# Dubbo实现单点登录

## Dubbo负载均衡特性

### 业务需求

说明:使用dubbo框架通信的方式采用**RPC**通信.如果需要搭建提供者的集群来实现抗击高兵发的业务处理.不能使用Nginx实现负载均衡.应该如何处理

答案:Dubbo框架内部基于多个提供者实现了负载均衡策略.自己实现不需要第三方支持.

### 测试用例

说明:面向同一个接口准备2个服务提供者.

注意事项:

1).服务名称一定是相同的,因为实现同一个接口.

2)服务端口号一定不相同的,一个服务占用一个端口.

3).实现同一个接口业务逻辑必然相同.

### 负载均衡说明

说明:dubbo默认的负载均衡策略是随机发送!!!!!

策略:

1. RandomLoadBalance 随机发起请求 该配置是默认的
2. RoundRobinLoadBalance 权重相同时轮询策略
3. LeastActiveLoadBalance 根据响应时间的长短实现负载均衡.如果服务器响应时间越短,则用户会将大量的请求发送给该服务器.
4. ConsistentHashLoadBalance

根据hash算法实现负载均衡.实现服务器绑定.

配置方式:

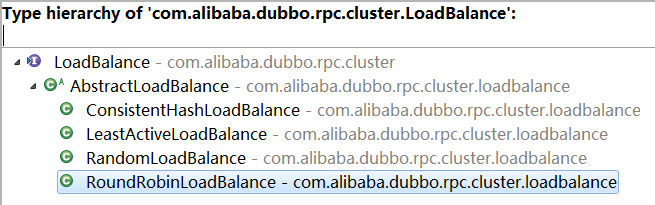
可以在服务端/客户端通过注解的形式配置.**引用时将负载均衡类名前缀小写即可.**

@Reference(timeout = 3000,check=false,loadbalance ="random")

@Reference(timeout = 3000,check=false,loadbalance ="roundrobin")

@Reference(timeout = 3000,check=false,loadbalance ="leastactive")

@Reference(timeout = 3000,check=false,loadbalance ="consistenthash")



### Dubbo面试题

问题:当注册中心宕机后,dubbo能否继续提供服务.

答案:依然可以正确访问.因为消费者启动时已经将服务列表数据缓存到本地.

# 用户注册

## 实现用户注册

### 导入jar包

<!--引入dubbo配置 -->

<dependency>

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

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

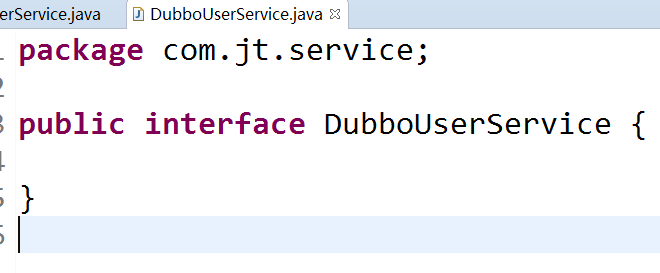
<version>0.2.0</version>

</dependency>

## 定义中立接口

### 编辑接口文件

说明:在jt-common中添加userService接口



## 编辑服务提供者

### 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

#Springboot整合dubbo

dubbo:

scan:

basePackages: com.jt #包扫描dubbo的service注解

application:

name: provider-user #提供者服务名称

registry: #链接zookeeper

address: zookeeper://192.168.175.129:2181?backup=192.168.175.129:2182,192.168.175.129:2183

protocol:

name: dubbo

port: 20880 #服务通信端口 消费者链接提供者端口

#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

### 编辑业务接口实现

//该类是dubbo的实现类

@Service(timeout =3000)

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

@Autowired

**private** UserMapper userMapper;

}

## 定义消费者

### 编辑yml配置文件

server:

port: 8092

spring: #定义springmvc视图解析器

mvc:

view:

prefix: /WEB-INF/views/

suffix: .jsp

dubbo:

scan:

basePackages: com.jt

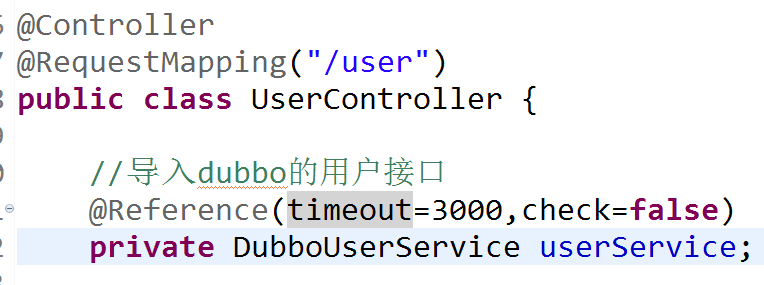
application:

name: consumer-jt-web #服务名称"唯一"

registry:

address: zookeeper://192.168.175.129:2181?backup=192.168.175.129:2182,192.168.175.129:2183

### 编辑前台Controller



## 实现用户新增业务

### 编辑Controller

//使用dubbo形式实现业务调用

@RequestMapping("/doRegister")

@ResponseBody

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

**try** {

userService.saveUser(user);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

### 编辑Service

//该类是dubbo的实现类

@Service(timeout =3000)

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

@Autowired

**private** UserMapper userMapper;

@Transactional //添加事务控制

@Override

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

//1.将密码加密

//2.补齐入库数据 email暂时使用电话代替

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

user.setPassword(md5Pass)

.setEmail(user.getPhone())

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

.setUpdated(user.getCreated());

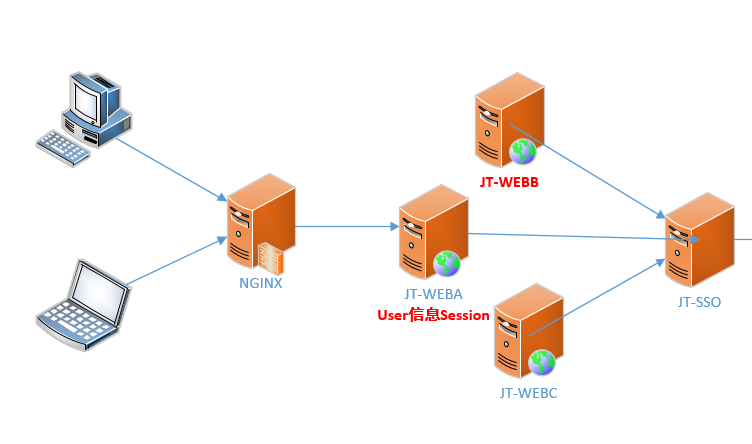
userMapper.insert(user);

}

}

## 单点登录实现

### 分布式项目中登录问题



问题说明:

当采用分布式的部署之后,jt-web服务将来部署集群,如果将用户信息保存到Session中时.由于Session不能共享数据.则可能会导致用户经过nginx负载均衡时每次访问不同的服务器.这时用户做敏感操作时例如:支付/查询订单.必须要求用户登录.这样用户体验较差!!!

问题2:

在一个业务系统中登录之后,访问其他启动时应该也是免密登录的.不应该重复登录.使用Session不能实现该操作?

### 实现思路

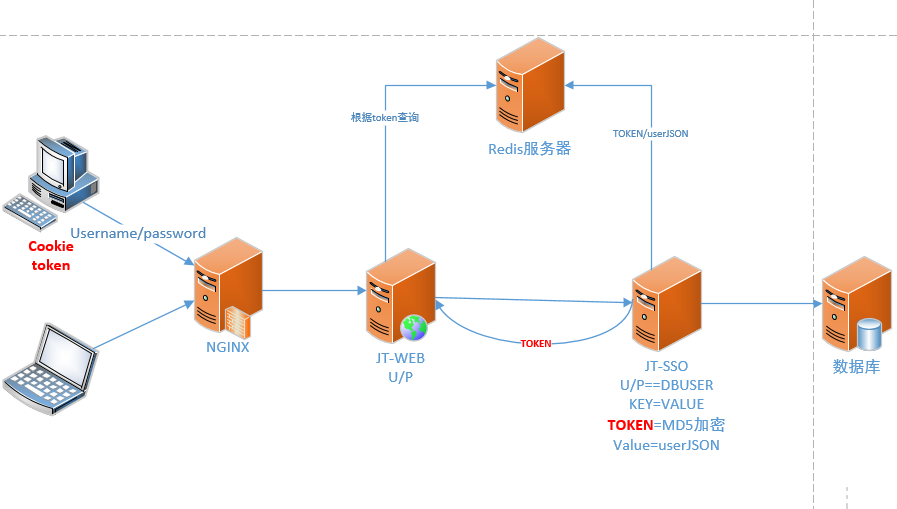
由于之前的方案将用户信息与服务器系统绑定在一起.而导致数据信息不能共享.

## SSO单点登录介绍

### SSO介绍

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。

### SSO单点登录架构图设计



1. 当用户通过浏览器登录时发送用户名和密码进行校验
2. 当jt-web服务器获取到用户信息时.由于分布式的原因前台不能直接校验用户信息.所以访问sso单点登录业务系统.
3. 当jt-sso业务系统接收前台jt-web传递的用户信息(用户名/密码)

如果根据信息查询数据库没有获取数据库数据.则说明用户名或者密码错误. 如果后台查询失败则通知前台.前台通过友好提示告知用户用户名或密码错误.(status=201)

当查询数据库时数据存在.说明用户输入的数据正确.

1. 如果信息正确则开始生成秘钥token/ticket,将用户信息转化为json一起保存到redis中.7天有效.
2. 当上述操作完成之后,将token信息秘钥交给用户自己保存.将数据保存到Cookie中 7天有效.

同时保证Cookie在相互信任的系统中共享!!!!

## 单点登录业务实现

### 页面分析

#### URL参数问题

目的:Controller中拦截请求路径,参数名称

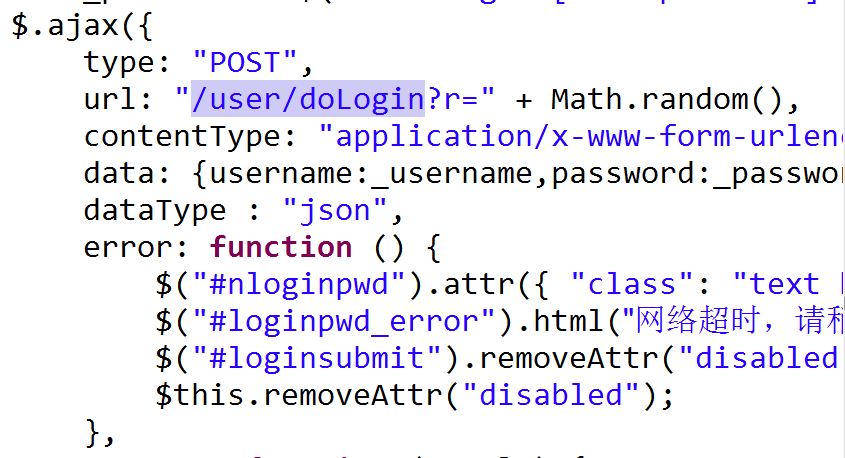
url: <http://www.jt.com/user/doLogin?r=0.6489583120288842>

参数: username:xxxx

Password:xxxx

#### JS分析

目的:了解请求是如何发起的.了解返回值数据的要求.



### 编辑Controller

/\*\*

\* 实现用户登录

\*

\* 说明:利用Response对象将Cookie数据写入客户端

\* cookie生命周期:

\* cookie.setMaxAge(0); 立即删除

\* cookie.setMaxAge(值>0) 能够存活多久 单位/秒

\* cookie.setMaxAge(-1) 当会话结束后删除

\*

\* cookie的权限 一般都写"/"

\* cookie.setPath("/");

\*

\* 页面位置:

\* www.jd.com/abc/a.html

\* www.jd.com/b.html

\* **@param** user

\* **@return**

\*/

@RequestMapping("/doLogin")

@ResponseBody

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

**try** {

//调用sso系统获取秘钥

String token =

userService.findUserByUP(user);

//如果数据不为null时,将数据保存到Cookie中

//cookie中的key固定值 JT\_TICKET

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

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

cookie.setMaxAge(7\*24\*3600);//生命周期

cookie.setPath("/");

response.addCookie(cookie);

**return** SysResult.*ok*();

}

} **catch** (Exception e) {

e.printStackTrace();

}

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

}

### 编辑Service

/\*\*

\* 1.校验用户名或密码是否正确

\* 2.如果数据不正确 返回null

\* 3.如果数据正确.

\* 1.生成加密秘钥: MD5(JT\_TICKET\_+username+当前毫秒数)

\* 2.将userDB数据转化为userJSON

\* 3.将数据保存到redis中 7天超时

\* 4.返回token

\*/

@Override

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

String token = **null**;

//1.将密码进行加密

String md5Pass =

DigestUtils.*md5DigestAsHex*(

user.getPassword().getBytes());

user.setPassword(md5Pass);

QueryWrapper<User> queryWrapper =

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

User userDB = userMapper.selectOne(queryWrapper);

//2.判断数据正确执行下列代码

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

token = "JT\_TICKET\_"+userDB.getUsername()

+ System.*currentTimeMillis*();

token = DigestUtils

.*md5DigestAsHex*(token.getBytes());

//脱敏处理 一切从业务出发.

userDB.setPassword("你猜猜!!!!");

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

//将数据保存到redis中

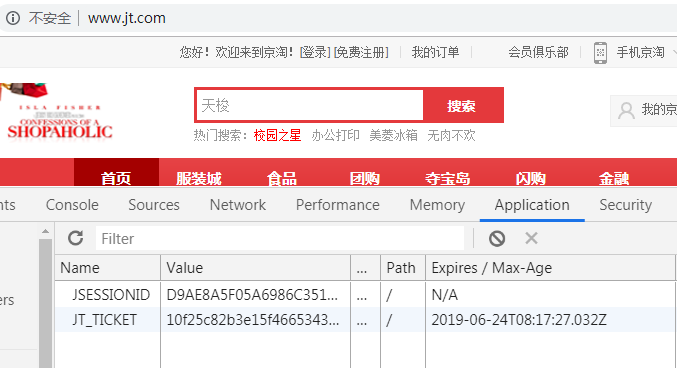
jedisCluster.setex(token,7\*24\*3600, userJSON);

}

**return** token;

}

### 页面效果



### 实现Cookie数据共享

说明:单点登录要求,当用户进行一次登录后,所有相互信任的系统都能实现免密登录.前提条件是所有的信任系统都可以获取token数据,前提实现Cookie的共享!!!!.

实现步骤:

//cookie中的key固定值 JT\_TICKET

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

Cookie cookie =

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

cookie.setMaxAge(7\*24\*3600);//生命周期

**cookie.setDomain("jt.com"); //实现数据共享**

cookie.setPath("/");

response.addCookie(cookie);

**return** SysResult.*ok*();

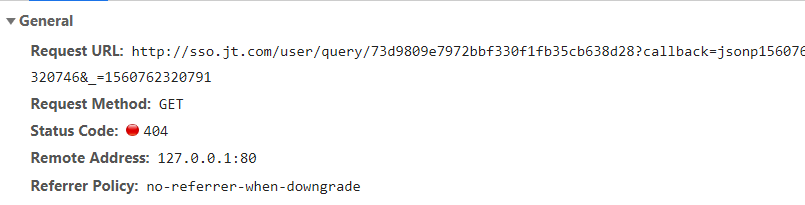
}

## 用户信息回显

### 业务说明

当用户登录成功之后.需要回显用户信息.

1.页面url地址



2.页面js

checkLogin : **function**(){

**var** \_ticket = $.cookie("JT\_TICKET");

**if**(!\_ticket){

**return** ;

}

//当dataType类型为jsonp时，jQuery就会自动在请求链接上增加一个callback的参数

$.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SSO的Controller

/\*\*

\* 利用跨域实现用户信息回显

\* http://sso.jt.com/user/query/73d9809e7972bbf330f1fb35cb638d28?callback=jsonp1560762546338&\_=1560762546390

\* **@return**

\*/

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

**public** JSONPObject findUserByTicket(@PathVariable String ticket,String callback) {

String userJSON = jedisCluster.get(ticket);

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

//回传数据需要经过200判断 SysResult对象

**return** **new** JSONPObject(callback, SysResult.*fail*());

**else**

**return** **new** JSONPObject(callback, SysResult.*ok*(userJSON));

}

### 页面效果



### 作业

1. 完成用户登出操作
2. 当点击登出按钮时.页面跳转到系统首页 转发/重定向
3. 删除cookie中的数据…
4. 删除redis中的数据…. ~~~~token ~~~~~获取cookie~~~~request对象获取.