目　录

单点登录

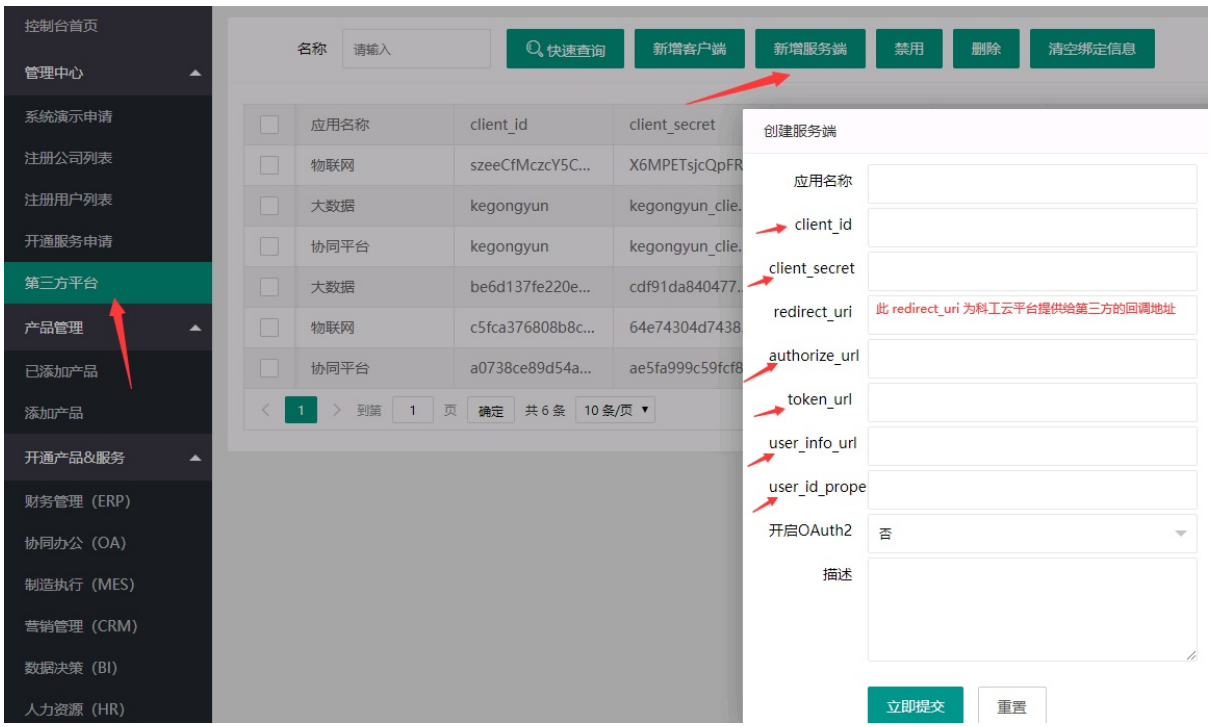
单点登录客户端

科工云登录第三方平台

登录科工云控制台，实现导航栏单点登录

第一步：添加服务端配置参数

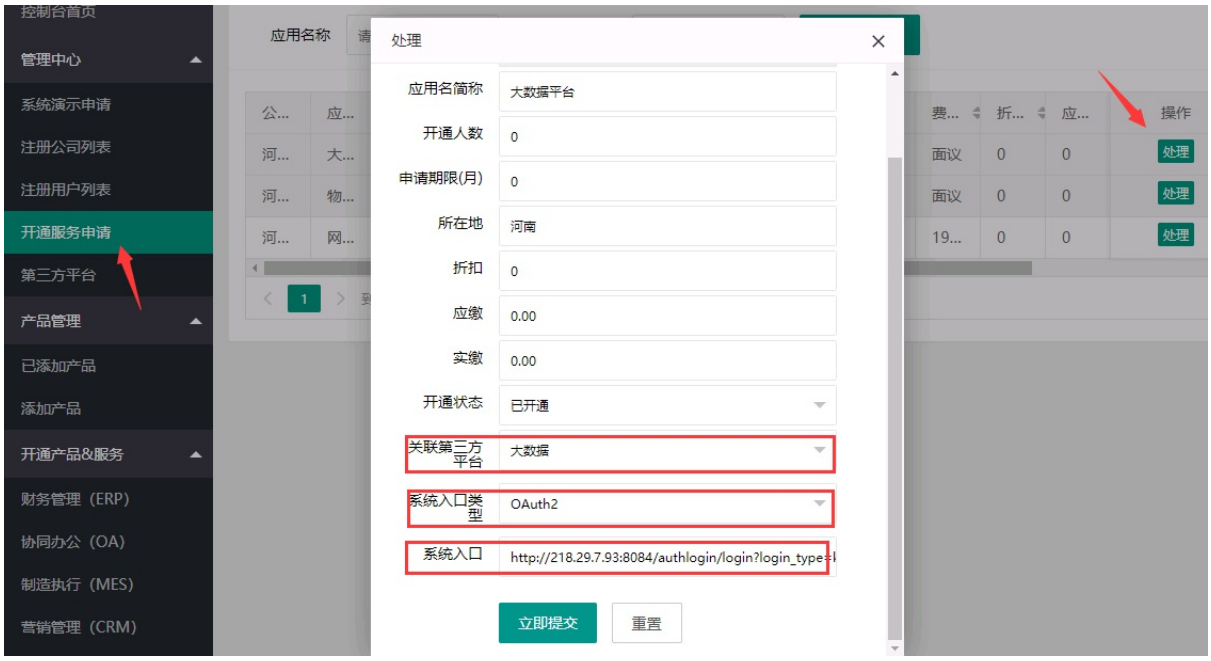
1.进入系统: 第三方平台-新增服务端。
2.输入第三方提供的配置参数和科工云平台提供给第三方平台的回调地址（redirect_uri）。
3.点击立即提交进行保存。



第二步：第三方应用关联科工云开通服务

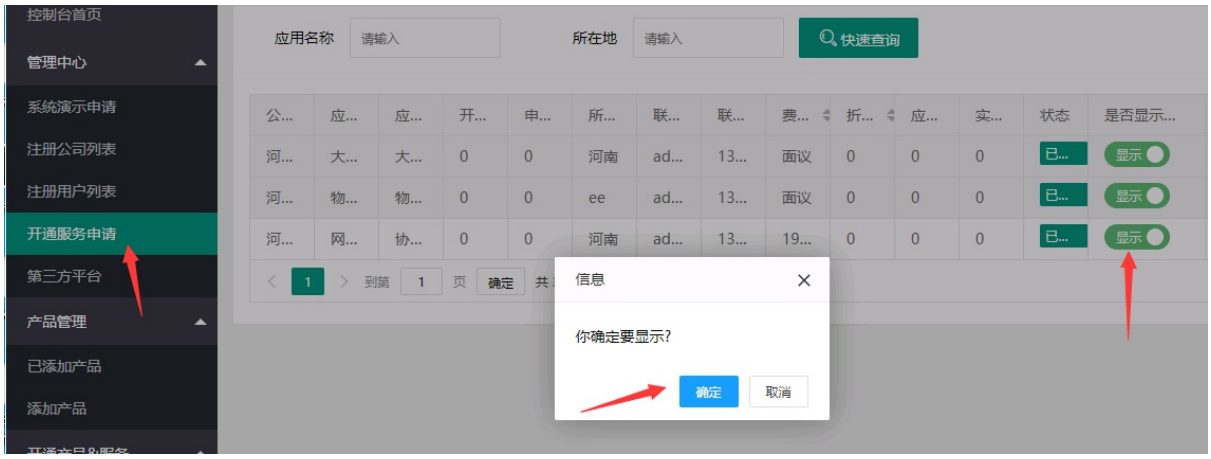
1.进入系统: 开通服务申请-操作-处理。

2.选择关联的第三方平台，系统接口类型为OAuth2,填写第三方提供的系统入口链接
3.点击立即提交进行保存。

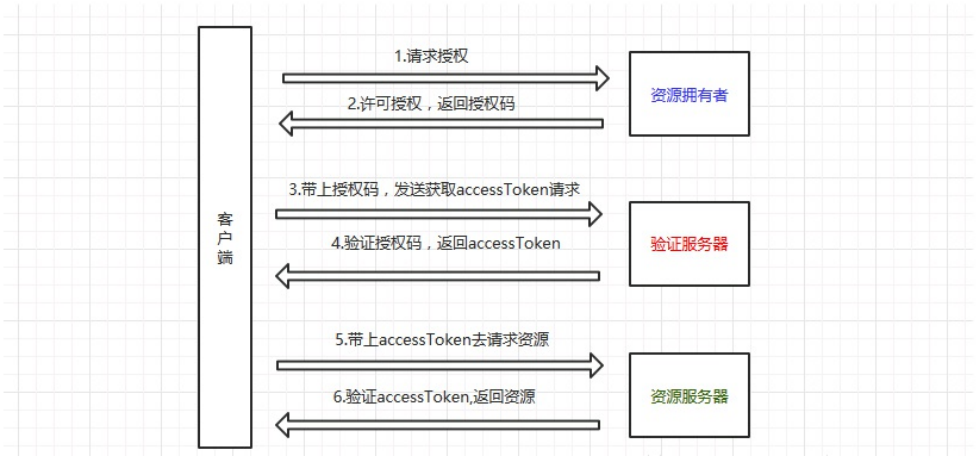


第三步：第三方应用显示在导航栏

1.进入系统:该服务处于开通状态的情况下，在列表是否显示在导航栏点击按钮出现弹框
2.点击确定按钮即可显示在导航栏



科工云作为客户端授权登录流程



科工云客户端单点登录客户端C#代码示例

```
public ActionResult BigData_CallBack()
        {
            var member = LoginMember.Login();
            //第一步，请求获取code（请求OAuth服务器）
            string code = Request["code"] ?? "";
            if (string.IsNullOrEmpty(code))
            {
               //请求获取code为null 生成请求链接（请求OAuth服务器的code）
                List<tb_oauth_server> list = dal_oauth_server.GetList(0, "
type=1", "").ToList();
                tb_oauth_server data_set = list.Where(c => c.name == "bigda
ta").FirstOrDefault();
                Random rd = new Random();
                string response_type = "code";
                string scope = "user_info";
                string state = Utils.MD5(rd.Next(1, 100000).ToString());

                BigDataOAuth e = new BigDataOAuth();
                var authorize  = e.GetAuthorizeUrl(data_set.authorize_url,
data_set.client_id, data_set.redirect_uri);
                //生成请求链接重定向到OAuth服务器
                Response.Redirect(authorize);
                return null;
            }
            else
            {
                //第二步，获取code之后请求获取token（请求OAuth服务器）
                //获取应用的配置信息
```

```
                    List<tb_oauth_server> list = dal_oauth_server.GetList(0
, " type=1", "").ToList();
                    tb_oauth_server data_set = list.Where(c => c.name == "b
igdata").FirstOrDefault();
                //此步骤包括根据授权码code获取accessToken,根据返回的accessTo
ken获取用户信息
                    BigDataOAuth e = new BigDataOAuth();
                    BigDataModel authorize = e.GetModel(code,data_set.clien
t_id,data_set.client_secret,data_set.redirect_uri,data_set.token_url,data_s
et.user_info_url);

                    var name = authorize.login;
                    var id = authorize.id;

                //保存用户信息到本地服务器
                    Session["name"] = name;
                    Session["id"] = id;
                    string proviter = "大数据";
                    Session["proviter"] = proviter;
                    //获取第三方在科工云的绑定信息
                    tb_member_oauth auth = dal_member_auth.GetModelByOauthI
d(authorize.id);
                    if (auth == null && member == null)
                    {
                        Response.Redirect("/sso/login");
                    }
                    else if (member != null && auth == null)
                    {
                        tb_member_oauth auth_model = new tb_member_oauth();
                        auth_model.user_id = member.id;
                        auth_model.oauth_name = name;
                        auth_model.proviter = proviter;
                        auth_model.bind_time = DateTime.Now;
                        auth_model.oauth_id = id;
                        if (dal_member_auth.Insert(auth_model) > 0)
                        {
                            //tb_member new_model = dal_member.GetModel(mem
_id);
                            var login_name = "Member" + Utils.MD5(member.lo
ginname);
                            Memcache.Set(login_name, member, 60 * 2);
                            TYSystem.Public.Cookie.CreatCookie("_cookie_mem
ber_login", login_name, 2, Utils.GetTopDomainName(Request.Url.ToString()));
                        }

                        return RedirectToAction("Index", "Home");
```

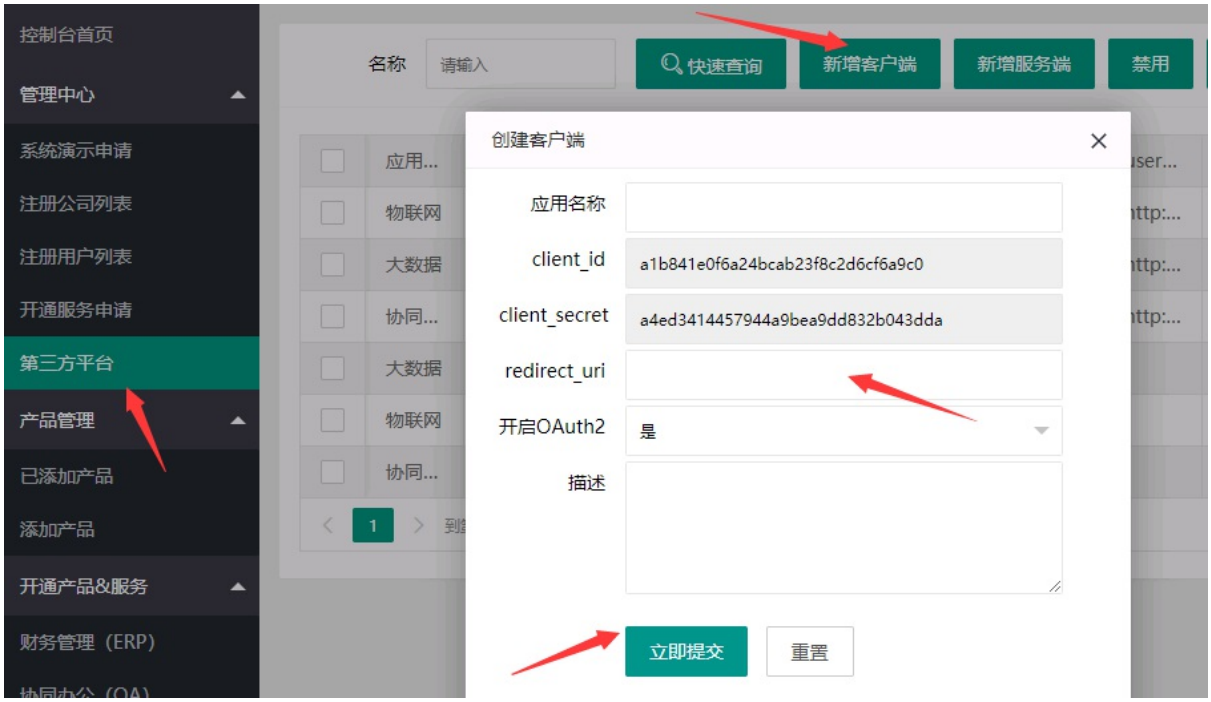
```
                    }
                    else
                    {

                        var mem = dal_member.GetModel(auth.user_id);
                        var login_name = "Member" + Utils.MD5(mem.phone);
                        Memcache.Set(login_name, mem, 60 * 2);
                        TYSystem.Public.Cookie.CreatCookie("_cookie_member_
login", login_name, 2, Utils.GetTopDomainName(Request.Url.ToString()));
                        return RedirectToAction("Index", "Home");
                    }
                    return RedirectToAction("Index", "Home");
                }
```

单点登录服务端

第三方平台登录科工云

一.生成第三方配置

1.进入系统: 第三方平台-新增客户端。
2.会自动生成client_id，client_secret，填写第三方提供的回调地址（redirect_uri）。
3.点击立即提交进行保存。



二．科工云单点登录配置

科工云服务端配置

1.授权：http://auth.kegongyun.com/Oauth/Authorize
2.令牌：http://auth.kegongyun.com/Oauth/Token
3.用户：http://auth.kegongyun.com/Oauth/User

三．科工云作为服务端登录流程

3.1 第一步，开始授权验证，并且跳转到指定的授权页面。

```
        /// <summary>
        /// 第一步，开始授权验证
```

```
        /// </summary>
        /// <param name="client_id"></param>
        /// <param name="response_type"></param>
        /// <param name="redirect_uri"></param>
        /// <param name="state"></param>
        /// <param name="scope"></param>
        /// <returns></returns>
        public ActionResult Authorize(string client_id, string response_typ
e, string redirect_uri, string scope, string state = "")
        {
            if ("code".Equals(response_type))
            {
                //判断client_id是否可用
                if (!_oAuth2ServerServices.IsClientIdValied(client_id))
                    return this.Json("client_id 无效", JsonRequestBehavior.
AllowGet);

                //保存用户请求的所有信息到指定容器（session）
                Session.Add("client_id", client_id);
                Session.Add("response_type", response_type);
                Session.Add("redirect_uri", redirect_uri);
                Session.Add("state", state);
                Session.Add("scope", scope);
                //科工云未登录时需要先登录，去掉了授权页面，默认同意授权
                 if (member != null)
                {
                    ViewBag.memberName = member.phone;
                    //return View();去掉授权页面

                    return RedirectToAction("Authenticate", "Oauth");
                }
                else
                {
                    return RedirectToAction("index", "login");
                }
            }
            return View("Error");
        }
```

客户端默认授权后会请求授权验证方法

首先，验证client_id是否可用，这里的client_id是为了保证安全性，确保请求端是服务端给予请求或者授权的权利。简单地说，就是请求端在用此服务端之前要申请唯一的一个client_id；

然后，在把客户端传过来的信息保存在Session（你也可以保存在其他地方）；

最后，跳转到同意授权后的操作

3.2 默认同意授权后返回code到请求端

生成code，并且设定code的生存时间，默认是30秒。（code只能用一次，之后要删除）；

再绑定code与用户信息；

最后，重定向回redirect_uri请求的地址，并且返回code与**state**。（**state**是请求端那边想要用于处理一些业务逻辑所用到的，当然可以为空）

```
        /// <summary>
        /// 第二步，用户确认授权后的操作。
        /// 用户确认授权后，则返回code、access_token，并重定向到redirect_uri所指定的页面
        /// </summary>
        /// <returns></returns>
        public ActionResult Authenticate()
        {

             var member = LoginMember.Login();
            //取得重定向的信息
            var redirect_uri = Session["redirect_uri"] ?? "";
            var state = Session["state"] ?? "";
            string code = TokenCodeUtil.GetCode();

            //保存code到DB/Redis，默认存在30秒
            _oAuth2ServerServices.SaveCode(code);
            //绑定code与member，因为后面查询用户信息的时候要用到，默认存在30秒
            _oAuth2ServerServices.BingCodeAndUser(member, code);
            //重定向
            string url = string.Format(HttpUtility.UrlDecode(redirect_uri.ToString()) + "?code={0}&state={1}", code, state);
            Response.Redirect(url);

            return null;
        }
```

3.3 获取token

在请求端获取到code之后，请求端要获取token，因为获取了token请求端才能获取到用户信息等资料。

首先，把token设置成不能保持**cache**的状态，为了保证安全性；

然后，判断是获取token还是刷新token的状态；

再验证code是否过期，验证client_id、client_secret是否正确；

再生成token，把token存入容器（DB、Redis、**Memory**等）中；

在通过code来获取用户的信息，把用户信息（主键）与token做绑定；

最后，把code删除（code只能用一次，如果想再获取token只能第一步开始重新做），返回token。

```
        /// <summary>
        /// 获取或刷新token。
        /// token可能保存在DB/Redis等
        /// </summary>
        /// <param name="code"></param>
        /// <param name="grant_type"></param>
        /// <param name="client_id"></param>
        /// <param name="client_secret"></param>
        /// <returns></returns>
        public ActionResult Token(string code, string grant_type, string client_id, string client_secret)
        {
            Response.ContentType = "application/json";
            Response.AddHeader("Cache-Control", "no-store");

            //获取token
            if (grant_type == "authorization_code")
            {
                //判断code是否过期
                if (!_oAuth2ServerServices.IsCodeValied(code, DateTime.Now))
                    return this.Json("code 过期", JsonRequestBehavior.AllowGet);
                //判断client_id与client_secret是否正确
                if (!_oAuth2ServerServices.IsClientValied(client_id, client_secret))
                    return this.Json("client_id、client_secret不正确", JsonRequestBehavior.AllowGet);
                //新建token
```

```
                string access_token = TokenCodeUtil.GetToken();
                //保存token，默认是30分钟
                _oAuth2ServerServices.SaveToken(access_token);
                //通过code获取userid，然后用token与userid做绑定，最后把code设
置成消失（删除）
                string userId = _oAuth2ServerServices.GetUserIdFromCode(cod
e);
                if (string.IsNullOrEmpty(userId))
                    return this.Json("code过期", JsonRequestBehavior.AllowG
et);
                _oAuth2ServerServices.BingTokenAndUserId(access_token, user
Id);
                _oAuth2ServerServices.RemoveCode(code);

                //返回token
                return this.Json(access_token, JsonRequestBehavior.AllowGet
);
            }
            //刷新token
            else if (grant_type == "refresh_token")
            {
                //新建token
                string new_access_token = TokenCodeUtil.GetToken();
                //替换保存新的token，默认是30分钟

                //返回新建的token
                return this.Json(new_access_token, JsonRequestBehavior.Allo
wGet);
            }
            return this.Json("error grant_type=" + grant_type, JsonRequestB
ehavior.AllowGet);
        }
```

3.4 通过token获取用户信息

上面请求端已经获取到了token，所以这里只需要验证token，token验证通过就直接返回用户信息。

验证token包括验证是否存在、验证是否过期。

```
        /// <summary>
        /// 通过token获取用户信息
        /// </summary>
        /// <param name="oauth_token"></param>
        /// <returns></returns>
```

```
        public ActionResult User(string oauth_token)
        {
            if(!_oAuth2ServerServices.IsTokenValied(oauth_token, DateTime.N
ow))
                return this.Json("oauth_token无效", JsonRequestBehavior.All
owGet);
            UserInfo u = _oAuth2ServerServices.GetUserInfoFromToken(oauth_t
oken);
            return this.Json(u, JsonRequestBehavior.AllowGet);
        }
```