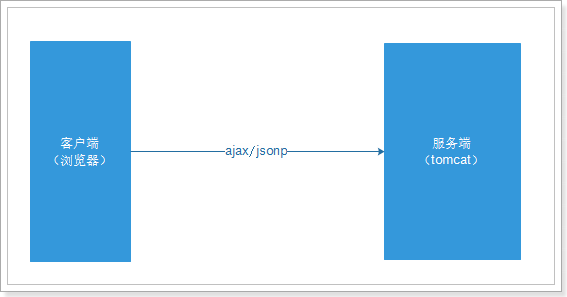
# 今日大纲

1. Dubbo入门学习
2. 使用dubbo优化单点登录系统

# 系统间服务调用方式

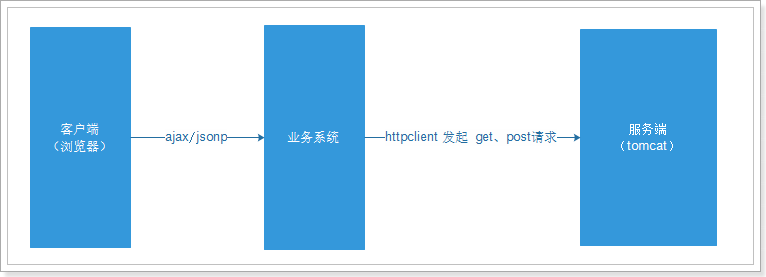
## 浏览器直接访问

浏览器发起请求，通过ajax或jsonp方式请求：



## Httpclient方式

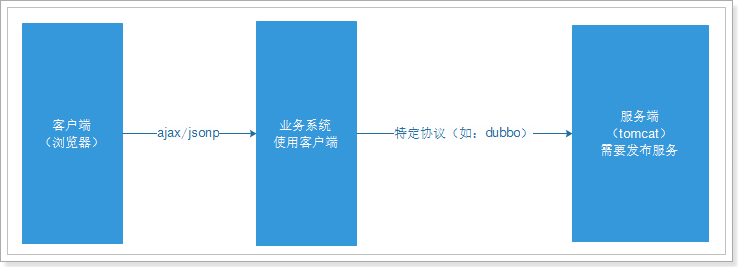
系统与系统之间通过Httpclient发起http请求来请求数据：



http协议是短连接。

## RPC方式





采用长连接方式。

# 单点系统中存在的问题

在单点登录系统中的功能中，根据token查询用户信息的功能对系统性能要求最高，如果我们想单独调整该功能的性能是不可能的，因为该功能和其它的功能耦合再一起。

要想单独优化该功能的性能就必须把该功能单独出来，我们就可以借助与dubbo框架完成。

# Dubbo入门

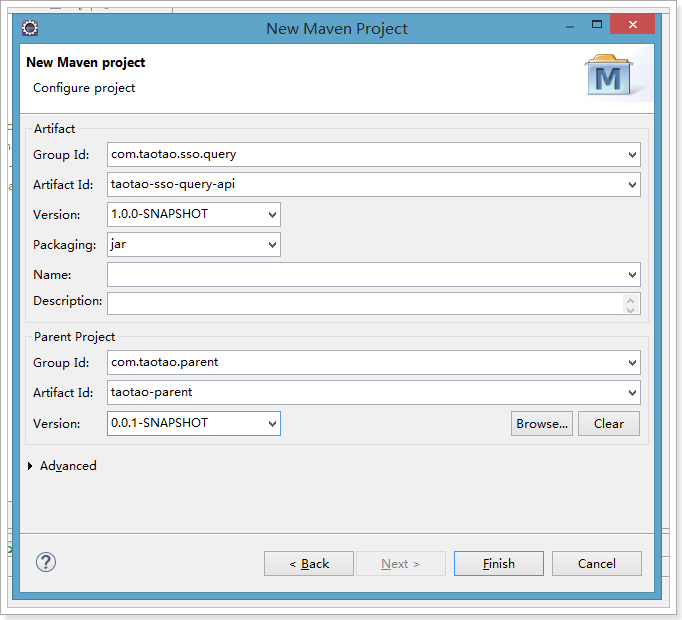
具体参考课前资料中的《dubbo入门教程.docx》

# 使用dubbo优化单点系统的查询功能

## 创建taotao-sso-query-api工程

### 创建maven工程

该工程定义查询接口。



### 导入依赖

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>com.taotao.parent</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao.sso.query</groupId>

<artifactId>taotao-sso-query-api</artifactId>

<version>1.0.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>com.taotao.common</groupId>

<artifactId>taotao-common</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

</dependencies>

</project>

### 将taotao-sso中的User对象拷贝到该工程

拷贝代码并且做一些修改，将jpa的注解删除：

**package** com.taotao.sso.query.bean;

**import** java.util.Date;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

**public** **class** User {

**private** Long id;

**private** String username;

@JsonIgnore

// 将对象序列化json字符串时忽略该字段

**private** String password;

**private** String phone;

**private** String email;

**private** Date created;

**private** Date updated;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** Date getCreated() {

**return** created;

}

**public** **void** setCreated(Date created) {

**this**.created = created;

}

**public** Date getUpdated() {

**return** updated;

}

**public** **void** setUpdated(Date updated) {

**this**.updated = updated;

}

}

### 定义服务接口

**package** com.taotao.sso.query.api;

**import** com.taotao.sso.query.bean.User;

**public** **interface** UserQueryService {

/\*\*

\* 根据token查询User对象

\*

\* **@return**

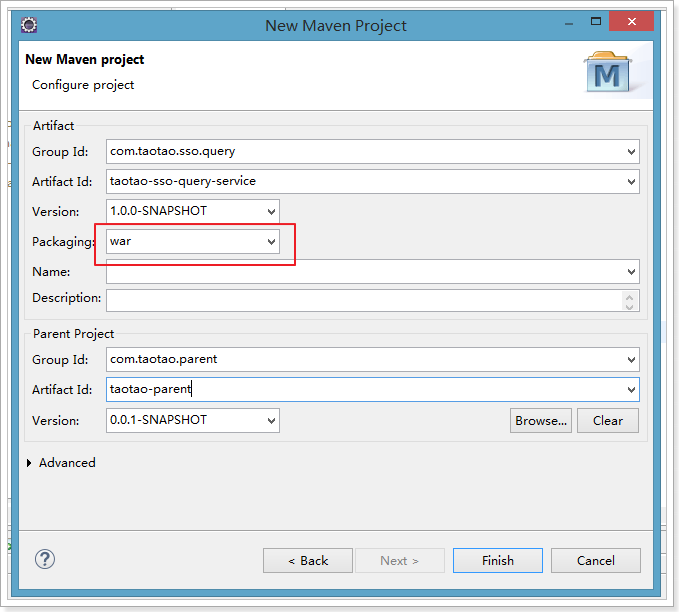
\*/

**public** User queryUserByToken(String token);

}

## 创建taotao-sso-query-service

### 创建maven工程



### 导入依赖

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>com.taotao.parent</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<groupId>com.taotao.sso.query</groupId>

<artifactId>taotao-sso-query-service</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<dependency>

<groupId>com.taotao.common</groupId>

<artifactId>taotao-common</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>com.taotao.sso.query</groupId>

<artifactId>taotao-sso-query-api</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- dubbo采用spring配置方式，所以需要导入spring容器依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.5.3</version>

<exclusions>

<exclusion>

<!-- 排除传递spring依赖 -->

<artifactId>spring</artifactId>

<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<port>8087</port>

<path>/</path>

</configuration>

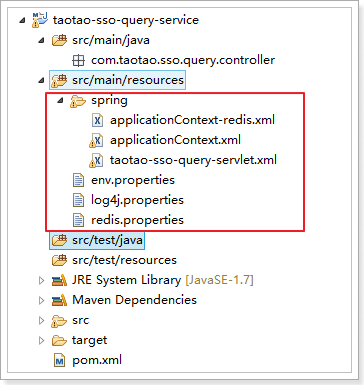
</plugin>

</plugins>

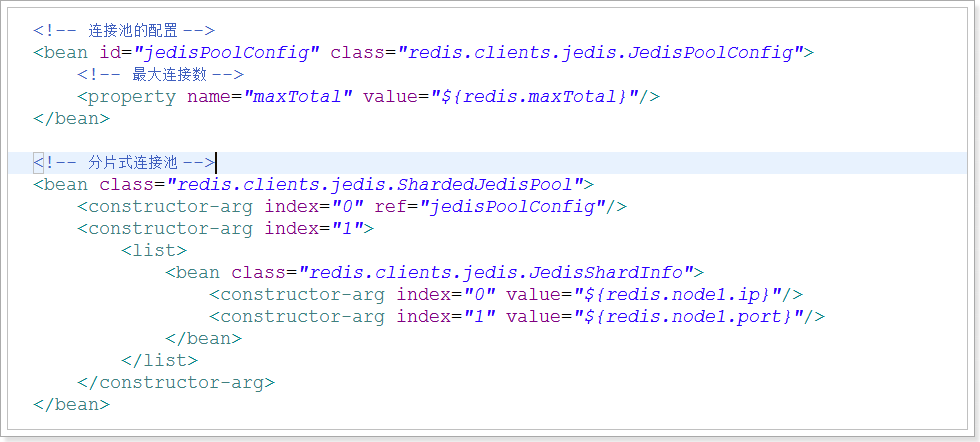
</build>

</project>

### 编写配置文件



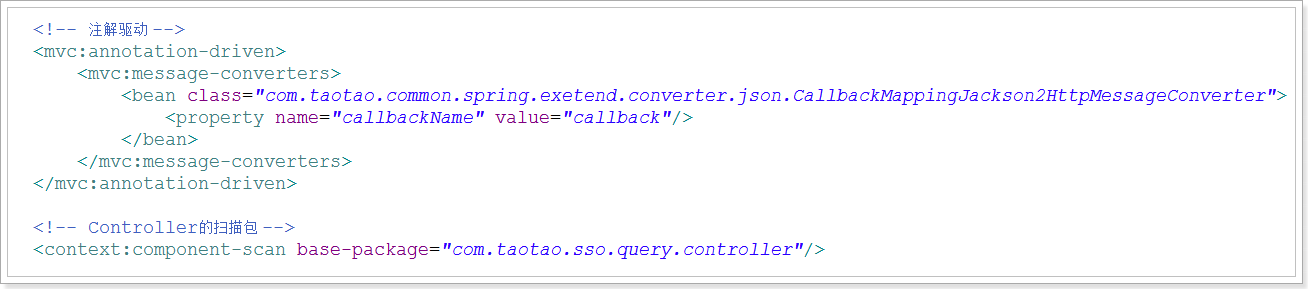
applicationContext-redis.xml：



applicationContext.xml：



taotao-sso-query-servlet.xml：



### 编写web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>taotao-sso-query-service</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/applicationContext\*.xml</param-value>

</context-param>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 编码过滤器，以UTF8编码 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置SpringMVC框架入口 -->

<servlet>

<servlet-name>taotao-sso</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/taotao-sso-query-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>taotao-sso</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

## 编写UserQueryService的实现

**package** com.taotao.sso.query.service;

**import** org.apache.commons.lang3.StringUtils;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** com.taotao.common.service.RedisService;

**import** com.taotao.sso.query.api.UserQueryService;

**import** com.taotao.sso.query.bean.User;

@Service

**public** **class** UserQueryServiceImpl **implements** UserQueryService {

@Autowired

**private** RedisService redisService;

**private** **static** **final** String ***REDIS\_KEY*** = "TOKEN\_";

**private** **static** **final** Integer ***REDIS\_TIME*** = 60 \* 30;

**private** **static** **final** ObjectMapper ***MAPPER*** = **new** ObjectMapper();

@Override

**public** User queryUserByToken(String token) {

String key = ***REDIS\_KEY*** + token;

**try** {

String jsonData = **this**.redisService.get(key);

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

// 登录超时

**return** **null**;

}

// 重新刷新用户的生存时间

**this**.redisService.expire(key, ***REDIS\_TIME***);

**return** ***MAPPER***.readValue(jsonData, User.**class**);

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **null**;

}

}

## 实现RESTful web service

**package** com.taotao.sso.query.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** com.taotao.sso.query.api.UserQueryService;

**import** com.taotao.sso.query.bean.User;

@RequestMapping("user")

@Controller

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

@Autowired

**private** UserQueryService userQueryService;

/\*\*

\* 根据token查询用户信息

\*

\* **@param** token

\* **@return**

\*/

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

**public** ResponseEntity<User> queryUserByToken(@PathVariable("token") String token) {

**try** {

User user = **this**.userQueryService.queryUserByToken(token);

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

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

}

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

} **catch** (Exception e) {

e.printStackTrace();

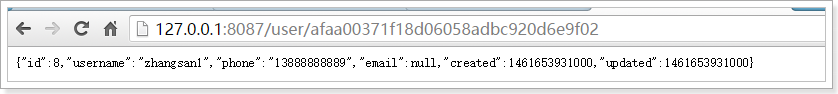
}

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

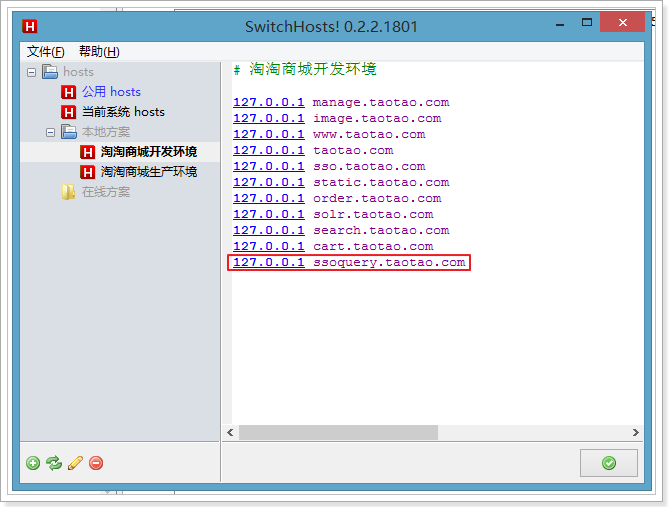
}

}

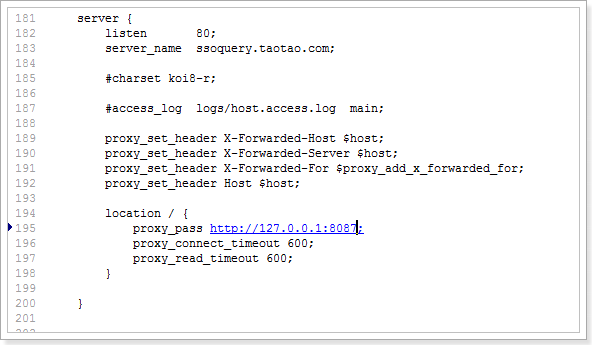
### 测试



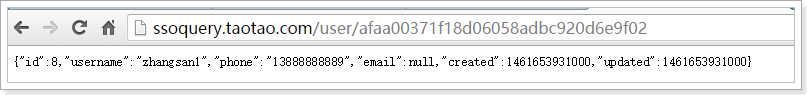
### 修改为域名访问



配置nginx：

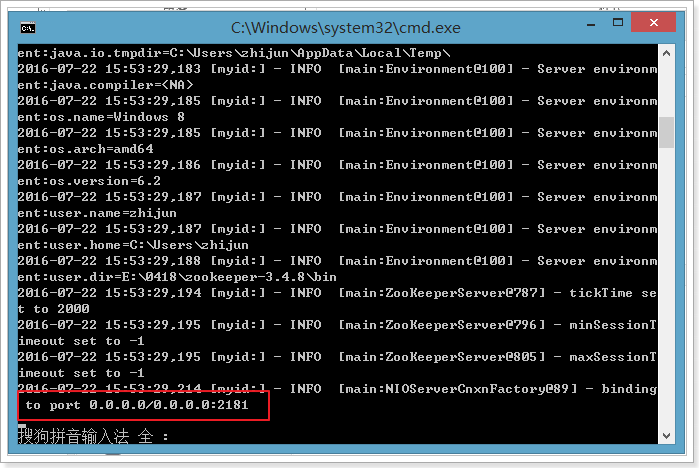


测试：



## 将查询服务发布到dubbo中

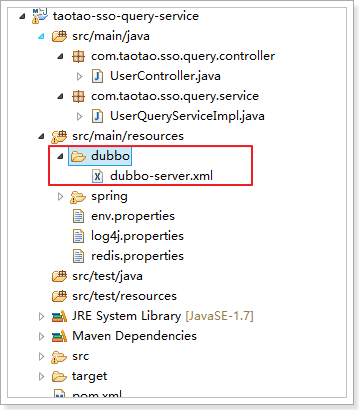
### 启动zookeeper服务



### 导入zookeeper客户端依赖



### 编写dubbo配置文件



<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name=*"dubbo-sso-query-server"* />

<!-- 这里使用的注册中心是zookeeper -->

<dubbo:registry address=*"zookeeper://127.0.0.1:2181"* client=*"zkclient"*/>

<!-- 用dubbo协议在20880端口暴露服务 -->

<dubbo:protocol name=*"dubbo"* port=*"20880"* />

<!-- 将该接口暴露到dubbo中 -->

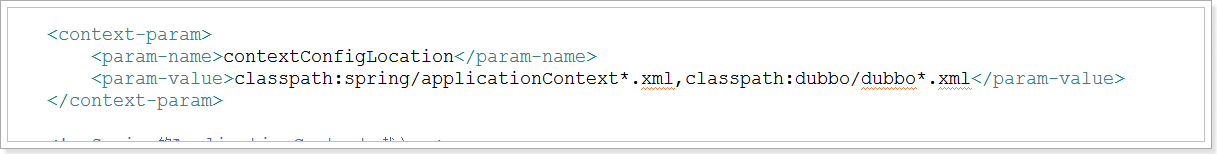
<dubbo:service interface=*"com.taotao.sso.query.api.UserQueryService "* ref=*"userQueryServiceImpl"* />

<!-- 将具体的实现类加入到Spring容器中 -->

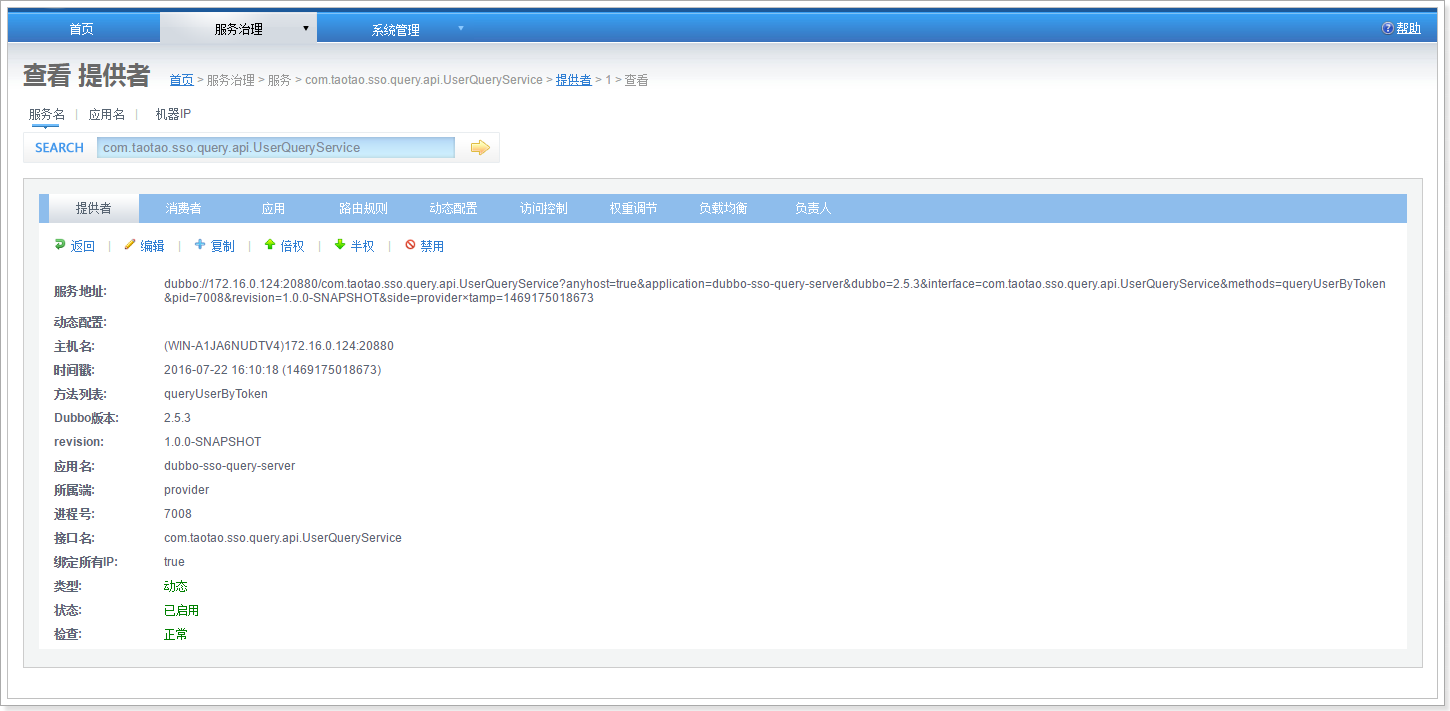
<bean id=*"userQueryServiceImpl"* class=*"com.taotao.sso.query.service.UserQueryServiceImpl"* />

</beans>

### 在web.xml中配置Spring扫描dubbo配置文件



## 启动tomcat并且测试



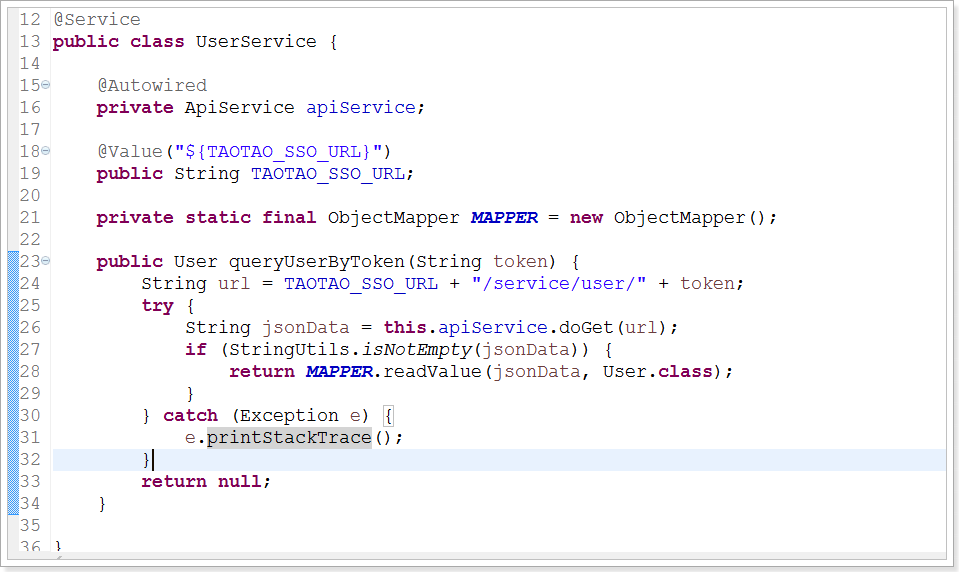
## 禁用原有的taotao-sso中的服务



## 改造前台系统的调用方式

### 之前调用方式

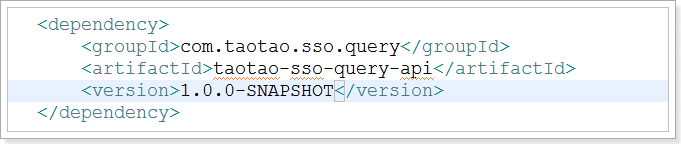
之前前台系统是通过Httpclient方式调用sso系统的restful接口，现在我们改造成dubbo调用方式。



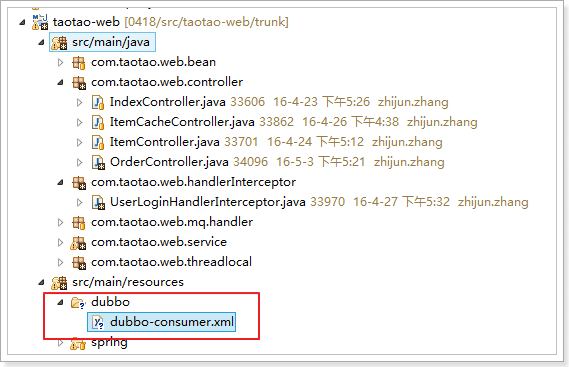
### 加入dubbo依赖



### 导入api依赖



### 编写dubbo配置文件



<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name=*"dubbo-web-consumer"* />

<!-- 这里使用的注册中心是zookeeper -->

<dubbo:registry address=*"zookeeper://127.0.0.1:2181"* client=*"zkclient"*/>

<!-- 用dubbo协议在20880端口暴露服务 -->

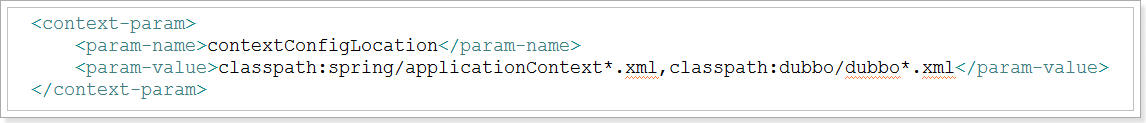
<dubbo:protocol name=*"dubbo"* port=*"20880"* />

<!-- 从注册中心中查找服务 -->

<dubbo:reference id=*"userQueryService"* interface=*"com.taotao.sso.query.api.UserQueryService"*/>

</beans>

### 在web.xml中配置Spring容器扫描dubbo



### 测试

功能一切正常。

### 作业

1. 废弃原有sso系统中的查询功能；
2. 改造所有系统中的调用方式；