微信授权登录

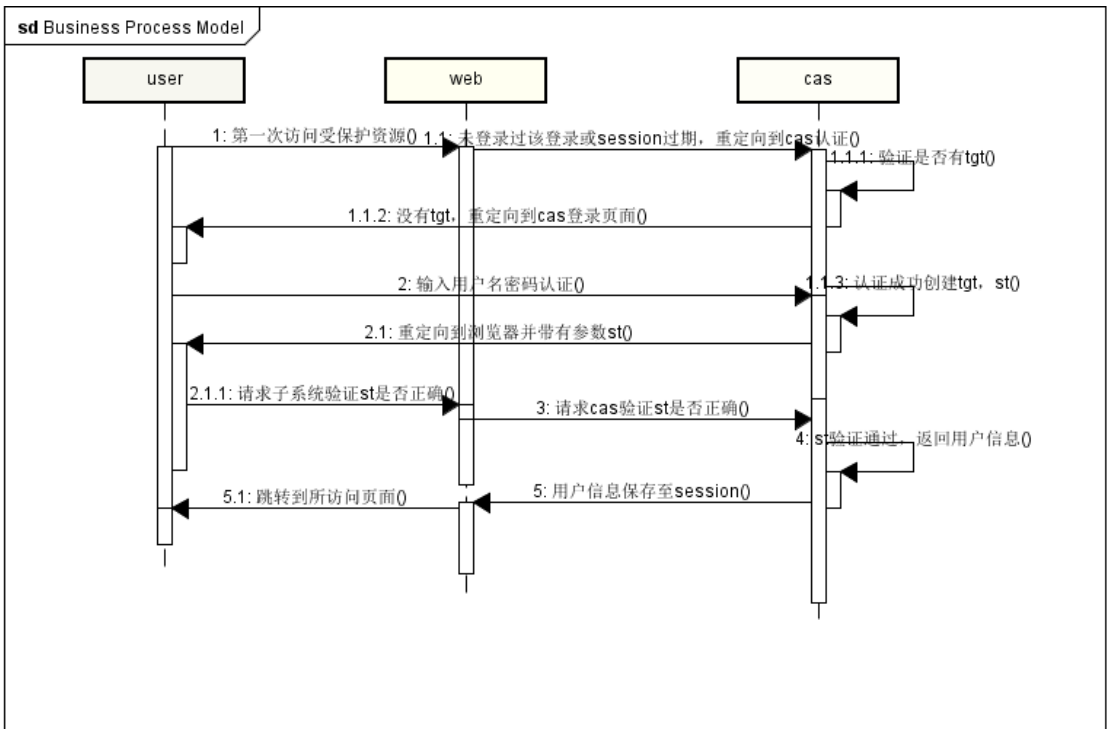
一、 简单介绍一下微信平台

1. 微信平台包括公众平台、开放平台、商户平台，其分别负责完成的功能如下：
   公众平台：主要是负责微信客户端。
   开放平台：主要是负责 pc 端和 app。
   商户平台：主要负责微信支付相关的内容，包括所有的客户端类型。
2. 注册微信账号包括订阅号、服务号、企业号，不同的账号类型对应获取的权限接口相应有一些区别，在新建账号的时候可以自己根据所需接口选择相应的账号类型。

3. 微信注册成功之后可获得当前微信号的 appId 和 appSecret，分别表示应用 ID 和秘钥（很重要），是当前微信号的唯一标识，在微信授权登录、微信支付、微信签名等地方都需带上该参数。

二、 简单介绍一下 CAS 单点登录（SSO）

1. **What is CAS?**
   CAS （ Central Authentication Service ）是 Yale 大学发起的一个企业级的、开源的项目，旨在为 Web 应用系统提供一种可靠的单点登录解决方法（属于 Web SSO ）。
2. **What is SSO?**
   单点登录（ Single Sign-On , 简称 SSO ）是目前比较流行的服务于企业业务整合的解决方案之一， SSO 使得在多个应用系统中，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。
3. **CAS 原理**
   用户第一次访问受保护资源的 cas 授权的时序图如下



简单的讲，CAS client 将用户转向 CAS server，用户在 CAS server 上登陆，产生了 ticket。CAS server 将用户转回 CAS client 页面，带着 ticket 参数。CAS client 悄悄的使用 httpClient 向 CAS server 索取用户全部信息，凭证就是 ticket。CAS server 将用户信息返回给 CAS client，client 对用户进行检验并重定向到原始请求的页面。

4. **关于我们平台的 CAS（包括了 security 说明）**

- 平台的每一个子系统都整合了 Spring Security（这是一种基于 Spring AOP 和 Servlet 过滤器的安全框架，提供在 web 请求时处理身份确认和授权）。下面简单讲解一下 Spring Security

  首先在 web.xml 中进行了如下配置：

```
<!-- 启动Spring的安全框架 -->
<filter id="Filter_security">
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping id="Filter_Mapping_Security">
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

  对用户的所有请求都进了过滤，委托给 Spring 进行处理。

  那么该过滤器是怎么初始化的呢？

  在系统中我们有如下配置，在系统初始化时会启动该配置：

```
<security:http auto-config="false" use-expressions="false" entry-point-ref="cas_loginUrlAuthEntrPoint">
    <security:intercept-url pattern="/statics/**" filters="none" />
    <security:intercept-url pattern="/remote/**" filters="none" />
    <security:intercept-url pattern="/home/admin/spaces/**" access="ROLE_SUPERMAN,ROLE_SPACE_ADMIN"/>
    <security:intercept-url pattern="/home/admin/resourcebanks/**" access="ROLE_SUPERMAN,ROLE_SPACE_ADMIN"/>
    <security:intercept-url pattern="/home/admin/**" access="ROLE_SUPERMAN"/>
    <security:intercept-url pattern="/home/**" access="ROLE_LOGINUSER"/>
    <security:logout logout-url="/cas_security_logout" invalidate-session="true" success-handler-ref="cas_lo
    <security:custom-filter ref="cas_authenticationProcessingFilter" position="FORM_LOGIN_FILTER" />
    <security:session-management>
        <security:concurrency-control max-sessions="3" session-registry-ref="cas_account_sessionRegistryMana
    </security:session-management>
</security:http>
```

配置说明：

http 标签	auto-config 配置为 true 表示使用系统默认设置。自定义过滤器失效。
	entry-point-ref 表示认证失败后的处理
	use-expression：是否使用 spEL，spEL 如：“hasRole(ROLE_SUPERMAN)”等
Intercept-url 标签	表示访问指定连接需要的权限
logout 标签	表示退出登录时的操作,退出成功清空会话，这个地方有 bug。
custom-filter 标签	表示自定义的过滤器，其中 position 表示过滤器执行的顺序，表示替换，相应的还有 after 和 before。
session-managger 标签	max-sessions 表示一个用户在该子系统登录的最多次数 session-registry-ref 维护当前子系统的 session，实现了 sessionRegisterImpl（spring），其中当前应用的在线统计等功能是在这里做的。

其中 custom-filter 实现如下：

```
<bean id="cas_authenticationProvider"
    class="cn.com.eduedu.jee.security.cas.CASAuthenticationProvider">
    <property name="casValidateServiceUrl" value="${security.cas.validate.url}" />
    <property name="authService" ref="sec_md5AuthenticationClientService" />
    <property name="serviceUrl" value="${security.login.url}" />
</bean>
<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider
        ref="cas_authenticationProvider" />
</security:authentication-manager>
<bean id="cas_authenticationResultHandler"
    class="cn.com.eduedu.jee.security.cas.CASAuthenticationResultHandler">
    <property name="defaultTargetUrl" value="${security.login.default.target.url}" />
    <property name="alwaysUseDefaultTarget" value="${security.login.use.default.target}" />
</bean>
<bean id="cas_authenticationProcessingFilter"
    class="cn.com.eduedu.jee.security.cas.CASAuthenticationProcessingFilter">
    <property name="authenticationManager" ref="authenticationManager"></property>
    <property name="authenticationSuccessHandler" ref="cas_authenticationResultHandler" />
    <property name="authenticationFailureHandler" ref="cas_authenticationResultHandler" />
</bean>
```

这里实现了认证处理，以及认证成功和失败的处理，其中若认证失败会抛出

异常。认证过程就是去请求 cas，并且带上 st 参数,st 是针对每一个小系统。

1. 若认证失败抛出异常（没有 st 或还未登陆），进入上一次配置的 entry-point-ref，在这里会去检查 cas 会话中的 tgt（判断用户是否在 cas 登陆）。
   若已登录，则 cas 创建 st，返回给 security 再次认证（即下面的 2）。
   若未登录，则跳转到 cas 登录页面，并记录访问的 url（因为登录成功会跳转到指定页面会再次验证 st 的正确性，即下面的 2）。
2. 若认证成功，保存用户信息进 session，跳转到访问页面。

注意：这里只对我们平台中的配置进行了一下说明，其实 Spring Security 的配置远非如此，若有兴趣还需小伙伴没自行下来研究。

- 下面对请求 cas 的登录过程进行分析
  访问 cas(会先记录租户编码和访问的 url)

当用户输入用户名密码，且登录成功，会在这时会创建 tgt（按指定规则，需要使用用户的基本信息创建），并保存至会话。
后面的流程和上面一致,cas 创建 st，security 根据 st 再次请求 cas，若成功跳转到指定页面。

三、 简单介绍一下微信 OAuth2.0 网页授权

1. What is OAuth2.0

OAuth 是一个开放协议，允许用户让第三方应用以安全且标准的方式获取该用户在某一网站、移动或桌面应用上存储的私密的资源（如用户个人信息、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。

OAuth 2.0 是 OAuth 协议的下一版本，但不向后兼容 OAuth 1.0，OAuth 2.0 关注客户端开发者的简易性，同时为 Web 应用、桌面应用、手机、和起居室设备提供专门的认证流程。新浪微博、腾讯 QQ 等目前授权也使用的 OAuth2.0 认证。

2. 微信授权类型

分为网微信客户端内授权、pc 端扫码授权、app 授权。

3. unionID 和 openID

openID：网站上或应用中唯一对应用户身份的标识，可存于本地数据库用于标识微信用户的唯一性。

unionID：如果开发者拥有多个移动应用、网站应用和公众帐号，需要前往微信开放平台绑定公众号，可通过获取用户基本信息中的 unionid 来区分用户的唯一性，因为同一用户，对同一个微信开放平台下的不同应用（移动应用、网

站应用和公众帐号），unionid 是相同的。

4. **微信授权作用域 scope**

表示用户授权的范围，例如：微信登录——获取用户基本信息
QQ 登录——获取用户基本信息，分享至空间等
微博登录——授权访问微博账号

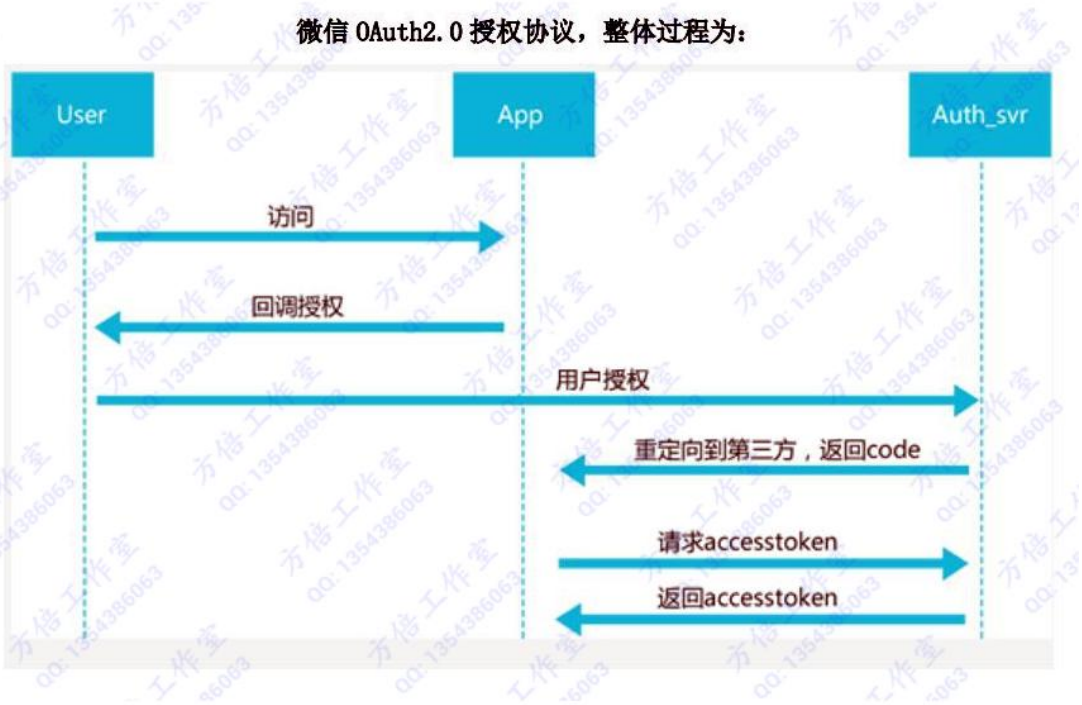
微信登录的授权作用域 scope 分为 snsapi_base、snsapi_userinfo、snsapi_login。

snsapi_base：静默授权，用户无感知，仅能获取用户的 openID 使用于微信公众号，服务号认证后可获得。

snsapi_userinfo：需要用户手动授权，表现形式为弹出框让用户授权，使用于微信公众号，若用户已关注该公众号是没有弹出框提示的，服务号认证后可获得。

snsapi_login：需用户扫描二维码授权，能获取用户的基本信息，使用于网站应用，认证后可获得。

5. **整体授权流程简介**



a) 引导用户进入授权页面同意授权，获取 code
b) 通过 code 换取网页授权 access_token（与基础支持中的 access_token 不同）
c) 如果需要，开发者可以刷新网页授权 access_token，避免过期
d) 通过网页授权 access_token 和 openid 获取用户基本信息（支持 UnionID 机制）

四、 微信网页授权获取用户基本信息与项目的集成

微信授权只是相当于把 cas 用户输入用户名密码的验证方式替换为微信用户授权的方式，创建 tgt，其他地方无需改动。

1. 在进行开发之前，需在微信公众平台或开放平台官网配置授权回调域名，配置之后该域名下的所有页面都可以进行授权登录。这里填写的是域名（是一个字符串），而不是 URL，因此请勿加 http://等协议开头。当授权成功回调的时候

会先去检查回调地址是否在该域名下，若不在该域下将会跳转不成功。

2. 配置租户的 appID 和 appSecret(包括网站应用和微信公众号)。当用户登录的时候会去验证是微信客户端还是 PC 端，从而选择不同的授权方式。若未配置这两个参数在页面上是不显示微信登录按钮的。

3. 用户点击使用微信登录按钮，跳转到微信用户授权的界面（二维码或弹出框）。跳转到二维码链接为：

https://open.weixin.qq.com/connect/qrconnect?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=snsapi_login&state=STATE#wechat_redirect

跳转到弹出框的链接为：

https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect

参数说明

参数	是否必须	说明
appid	是	公众号的唯一标识
redirect_uri	是	授权后重定向的回调链接地址，请使用urlencode对链接进行处理
response_type	是	返回类型，请填写code
scope	是	应用授权作用域，snsapi_base（不弹出授权页面，直接跳转，只能获取用户openid），snsapi_userinfo（弹出授权页面，可通过openid拿到昵称、性别、所在地。并且，**即使在未关注的情况下，只要用户授权，也能获取其信息**）
state	否	重定向后会带上state参数，开发者可以填写a-zA-Z0-9的参数值，最多128字节
#wechat_redirect	是	无论直接打开还是做页面302重定向时候，必须带此参数

这时会返回弹出框（微信客户端）或二维码（PC）

4. 当用户在微信客户端确认授权成功之后（用户操作截止，其后都是系统和微信服务器的交互），微信服务器会回调访问在上一步设置的回调地址，除了带有

上一步设置的 STATE 参数之外，还会带有 code 参数。code 作为换取 access_token 的票据，每次用户授权带上的 code 将不一样，code 只能使用一次，5 分钟未被使用自动过期。

5. 根据 code 获取 access_token

公众号可通过下述接口来获取网页授权 access_token。如果网页授权的作用域为 snsapi_base，则本步骤中获取到网页授权 access_token 的同时，也获取到了 openid，snsapi_base 式的网页授权流程即到此为止。access_token 有效期为两小时。

请求链接如下：

https://api.weixin.qq.com/sns/oauth2/access_token?appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code

参数说明：

参数	是否必须	说明
appid	是	公众号的唯一标识
secret	是	公众号的appsecret
code	是	填写第一步获取的code参数
grant_type	是	填写为authorization_code

返回值说明：

正确时返回的JSON数据包如下：

```
{
  "access_token":"ACCESS_TOKEN",
  "expires_in":7200,
  "refresh_token":"REFRESH_TOKEN",
  "openid":"OPENID",
  "scope":"SCOPE",
  "unionid":"o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
expires_in	access_token接口调用凭证超时时间，单位（秒）
refresh_token	用户刷新access_token
openid	用户唯一标识，请注意，在未关注公众号时，用户访问公众号的网页，也会产生一个用户和公众号唯一的OpenID
scope	用户授权的作用域，使用逗号（,）分隔
unionid	当且仅当该公众号已获取用户的userinfo授权，并且该公众号已经绑定到微信开放平台帐号时，才会出现该字段

获取到 unionID 后,根据此 unionID 查询数据库

- 若该用户已经登录，获取该用户信息，创建令牌，登录成功，结束；
- 若不存在，继续向下执行

6. 拉取用户信息

如果网页授权作用域为 snsapi_userinfo 或 snsapi_login，则此时开发者可以通过 access_token 和 openID 拉取用户信息。

请求方法如下：

https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN

参数说明：

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
openid	用户的唯一标识
lang	返回国家地区语言版本，zh_CN 简体，zh_TW 繁体，en 英语

返回说明：


```
{
  "openid":" OPENID",
  " nickname": NICKNAME,
  "sex":"1",
  "province":"PROVINCE"
  "city":"CITY",
  "country":"COUNTRY",
   "headimgurl":
"http://wx.qlogo.cn/mmopen/g3MonUZtNHkdmzicIlibx6iaFqAc56vxLSUfpb6n5WKSYVY0ChQKkiaJSgQ1dZuTOgvLLrhJbERQQ4eMsv84eavHiaiceqxibJxCfHe/46",
      "privilege":[
      "PRIVILEGE1"
      "PRIVILEGE2"
   ],
   "unionid": "o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```


参数	描述
openid	用户的唯一标识
nickname	用户昵称
sex	用户的性别，值为1时是男性，值为2时是女性，值为0时是未知
province	用户个人资料填写的省份
city	普通用户个人资料填写的城市
country	国家，如中国为CN
headimgurl	用户头像，最后一个数值代表正方形头像大小（有0、46、64、96、132数值可选，0代表640*640正方形头像），用户没有头像时该项为空。若用户更换头像，原有头像URL将失效。
privilege	用户特权信息，json 数组，如微信沃卡用户为（chinaunicom）
unionid	只有在用户将公众号绑定到微信开放平台帐号后，才会出现该字段。详见：

获取用户的所在地、昵称、账号等信息保存至数据库，创建登录令牌，登录成功。

引用

http://projects.spring.io/spring-security spring security
https://mp.weixin.qq.com 微信公众平台
https://open.weixin.qq.com/ 微信开放平台
http://www.coin163.com/java/cas/cas.html cas
http://www.cnblogs.com/txw1958/p/weixin71-oauth20.html 微信授权