**淘淘商城**

**第八天**

# 课程计划

1. 单点登录接口实现
2. 实现跳转注册登录页面
3. 跨域问题和jsonp

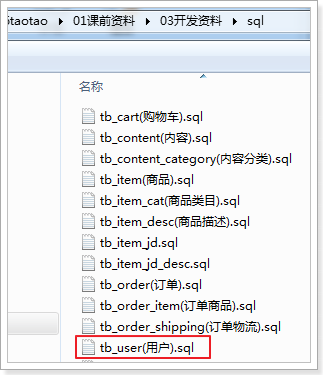
4. 实现用户注册功能

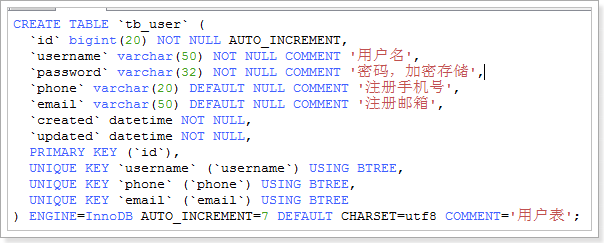
5. 实现用户登录功能

# 单点登录接口实现

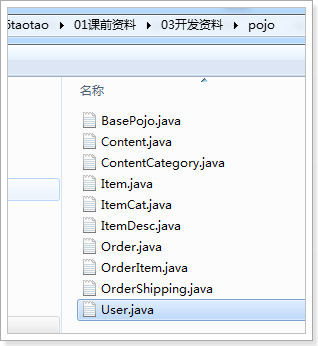
## 数据库表分析

导入数据库表

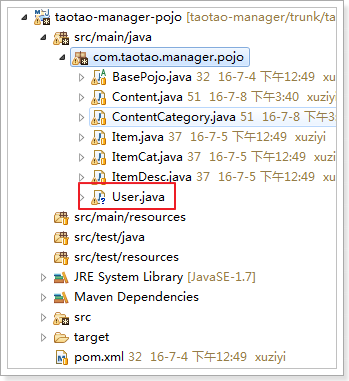




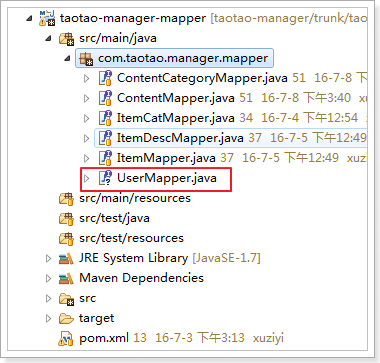
### 创建pojo



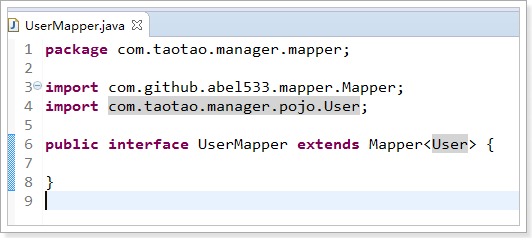
加入pojo到taotao-manager-pojo中



### 创建Mapper

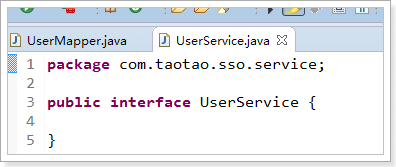


在taotao-manager-pojo创建UserMapper



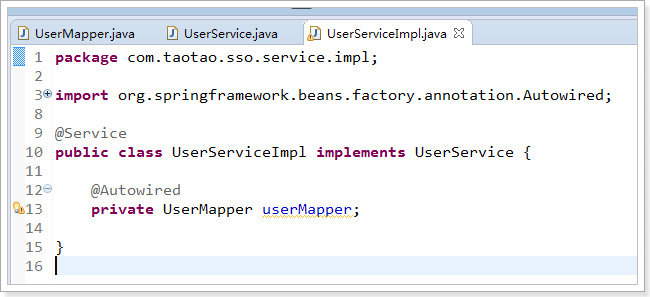
### 创建Service接口

在taotao-sso-interface创建接口



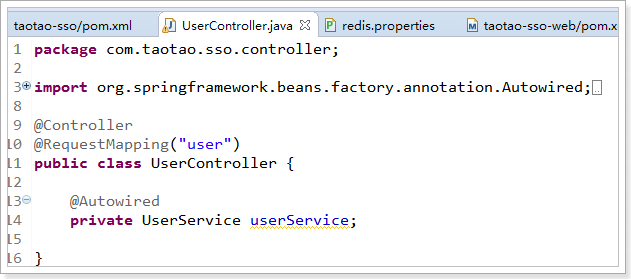
### 创建Service接口实现类

在taotao-sso-service创建实现类



### 创建Controller

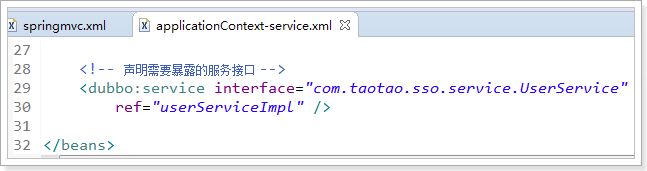
在taotao-sso-web编写



### 发布服务服务

#### 声明服务的发布

在taotao-sso-service发布服务



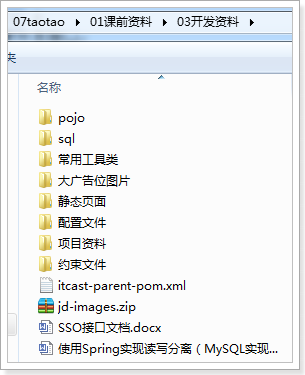
#### 声明服务的调用

在taotao-sso-web调用服务

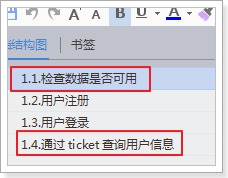


需要根据接口文档开发接口

SSO接口文档的位置：



实现两个接口



## 实现检查数据是否可用

### 开发分析

接口文档

|  |  |
| --- | --- |
| 请求方法 | GET |
| URL | http://sso.taotao.com/user/check/{param}/{type} |
| 参数说明 | 格式如：zhangsan/ 1，其中zhangsan是校验的数据，type为类型，可选参数1、2、3分别代表username、phone、email |
| 示例 | http://sso.taotao.com/user/check/zhangsan/1 |
| 返回值 | true：数据可用，false：数据不可用 |

接口是校验username、phone、email三个字段是否可用

用户要输入的这三个字段必须是没有被别人使用过的

我们可以把这个字段作为条件进行查询，如果查询到了，则表示已经有人用了，那么新用户就不能使用，数据不可用，返回false

如果没有查询到，则表示没有被别人用，新用户可以使用，数据可用，true

### 实现Controller

在taotao-sso-web实现

@Controller

@RequestMapping("user")

**public** **class** UserController {

@Autowired

**private** UserService userService;

// http://sso.taotao.com/user/check/{param}/{type}

/\*\*

\* 检查数据是否可用

\*

\* **@param** param

\* **@param** type

\* **@return**

\*/

@RequestMapping(value = "check/{param}/{type}", method = RequestMethod.***GET***)

@ResponseBody

**public** ResponseEntity<Boolean> check(@PathVariable("param") String param, @PathVariable("type") Integer type) {

**try** {

Boolean bool = **this**.userService.check(param, type);

// 200

**return** ResponseEntity.*ok*(bool);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

// 500

**return** ResponseEntity.*status*(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(**null**);

}

}

### 实现Service接口

在taotao-sso-interface实现

**public** **interface** UserService {

/\*\*

\* 检查数据是否可用

\*

\* **@param** param

\* **@param** type

\* **@return**

\*/

Boolean check(String param, Integer type);

}

### 实现Service接口实现类

在taotao-sso-service实现

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** Boolean check(String param, Integer type) {

User user = **new** User();

**switch** (type) {

**case** 1:

user.setUsername(param);

**break**;

**case** 2:

user.setPhone(param);

**break**;

**case** 3:

user.setEmail(param);

**break**;

**default**:

**break**;

}

**int** count = **this**.userMapper.selectCount(user);

// 根据条件查询，如果查到值，则表示数据已经使用了，返回false，反之返回true

**if** (count > 0) {

**return** **false**;

} **else** {

**return** **true**;

}

}

## 实现根据ticket查询用户接口

### 开发分析

|  |  |
| --- | --- |
| 请求方法 | GET |
| URL | http://sso.taotao.com/user/{ticket} |
| 参数 | ticket //用户登录凭证 |
| 参数说明 |  |
| 示例 | <http://sso.taotao.com/user/fe5cb546aeb3ce1bf37abcb08a40493e> |
| 返回值 | {"id":1,"username":"xuziyi","phone":"15800807944",  "email":"11223344@qq.com","created":1414119176000,  "updated":1414119179000}"} |

根据ticket查询用户

用户信息保存在redis中，返回的数据是json格式，直接把从数据库中查到的json数据返回

### 实现Controller

在taotao-sso-web实现

// 请求方法 GET

// URL http://sso.taotao.com/user/{ticket}

/\*\*

\* 根据ticket查询用户数据

\*

\* **@param** ticket

\* **@return**

\*/

@RequestMapping(value = "{ticket}", method = RequestMethod.***GET***)

**public** ResponseEntity<User> queryUserByTicket(@PathVariable("ticket") String ticket) {

**try** {

User user = **this**.userService.queryUserByTicket(ticket);

// 判断user是否为空

**if** (user != **null**) {

// 如果不为空 200

**return** ResponseEntity.*ok*().body(user);

} **else** {

// 如果为空 404

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(**null**);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

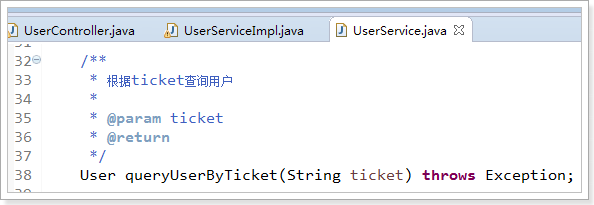
// 500

**return** ResponseEntity.*status*(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(**null**);

}

### 实现Service接口

在taotao-sso-interface实现



### 实现Service接口实现类

在taotao-sso-service实现

@Override

**public** User queryUserByTicket(String ticket) {

// 1. 从redis中查询数据，redis存放的数据都是字符串类型的json格式数据

String json = **this**.redisUtils.get(**this**.TAOTAO\_TICKET\_KEY + ticket);

**try** {

// 2. 判断数据是否为非空，如果不为空，把json转为对象

**if** (StringUtils.*isNotBlank*(json)) {

User user = ***MAPPER***.readValue(json, User.**class**);

// 如果查询了用户信息，表示该用户是活动的，需要重置有效时间

**this**.redisUtils.expire(**this**.TAOTAO\_TICKET\_KEY + ticket, 60 \* 30);

**return** user;

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

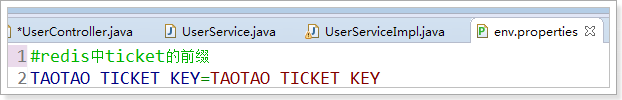
}

// 3.如果为空，或者转换异常，返回null

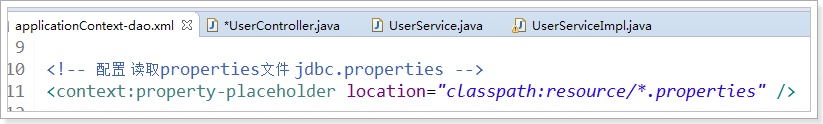
**return** **null**;

}

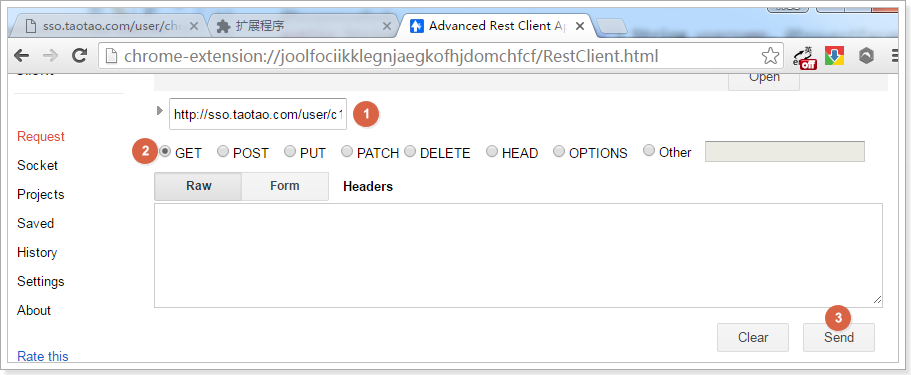
### 加入配置文件



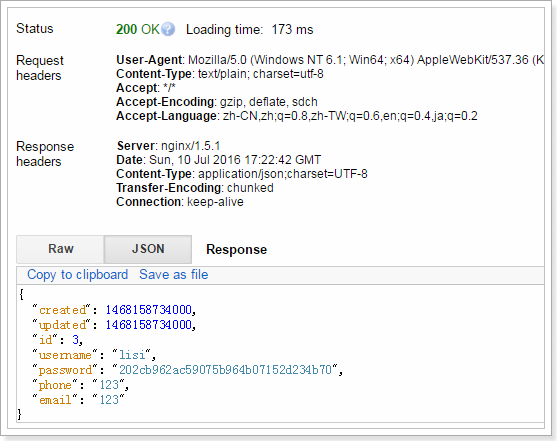
加载配置文件



### 测试



查询成功



# 首页跳转注册、登录页面

### 开发分析

打开商城首页



首页顶部有登录、免费注册按钮，点击按钮可以进入登录、注册页面

登录

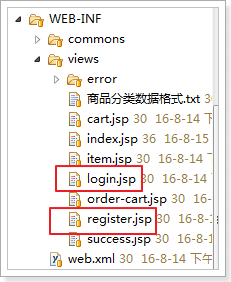


注册



分析发现，登录和注册都是差不多的请求，可以编写通用方法进行跳转

在taotao-portal门户系统的WEB-INF/views里面可以看到有对应的页面



### 实现通用跳转方法

在taotao-portal实现

@Controller

@RequestMapping("page")

**public** **class** PageController {

/\*\*

\* 通用页面跳转方法

\* **@param** pageName

\* **@return**

\*/

@RequestMapping(value = "{pageName}")

**public** String toPage(@PathVariable("pageName") String pageName) {

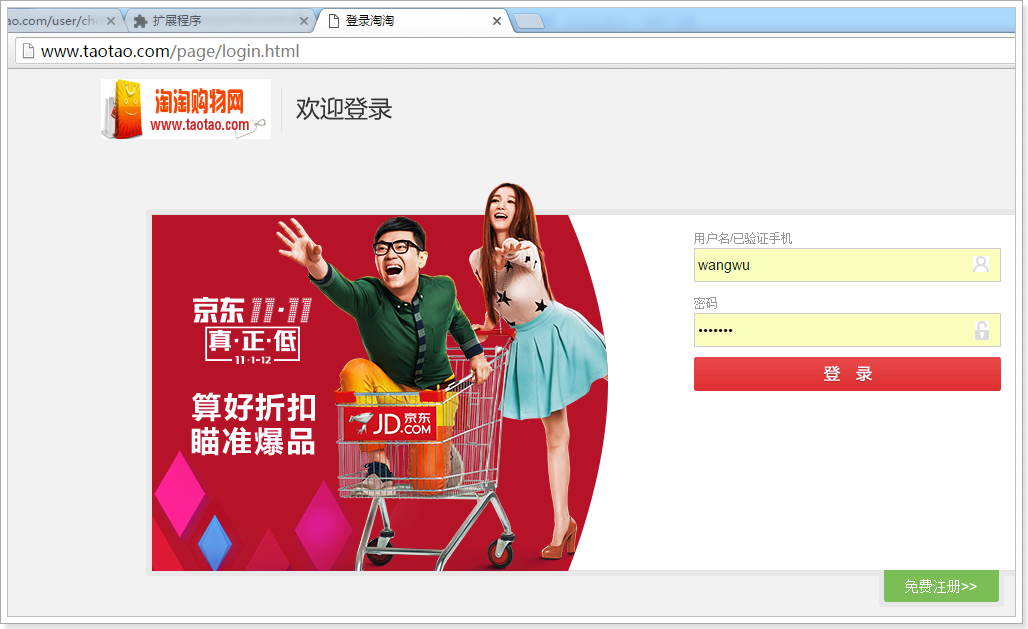
**return** pageName;

}

}

### 效果

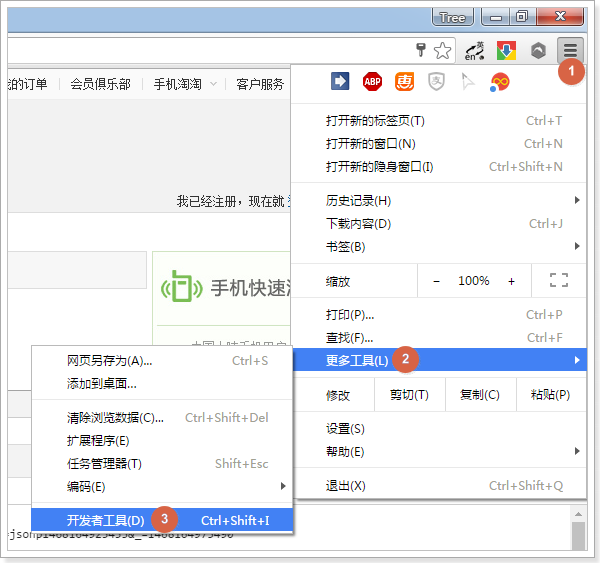




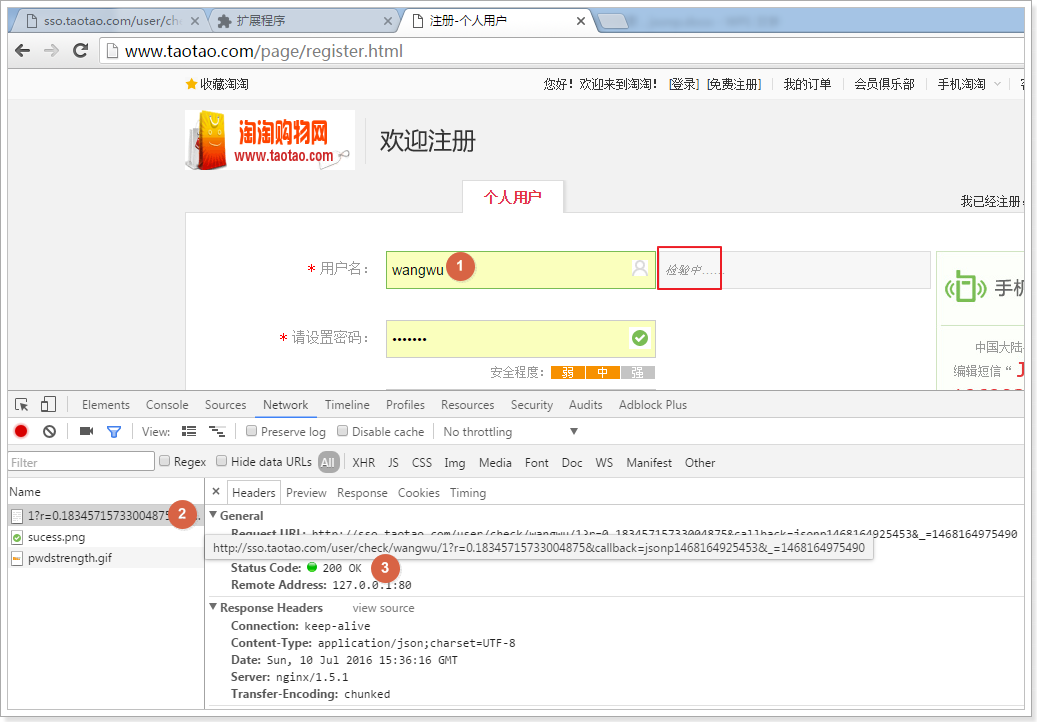
# jsonp

## 用户注册问题

使用谷歌浏览器，打开开发者助手，可以查看请求信息

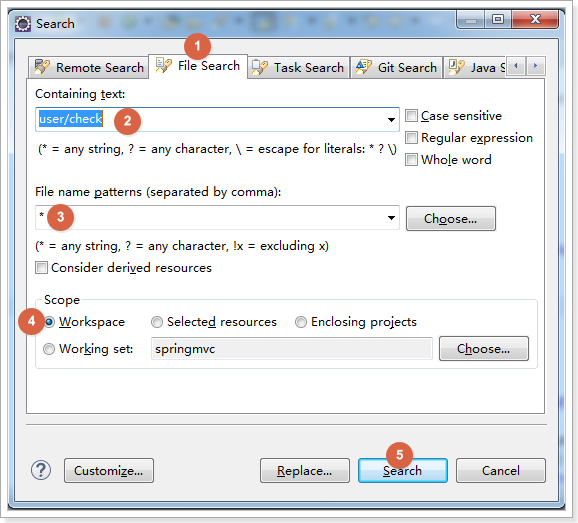


输入用户名，一直是“检测中”，但是返回的状态码是200，表示请求成功

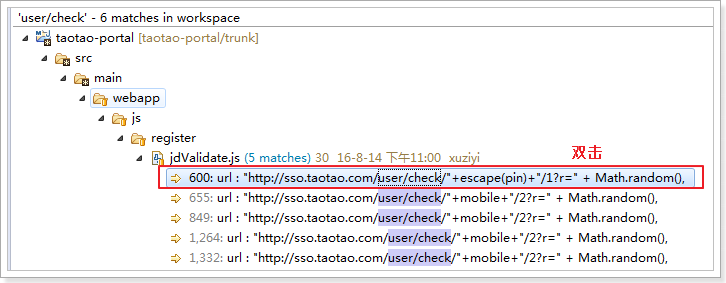


根据响应的url在eclipse查询发起请求的逻辑

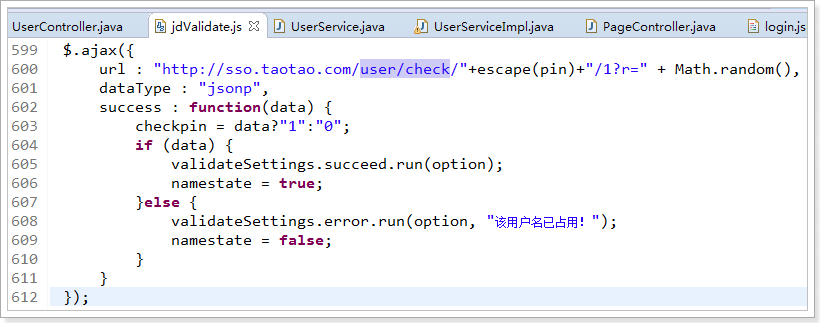
在eclipse中按Ctrl+h调出搜索界面



查询结果：



查看发起请求的逻辑：



分析发现使用的dataType是jsonp，这里使用的是jsonp技术

为什么要使用jsonp?

为了解决js跨域调用的问题

## 跨域问题

什么是跨域问题？

js使用Ajax请求进行跨域请求，无法返回数据（json、xml等）。

什么是跨域？

跨域就是跨域名或跨端口号进行调用

例如:

www.taotao.com >> www.jd.com 是跨域

www.taotao.com >> sso.taotao.com 是跨域

www.taotao.com >> www.taotao.com:8080 是跨域

www.taotao.com >> www.taotao.com 不是跨域

只有请求者和被请求者的域名和端口号完全一致，才不是跨域

为什么会有跨域问题？

浏览器基于安全考虑，不允许Ajax请求跨域调用数据（json、xml等）

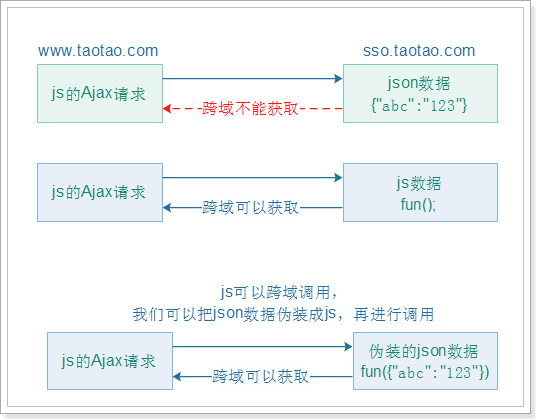
我们现在碰到的问题就是，www.taotao.com请求sso.taotao.com，因为跨域问题，是获取不到数据的

## jsonp原理

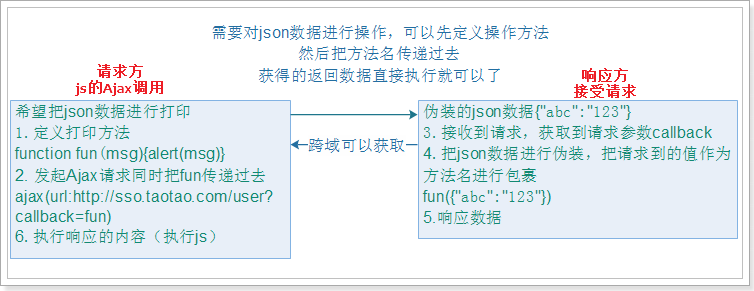
jsonp核心原理：

js使用Ajax无法跨域调用数据（xml，json等），但是可以跨域调用js数据

js的Ajax请求数据和请求js数据的不同如下图：



解决js使用Ajax无法跨域获取数据的方法：



上图，我们可以通过请求js的方式，进行跨域调用

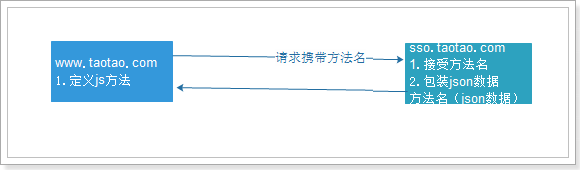
## 单点登录支持跨域

### 分析

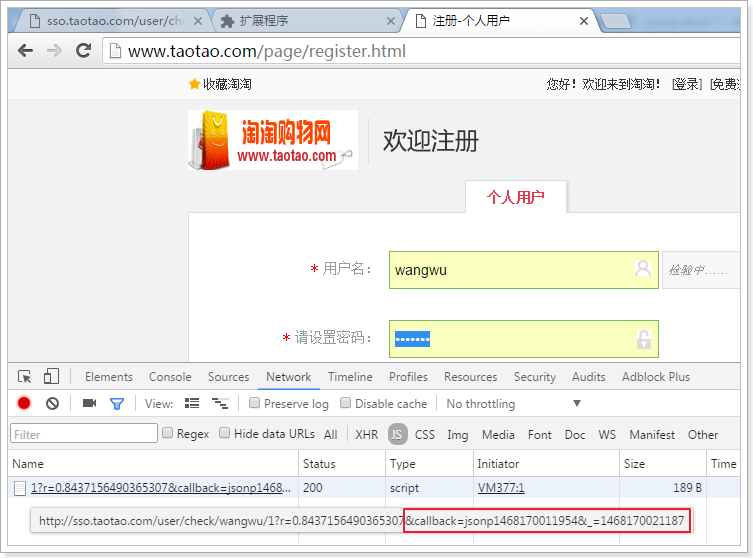
修改单点登录接口，使其支持跨域

其实就是使用request获取方法名，再把返回值进行包裹

我们需要做的事，如下图



分析发现，我们需要的请求参数名其实就是callback



### 改造

项目中校验数据是否可用和根据ticket查询用户信息两个接口会使用jsonp支持

对这两个接口进行改造

/\*\*

\* 校验数据是否可用

\*

\* **@param** param

\* **@param** type

\* **@return**

\*/

@RequestMapping(value = "check/{param}/{type}", method = RequestMethod.***GET***)

@ResponseBody

**public** ResponseEntity<String> check(HttpServletRequest request, @PathVariable("param") String param,

@PathVariable("type") Integer type) {

**try** {

Boolean bool = **this**.userService.check(param, type);

// 1 获取callback参数

String callback = request.getParameter("callback");

// 2. 判断callback是否为空，

**if** (StringUtils.*isNotBlank*(callback)) {

// 如果不为空，表示请示使用的jsonp进行，我们就需要把返回的json数据进行包裹，使用callback

String result = callback + "(" + bool + ")";

**return** ResponseEntity.*ok*(result);

} **else** {

// 如果为空，表示请求不是使用的jsonp，和原来一样，直接返回

**return** ResponseEntity.*ok*(bool + "");

}

// 200

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

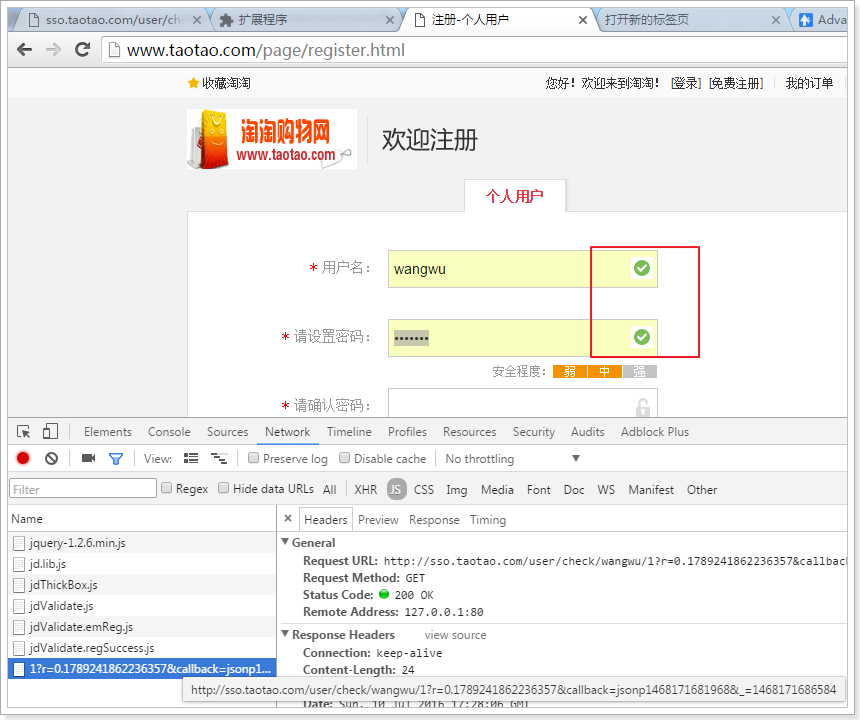
}

// 500

**return** ResponseEntity.*status*(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(**null**);

}

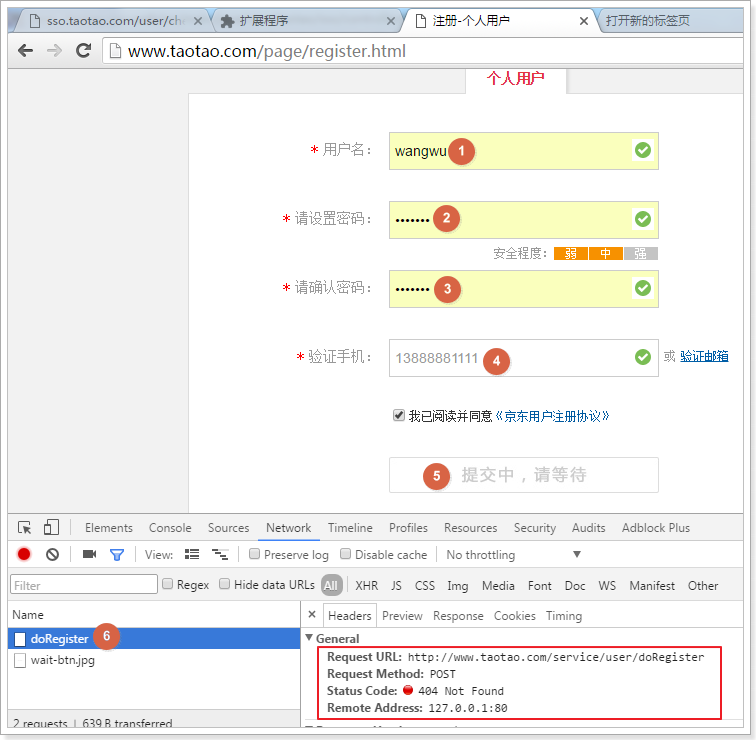
### 注册校验成功



# 实现用户注册

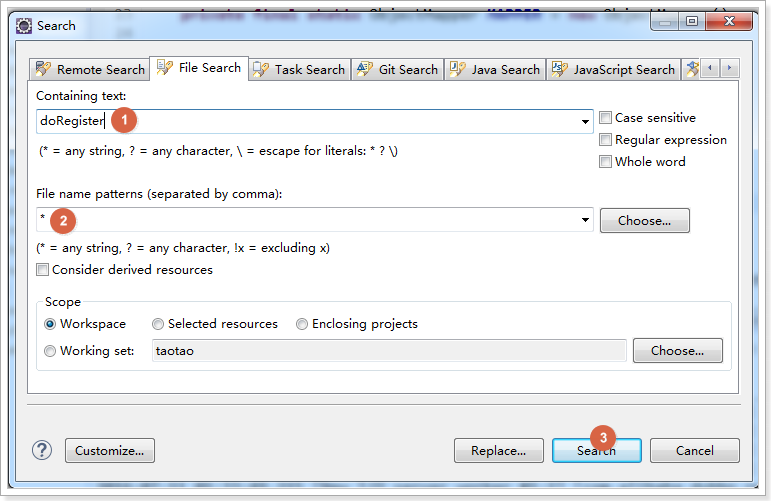
## 开发分析

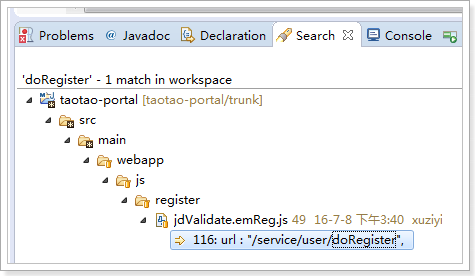
输入用户信息，点击注册发现404错误



后台方法没有编写，404错误正常，需要分析前端，再编写后端

在eclipse按ctrl+h进行搜索，搜索doRegister





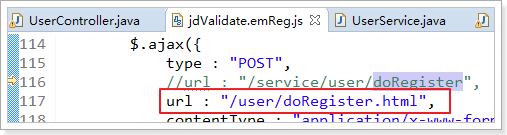
只有一个结果，打开页面分析前端实现



页面有一个注册成功的判断，需要返回值必须有status的属性，而且属性值为200

数据类型是json格式数据，key-value，我们可以使用类似数据结构的map来封装

请求路径不符合我们现在的情况，修改url为我们现在的方式

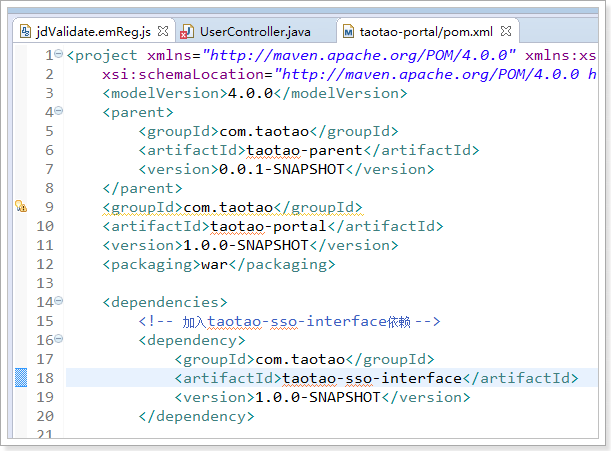


## 添加依赖和声明服务

需要调用单点登录的服务实现用户注册，所以需要加入单点登录的接口依赖，和声明服务调用

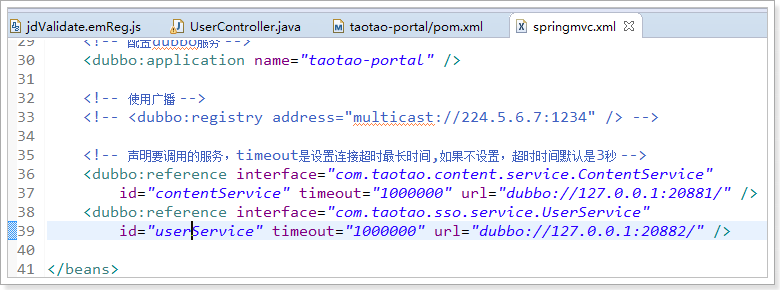
### 添加依赖

在taotao-portal添加taotao-sso-interface依赖



### 声明服务调用

在taotao-portal声明



## controller实现

在taotao-portal实现

@Controller

@RequestMapping("user")

**public** **class** UserController {

@Autowired

**private** UserService userService;

/\*\*

\* 用户注册

\*

\* **@param** user

\* **@return**

\*/

@RequestMapping(value = "doRegister", method = RequestMethod.***POST***)

@ResponseBody

**public** Map<String, Object> doRegister(User user) {

**this**.userService.doRegister(user);

Map<String, Object> map = **new** HashMap<>();

map.put("status", "200");

**return** map;

}

}

## 实现Service接口

在taotao-sso-interface编写

/\*\*

\* 用户注册

\*

\* **@param** user

\*/

**void** doRegister(User user);

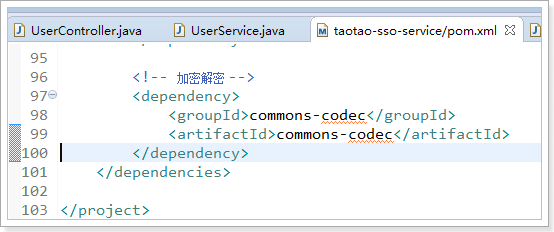
## 实现Service接口实现类

在taotao-sso-service编写

用户密码需要保证安全，我们需要给密码进行加密。

可以使用MD5加密，使用commons-codec进行加密

需要把commons-codec的依赖加入到taotao-sso-service中



编写注册方法

@Override

**public** **void** register(User user) {

user.setCreated(**new** Date());

user.setUpdated(user.getCreated());

// 需要给用户密码进行加密，保证密码安全，我们使用MD5加密

user.setPassword(DigestUtils.*md5Hex*(user.getPassword()));

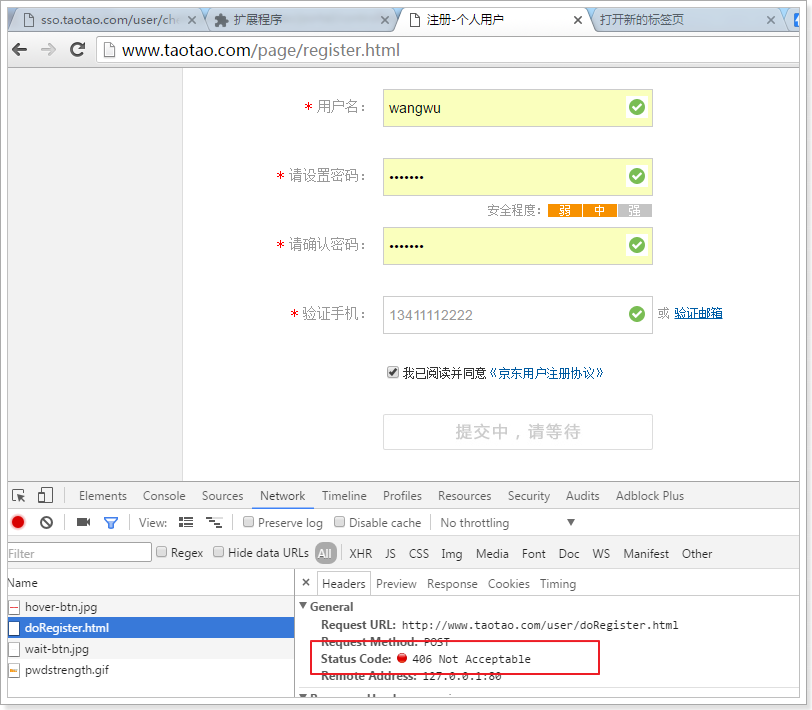
// 保存用户

**this**.userMapper.insert(user);

}

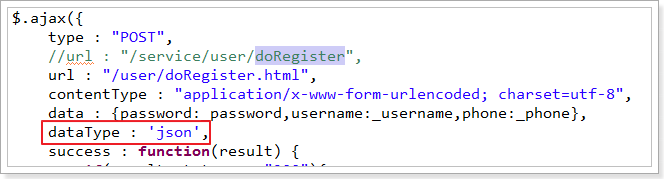
## 406错误

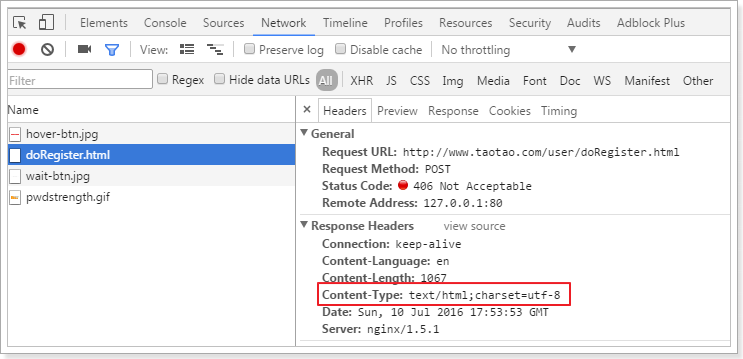
### 出现406错误



### 406的原因

Ajax需要返回的是json格式的响应，但我们返回的是文本响应



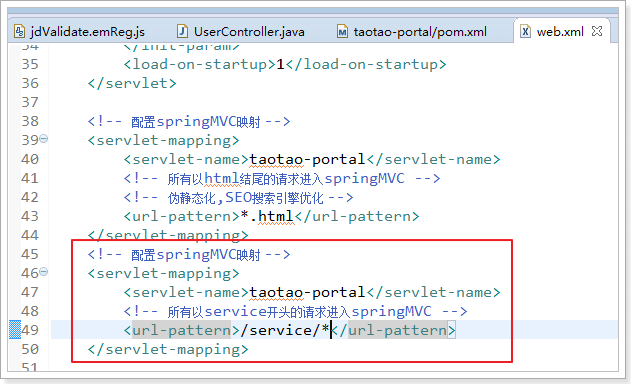


原因：springMVC看到html结尾的请求，就会返回文本的响应

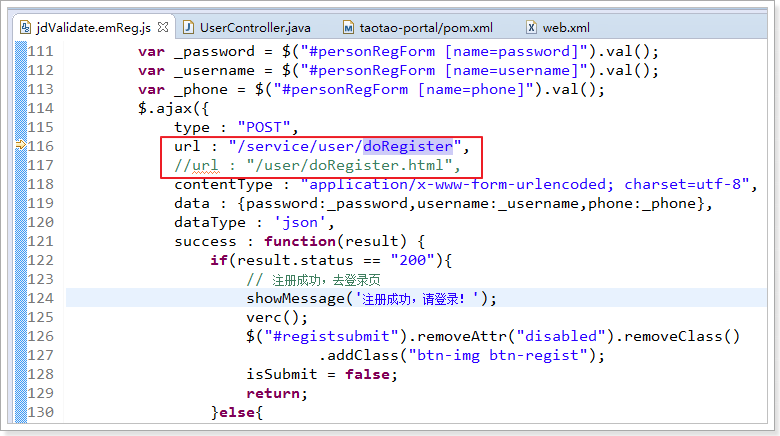
### 解决方案

请求的时候，不使用html结尾即可，可以使用service开头

修改taotao-portal的web.xml



把taotao-portal的前台Ajax请求url改为原来的路径



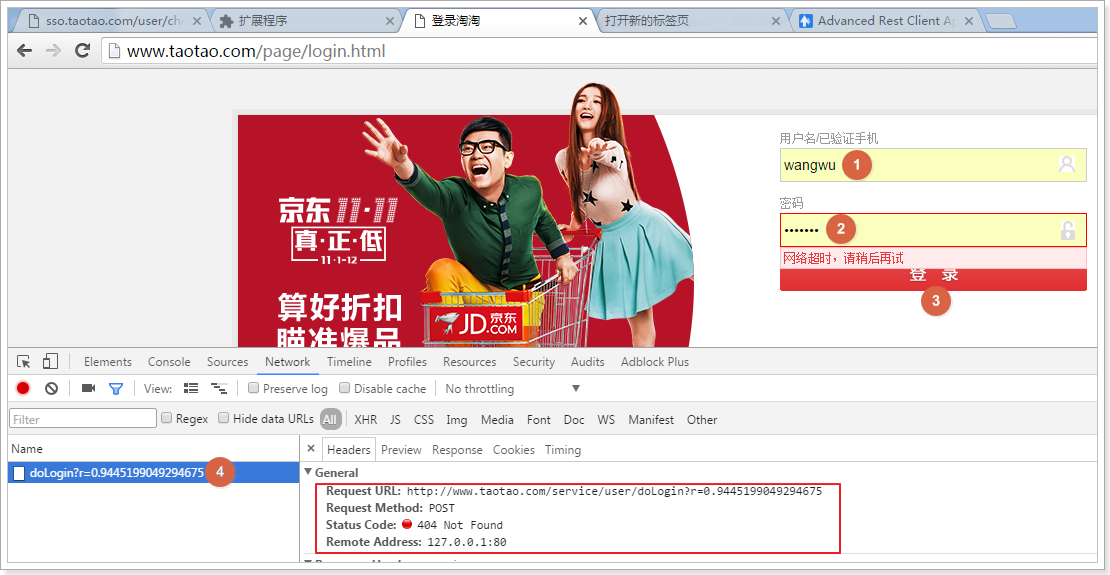
## 实现效果



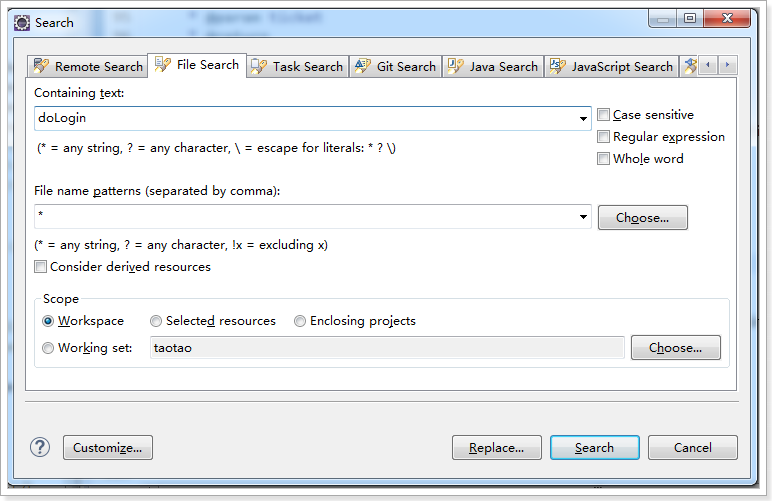
# 实现用户登录

## 开发分析

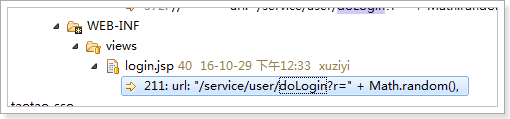
进入登录页面，点击“立刻注册”出现404错误



在eclipse里按ctrl+h搜索doLogin



搜索到一条结果



根据前台请求实现后台逻辑

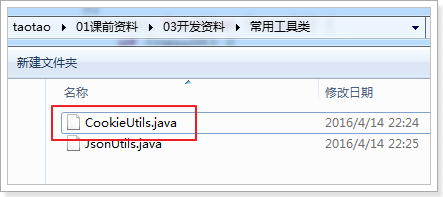
返回值要有status且为200，前台才认为登录成功

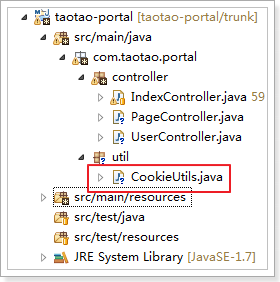


## 实现Controller

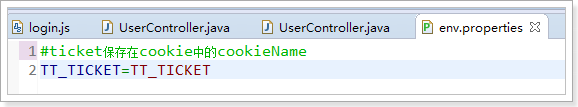
在taotao-portal实现

加入CookieUtils工具类，操作cookie

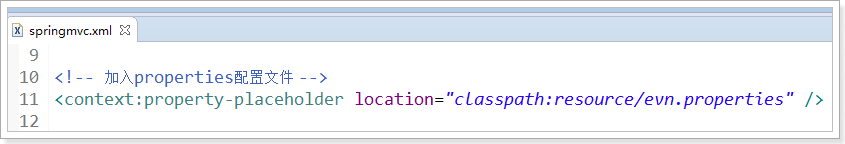




修改配置文件env.properties



在taotao-portal中加入env.properties



Controller实现登录逻辑

/\*\*

\* 用户登录

\*

\* **@param** user

\* **@param** request

\* **@param** response

\* **@return**

\*/

@RequestMapping(value = "doLogin", method = RequestMethod.***POST***)

@ResponseBody

**public** Map<String, Object> doLogin(User user, HttpServletRequest request, HttpServletResponse response) {

// 调用服务进行用户登录，需要返回ticket，目的是放到cookie中给用户

String ticket = **this**.userService.doLogin(user);

// 判断ticket是否为空，如果不为空表示用户登录成功

**if** (StringUtils.*isNotBlank*(ticket)) {

// 如果登录成功ticket需要放到cookie中

CookieUtils.*setCookie*(request, response, **this**.TT\_TICKET, ticket, 60 \* 60 \* 24, **true**);

// 封装返回数据

Map<String, Object> map = **new** HashMap<>();

map.put("status", "200");

**return** map;

}

**return** **null**;

}

## 实现Service接口

在taotao-sso-interface实现

/\*\*

\* 用户登录

\*

\* **@param** user

\* **@return**

\*/

String doLogin(User user) **throws** Exception ;

## 实现Service接口实现类

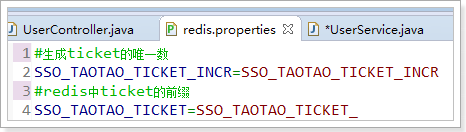
登录成功，需要返回ticket，ticket需要没有意义且唯一的，我们可以使用用户id+唯一数字，再使用MD5加密即可

登录成功后需要把用户信息保存在redis服务器中，

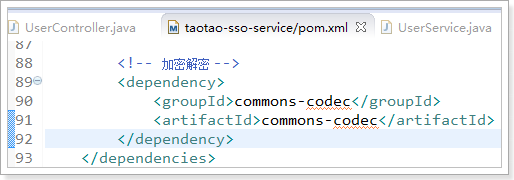
key为生成的ticket，value为用户信息。我们使用字符串类型存储，需要把User对象转为json数据

在taotao-sso-service实现

添加配置文件



需要对密码进行加密，添加依赖



实现方法

@Override

**public** String doLogin(User user) {

// 根据用户名和密码查询用户

// 设置密码加密，因为数据是用md5加密的

user.setUsername(user.getUsername());

user.setPassword(DigestUtils.*md5Hex*(user.getPassword()));

User result = **this**.userMapper.selectOne(user);

// 判断用户是否为空，如果不为空表示登录成功

**if** (result != **null**) {

**try** {

// 生成唯一数ticket,可是使用redis的唯一数+用户id

String ticket = "" + **this**.redisUtils.incr(**this**.TAOTAO\_SSO\_TICKET\_INCR) + result.getId();

// 把ticket和用户数据放到redis中,模拟session，原来的session有效时间是半小时

**this**.redisUtils.set(**this**.TAOTAO\_SSO\_KEY + ticket, ***MAPPER***.writeValueAsString(result), 60 \* 30);

// 返回ticket

**return** ticket;

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// 如果查询的用户为空，返回空

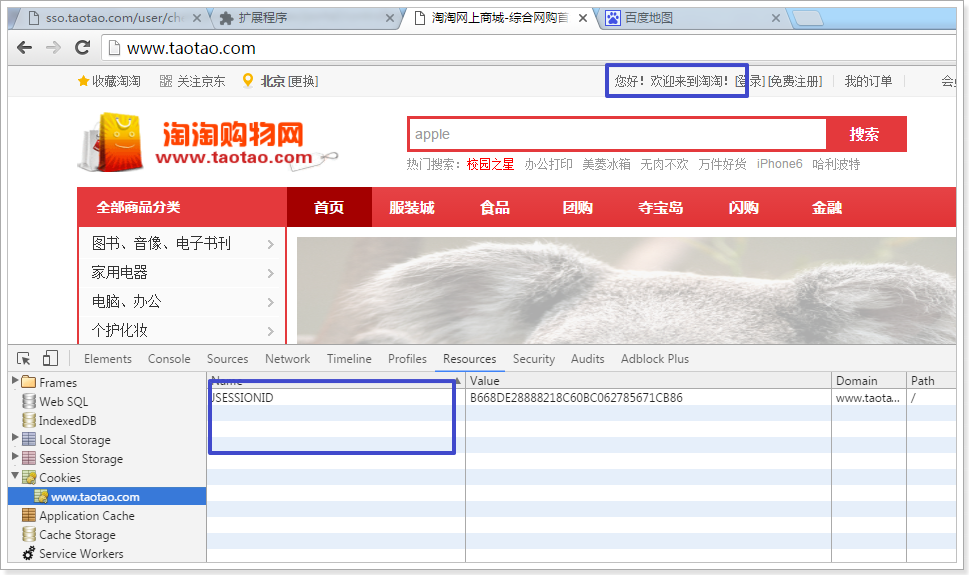
**return** **null**;

}

## 登录的问题

点击登录，直接跳转到首页，表示登录成功，但是在首页发现两个问题：

1. 没有欢迎用户的语句
2. cookie中没有ticket的信息

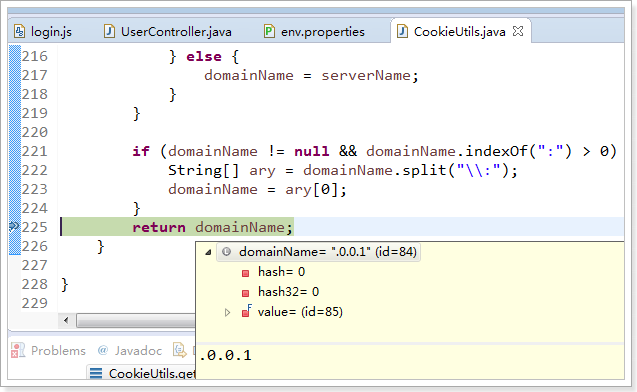


cookie没有数据，应该是保存cookie的问题，我们只有一个CookiUtils的方法跟保存cookie有关。用debug断点查看CookieUtils的保存方法，

发现获取ServerName的时候，不是域名而是ip端口号



返回domainName的时候是.0.0.1



发现问题：

1. 我们使用的是域名访问，但是这里是ip访问
2. domainName是.0.0.1

cookie中domain的作用：

每个cookie都是对应自己的域名，不同的域名访问时保存的信息放在不同的cookie中，使用domain进行区分。

查看淘宝、京东、百度等。他们使用的都是自己的domain

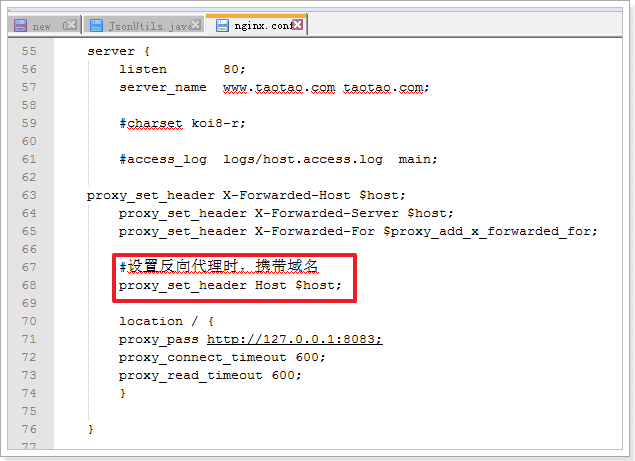
我们这里也应该是一样，把信息都保存在.taotao.com上

原因：

我们使用Nginx反相代理，Nginx在反相代理成功后，把域名丢失了，没有携带域名进行反向代理

## 解决Nginx没有携带域名的问题

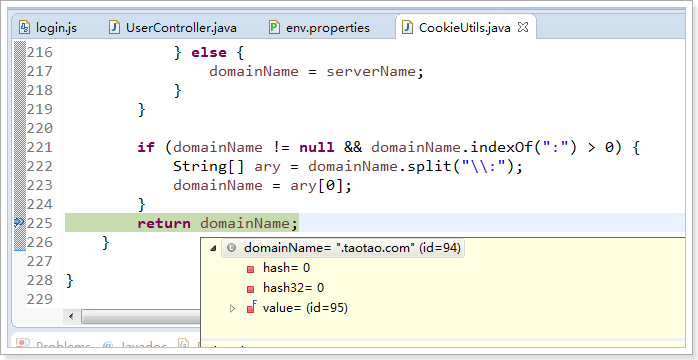
可以修改Nginx配置，让Nginx进行反向代理时，携带域名



再次debug查看效果



拿到.taotao.com的domain



返回首页正常

