# 结合redis完成用户登录功能

# 登录

## 1.1 登录逻辑

### 1.1.1功能需求演示

图形用户界面, 应用程序

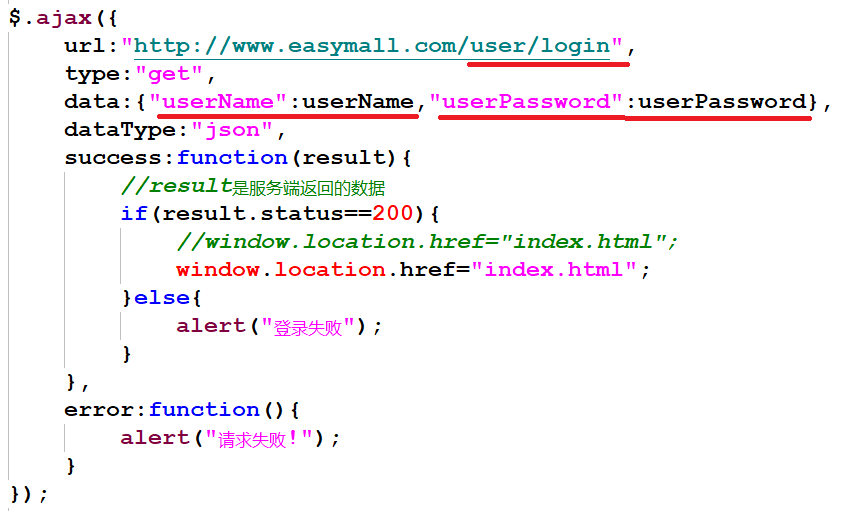
描述已自动生成

登录成功转向首页：

图形用户界面, 文本

中度可信度描述已自动生成

### 1.1.2 js代码展示（login\_ajax.js）

9

### 1.1.3 js代码解释

点击登录发起ajax请求,将**用户名**传递给后台处理,根据**响应结果**判断处理,如果**成功**跳转向首页.

1. **Nginx配置**

图形用户界面, 文本, 应用程序

描述已自动生成

1. **Js请求地址：**

[http://www.easymall.com](http://www.easymall.com/user/login)**[/user](http://www.easymall.com/user/login)**[/login](http://www.easymall.com/user/login)

1. **Nginx转发**

[http://127.0.0.1:9005/zuul-user/](http://127.0.0.1:9005/zuul-user/user/manage/login%20)**[user/manage/login](http://127.0.0.1:9005/zuul-user/user/manage/login%20)**

1. **网关路由配置：**

zuul.routes.zuul-user.path=/zuul-user/\*\*

zuul.routes.zuul-user.serviceId=userservice

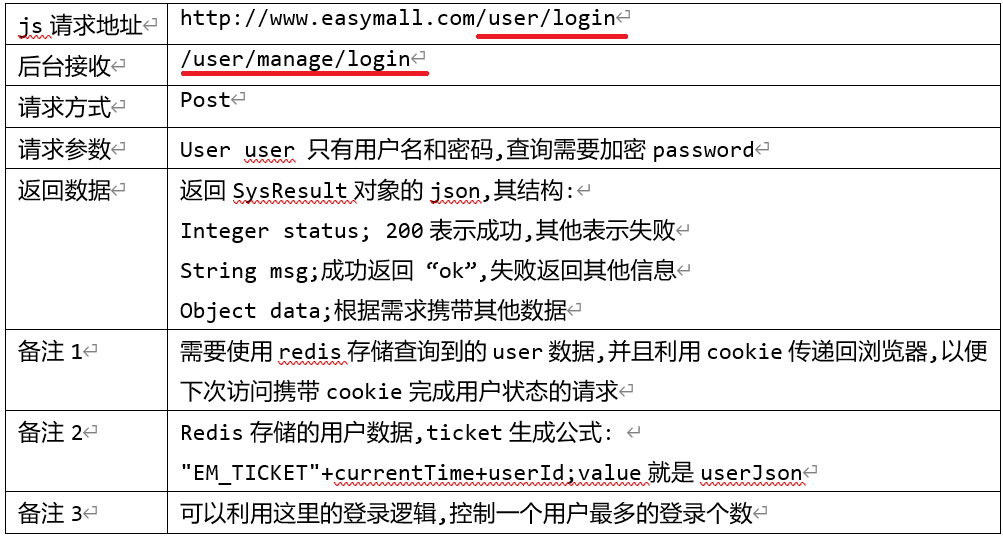
1. **路由转发：**

**http://userservice**/**user/manage/login**

1. **ribbon抓取**

[http://127.0.0.1:10003/](http://127.0.0.1:10003/user/manage/login)**[user/manage/login](http://127.0.0.1:10003/user/manage/login)**

### 1.1.4 接口文件：



## 1.2 结合redis完成登录功能（第一版）

### 1.2.1 在用户中心的pom文件中添加redis的依赖

<!-- 添加支持redis的简化依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-redis</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

### 1.2.2 代码实现

#### 1.控制器UserController

文本

描述已自动生成

#### 2.Service层（UserService）

图形用户界面, 文本, 应用程序

描述已自动生成

#### 3.持久层UserMapper

徽标

描述已自动生成

#### 4.映射文件UserMapper.xml

文本

描述已自动生成

#### 5.直接测试

启动user项目，由于没有启动eureka，所以会报错，不用理它，直接在浏览输入:

<http://localhost:10003/user/manage/login?userName=aa&userPassword=123>

由于数据库存在aa，密码123。所以登录成功，进入F12，可以查看到EM\_TICKET已经存入到cookie中。

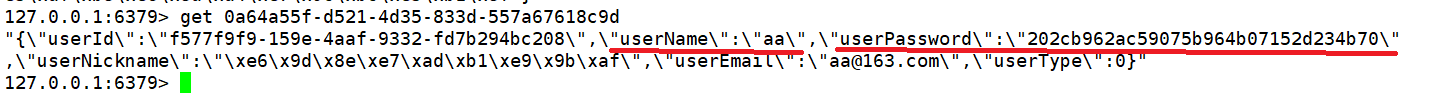
图形用户界面, 应用程序, Word

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

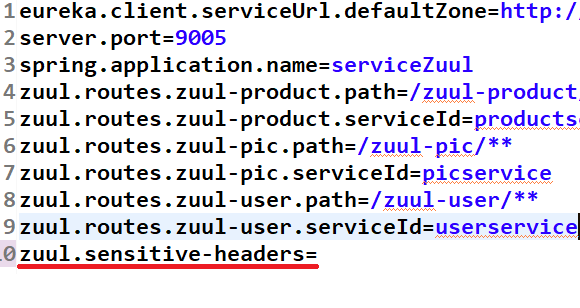
复制cookie的值，到redis客户端查找，可以查找到对应的user对象。



## 1.3 zuul的cookie拦截

启动eureka、zuul网关、product和ngin后，对登录进行测试，发现第一个登录功能在登录成功时，主页上没有显示登录账号，F12查看，发现没有存cookie，浏览器并不能接收cookie，原因是Zuul网关组件.默认对Cookie的头敏感，将cookie过滤掉了，我们只需要在Zuul网关的application.properties中添加敏感头为空的配置：

zuul.sensitive-headers=



登录成功后返回首页，F12，点击index.html，查看cookie不存在，进入登录界面，输入用户名密码，登录，返回主页，再次查看index.html，可以查看到cookie的值，说明登录成功后返回cookie。

表格

描述已自动生成

访问全部商品、商品详情、注册等页面，cookie一直都携带着。

## 1.4登录功能的第二个接口方法

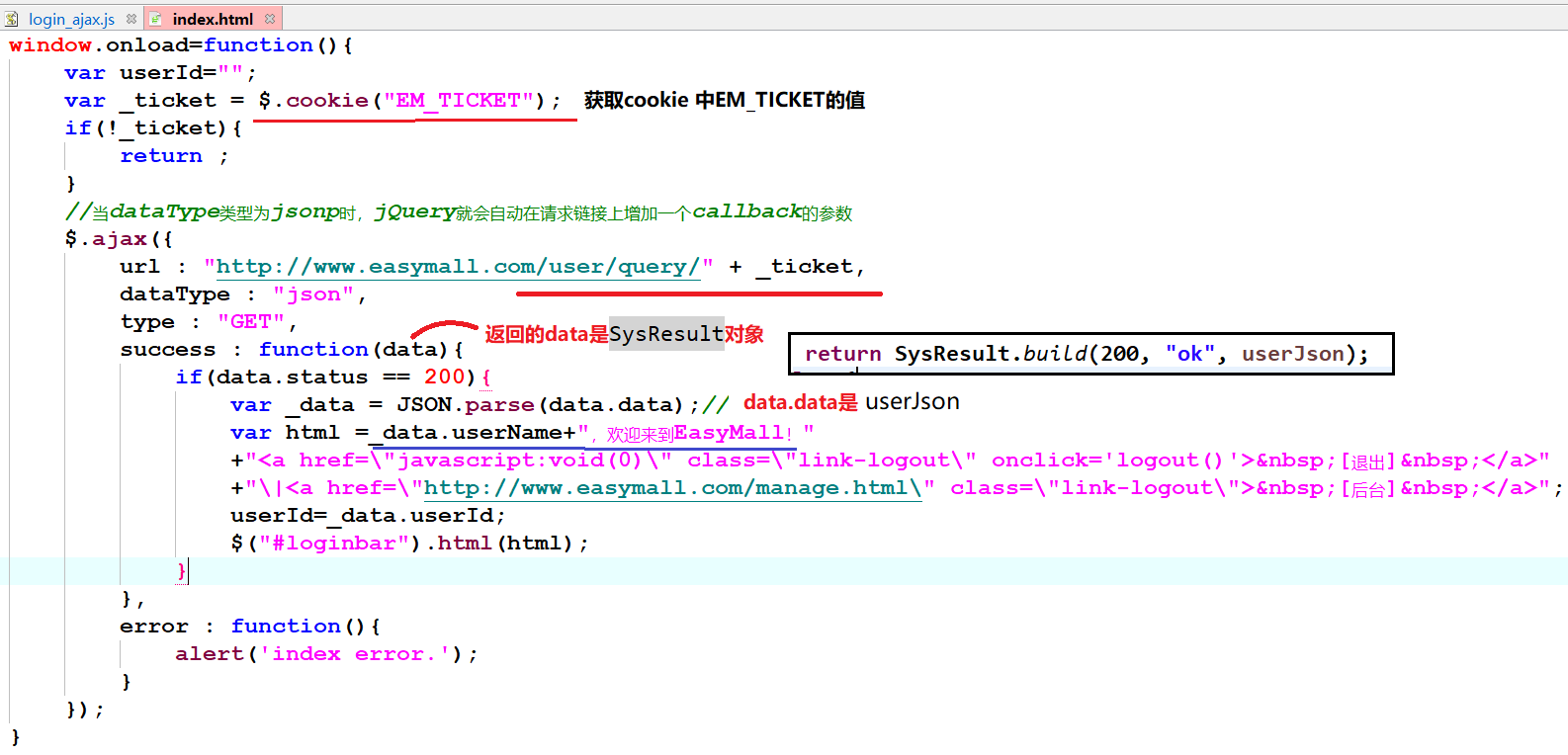
图示

描述已自动生成

### 4.1 Js发起请求

查看index.html的window.onload事件

通过js发起请求,到user用户系统访问redis，通过cookie的EM\_TICKET的值作为key到redis获取userJson数据。



图形用户界面, 文本, 应用程序

描述已自动生成

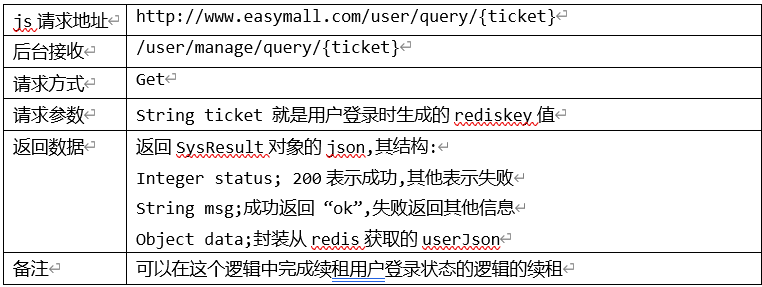
请求地址：

[http://www.easymall.com/use](http://www.easymall.com/user/query/%7bticket%7d)[r/query/{ticket}](http://www.easymall.com/user/query/%7bticket%7d)

Nginx转发

http://127.0.0.1:9005/zuul-user/**user/manage/query/{ticket}**

### 接口文件



### 代码实现

#### 1.UserController

图形用户界面, 文本

描述已自动生成

#### 2.UserService

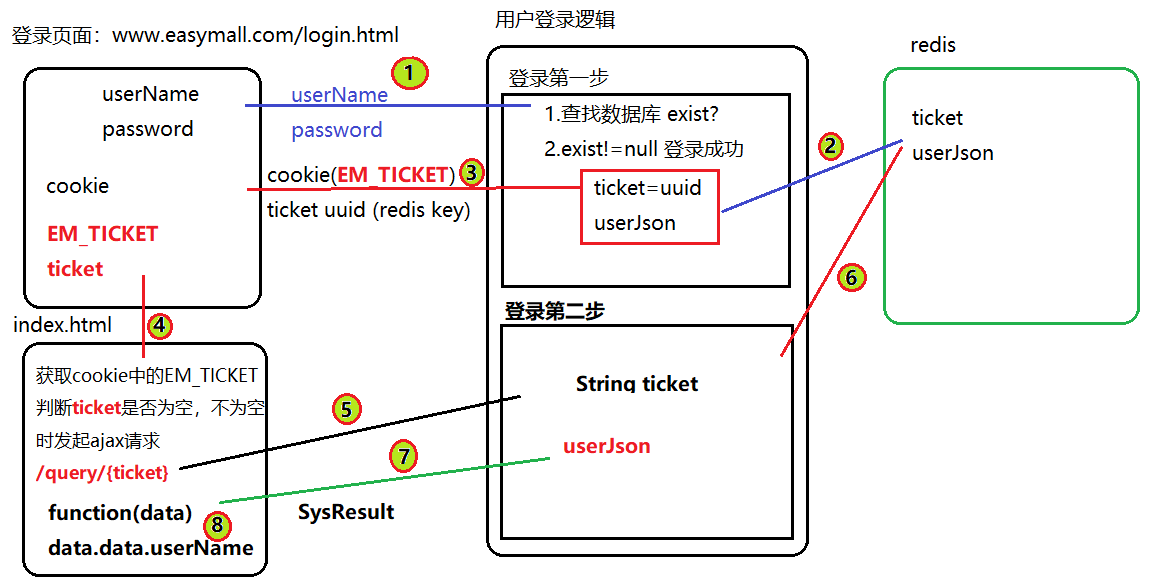
文本

描述已自动生成

测试：



## 1.5登录过程配合携带状态访问整体流程



代码存在的问题：

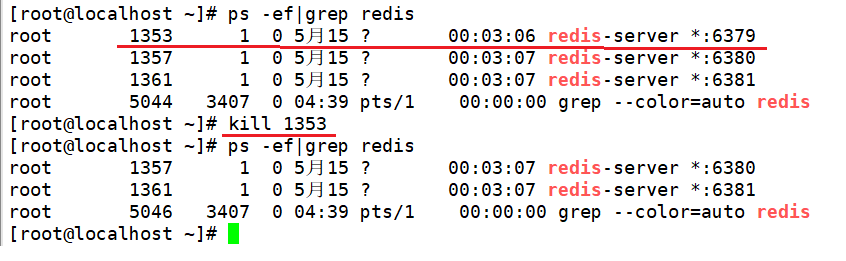
1. 代码中使用了new,登录并发量多，导致内存溢出。



1. 访问的redis是单机节点，节点down了，任何人都无法登录

退出redis-server，查看进程，杀掉6379节点所在进程（**因为我们前面设置为后端运行，所以退出还不行，要杀掉进程才真正停止redis-server**）：进程杀掉后再次刷新主要，此时无法正常登录。

所以单节点如果节点down了，任何人无法登录。



携带ticket去访问redis，失败，报500错。



1. 单节点，容易达到内存容量上限