一、oauth2与单点登陆的区别

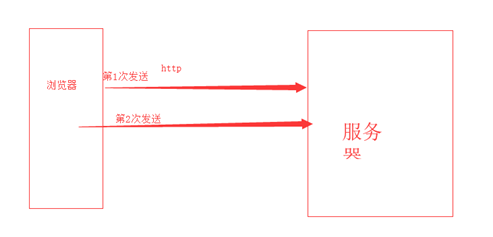
1、oauth2，不同的企业之间的登陆，应用之间的信任度较低

2、单点登陆，是同一企业的产品系列间的登陆，相互信任度较高

二、session-cookie机制

1、session-cookie机制出现的根源， http连接是无状态的连接

-------- 同一浏览器向服务端发送多次请求，服务器无法识别，哪些请求是同一个浏览器发出的



2、为了标识哪些请求是属于同一个人 ---------- 需要在请求里加一个标识参数

方法1-----------直接在url里加一个标识参数(对前端开发有侵入性)，如: token

方法2-----------http请求时，自动携带浏览器的cookie（对前端开发无知觉），如：jsessionid=dfdfdfdfdf

3、浏览器标识在网络上的传输，是明文的，不安全的

-----------安全措施：改https来保障

4、cookie的使用限制---依赖域名

-------------- 顶级域名下cookie，会被二级以下的域名请求，自动携带

-------------- 二级域名的cookie，不能携带被其它域名下的请求携带

5、在服务器后台，通过解读标识信息（token或jsessionid），来对应会话是哪个session

--------------- 一个tomcat，被1000个用户登陆，tomcat里一定有1000个session -------》存储格式map《sessionid，session对象》

--------------- 通过前端传递的jsessionid，来对应取的session ------ 动作发生时机request.getsession

三、session共享方式，实现的单点登陆

1、多个应用共用同一个顶级域名，sessionid被种在顶级域名的cookie里

2、后台session通过redis实现共享，即每个tomcat都在请求开始时，到redis查询session;在请求返回时，将自身session对象存入redis

3、当请求到达服务器时，服务器直接解读cookie中的sessionid，然后通过sessionid到redis中查找到对应会话session对象

4、后台判断请求是否已登陆，主要校验session对象中，是否存在登陆用户信息

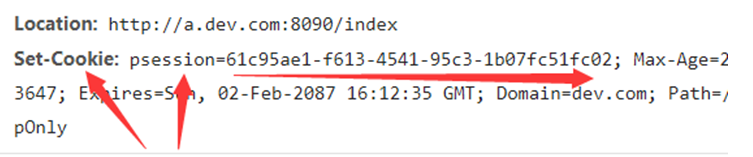
5、整个校验过程，通过filter过滤器来拦截切入，如下图：



6、登陆成功时，后台需要给页面种cookie方法如下：



response里，反映的种cookie效果如下：



7、为了request.getsession时，自动能拿到redis中共享的session，

我们需要重写request的getsession方法（使用HttpServletRequestWrapper包装原request）

四、cas单点登陆方案

1、对于完全不同域名的系统，cookie是无法跨域名共享的

2、cas方案，直接启用一个专业的用来登陆的域名（比如：cas.com）来供所有的系统登陆。

3、当业务系统（如b.com）被打开时，借助cast系统来登陆，过程如下：

cas登陆的全过程：

（1）、b.com打开时，发现自己未登陆 ----》 于是跳转到cas.com去登陆

（2）、cas.com登陆页面被打开，用户输入帐户/密码登陆成功

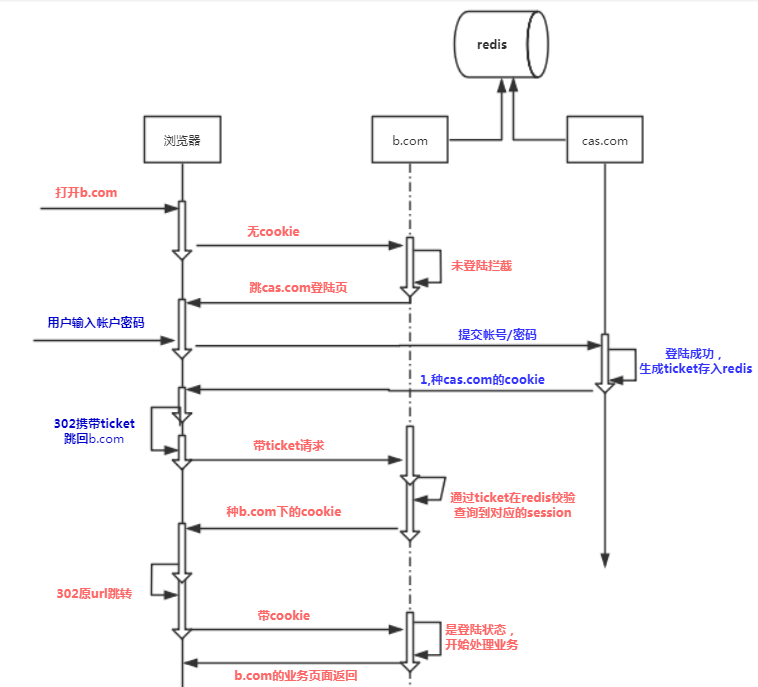
（3）、cas.com登陆成功，种cookie到cas.com域名下 -----------》把sessionid放入后台redis《ticket，sesssionid》---页面跳回b.com

（4）、b.com重新被打开，发现仍然是未登陆，但是有了一个ticket值

（5）、b.com用ticket值，到redis里查到sessionid，并做session同步 ------ 》种cookie给自己，页面原地重跳

（6）、b.com打开自己页面，此时有了cookie，后台校验登陆状态，成功

（7）整个过程交互，列图如下：



4、cas.com的登陆页面被打开时,如果此时cas.com本来就是登陆状态的,则自动返回生成ticket给业务系统

整个单点登陆的关键部位，是利用cas.com的cookie保持cas.com是登陆状态,此后任何第三个系统跳入,都将自动完成登陆过程

5,本示例中,使用了redis来做cas的服务接口,请根据工作情况,自行替换为合适的服务接口(主要是根据sessionid来判断用户是否已登陆)

6,为提高安全性,ticket应该使用过即作废(本例中,会用有效期机制)