# Dubbo项目改造

## 将jt-web/jt-sso改造为dubbo项目

### 改造JT-SSO

#### 引入jar包

<!--引入dubbo配置 -->

<dependency>

<groupId>com.alibaba.boot</groupId>

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

<version>0.2.0</version>

</dependency>

#### 编辑接口

**package** com.jt.service;

//定义中立的第三方接口

**public** **interface** DubboUserService {

}

#### 定义提供者实现类

//我是提供者的实现类

@Service(timeout = 3000) //阿里的包

**public** **class** DubboUserServiceImpl **implements** DubboUserService {

}

#### 编辑YML配置文件

server:

port: 8093

servlet:

context-path: /

spring:

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://127.0.0.1:3306/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

#mybatis-plush配置

mybatis-plus:

type-aliases-package: com.jt.pojo

mapper-locations: classpath:/mybatis/mappers/\*.xml

configuration:

map-underscore-to-camel-case: true

#引入日志信息

logging:

level:

com.jt.mapper: debug

#引入dubbo的配置

dubbo:

scan:

basePackages: com.jt

application:

name: provider-user #定义服务名称

registry: #定义注册中心链接

address: zookeeper://192.168.182.129:2181?backup=192.168.182.129:2182,192.168.182.129:2183

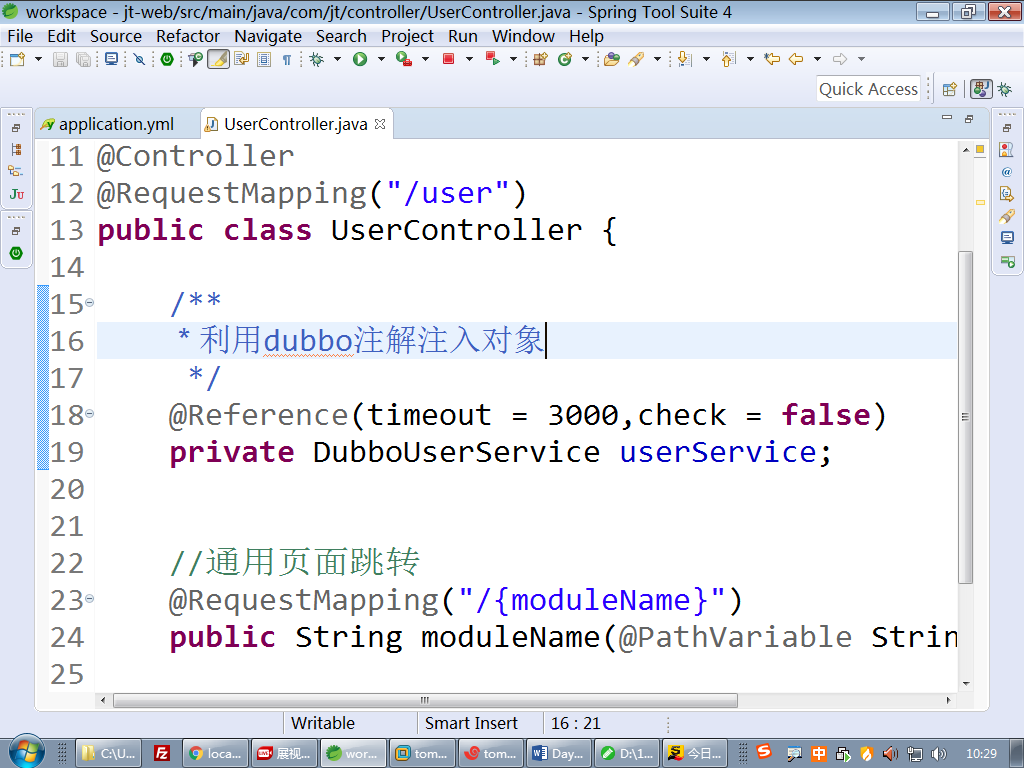
protocol:

name: dubbo #变量名称一般固定

port: 20880 #生产者的端口号

### 改造JT-WEB消费者

#### 编辑yml配置文件



#### 编辑消费者配置文件

server:

port: 8092

spring: #定义springmvc视图解析器

mvc:

view:

prefix: /WEB-INF/views/

suffix: .jsp

#引入dubbo的配置

dubbo:

scan:

basePackages: com.jt

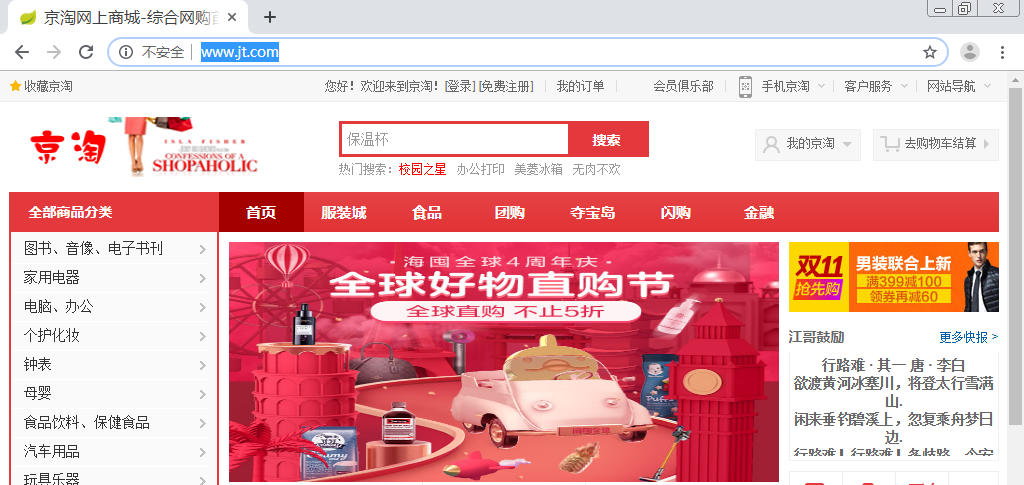
application:

name: consumer-user

registry:

address: zookeeper://192.168.182.129:2181?backup=192.168.182.129:2182,192.168.182.129:2183

#### 项目检查

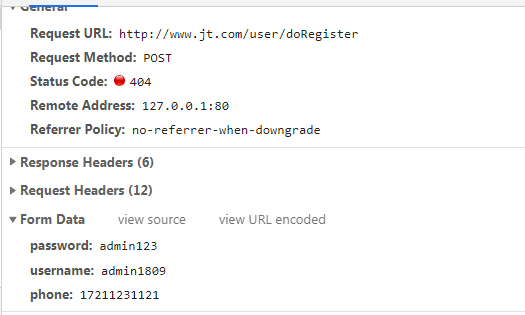


# 单点登录

## 用户注册

### 页面分析

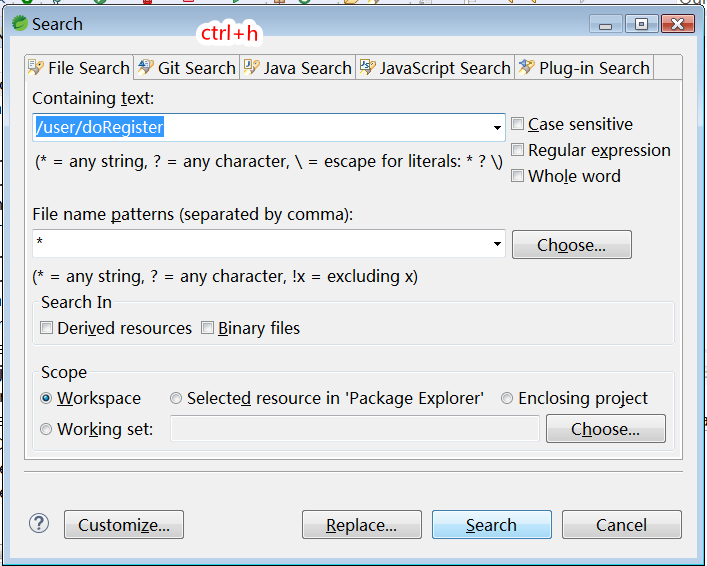
1.查看页面信息

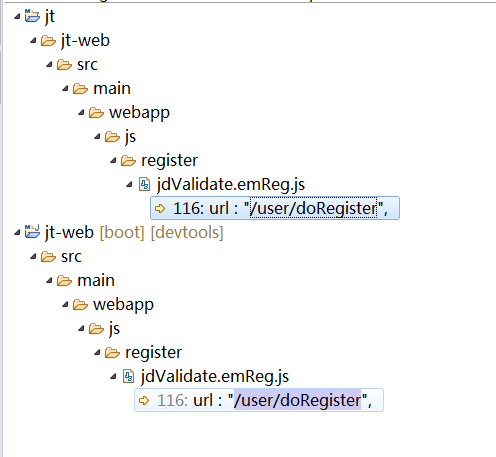


2.服务器返回数据是什么?

1.接口文档规范获取返回值信息.

2.如何快速获取返回值信息 检查js





3.JS分析

$.ajax({

type : "POST",

url : "/user/doRegister",

contentType : "application/x-www-form-urlencoded; charset=utf-8",

data : {password:\_password,username:\_username,phone:\_phone},

dataType : 'json',

success : **function**(result) {

**if**(result.status == "200"){

// 注册成功，去登录页

showMessage('注册成功，请登录！');

verc();

$("#registsubmit").removeAttr("disabled").removeClass()

.addClass("btn-img btn-regist");

isSubmit = **false**;

**return**;

}**else**{

showMessage('注册失败,请联系管理员!');

//alert('注册失败，请重新注册！ ' + result.data );

}

### 编辑Controller

@RequestMapping("/doRegister")

@ResponseBody

**public** SysResult saveUser(User user) {

//利用dubbo rpc协议完成远程过程调用

userService.saveUser(user);

**return** SysResult.*success*();

}

### 编辑Service

//我是提供者的实现类

@Service(timeout = 3000) //阿里的包

**public** **class** DubboUserServiceImpl **implements** DubboUserService {

@Autowired

**private** UserMapper userMapper;

@Override

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

//1.密码加密

String md5Pass =

DigestUtils.*md5DigestAsHex*

(user.getPassword().getBytes());

//2.封装数据

user.setEmail(user.getPhone())//暂时使用电话代替邮箱

.setPassword(md5Pass)

.setCreated(**new** Date())

.setUpdated(user.getCreated());

userMapper.insert(user);

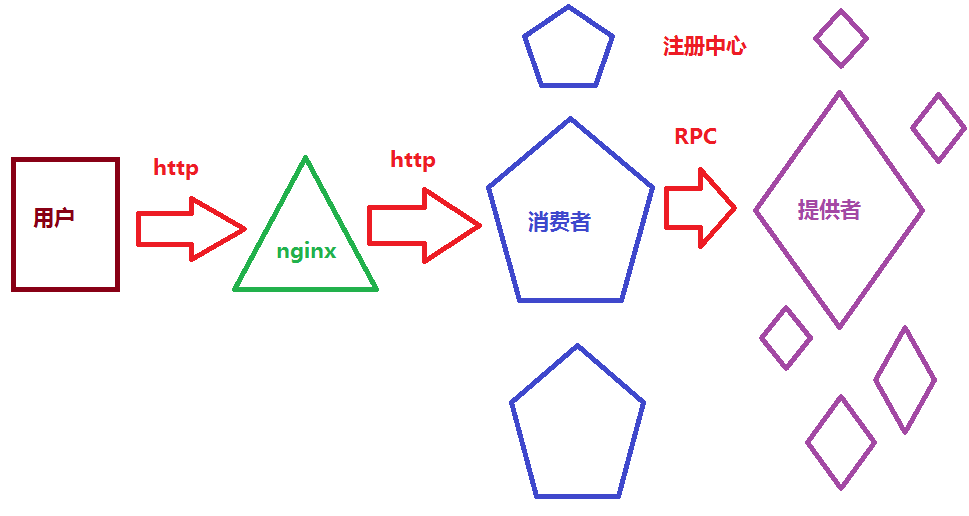
}

}

### 效果测试



### Nginx与dubbo关系



## 单点登录实现

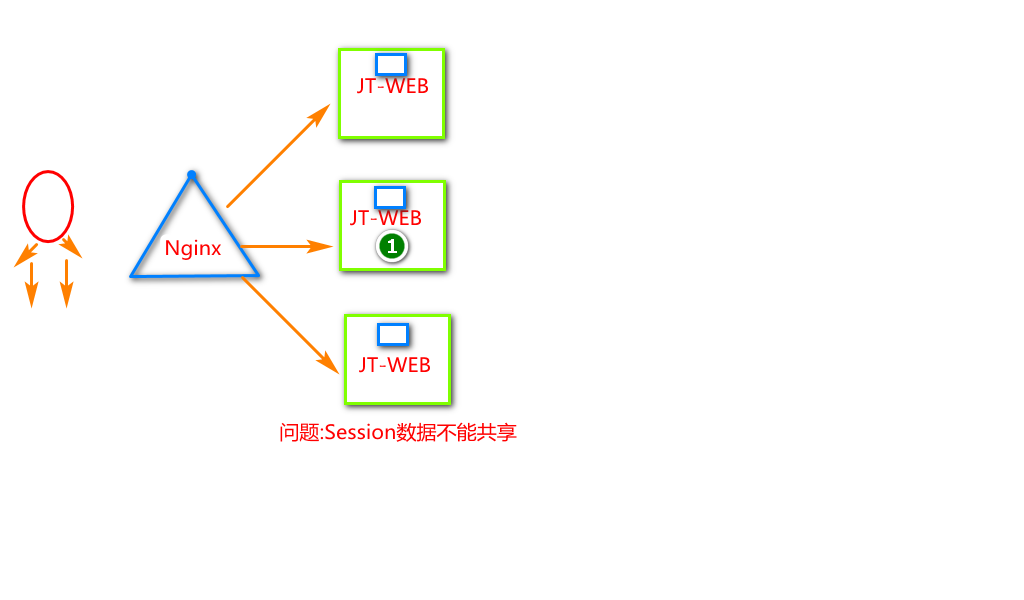
### 业务需求

当用户登录一次之后,下次可以免密登录.

知识回顾:Shiro将用户信息保存到Session中.(单台测试);

问题说明:

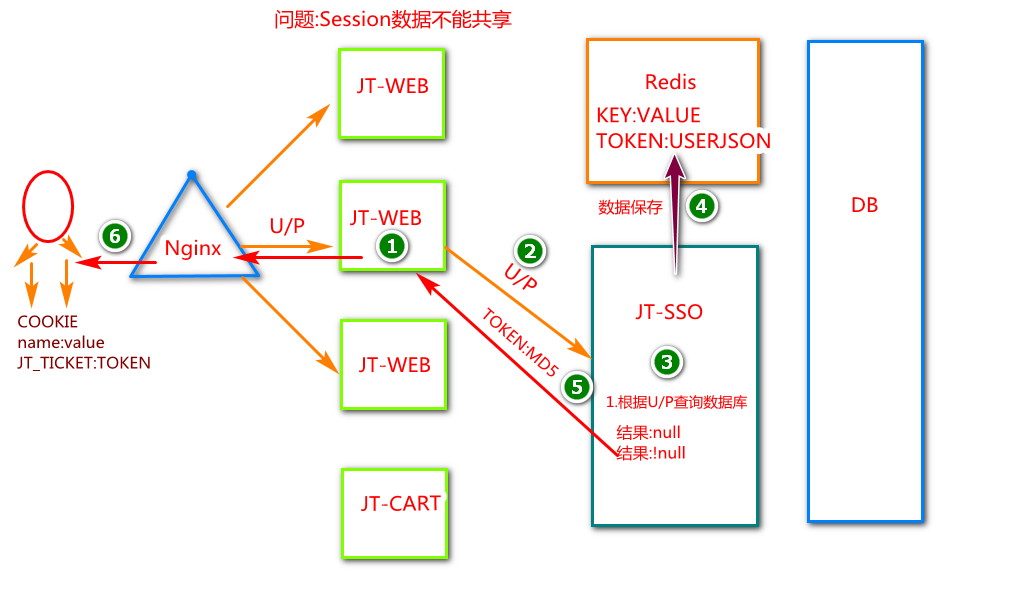
tomcat中不可以实现Session数据的共享.导致用户可能出现频繁登录的现象.



### SSO介绍

**单点登录(SingleSignOn，SSO)**，就是通过用户的一次性鉴别登录。当用户在身份认证服务器上登录一次以后，即可获得访问单点登录系统中其他联邦系统和应用软件的权限，同时这种实现是不需要管理员对用户的登录状态或其他信息进行修改的，这意味着在多个应用系统中，**用户只需一次登录就可以访问所有相互信任的应用系统**。这种方式减少了由登录产生的时间消耗，辅助了用户管理，是目前比较流行的.

### SSO单点登录业务实现



1. 用户通过浏览器输入username和密码进行登录操作.
2. JT-WEB服务将接受到的用户参数U/P.发送给后台服务器JT-SSO进行数据校验.
3. JT-SSO.接收请求之后通过U/P检索数据库.

如果检索的结果为null:表示用户名或密码错误.直接告知JT-WEB.

JT-WEB利用系统返回值SysResult告知用户密码有误.

如果用户名和密码正确则执行登录流程.

1. 利用MD5加密算法生产用户秘钥TOKEN数据.将User对象转化为json数据一起保存到redis中.
2. 将生成的TOKEN返回给JT-WEB.
3. JT-WEB获取服务器响应数据,之后校验 如果准确无误.将token数据写入用户的COOKIE中.方便下次访问.
4. 当用户登录成功之后.访问其他服务器系统.则利用**拦截器**校验用户是否登录.
5. 检查COOKIE
6. 检查redis

程序放行.实现了用户的免密登录.

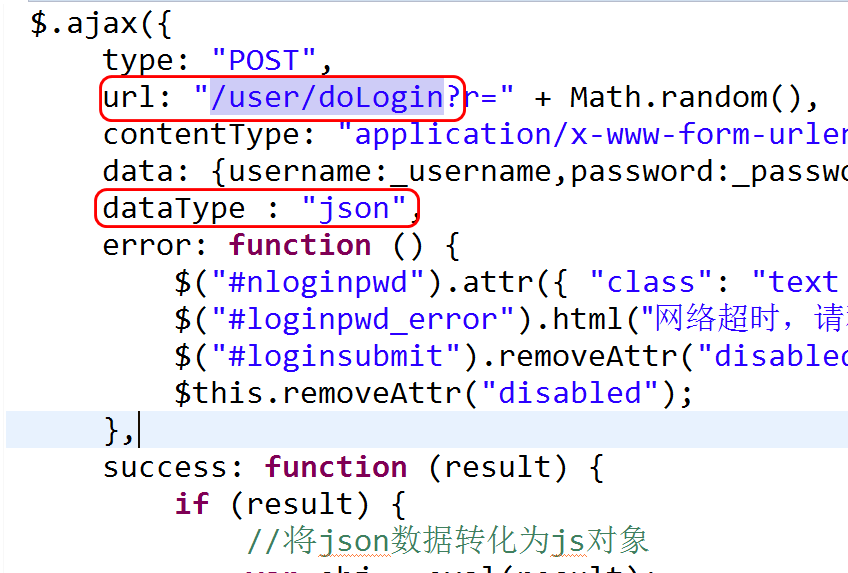
## 用户单点登录实现

### 页面分析

1. url分析



1. JS分析



### 编辑Controller

/\*\*

\* 1.关于Cookie声明周期问题

\* cookie.setMaxAge(>0); cookie数据存活100秒

\* cookie.setMaxAge(0); 表示删除cookie

\* cookie.setMaxAge(-1); 会话关闭cookie删除

\*

\* 2.表示Cookie使用的权限问题

\* www.baidu.com

\* cookie.setPath("/");

\* www.baidu.com/aa/1.html 可以访问

\* cookie.setpath("/bb")

\* www.baidu.com/aa/1.html 不可以访问

\*

\* 3.设定cookie共享

\* 规定:每一个网址都有自己固定的Cookie信息.默认不能共享

\* 需求:

\* www.jd.com 一级域名

\* item.jd.com 二级域名

\* 要求在一级域名与二级域名实现cookie数据共享.

\* 实现步骤:

\* 可以设定domain标签实现cookie共享.

\* **@param** user

\* **@param** response

\* **@return**

\*/

@RequestMapping("/doLogin")

@ResponseBody

**public** SysResult doLogin(User user,HttpServletResponse response) {

String token = userService.doLogin(user);

//校验远程服务器返回数据是否为null

**if**(StringUtils.*isEmpty*(token)) {

**return** SysResult.*fail*();

}

//将token数据写入Cookie

Cookie cookie = **new** Cookie("JT\_TICKET", token);

cookie.setMaxAge(7\*24\*3600); //cookie的存活时间

cookie.setPath("/");

cookie.setDomain("jt.com"); //让cookie共享

response.addCookie(cookie);

**return** SysResult.*success*();

}

### 编辑Service

/\*\*

\* 1.用户信息校验

\* 密码加密处理之后查询数据库

\* 2.校验用户数据.

\* 3.将数据保存到redis中.

\*/

@Override

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

String md5Pass = DigestUtils.*md5DigestAsHex*(user.getPassword().getBytes());

user.setPassword(md5Pass);

//将对象中不为null的属性当做where条件

QueryWrapper<User> queryWrapper = **new** QueryWrapper<User>(user);

User userDB = userMapper.selectOne(queryWrapper);

String token = **null**;

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

//1.将用户数据保存到redis中 生产key

String tokenTemp = "JT\_TICKET\_"+System.*currentTimeMillis*()+user.getUsername();

tokenTemp = DigestUtils.*md5DigestAsHex*(tokenTemp.getBytes());

//2.生成vlaue数据 userJSON

//为了安全 需要将数据进行脱敏处理

userDB.setPassword("123456你猜对吗??");

String userJSON = ObjectMapperUtil.*toJSON*(userDB);

jedisCluster.setex(tokenTemp,7\*24\*3600, userJSON);

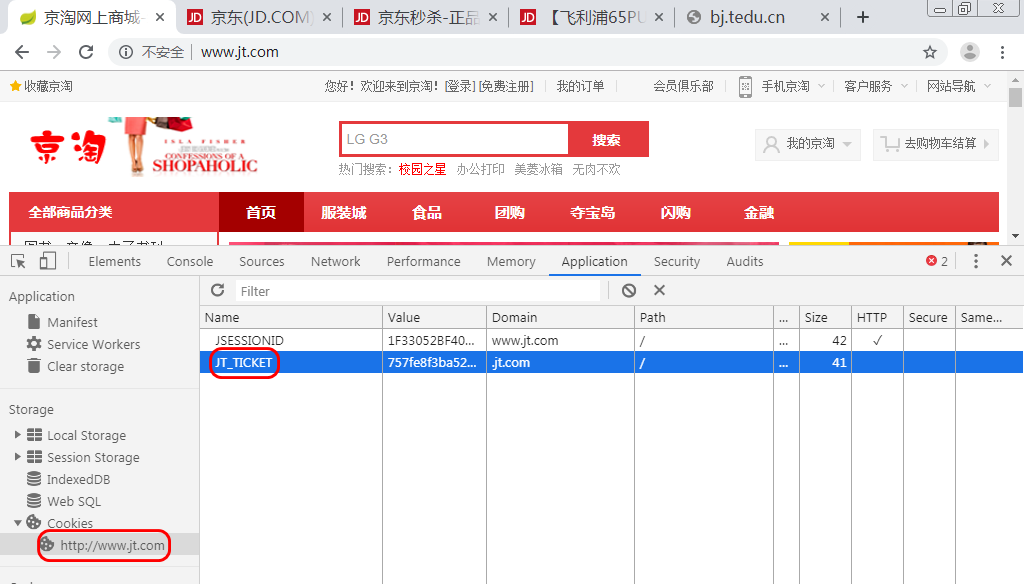
token = tokenTemp;

}

**return** token;

}

### 页面效果



## 实现用户信息回显

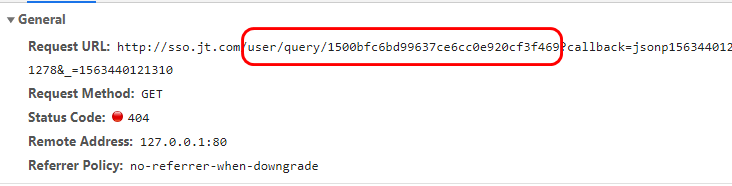
### 业务需求

说明:当用户登录成功之后,应该在网站首页部分展现用户信息.标识已经登录.



### 页面分析

1. url分析



1. js分析

$.ajax({

url : "http://sso.jt.com/user/query/" + \_ticket,

dataType : "jsonp",

type : "GET",

success : **function**(data){

**if**(data.status == 200){

//把json串转化为js对象

**var** \_data = JSON.parse(data.data);

**var** html =\_data.username+"，欢迎来到京淘！<a href=\"http://www.jt.com/user/logout.html\" class=\"link-logout\">[退出]</a>";

$("#loginbar").html(html);

}

}

});

### 编辑JT-SSO的Controller

@RequestMapping("/query/{token}")

**public** JSONPObject findUserByToken(String callback,@PathVariable String token) {

//1.根据秘钥查询用户信息

String userJSON = jedisCluster.get(token);

JSONPObject jsonpObject = **null**;

**if**(StringUtils.*isEmpty*(userJSON)) {

jsonpObject = **new** JSONPObject(callback,SysResult.*fail*());

}**else** {

//2.表示用户数据获取成功

jsonpObject = **new** JSONPObject(callback,SysResult.*success*(userJSON));

}

**return** jsonpObject;

}

### 页面效果展现

