AES，都指定Cookie.Domain = ".cg.com"，保证了Cookie同域共享，设置了HttpOnly避免XSS攻击。

两者区别在于：

options.Events.OnRedirectToLogin = BuildRedirectToLogin;

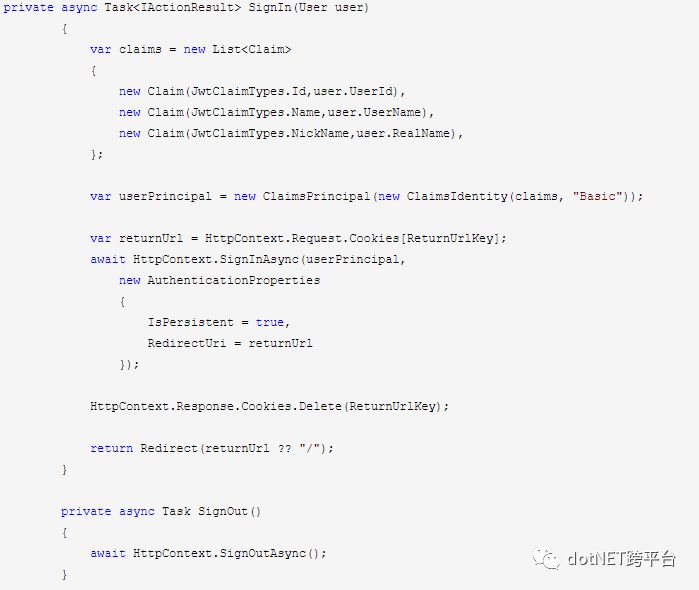
options.Events.OnSigningOut = BuildSigningOut;

这是为了让业务应用引导跳转到认证中心登录页面。OnRedirectToLogin是认证失败跳转。OnSigningOut是注销跳转。



## **登录注销**

认证中心与业务应用两者的登录注册基本一致。



### **HttpContext.SignInAsync的原理**

使用的是Cookie认证那么就是通过Microsoft.AspNetCore.Authentication.Cookies库的CookieAuthenticationHandler类的HandleSignInAsync方法进行处理的。

源码地址：<https://github.com/aspnet/Security/blob/master/src/Microsoft.AspNetCore.Authentication.Cookies/CookieAuthenticationHandler.cs>

从源码我们可以分析出流程：

根据ClaimsPrincipal的用户信息序列化后通过加密方式进行加密获得ticket。

（默认加密方式是的KeyRingBasedDataProtecto。源码地址：<https://github.com/aspnet/DataProtection）>

再通过之前的初始化好的CookieOption再AppendResponseCookie方法进行设置Cookie

最后通过Events.RedirectToReturnUrl进行重定向到ReturnUrl。

## **Ticket加密**

两种设置方式

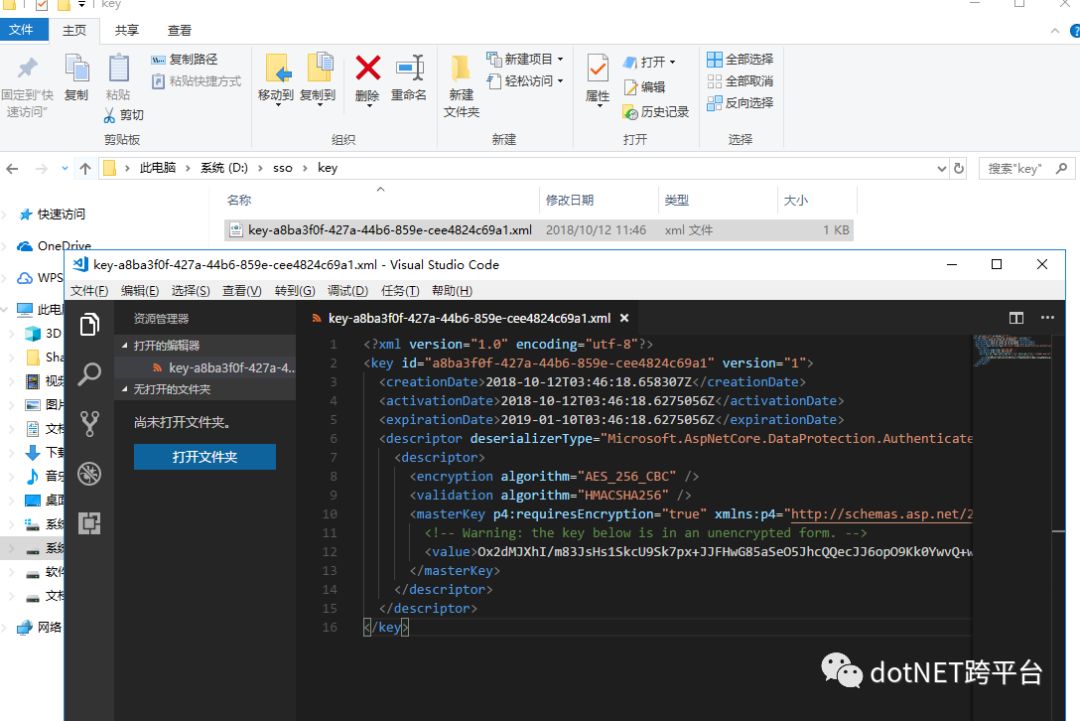
CookieAuthenticationOptions.DataProtectionProvider

CookieAuthenticationOptions.TicketDataFormat

### **DataProtectionProvider**

如果做了集群可以设置到共享文件夹，在第一个启动的应用则会创建如下图的文件

options.DataProtectionProvider = DataProtectionProvider.Create(new DirectoryInfo(@"D:\sso\key"));



### 

### **TicketDataFormat**

重写数据加密方式，本次demo使用了是AES.

options.TicketDataFormat = new TicketDataFormat(new AesDataProtector());

