云归档接口整理

1.

/Plugin/List

浙江-蜗牛（彪哥） 2018/12/11 14:00:12

这个地址下面是插件管理界面

浙江-蜗牛（彪哥） 2018/12/11 14:00:29

Admin

tomtaw

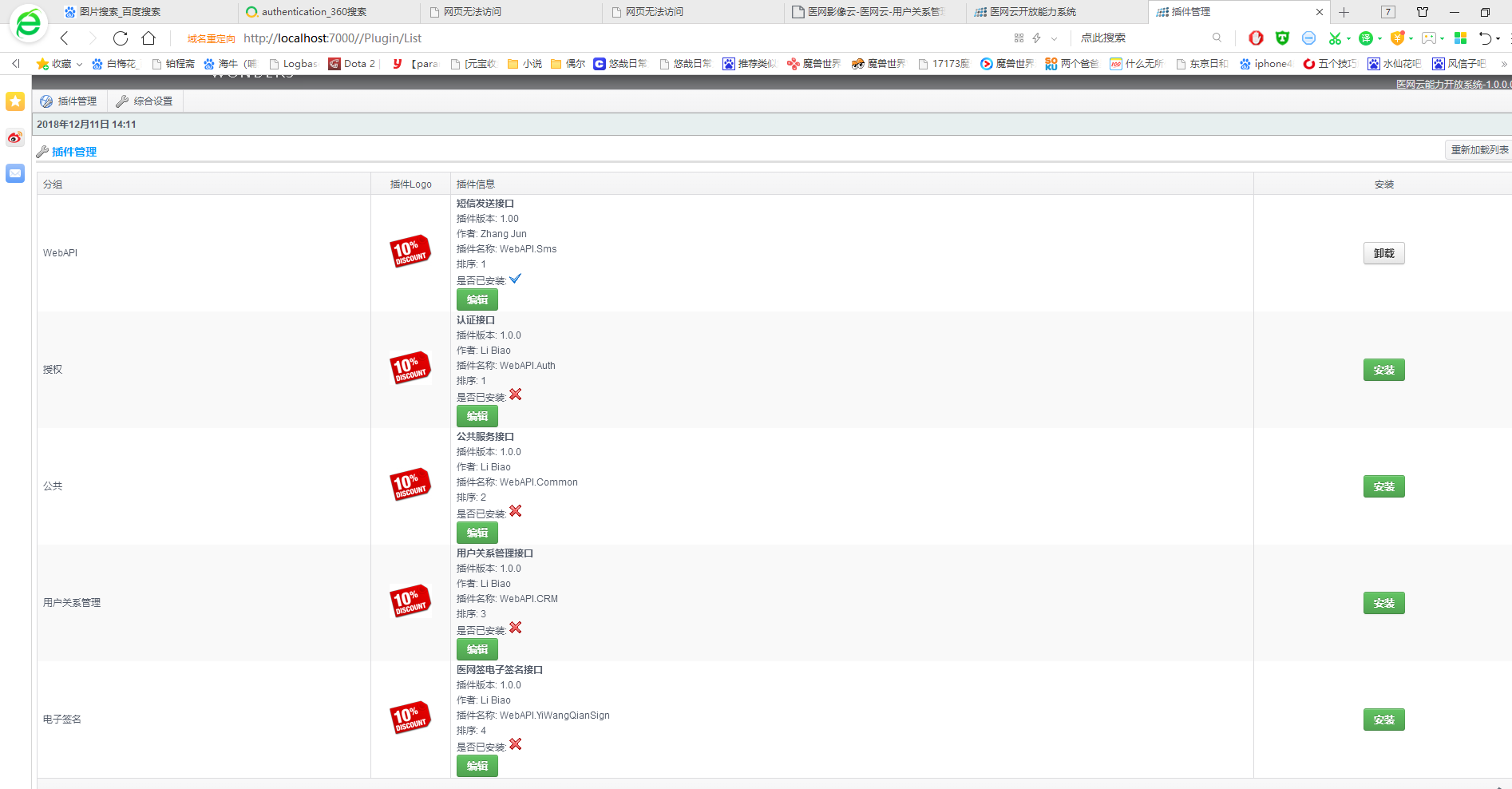
浙江-蜗牛（彪哥） 2018/12/11 14:00:31

这是账号

浙江-蜗牛（彪哥） 2018/12/11 14:00:55

检查下是不是没有安装插件

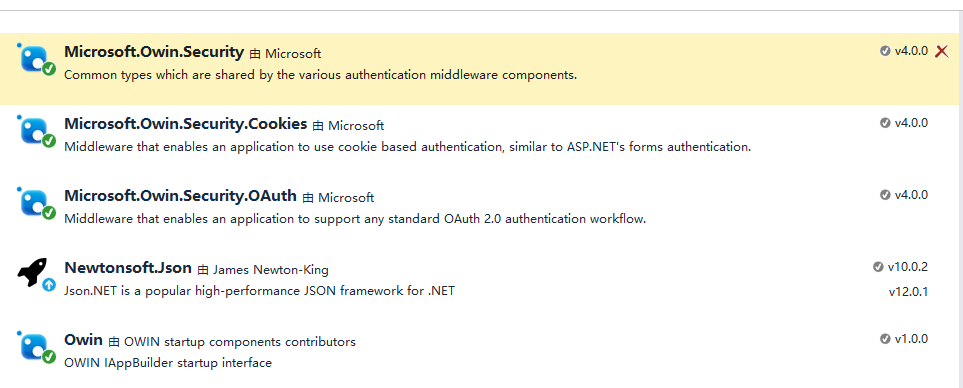
接下来进入插件页面



原来这里的能力开放平台，接口的呈现是以插件的形式。比方我的能力云归档整理，最后也是这么一个插件，下载安装了，就可以有权限调我的接口了。

2.

接口是以类库，以插件的形式。



在nuget中搜索到了这些内容：

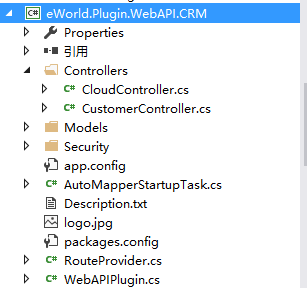


3.

在自定义一个云归档的接口

eWorld.Plugin.WebAPI.YunGuiDang

找彪哥确认下，是按照网上的nuget中安装mvc的方式，还是直接把彪哥这些东西复制过来，修改修改就可以。



曹兵那边应该是wcf整理好，我NuGet中直接安装，然后就可以调用了。

4.

OAuth 2.0框架

基于OAuth2.0实现开放平台授权中心

OAuth (开发授权)是一个开放标准，允许用户让第三方应用访问该用户在某一网站上存储的秘密的资源（如 照片、视频、联系人列表）而无需将用户名和密码提供给第三方应用

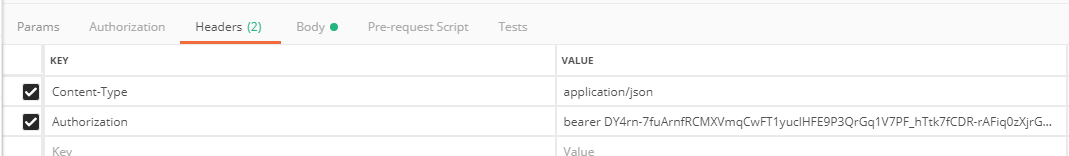
类似，登陆一些应用。比方微信小程序，获取微信的头像啊，信息啊，这些。

就是我们点统一的同时，也许我们的账号密码，身份验证了，腾讯返回一个access\_token，这样，其他的网站都可以通过这个token去微信拿到我们的身份信息了。

类似我从张军的api接口中，把加密后的账号密码发过去，就拿到了token，这样，接下来所有的api请求，我只要把token放在请求头的头部，就不需要每次都登陆，而服务端也能允许此次请求通过了。

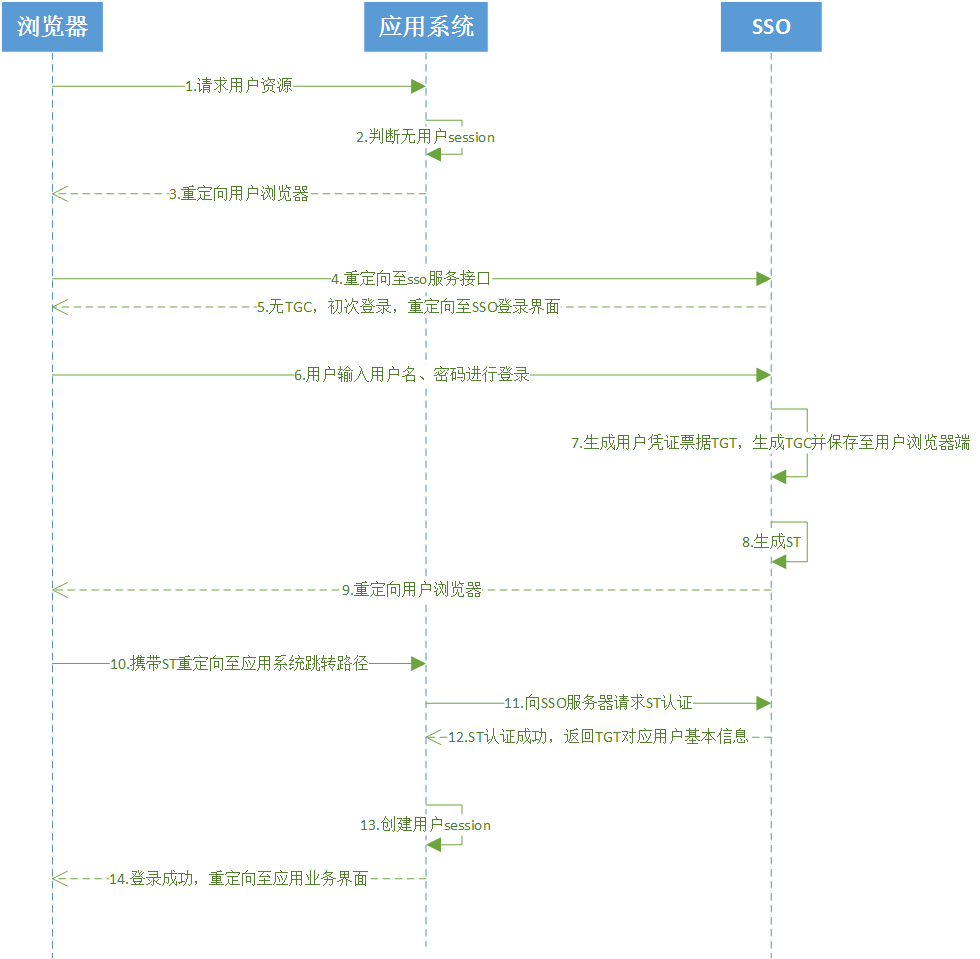
var result = WebHelper.LoginRequestUtile($"{param.ApiUrl.TrimEnd('/')}/api/login", WebRequestMethods.Http.Post, bodyData, "",

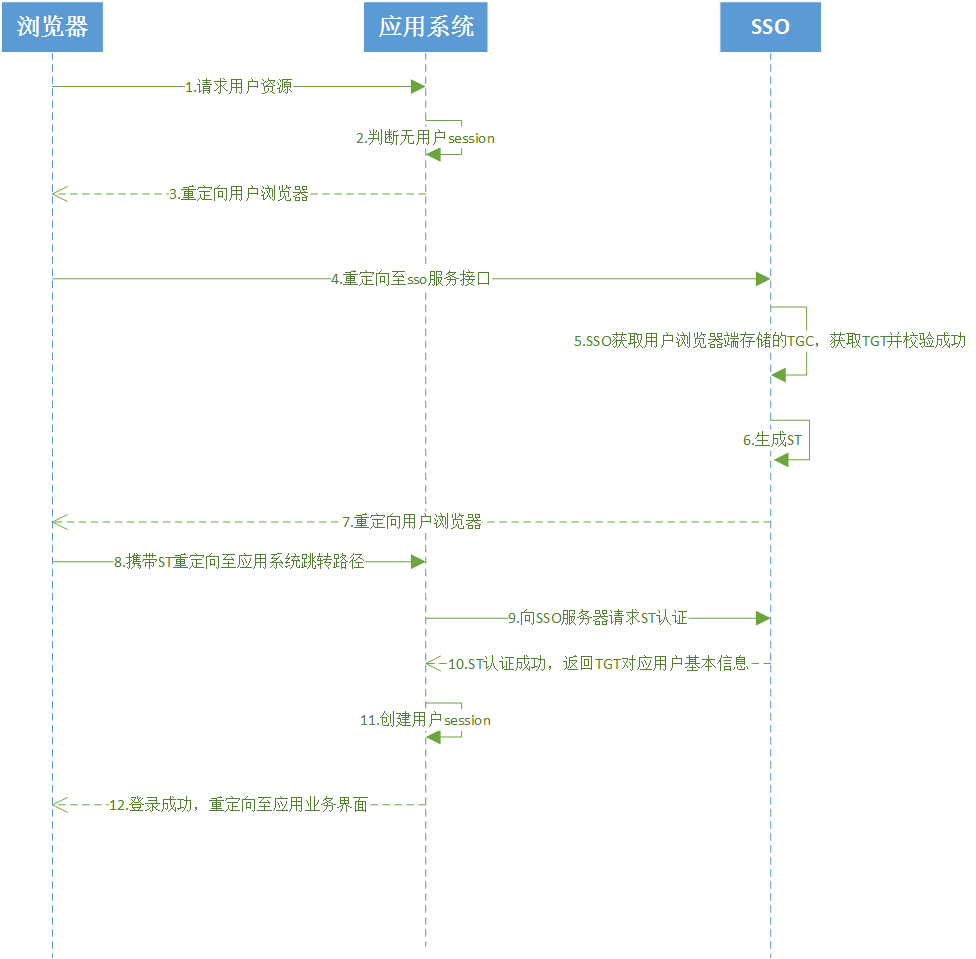
"text/plain");



5.

sso单点登陆的图，联想到了第三方登陆代码。各种Redirect跳转。





#region 第三方登录

/// <summary>

/// 第三方登录

/// </summary>

/// <param name="LoginName">用户名</param>

/// <param name="Password">密码</param>

/// <param name="ReturnUrl">返回页面</param>

/// <returns></returns>

public ActionResult ThirdLoginNoPassword(string LoginName, string Password, string ReturnUrl)

{

var crmUrl = \_crmSettings.Url;

var currentUrl = BaseCommon.WebSiteUrl + "/ThirdRedirect/ThirdLoginNoPassword";

var TimeStamp = DateTime.Now.ToString("yyyyMMddHHmmss");

var ThirdKey = \_platInfoSettings.ThirdKey;

var Signatrue = MD5Encrypt.GetUnicodeMD5(LoginName + TimeStamp + ThirdKey);

if (!\_ssoProvider.IsLogin)

{

var queryToken = HttpContext.Request["tk"];

if (IsNullOrEmpty(queryToken))

{

currentUrl += "?LoginName=" + LoginName + "&Password=" + Password + "&ReturnUrl=" + ReturnUrl;

//跳转到GetToken界面

return Redirect(\_ssoProvider.GetTokenUrl(crmUrl, currentUrl));

}

currentUrl += "?LoginName=" + LoginName + "&Password=" + Password + "&ReturnUrl=" + ReturnUrl;

crmUrl += "/Login/SingleLoginByUserName?LoginName=" + LoginName

+ "&Password=" + Password

+ "&BackUrl=" + HttpUtility.UrlEncode(currentUrl)

+ "&TimeStamp=" + TimeStamp

+ "&Signatrue=" + Signatrue;

if (queryToken == "$Token$")

{

return Redirect(crmUrl);

}

currentUrl = Regex.Replace(currentUrl, @"&signatrue=[^&]\*", string.Empty);

currentUrl = Regex.Replace(currentUrl, @"\?tk=[^&]\*&?", Empty);

Regex.Replace(currentUrl, @"&tk=[^&]\*&?", Empty);

//Redis凭据已删除，但CRM的Cookie仍在

if (\_ssoProvider.GetCustomerGuidByQueryToken(queryToken) == null)

{

return Redirect(crmUrl);

}

//Token写入Cookie中

\_ssoProvider.SubSiteLogin(queryToken);

}

return Redirect(BaseCommon.WebSiteUrl + $"{new RedirectHelper().Do("", ReturnUrl, "client")}");

}

#endregion

6.

协同的code，msg都是小写，彪哥这是是大写，所以最后做了小写处理。

namespace eWorld.Web.Framework.Models

{

/// <summary>

/// 无返回值数据模型

/// </summary>

public class BaseResultModel

{

public GlobalResultCode Code { get; set; }

public string Msg { get; set; }

}

怀疑是在global中，把统一格式给替换了。

GlobalConfiguration.Configuration.Formatters.Clear();

GlobalConfiguration.Configuration.Formatters.Add(new System.Net.Http.Formatting.JsonMediaTypeFormatter()

{

SerializerSettings = new JsonSerializerSettings()

{

ContractResolver = new LowerCasePropertyNameContractResolver()

}

});

继续查找new LowerCasePropertyNameContractResolver()：

namespace eWorld.Web.Framework.Mvc

{

public class LowerCasePropertyNameContractResolver: DefaultContractResolver

{

protected override string ResolvePropertyName(string propertyName)

{

return propertyName.ToLower();

}

}

}

7.

8.

9.

10.