# 认证服务

1. OAuth2.0
   1. 它是一个开放标准协议，用户可以跳过繁琐的注册流程，通过第三方社交软件进行访问登录此网站，并不会将QQ的账户和密码给此网站
   2. 流程是这样的
      1. 1. 用户点击微信图标发送一个认证请求
      2. 2. 用户本人就会输入那个账户或者密码（绕过那个网站），向微信服务器进行认证
      3. 3. 认证成功就返回一个令牌给自己的app
      4. 4. 可以拿那个令牌去获取微信头像，名字，性别等信息
      5. 5. 再去微信的服务器去认证一下令牌是否正确，正确就将这些信息返回
      6. 6. 与系统账户进行绑定
   3. 需要去设置授权回调页面与失败回调页
2. session原理
   1. 1. 浏览器第一次访问服务器的时候，服务器会创建一个session来保存这个用户信息，存入sessionManager这个保存（map形式）
   2. 2. 服务器会命令浏览器保存一个JsessionId=xxx的cookie
   3. 3. 之后浏览器再访问都会带上这个cookie
   4. 4. 浏览器关闭了，清除这个会话cookie
   5. 5. 下次再访问，没有这个JsessionId，就会重新进入步骤1了
3. 分布式下session的问题
   1. session不同步（集群环境）
      1. 集群环境下，有多个这样的服务是不，当第一次负载均衡的时候给服务器里面的session放了数据，第二次负载均衡到别的地方就没有这个数据了
      2. 解决方案
         1. 就其实跟分布式锁很像，当用本地锁的时候也是每个服务一把锁嘛，这个也是一样，每个服务都会有自己的session，那么我们也可以把他取出来，放redis等中间件就可以了
         2. 可以使用Spring Session的方式，统一存储到这个redis中
   2. 子域session不共享（不同微服务）
      1. 比如访问到A服务，A服务会给浏览器传一个JsessionId，他会有一个domain可以浏览器看，这是他的作用域，现在应该是[A.lcy.com](http://a.lcy.com/)
      2. 然后我跳到B服务，现在呢他的作用域其实是不一样的，业务B服务的是[B.lcy.com](http://b.lcy.com/)，所以当跳到B服务的时候去观察那个cookie，会发生根本没有JsessionId
      3. 解决方案
         1. 我们在A服务页面把他的这个作用域变成[lcy.com](http://lcy.com/)父域，再次跳转，就可以获取到了，所以我们可以在将session设到redis的时候，将它作用域也改大一点
   3. 首先来了解父域和子域：[lcy.com](http://lcy.com/)就是父域，[order.lcy.com](http://order.lcy.com/)为子域
4. Spring Session的使用流程
   1. 首先肯定是导入起步依赖不
   2. 加一个注解@EnableRedisHttpSession
   3. 需要配置一下：指定存储session的中间件类型为redis或者其他中间件
   4. 基本原理大概就是：它肯定会有一个redis操作session的组件，然后有一个拦截器之类的对请求进行一次包装
   5. 关于子域共享的问题：可以自定义一个配置类，去修改作用域（CookieSerializer）