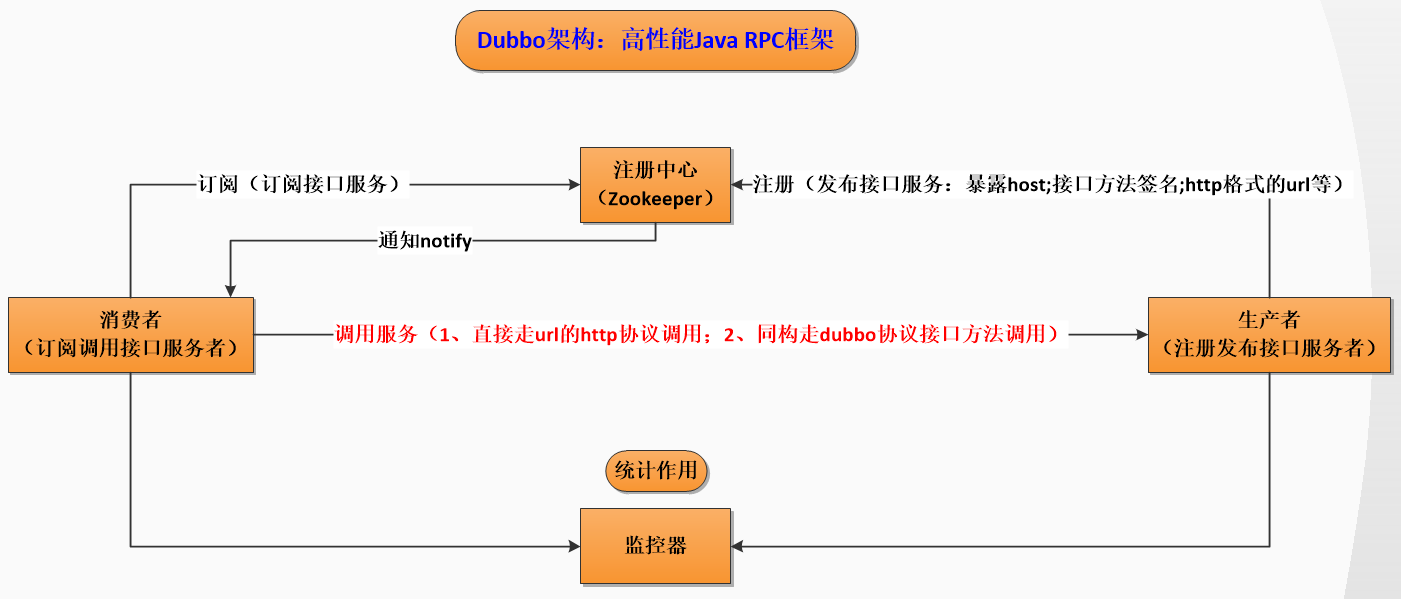
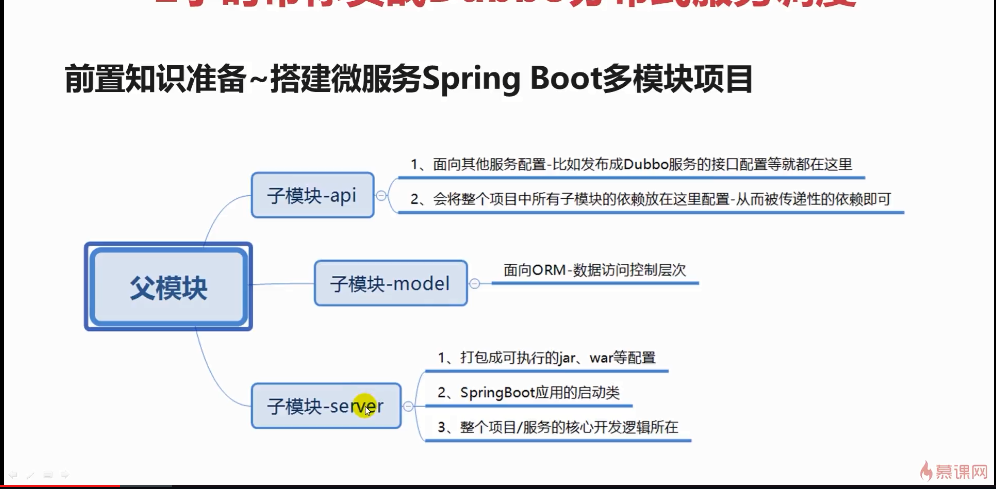
# springboot多模块构建





## 首先将父模块xml的包改成pom 如：

<packaging>pom</packaging>

新建三个模块，本例中为api>model>server依次依赖，在父xml可看到

<modules>  
 <module>api</module>  
 <module>model</module>  
 <module>server</module>  
</modules>

在三个子模块中添加依赖关系：（model，server中依赖）

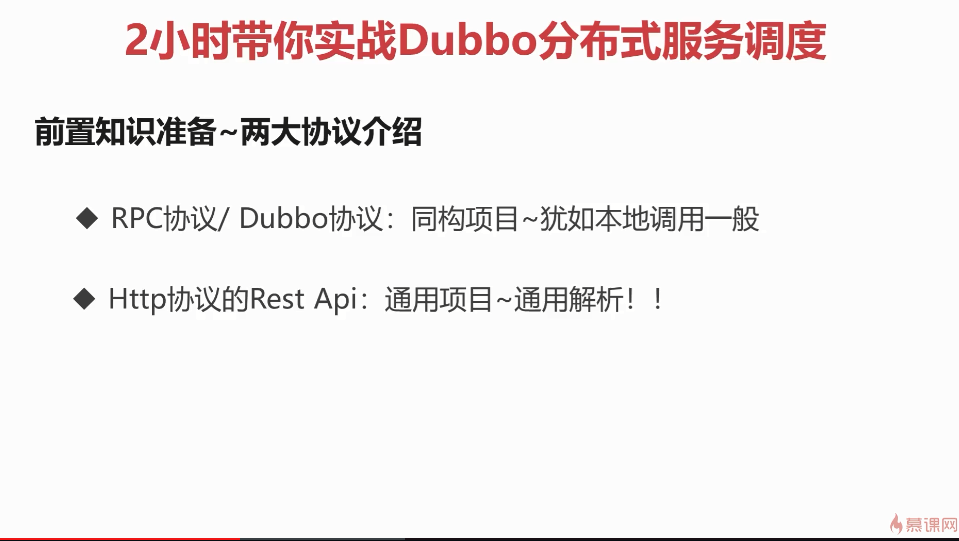
<dependency>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <artifactId>api</artifactId>  
 <version>${project.parent.version}</version>  
</dependency>

<!--model-->  
<dependency>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <artifactId>model</artifactId>  
 <version>${project.parent.version}</version>  
</dependency>

## 基本结构的多模块项目构建完成，可以删除dubboone的src目录

# 创建服务提供商模块dubboone





## 在上述多模块项目框架构建完成后引入相关依赖及完成配置文件：

### api pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<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">  
 <parent>  
 <artifactId>dubboone</artifactId>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <version>1.0.1</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>api</artifactId>  
  
 <properties>  
 <mybatis-pagehelper.version>4.1.2</mybatis-pagehelper.version>  
 <lombok.version>1.16.10</lombok.version>  
 </properties>  
  
 <dependencies>  
 <!--java校验 跟 hibernate校验-->  
 <dependency>  
 <groupId>javax.validation</groupId>  
 <artifactId>validation-api</artifactId>  
 <version>1.1.0.Final</version>  
 </dependency>  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-validator</artifactId>  
 <version>5.3.5.Final</version>  
 </dependency>  
  
 <!--for page-->  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>${mybatis-pagehelper.version}</version>  
 </dependency>  
  
 <!--lombok-->  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>${lombok.version}</version>  
 </dependency>  
  
 </dependencies>  
  
  
</project>

### model pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<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">  
 <parent>  
 <artifactId>dubboone</artifactId>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <version>1.0.1</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>model</artifactId>  
 <properties>  
 <mybatis-spring-boot.version>1.1.1</mybatis-spring-boot.version>  
 <mybatis-pagehelper.version>4.1.2</mybatis-pagehelper.version>  
 </properties>  
  
 <dependencies>  
 <!--api-->  
 <dependency>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <artifactId>api</artifactId>  
 <version>${project.parent.version}</version>  
 </dependency>  
  
 <!--spring-mybatis-->  
 <dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>${mybatis-spring-boot.version}</version>  
 </dependency>  
  
 </dependencies>  
  
</project>

Server pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<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">  
 <parent>  
 <artifactId>dubboone</artifactId>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <version>1.0.1</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>server</artifactId>  
 <packaging>jar</packaging>  
 <properties>  
 <start-class>com.cjy.dubbo.one.server.BootMoreApplication</start-class>  
  
 <!--<spring-boot.version>1.3.3.RELEASE</spring-boot.version>-->  
 <spring-boot.version>2.0.5.RELEASE</spring-boot.version>  
  
 <spring-session.version>1.2.0.RELEASE</spring-session.version>  
 <slf4j.version>1.7.13</slf4j.version>  
 <log4j.version>1.2.17</log4j.version>  
 <mysql.version>5.1.37</mysql.version>  
 <druid.version>1.0.16</druid.version>  
 <guava.version>19.0</guava.version>  
 <joda-time.version>2.9.2</joda-time.version>  
 <cglib.version>3.1</cglib.version>  
  
 <zookeeper.version>3.4.10</zookeeper.version>  
 <curator.version>2.12.0</curator.version>  
 <dubbo.version>2.8.4</dubbo.version>  
  
 <resteasy.version>3.0.7.Final</resteasy.version>  
 <okhttp.version>3.1.2</okhttp.version>  
 <gson.version>2.6.1</gson.version>  
 <httpclient.version>4.3.6</httpclient.version>  
  
 </properties>  
  
 <!-- 依赖管理 -->  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-dependencies</artifactId>  
 <version>${spring-boot.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
 <dependencies>  
  
 <!--model-->  
 <dependency>  
 <groupId>com.cjy.demo.dubboone</groupId>  
 <artifactId>model</artifactId>  
 <version>${project.parent.version}</version>  
 </dependency>  
  
 <!--log-->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>${slf4j.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>${slf4j.version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>${log4j.version}</version>  
 </dependency>  
  
 <!--guava-->  
 <dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>${guava.version}</version>  
 </dependency>  
  
 <!-- time -->  
 <dependency>  
 <groupId>joda-time</groupId>  
 <artifactId>joda-time</artifactId>  
 <version>${joda-time.version}</version>  
 </dependency>  
  
 <!--mysql-->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>${mysql.version}</version>  
 </dependency>  
  
 <!--druid-->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>${druid.version}</version>  
 </dependency>  
  
 <!--spring-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 <version>${spring-boot.version}</version>  
 <exclusions>  
 <exclusion>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.slf4j</groupId>  
 <artifactId>log4j-over-slf4j</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 <!-- jsp 支持 -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-tomcat</artifactId>  
 <!--<scope>provided</scope>-->  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.tomcat.embed</groupId>  
 <artifactId>tomcat-embed-jasper</artifactId>  
 <!--<scope>provided</scope>-->  
 </dependency>  
  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 </dependency>  
  
 <!-- dubbo -->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>dubbo</artifactId>  
 <version>${dubbo.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>curator-framework</artifactId>  
 <groupId>org.apache.curator</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>curator-client</artifactId>  
 <groupId>org.apache.curator</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>guava</artifactId>  
 <groupId>com.google.guava</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>spring-web</artifactId>  
 <groupId>org.springframework</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 <dependency>  
 <groupId>cglib</groupId>  
 <artifactId>cglib</artifactId>  
 <version>${cglib.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>asm</artifactId>  
 <groupId>org.ow2.asm</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 <!-- zookeeper start -->  
 <dependency>  
 <groupId>org.apache.zookeeper</groupId>  
 <artifactId>zookeeper</artifactId>  
 <version>${zookeeper.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>slf4j-log4j12</artifactId>  
 <groupId>org.slf4j</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.curator</groupId>  
 <artifactId>curator-framework</artifactId>  
 <version>${curator.version}</version>  
 </dependency>  
  
 <!--dubbo rest-->  
 <dependency>  
 <groupId>org.jboss.resteasy</groupId>  
 <artifactId>resteasy-jaxrs</artifactId>  
 <version>${resteasy.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>httpclient</artifactId>  
 <groupId>org.apache.httpcomponents</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.jboss.resteasy</groupId>  
 <artifactId>resteasy-netty</artifactId>  
 <version>${resteasy.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.jboss.resteasy</groupId>  
 <artifactId>resteasy-client</artifactId>  
 <version>${resteasy.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.jboss.resteasy</groupId>  
 <artifactId>resteasy-jackson-provider</artifactId>  
 <version>${resteasy.version}</version>  
 </dependency>  
  
  
 <!--for test-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- gson -->  
 <dependency>  
 <groupId>com.google.code.gson</groupId>  
 <artifactId>gson</artifactId>  
 <version>${gson.version}</version>  
 </dependency>  
  
 <!-- okhttp -->  
 <dependency>  
 <groupId>com.squareup.okhttp3</groupId>  
 <artifactId>okhttp</artifactId>  
 <version>${okhttp.version}</version>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <finalName>dubboOne-${project.parent.version}</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <version>${spring-boot.version}</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>repackage</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>2.4</version>  
 <configuration>  
 <failOnMissingWebXml>false</failOnMissingWebXml>  
 </configuration>  
 </plugin>  
 </plugins>  
  
 <resources>  
 <resource>  
 <directory>src/main/resources</directory>  
 <filtering>true</filtering>  
 </resource>  
 </resources>  
 </build>  
  
</project>

### application.properties

#profile  
#spring.profiles.active=production  
#spring.profiles.active=local  
  
server.port=8093  
#server.context-path=/dubboOne  
server.servlet.context-path=/dubboOne  
  
#logging  
logging.path=D:\\work\\github\\log\\dubboone  
logging.file=dubboOne  
  
logging.level.org.springframework = INFO  
logging.level.com.fasterxml.jackson = INFO  
logging.level.com.debug.mooc.dubbo.one = debug  
  
#json serialization  
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss  
spring.jackson.time-zone=GMT+8  
  
#spring.datasource.initialize=false  
spring.datasource.initialization-mode=never  
spring.jmx.enabled=false  
  
#\u6570\u636E\u6E90\u914D\u7F6E  
datasource.url=jdbc:mysql://127.0.0.1:3306/dubbo\_one?useUnicode=true&amp;characterEncoding=utf-8&zeroDateTimeBehavior=convertToNull  
datasource.username=root  
datasource.password=123456  
  
#mybatis  
mybatis.config-location=classpath:mybatis-config.xml  
mybatis.checkConfigLocation = true  
mybatis.mapper-locations=classpath:mappers/\*.xml  
  
  
#dubbo zookeeper\u914D\u7F6E\u4FE1\u606F  
dubbo.registry.address=zookeeper://127.0.0.1:2181

### mybatis-config.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
  
 <settings>  
 <!-- Globally enables or disables any caches configured in any mapper under this configuration -->  
 <setting name="cacheEnabled" value="true"/>  
 <!-- Sets the number of seconds the driver will wait for a response from the database -->  
 <setting name="defaultStatementTimeout" value="3000"/>  
 <!-- Enables automatic mapping from classic database column names A\_COLUMN to camel case classic Java property names aColumn -->  
 <setting name="mapUnderscoreToCamelCase" value="true"/>  
 <!-- Allows JDBC support for generated keys. A compatible driver is required.  
 This setting forces generated keys to be used if set to true,  
 as some drivers deny compatibility but still work -->  
 <setting name="useGeneratedKeys" value="true"/>  
 <!-- 设置控制台打印sql -->  
 <!--<setting name="logImpl" value="stdout\_logging" />-->  
 </settings>  
  
 <!-- Continue going here -->  
  
 <!--<typeAliases>-->  
 <!--<package name="cn.com.countrygarden.ucenter.entity"/>-->  
 <!--</typeAliases>-->  
 <plugins>  
 <!-- com.github.pagehelper为PageHelper类所在包名 -->  
 <plugin interceptor="com.github.pagehelper.PageHelper">  
 <!-- 4.0.0以后版本可以不设置该参数 -->  
 <property name="dialect" value="mysql"/>  
 <!-- 该参数默认为false -->  
 <!-- 设置为true时，会将RowBounds第一个参数offset当成pageNum页码使用 -->  
 <!-- 和startPage中的pageNum效果一样-->  
 <property name="offsetAsPageNum" value="true"/>  
 <!-- 该参数默认为false -->  
 <!-- 设置为true时，使用RowBounds分页会进行count查询 -->  
 <property name="rowBoundsWithCount" value="true"/>  
 <!-- 设置为true时，如果pageSize=0或者RowBounds.limit = 0就会查询出全部的结果 -->  
 <!-- （相当于没有执行分页查询，但是返回结果仍然是Page类型）-->  
 <property name="pageSizeZero" value="true"/>  
 <!-- 3.3.0版本可用 - 分页参数合理化，默认false禁用 -->  
 <!-- 启用合理化时，如果pageNum<1会查询第一页，如果pageNum>pages会查询最后一页 -->  
 <!-- 禁用合理化时，如果pageNum<1或pageNum>pages会返回空数据 -->  
 <property name="reasonable" value="false"/>  
 <!-- 3.5.0版本可用 - 为了支持startPage(Object params)方法 -->  
 <!-- 增加了一个`params`参数来配置参数映射，用于从Map或ServletRequest中取值 -->  
 <!-- 可以配置pageNum,pageSize,count,pageSizeZero,reasonable,orderBy,不配置映射的用默认值 -->  
 <!-- 不理解该含义的前提下，不要随便复制该配置 -->  
 <!--<property name="params" value="pageNum=pageHelperStart;pageSize=pageHelperRows;"/>-->  
 <!-- 支持通过Mapper接口参数来传递分页参数 -->  
 <property name="supportMethodsArguments" value="false"/>  
 <!-- always总是返回PageInfo类型,check检查返回类型是否为PageInfo,none返回Page -->  
 <property name="returnPageInfo" value="none"/>  
 </plugin>  
 </plugins>  
  
 <!--<mappers>-->  
 <!--&lt;!&ndash;<package name="cn.com.countrygarden.ucenter.mapper" />&ndash;&gt;-->  
 <!--<mapper resource="mappers/DiaryMapper.xml"/>-->  
 <!--<mapper resource="mappers/EmpEvaluateMapper.xml"/>-->  
 <!--<mapper resource="mappers/EmployeeMapper.xml"/>-->  
 <!--<mapper resource="mappers/LikesMapper.xml"/>-->  
 <!--<mapper resource="mappers/PicMapper.xml"/>-->  
 <!--<mapper resource="mappers/RepliesMapper.xml"/>-->  
 <!--<mapper resource="mappers/TeamMapper.xml"/>-->  
 <!--<mapper resource="mappers/UserMapper.xml"/>-->  
 <!--</mappers>-->  
</configuration>

### spring-dubbo.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 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.xsd  
  
 http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">  
  
 <!--发布出去的dubbo服务类所在包的注解-->  
 <dubbo:annotation package="com.cjy.dubbo.one.server.service.dubbo" />  
  
 <!--注册中心zookeeper配置信息-->  
 <dubbo:registry address="${dubbo.registry.address}" />  
  
 <!--支持两种协议的调用：rpc-dubbo协议；http协议-rest api-url调用-->  
 <dubbo:protocol name="rest" threads="500" contextpath="v1" server="tomcat" accepts="500"/>  
 <dubbo:protocol name="dubbo" />  
  
 <!--消费方信息配置-->  
 <dubbo:application name="dubboone-consume" owner="debug" organization="dubbox"/>  
  
  
</beans>

### spring-jdbc.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  
 " >  
  
 <!--dubboOne -->  
 <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close" primary="true" >  
 <!-- 基本属性 url、user、password -->  
 <property name="url" value="${datasource.url}" />  
 <property name="username" value="${datasource.username}" />  
 <property name="password" value="${datasource.password}" />  
  
 <!-- 配置初始化大小、最小、最大 -->  
 <property name="initialSize" value="10" />  
 <property name="minIdle" value="10" />  
 <property name="maxActive" value="20" />  
  
 <!-- 配置获取连接等待超时的时间 -->  
 <property name="maxWait" value="60000" />  
  
 <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->  
 <property name="timeBetweenEvictionRunsMillis" value="60000" />  
  
 <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->  
 <property name="minEvictableIdleTimeMillis" value="300000" />  
  
 <property name="validationQuery" value="SELECT 1 " />  
 <property name="testWhileIdle" value="true" />  
 <property name="testOnBorrow" value="false" />  
 <property name="testOnReturn" value="false" />  
  
 <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->  
 <property name="poolPreparedStatements" value="true" />  
 <property name="maxPoolPreparedStatementPerConnectionSize" value="20" />  
  
 <!-- 配置监控统计拦截的filters，去掉后监控界面sql无法统计 -->  
 <property name="filters" value="stat" />  
 </bean>  
  
 <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource"/>  
 </bean>  
  
</beans>

## 创建项目测试类测试项目是否能正常启动

在api模块里面新建两个公共类：

package com.cjy.dubbo.one.api.enums;*/\*\*  
 \* Created by Administrator on 2019/1/12.  
 \*/  
  
/\*\*  
 \** ***@Author:debug*** *(SteadyJack)  
 \** ***@Date:*** *2019/1/12 14:01  
 \*\*/*public enum StatusCode {  
  
 *Success*(0,"成功"),  
 *Fail*(-1,"失败"),  
 *InvalidParams*(200,"无效的参数"),  
 *ItemNotExist*(201,"商品不存在!");  
  
 private Integer code;  
 private String msg;  
  
 StatusCode(Integer code, String msg) {  
 this.code = code;  
 this.msg = msg;  
 }  
  
 public Integer getCode() {  
 return code;  
 }  
  
 public void setCode(Integer code) {  
 this.code = code;  
 }  
  
 public String getMsg() {  
 return msg;  
 }  
  
 public void setMsg(String msg) {  
 this.msg = msg;  
 }  
}

package com.cjy.dubbo.one.api.response;*/\*\*  
 \* Created by Administrator on 2019/1/12.  
 \*/*import com.cjy.dubbo.one.api.enums.StatusCode;  
  
import java.io.Serializable;  
  
*/\*\*  
 \** ***@Author:debug*** *(SteadyJack)  
 \** ***@Date:*** *2019/1/12 13:59  
 \*\*/*public class BaseResponse<T> implements Serializable{  
  
 private Integer code;  
 private String msg;  
 private T data;  
  
 public BaseResponse() {  
 }  
  
 public BaseResponse(Integer code, String msg, T data) {  
 this.code = code;  
 this.msg = msg;  
 this.data = data;  
 }  
  
 public BaseResponse(Integer code, String msg) {  
 this.code = code;  
 this.msg = msg;  
 }  
  
 public BaseResponse(StatusCode statusCode) {  
 this.code = statusCode.getCode();  
 this.msg = statusCode.getMsg();  
 }  
  
 public Integer getCode() {  
 return code;  
 }  
  
 public void setCode(Integer code) {  
 this.code = code;  
 }  
  
 public String getMsg() {  
 return msg;  
 }  
  
 public void setMsg(String msg) {  
 this.msg = msg;  
 }  
  
 public T getData() {  
 return data;  
 }  
  
 public void setData(T data) {  
 this.data = data;  
 }  
}

在server模块创建BaseController

package com.cjy.dubbo.one.server.controller;*/\*\*  
 \* Created by Administrator on 2019/1/12.  
 \*/*import com.cjy.dubbo.one.api.enums.StatusCode;  
import com.cjy.dubbo.one.api.response.BaseResponse;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
*/\*\*  
 \** ***@Author:debug*** *(SteadyJack)  
 \** ***@Date:*** *2019/1/12 14:02  
 \*\*/*@RestController  
public class BaseController {  
  
 private static final Logger *log*= LoggerFactory.*getLogger*(BaseController.class);  
  
 private static final String *prefix*="base";  
  
 */\*\*  
 \* 测试的hello world  
 \** ***@param*** *param  
 \** ***@return*** *\*/* @RequestMapping(value = *prefix*+"/one",method = RequestMethod.*GET*)  
 public BaseResponse one(@RequestParam String param){  
 BaseResponse response=new BaseResponse(StatusCode.*Success*);  
 try {  
 response.setData(param);  
  
 }catch (Exception e){  
 e.printStackTrace();  
 response=new BaseResponse(StatusCode.*Fail*);  
 }  
 return response;  
 }  
  
}

## 启动项目后通过该路径发起请求测试：[http://localhost:8093/dubboOne/base/one?param="hello,world](http://localhost:8093/dubboOne/base/one?param=\"hello,world)"

# 项目搭建终于到的坑

项目启动报错：

无法读取文档'http://code.alibabatech.com/schema/dubbo/dubbo.xsd'

，解决办法是把xml里面的dubbo版本号改一下，改成2.6.0就可以了

# 业务实战

