# 京淘单点登录

## 知识回顾

### HttpClient和JSONP区别

说明:JSONP是解决主流浏览器跨域问题的 (域名)

httpClient可以通过java代码访问任意的地址的数据

区别:

1. 请求发起的位置不同

JSONP:浏览器解析ajax发起的请求

HttpClient:java代码中模拟发起请求

1. 安全性不同

JSONP:浏览器可以记录JSONP的全部请求过程.

HttpClient: 因为请求是经过2次处理的,所以相对安全

1. 请求方式不同:

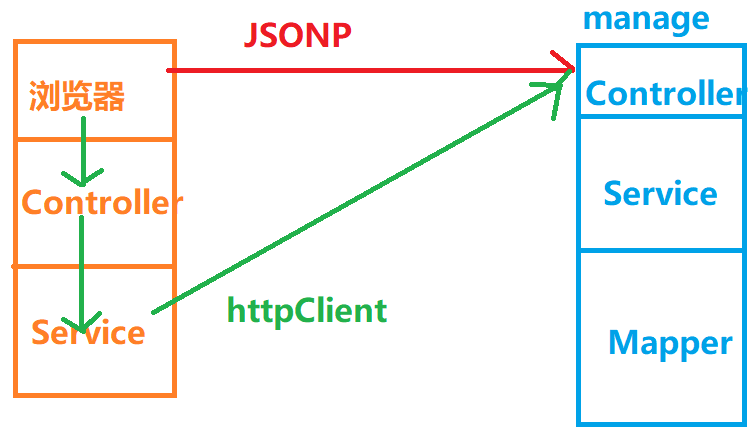
JSONP:发起GET请求,一般以查询为主.

httpClient:可以发起get/post请求.

1. 代码调用层级不同:

JSONP:3级调用

HttpClient:5级调用



### SpringMVC参数接收

1. SpringMVC可以接收String和基本类型

@RequestMapping("/doRegister")

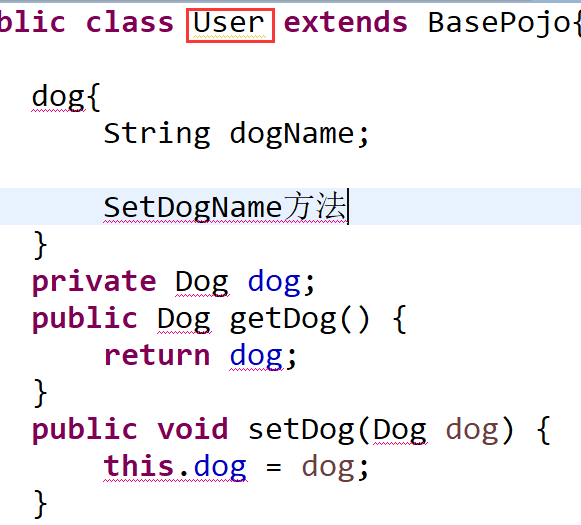
**public** SysResult saveUser(String password)

1. SpringMVC可以接收对象类型

先通过对象的get方法获取数据,之后通过set方法为属性赋值.

1. SpringMVC可以为对象的引用赋值!!!

页面:<input name=”dog.dogName” value=”旺财”/>



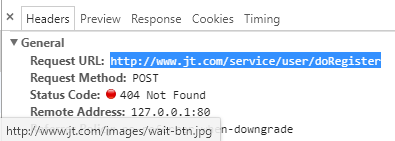
## 用户信息注册

### 用户注册步骤

1. 当用户点击注册按钮时,进行form表单的post提交.
2. 前台Controller接收页面提交的全部参数.
3. 接收数据后,进行业务处理,通过httpClient **Post提交**,实现数据远程传输.
4. 后台通过Controller接收参数.
5. 之后进行业务处理,最终入库保存.并且返回数据回执,告知前台入库情况.
6. 前台接收后台的回执后,进行判断,如果是200则正确跳转.如果201则友好提示,报错返回.

### 前台页面分析

1. 前台页面分析



1. 页面JS

$.ajax({

type : "POST",

url : "/service/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**{

alert('注册失败，请重新注册！ ' + result.data );

}

}

### 编辑前台Controller

/\*\*

\* SpringMVC,可以使用对象接收参数.

\* 1.password

\* **@param** user

\* **@return**

\*/

@RequestMapping("/doRegister")

@ResponseBody

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

**try** {

userService.saveUser(user);

**return** SysResult.*oK*(); //表示程序执行正确

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"用户注册失败");

}

### 编辑前台Service

@Service

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

@Autowired

**private** HttpClientService httpClient;

@Autowired

**private** ObjectMapper objectMapper;

//发起的httpClient Post请求

@Override

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

String url = "http://sso.jt.com/user/register";

Map<String,String> params =

**new** HashMap<>();

params.put("username",user.getUsername());

String md5Pass =

DigestUtils.*md5DigestAsHex*

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

params.put("password",md5Pass);

params.put("phone", user.getPhone());

params.put("email", user.getEmail());

//发送请求,获得回执

String result = httpClient.doPost(url, params);

//获取数据后,校验后台执行情况 200表示正确,201表示失败

**try** {

SysResult sysResult =

objectMapper.readValue(result,SysResult.**class**);

**if**(sysResult.getStatus() !=200) {

System.***out***.println("表示后台执行有错:"+result);

**throw** **new** RuntimeException();

}

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException();

}

}

}

### 编辑SSO后台Controller

//url http://sso.jt.com/user/register

@RequestMapping("/register")

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

**try** {

userService.saveUser(user);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"用户新增失败");

}

### 编辑SSO的Service

@Override

@Transactional //添加事务控制

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

//为了让后台不报错,则暂时使用电话代替email

user.setEmail(user.getPhone());

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

user.setUpdated(user.getCreated());

userMapper.insert(user);

System.***out***.println("用户信息入库成功!!!");

}

### 页面效果

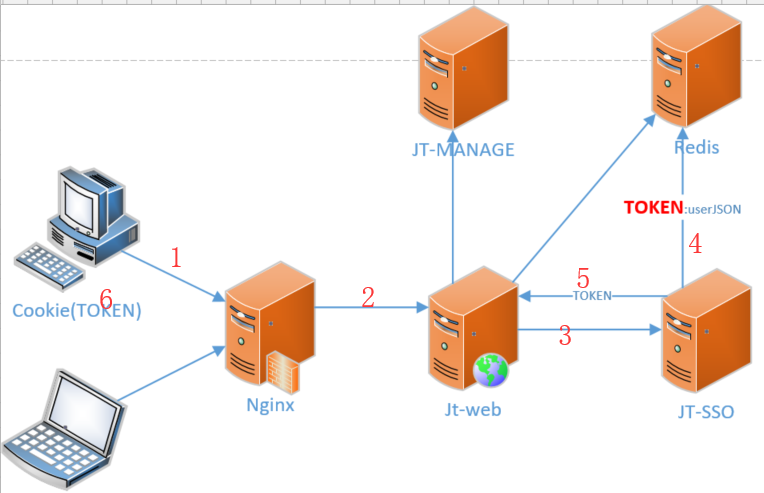


## SSO单点登录实现

### 单点登录定义

SSO英文全称Single Sign On，[单点登录](https://baike.baidu.com/item/%E5%8D%95%E7%82%B9%E7%99%BB%E5%BD%95/4940767" \t "_blank)。SSO是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。它包括可以将这次主要的登录映射到其他应用中用于同一个用户的登录的机制。它是目前比较流行的企业业务整合的解决方案之一

### 单点登录实现



实现步骤:

1. 当用户点击登陆按钮时,携带username和password实现登录操作.将请求发送给jt-web.
2. Jt-web接收用户请求.之后将参数进行封装.通过httpClient中的Post请求将数据发送给jt-sso.单点登录系统.
3. Jt-sso接收jt-web发送的用户登陆数据.通过数据库查询校验用户数据是否正确!

如果用户名和密码不正确,直接201报错返回

如果用户名和密码正确.

1. 经过加密的算法生成秘钥TOKEN
2. 将用户信息转化为JSON数据.

4.将秘钥token和JSON保存到redis中.

5.将后台生成的TOKEN返回给jt-web

6.jt-web接收token数据后,通过request对象操作Cookie.将用户token信息保存到Cookie中(默认7天).

7.如果之前的一系列操作执行正确.如果用户做敏感操作时,则通过token检查redis中是否有数据即可.该操作之后完善!!!

### 前台页面分析

1. url分析



1. JS分析

$.ajax({

type: "POST",

url: "/service/user/doLogin?r=" + Math.random(),

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

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

dataType : "json",

success: **function** (result) {

**if** (result) {

//eval将串转化为js对象

**var** obj = eval(result);

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

obj.success = "http://www.jt.com/index.html";

**var** isIE = !-[1,];

**if** (isIE) {

**var** link = document.createElement("a");

link.href = obj.success;

link.style.display = 'none';

document.body.appendChild(link);

link.click();

} **else** {

window.location = obj.success;

}

**return**;

}

### 编辑前台Controller

//实现京淘用户登陆

@RequestMapping("/doLogin")

@ResponseBody //http协议规定传输串

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

**try** {

//调用业务层方法,返回token数据

String token =

userService.findUserByUP(user);

//判断返回值token不为null时,.保存Cookie

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

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

cookie.setMaxAge(7 \* 24 \* 3600);

//cookie使用权限配置

cookie.setPath("/");

response.addCookie(cookie);

//操作无误后,正确返回

**return** SysResult.*oK*();

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"用户登陆失败");

}

### 编辑前台Service

@Override

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

String token = **null**;

String url = "http://sso.jt.com/user/login";

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

params.put("username", user.getUsername());

params.put("password", user.getPassword());

String result = httpClient.doPost(url, params);

**try** {

SysResult sysResult =

objectMapper.readValue(result,SysResult.**class**);

**if**(sysResult.getStatus() == 200) {

token = (String) sysResult.getData();

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** token;

}

### 编辑后台的Controller

//实现用户信息校验

@RequestMapping("/login")

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

**try** {

String token = userService.findUserByUP(user);

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

**return** SysResult.*oK*(token);

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"用户登陆失败");

}

### 编辑SSO后台Service

@Override

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

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

queryWrapper.eq("username",user.getUsername())

.eq("password",user.getPassword());

User userDB = userMapper.selectOne(queryWrapper);

String token = **null**;

**try** {

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

//1.生成加密的TOKEN

String key = "JT\_TICKET\_" +

System.*currentTimeMillis*()+

user.getUsername();

token =

DigestUtils.*md5DigestAsHex*(key.getBytes());

String userJSON =

objectMapper.writeValueAsString(userDB);

//将数据保存到redis中.添加超时时间

jedisCluster.set(token, userJSON);

jedisCluster.expire(token, 24 \* 7 \* 3600);

}

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException();

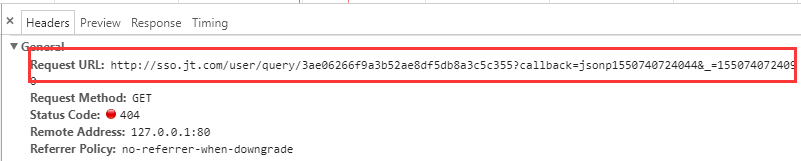
}

**return** token;

}

### 作业

1. 完成登录的数据回显



1. 实现登录后登出操作
2. 删除cookie数据
3. 删除redis数据.