Session保存登录状态

一、前言

1.B/S架构间通信使用的是无状态的HTTP协议，但是现在B/S的交互大多需要保存状态，这在服务端引入了session机制，在客户端引入了cookie机制

1. 纯cookie:在cookie保存用户名和密码，进行一定的加密
2. cookie+session：
   1. 在cookie保存sessionID、用户名和密码，进行一定的加密，session端验证ip等
   2. 在cookie保存sessionID、用户名，把登录状态放在session管理，session设置超时，且session在少使用是容器会将其钝化（序列化到硬盘中），不占用内存空间。
3. Token

二、LiveInYouth将采用服务器持久session的方案

1.此方案大部分工作在服务端完成

后端代码：

/\*

\* 自动登录设置模块

\*/

**if** ((**boolean**) loginInfo[1]) {

User existUser = **new** LoginDao().getUserByName(user.getUserName());//完整的用户信息

//状态在B/S双向保存，状态只包括 JSESSIONID 和 userName

**if** (existUser != **null** && user.isAutoLogin()) {

// session持久化

userNameSet.add(existUser.getUserName());//存放不重复用户名的Set，作为session的属性

session.setAttribute("user", userNameSet);

session.setMaxInactiveInterval(30 \* 24 \* 60 \* 60);//设置服务端session的会话时间

// cookie持久化

Cookie nameCookie = **new** Cookie("userName", existUser.getUserName());

Cookie sessIdCookie = **new** Cookie("JSESSIONID", session.getId());//重写主要是为了覆盖原来的值，然后才能使用setMaxAge来设定cookie端的存活时间

nameCookie.setMaxAge(30 \* 24 \* 60 \* 60);

sessIdCookie.setMaxAge(30 \* 24 \* 60 \* 60);//设置浏览器端的cookie的存活时间（以匹配服务端的会话）

// cookie输出到响应头

nameCookie.setPath("/LiveInYouth");

sessIdCookie.setPath("/LiveInYouth");//表示作用的范围

response.addCookie(nameCookie);

response.addCookie(sessIdCookie);

}

}

2.简单来说就是：

1）重写发给cookie的JSESSIONID，以便设置过期时间

2）写入含有userName的cookie，以区分账号，同时设置过期时间

3）设置后端session的过期时间（不使用时容器会将其持久化，直到过期）

4）后端在调用request.getSession的时候，会通过request报文header的Cookie字段的JSESSIONID，查找对应的session；

找到session的时候，session再通过user找对应user的状态，如果状态是自动登录中，那么给相关信息

3.那么每次请求用户相关信息的时候，后台都要经过这道判断，所以这个if放在最前面，所以又给了login-Controller新的任务，调用渲染个人页面所需要的各种数据。

4. 密码为空时，查看session时候存在，存在的时候查看session的userMap属性，看在map中能否找到对应的username，找到就返回登录成功的json

一个session服务一个用户多个账户，账户使用map存起来，作为属性放在session中