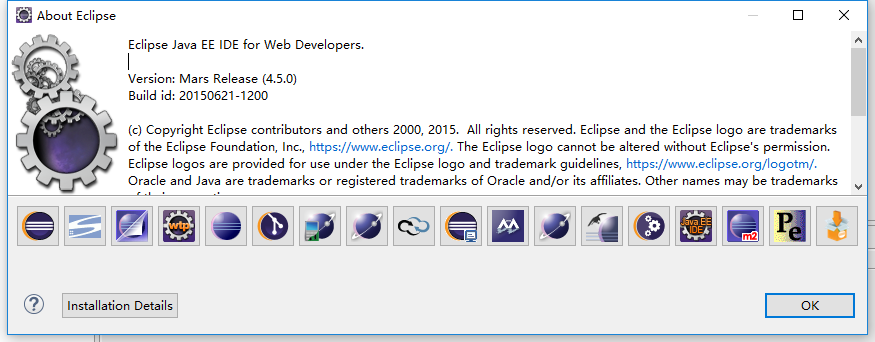
分布式系统架构解决方案—Dubbo

本文使用开发软件：



# 概述

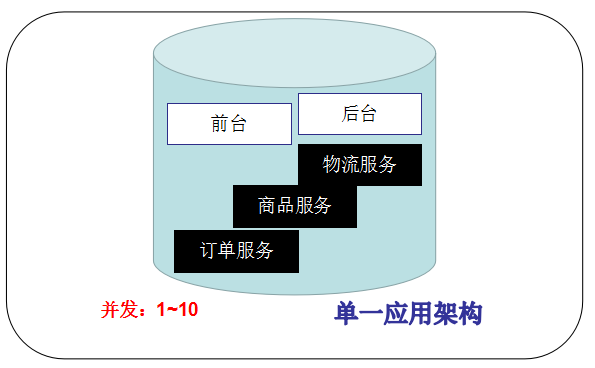
## 什么是SOA？

**S**ervice **O**riented **A**rchitecture 支持面向服务的架构

### 1.1单一应用架构

当网站流量很小时，只需要一个应用，将所有的功能部署到一起，从而减少部署节点和成本；

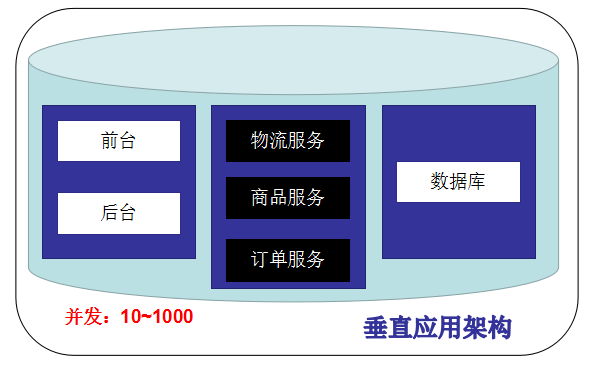
此时，用于简化 增删改查 工作量的数据访问框架 （**ORM**）是关键；



### 1.2垂直应用架构

当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成几个互不相干的几个应用,以提高效率；

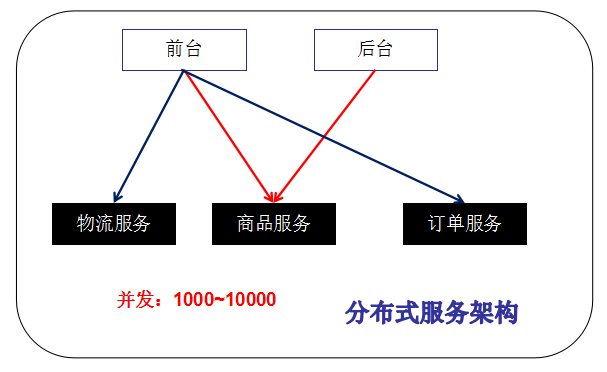
此时，用于加速前端页面开发的web框架（**MVC**）是关键；



### 1.3分布式服务架构

当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的业务，逐渐形成稳健的服务中心，使前端应用能更快速的响应多变的市场需求；

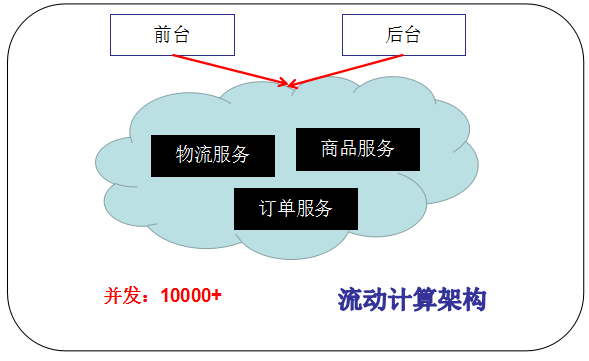
此时，用户提高业务复用及整合的分布式服务框架（**RPC**）远程调用是关键；



### 1.4流动计算架构

当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐呈现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率；

此时，用于提高机器利用率的资源调度和治理中心（**SOA**）是关键；



## Dubbo简介

* Dubbo是分布式服务框架，是阿里巴巴的开源项目，被国内电商及互联网项目广泛应用；
* Dubbo致力于提高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案；
* 简单来说，dubbo是个服务框架，如果没有分布式的需求，是不需要用的；
* 只有分布式的需求，才会用到dubbo；
* 说白了，就是个远程服务调用的分布式框架；

### 2.1架构

### 2.2节点角色

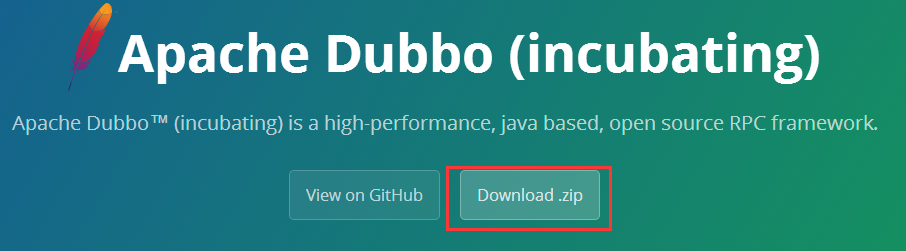
|  |  |
| --- | --- |
| **节点** | **角色说明** |
| Provider | 暴露服务的服务方 |
| Consumer | 调用远程服务的消费方 |
| Registry | 服务注册与发现的注册中心 |
| Monitor | 统计服务的调用次调和调用时间的监控中心 |
| Container | 服务运行容器 |

### 2.3调用关系

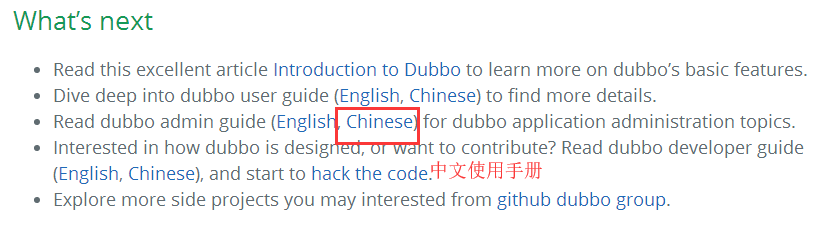
1. 服务容器负责启动，加载，运行服务提供者；
2. 服务提供者在启动时，向注册中心注册自己提供的服务；
3. 服务消费者在启动时，向注册中心订阅自己所需的服务；
4. 在注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者；
5. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用；
6. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心；

# 快速入门

<http://dubbo.apache.org/>



首页下滚，找到连接。



详细操作在其中



## 注册中心

### Zookeeper介绍

* 官方推荐使用zookeeper注册中心；
* 注册中心负责服务地址的注册与查找，相当于目录服务；
* 服务提供者和消费者只在启动时与注册中心交互，注册中不转发请求，压力较小；
* Zookeeper是apache hadoop的子项目，是一个树形的目录服务，支持变更推送，适合作为dubbox的服务注册中心，工业强度较高，可用于生产环境；
* **duboo是烤串店，zookeeper就是烧烤一条街；**

### 1.2下载

各种版本下载：<http://archive.apache.org/dist/zookeeper/>

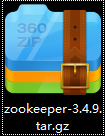


### 1.3 安装到linux

#### 安装jdk

省略。。。

#### 上传zookeeper压缩包到linux系统

通过ftp上传到linux

或者

|  |
| --- |
| wget http://archive.apache.org/dist/zookeeper/zookeeper-3.3.3/zookeeper-3.3.3.tar.gz |
|  |

#### 解压tar包

|  |
| --- |
| tar -zxvf zookeeper-3.3.3.tar.gz |
|  |

#### 进入zookeeper目录,创建data目录

|  |
| --- |
| mkdir data |
|  |

#### 修改zoo\_sample.cfg为zoo.cfg

|  |
| --- |
| cd conf  mv zoo\_sample.cfg zoo.cfg |
|  |

#### 配置zookeeper

主要配置**dataDir**和**clientPort**参数，clientPort参数使用默认值2181即可；

注意：dataDir目录中的data末级目录如果不存在,也会自动生成；

|  |
| --- |
|  |

1. 指定zookeeper的数据目录
2. 指定客户端连接的端口
3. zookeeper集群时候的配置
   1. 【server.1】中的1表示集群中集群编号
   2. 【slave1】集群中机器的域名（可以用ip）
   3. 【2888】集群内机器通信的端口
   4. 【3888】集群选举时使用的端口

#### 关闭防火墙

|  |
| --- |
| systemctl stop firewalld.service |
|  |

### 1.4启动Zookeeper

#### 1.4.1启动

|  |
| --- |
| ./zkServer.sh start |
|  |

#### 1.4.2查看运行状态

|  |
| --- |
| zkServer.sh status |
|  |

#### 1.4.3查看zookeeper的端口是否运行

|  |
| --- |
| netstat -apn | grep 2181 |
|  |

#### 1.4.4连接zookeeper

|  |
| --- |
| sh zkCli.sh -server 127.0.0.1:2181 |
|  |

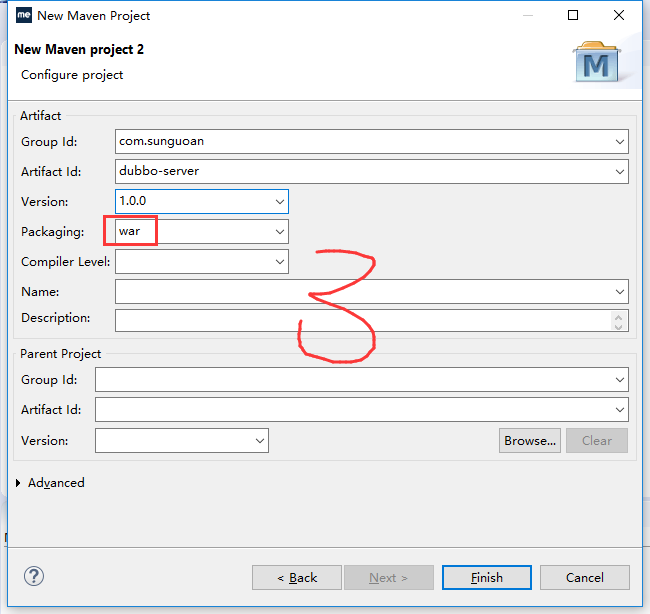
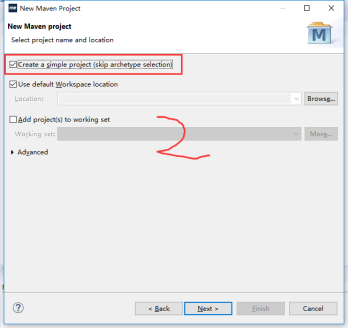
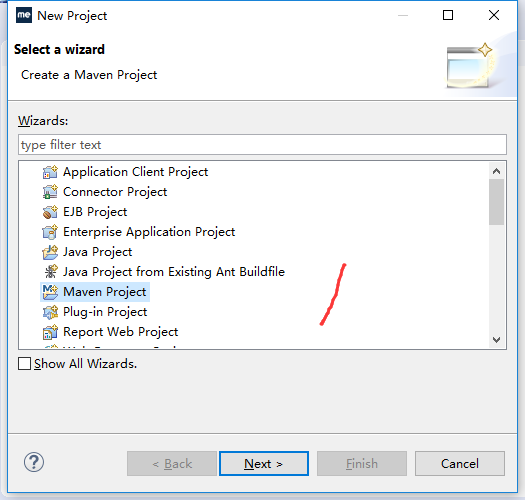
#### 1.4.5停止服务

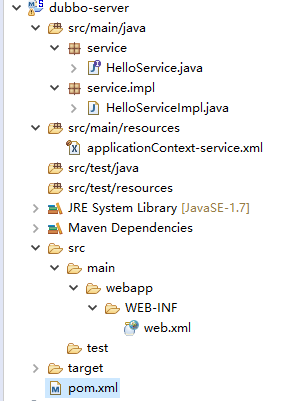
|  |
| --- |
| ./zkServer.sh stop |
|  |

## 服务提供方

创建mavenproject（本次使用myeclipse，将dubbo和spring整合进行测试）

IDEA会在springBoot中使用





### 2.1服务方的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/maven-v4\_0\_0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.sunguoan</groupId>  <artifactId>dubbo-server</artifactId>  <version>1.0.0</version>  <packaging>war</packaging>  <properties>  <spring.version>5.0.6.RELEASE</spring.version>  <webVersion>3.0</webVersion>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>${spring.version}</version>  </dependency>  <!--dubbo -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.5.7</version>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>3.4.6</version>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  <version>0.1</version>  </dependency>  <dependency>  <groupId>javassist</groupId>  <artifactId>javassist</artifactId>  <version>3.11.0.GA</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.tomcat.maven </groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>**8080**</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

### 2.2服务方接口

|  |
| --- |
| **public** **interface** HelloService {  String sayHello(String name);  } |

### 2.3服务方实现

|  |
| --- |
| @com.alibaba.dubbo.config.annotation.Service  **public** **class** HelloServiceImpl **implements** HelloService{  @Override  **public** String sayHello(String name) {  **return** "Hello,"+name+"!!!";  } |

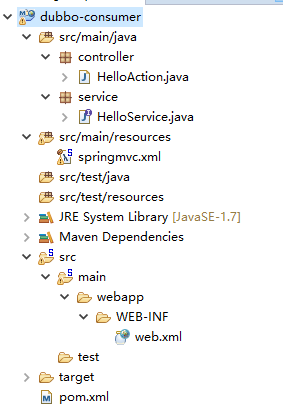
### 2.4服务方的配置文件applicationContext-service.xml

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!--Dubbo的应用名称，通常使用项目名-->  <dubbo:application name=*"dubbo-server"*/>  <!--配置Dubbo的注册中心地址-->  <dubbo:registry address=*"zookeeper://192.168.46.246:2181"*/>  <!--配置Dubbo扫描类，将这个类作为服务进行发布-->  <dubbo:annotation package=*"service.impl"*/>  </beans> |
| <dubbo>标签如果报错，说明加载xsd验证文件失败。  一般情况下，如果有网络，会自动去官网加载完毕，并且自行修改报错；  但，如果不行的话 。我们可以通过一个设置：  xsd文件只要解压dubbo.jar文件，就可以得到 |

### 2.5服务方的web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <display-name>SSM\_CRUD</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <!-- spring -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext-\*.xml</param-value>  </context-param>  </web-app> |

## 服务消费方



### 3.1消费方的pom.xml

|  |
| --- |
| **与服务方一致，只需要修改tomcat的端口为8081** |

### 3.2消费方的Controller

|  |
| --- |
| @RestController  **public** **class** HelloAction {  @com.alibaba.dubbo.config.annotation.Reference  **private** HelloService hs;  @RequestMapping("hello")  **public** String hello(){  **return** hs.sayHello("Tiger");  }  } |

### 3.3消费方的接口

**注意：**

controller中要依赖HelloService，所以我们创建一个接口；

这里是消费方，不需要实现，因为实现会让服务方为我们搞定！

|  |
| --- |
| **public** **interface** HelloService {  String sayHello(String name);  } |

### 3.3消费方的web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <display-name>dubbo-consumer</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharachtersEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharachtersEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>\*.spring</url-pattern>  </servlet-mapping>  </web-app> |

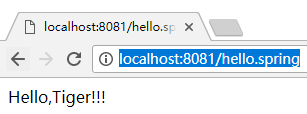
### 3.4消费方的springmvc.xml

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  DF  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!--Dubbo的应用名称，通常使用项目名 -->  <dubbo:application name=*"dubbo-consumer"* />  <!--配置Dubbo的注册中心地址 -->  <dubbo:registry address=*"zookeeper://192.168.46.246:2181"* />  <!--配置Dubbo扫描类，将这个类作为服务进行发布 -->  <dubbo:annotation package=*"controller"* />  </beans> |

## 启动服务测试

首先启动服务方，在启动消费方。

访问：<http://localhost:8081/hello.spring>



# 管理端

## 管理端安装

编译源码，得到war包

资源中有个dubbo-dubbo-2.5.7.zip，这个是dubbo的源码，我们可以使用maven命令编译源码得到“管理端”的war包；

将此包解压，在cmd中进入dubbo-admin目录，输入maven命令

|  |
| --- |
| mvn package -Dmaven.skip.test=true |
| 第一次打包，要下载好多    最后 |

|  |
| --- |
| 上传linux服务器的tomcat下，启动tomcat，即可访问  <http://192.168.46.246:8080/dubbo-admin-2.5.7> |
|  |

## 管理端使用

|  |
| --- |
| 启动服务方，将服务注册到zookeeper |
|  |

|  |
| --- |
| 刷新管理端，服务注册成功，只是没有消费者 |
|  |

|  |
| --- |
| 提供者页面 |
|  |

|  |
| --- |
| 把消费者也运行起来 |
|  |

|  |
| --- |
| 刷新服务，显示正常 |
|  |

|  |
| --- |
| 查看消费者 |
|  |

# 综合案例

## 1需求

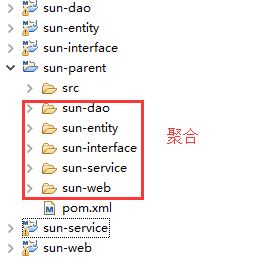
运行SpringMVC+MyBatis+Dubbo分布式实现注册

## 2创建数据库

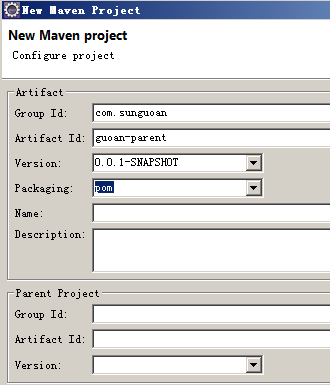
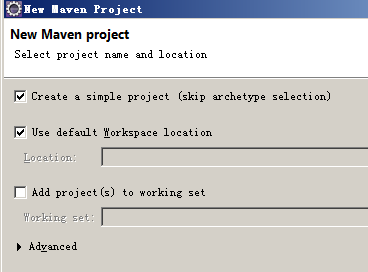
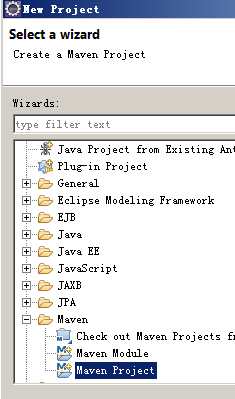
|  |
| --- |
| CREATE DATABASE smd  USE smd  CREATE TABLE users(  uid INT(11) AUTO\_INCREMENT PRIMARY KEY,  username VARCHAR(50) NOT NULL,  PASSWORD VARCHAR(50) NOT NULL,  phone VARCHAR(50) NOT NULL,  createtime VARCHAR(50) NOT NULL  ) |

## 3搭建项目

使用eclipse-mars开发工具，myeclipse2017创建maven项目有问题！



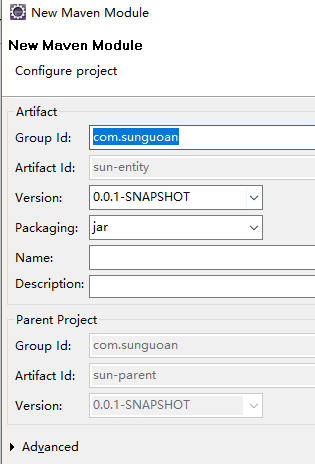
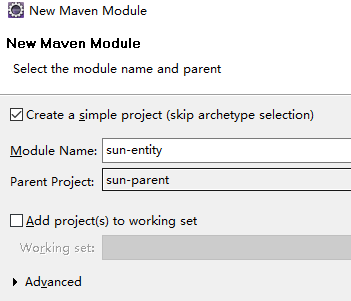
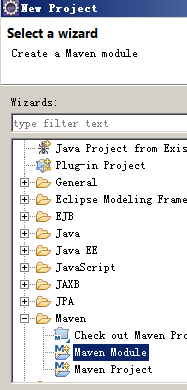
### 3.1父项目-聚合工程

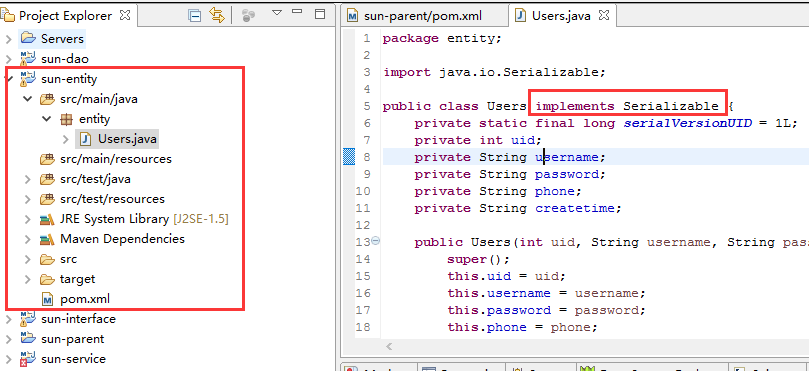


pom.xml

|  |
| --- |
| <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>  <groupId>com.sunguoan</groupId>  <artifactId>sun-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>pom</packaging>  <modules>  <module>sun-web</module>  <module>sun-entity</module>  <module>sun-dao</module>  <module>sun-interface</module>  <module>sun-service</module>  </modules>  <properties>  <spring.version>5.0.6.RELEASE</spring.version>  </properties>  <!-- 所有的依赖(引jar包) -->  <dependencies>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <scope>provided</scope>  <version>2.5</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <scope>provided</scope>  <version>2.0</version>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- Mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.2.8</version>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.2.2</version>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.0.9</version>  </dependency>  <!-- 数据库 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.32</version>  </dependency>  <!--dubbo -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.5.7</version>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>3.4.6</version>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  <version>0.1</version>  </dependency>  <dependency>  <groupId>javassist</groupId>  <artifactId>javassist</artifactId>  <version>3.11.0.GA</version>  </dependency>  <!-- fastjson -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.47</version>  </dependency>  <!-- junit -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>${spring.version}</version>  <scope>test</scope>  </dependency>  </dependencies>  </project> |

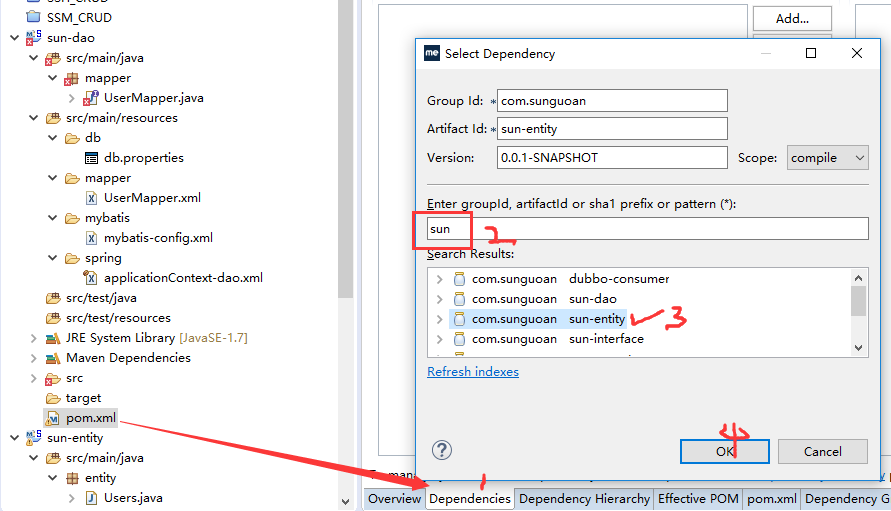
### 3.2模块—实体类



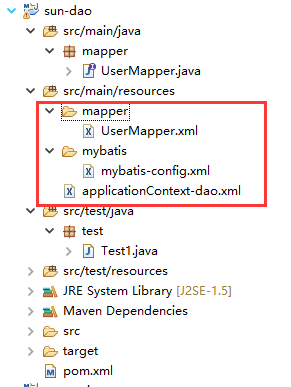


### 3.3模块—dao

goan-dao参考上一步,dao层依赖与实体类，所以，我们应该将sun-entity依赖进来



#### 模块架构



#### pom.xml

|  |
| --- |
| <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.sunguoan</groupId>  <artifactId>sun-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>sun-dao</artifactId>  <dependencies>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-entity</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  <build>  <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.xml</include>  </includes>  </resource>  <resource>  <directory>src/main/resources</directory>  <includes>  <include>\*\*/\*.xml</include>  </includes>  </resource>  </resources>  </build>  </project> |

#### UserMapper.java

|  |
| --- |
| **package** mapper;  **import** org.springframework.stereotype.Service;  **import** entity.Users;  @Service  **public** **interface** UserMapper {  **void** register(Users user);  } |

#### UserMapper.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  <mapper namespace=*"mapper.UserMapper"* >  <insert id=*"register"*>  insert into users (username,password,phone,createtime)  values  (#{username},#{password},#{phone},#{createtime})  </insert>  </mapper> |

#### 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>  <setting name=*"logImpl"* value=*"STDOUT\_LOGGING"* />  </settings>  </configuration> |

#### applicationContext-dao.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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!-- 加载配置文件-->  <bean id=*"datasource"* class=*"com.alibaba.druid.pool.DruidDataSource"* destroy-method=*"close"*>  <property name=*"url"* value=*"jdbc:mysql://192.168.46.246:3306/smd?characterEncoding=utf-8"* />  <property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"username"* value=*"root"* />  <property name=*"password"* value=*"123123"* />  <property name=*"maxActive"* value=*"10"* />  <property name=*"minIdle"* value=*"5"* />  </bean>  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"datasource"* />  <property name=*"configLocation"* value=*"classpath:mybatis/mybatis-config.xml"* />  </bean>  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"mapper"* />  </bean>  </beans> |

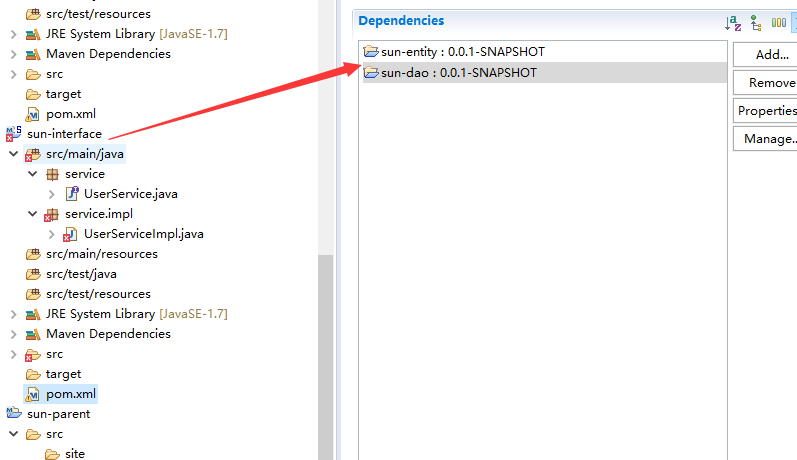
#### Test1.java

|  |
| --- |
| **package** test;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.test.context.ContextConfiguration;  **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  **import** entity.Users;  **import** mapper.UserMapper;  @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations={"classpath:applicationContext-dao.xml"})  **public** **class** Test1 {  @Autowired  **private** UserMapper userMapper;  @Test  **public** **void** test1(){  Users u = **new** Users();  u.setUsername("a3");  u.setPassword("333");  u.setPhone("113");  u.setCreatetime( **new** SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(**new** Date()) );  userMapper.register(u);  }  } |

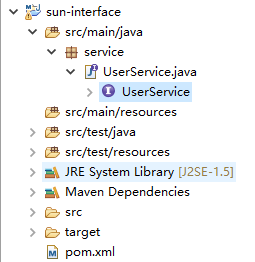
### 3.4模块—接口

sun-interface参考上一步，因为消费方要依赖接口

本层依赖sun-entity和sun-dao



#### 模块架构



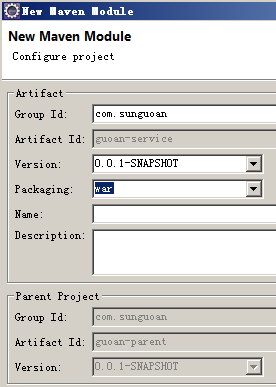
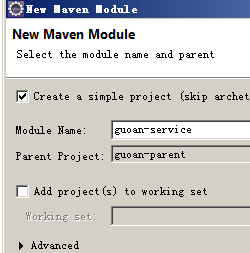
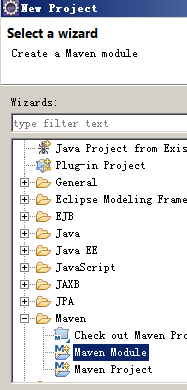
#### UserService.xml

|  |
| --- |
| **package** service;  **import** entity.Users;  **public** **interface** UserService {  **void** register(Users user);  } |

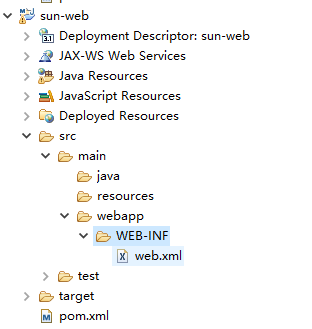
#### pom.xml

|  |
| --- |
| <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.sunguoan</groupId>  <artifactId>sun-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>sun-interface</artifactId>  <dependencies>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-dao</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-entity</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

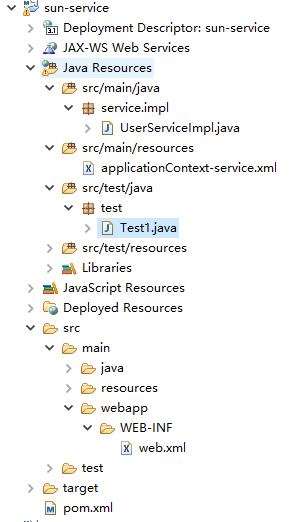
### 3.5模块—service



war模块，需要引入WEB-INF/web.xml解决报错



#### 模块结构



#### pom.xml

|  |
| --- |
| <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.sunguoan</groupId>  <artifactId>sun-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>sun-service</artifactId>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-dao</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-interface</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.tomcat.maven </groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8080</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

#### web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <!-- spring -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath\*:applicationContext\*.xml</param-value>  </context-param>  </web-app> |

#### UserServiceImpl.java

|  |
| --- |
| **package** service.impl;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** com.alibaba.dubbo.config.annotation.Service;  **import** entity.Users;  **import** mapper.UserMapper;  **import** service.UserService;  @Service  **public** **class** UserServiceImpl **implements** UserService {  @Autowired  **private** UserMapper userMapper;    @Override  **public** **void** register(Users user) {  userMapper.register(user);  }  } |

#### applicationContext-service.xml

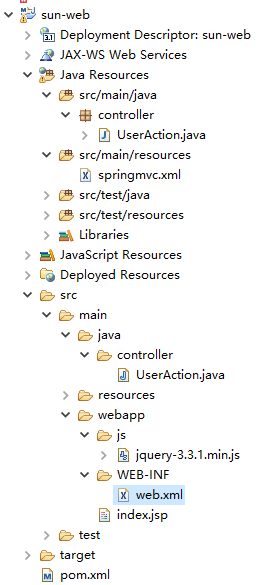
|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>    <!--Dubbo的应用名称，通常使用项目名-->  <dubbo:application name=*"sun-service"*/>  <!--配置Dubbo的注册中心地址-->  <dubbo:registry address=*"zookeeper://192.168.46.246:2181"*/>  <!--配置Dubbo扫描类，将这个类作为服务进行发布-->  <dubbo:annotation package=*"service.impl"*/>  </beans> |

#### Test1.java

|  |
| --- |
| **package** test;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.test.context.ContextConfiguration;  **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  **import** entity.Users;  **import** service.UserService;  @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations = { "classpath:applicationContext-dao.xml",  "classpath:applicationContext-service.xml" })  **public** **class** Test1 {  @Autowired  **private** UserService userService;  @Test  **public** **void** test1() {  Users u = **new** Users();  u.setUsername("a4");  u.setPassword("444");  u.setPhone("114");  u.setCreatetime(**new** SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(**new** Date()));  userService.register(u);  }  } |

### 3.6模块—web

#### 模块结构



#### pom.xml

|  |
| --- |
| <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.sunguoan</groupId>  <artifactId>sun-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>sun-web</artifactId>  <packaging>war</packaging>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.tomcat.maven </groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8081</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  <dependencies>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <type>war</type>  </dependency>  <dependency>  <groupId>com.sunguoan</groupId>  <artifactId>sun-entity</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

#### web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*  id=*"WebApp\_ID"* version=*"3.1"*>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharachtersEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharachtersEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>\*.spring</url-pattern>  </servlet-mapping>  </web-app> |

#### index.jsp

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  </head>  <script src=*"js/jquery-3.3.1.min.js"*></script>  <script>  $(**function**(){  $("#btn").click(**function**(){  $.post("register.spring",$("form").serialize(),**function**(r){  alert(r);  },"json");  });  });  </script>  <body>  <form>  <p>username:<input name=*"username"*/></p>  <p>password:<input name=*"password"*/></p>  <p>phone:<input name=*"phone"*/></p>  <p><input type=*"button"* id=*"btn"* value=*"register"* /></p>  </form>  </body>  </html> |

#### UserAction.java

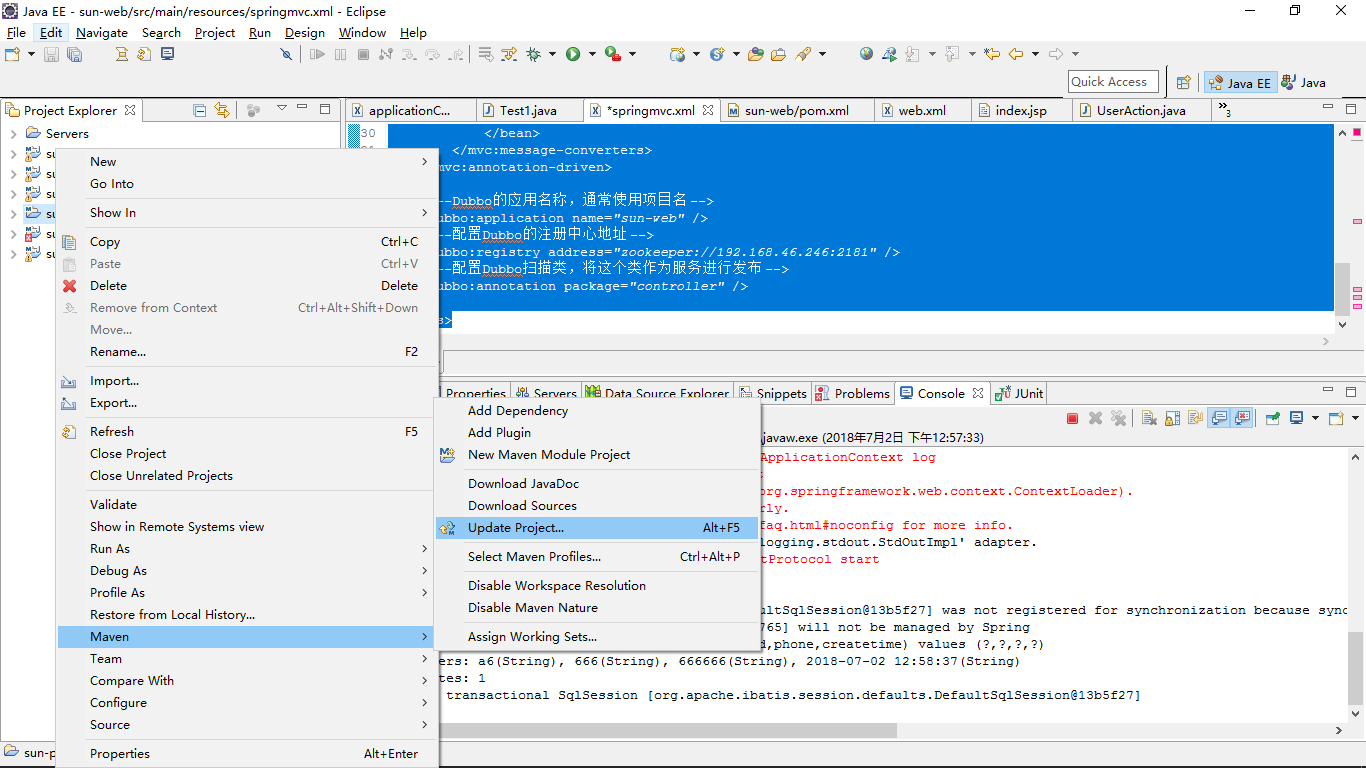
|  |
| --- |
| **package** controller;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.alibaba.dubbo.config.annotation.Reference;  **import** entity.Users;  **import** service.UserService;  @RestController  **public** **class** UserAction {  @Reference  **private** UserService us;    @RequestMapping("register")  **public** String register(Users user){  **try** {  user.setCreatetime(**new** SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(**new** Date()));  us.register(user);  **return** "注册成功！";  } **catch** (Exception e) {  e.printStackTrace();  **return** "注册shibai！";  }  }  } |

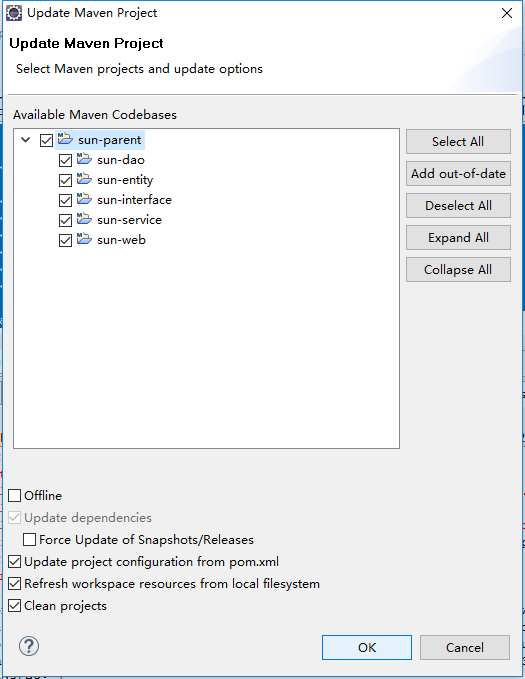
#### springmvc.xml

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  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://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo*  *http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>    <!-- json转换器 -->  <mvc:annotation-driven>  <mvc:message-converters register-defaults=*"true"*>  <bean  class=*"com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"*>  <property name=*"supportedMediaTypes"* value=*"application/json"* />  <property name=*"features"*>  <array>  <value>WriteMapNullValue</value>  <value>WriteDateUseDateFormat</value>  </array>  </property>  </bean>  </mvc:message-converters>  </mvc:annotation-driven>    <!--Dubbo的应用名称，通常使用项目名 -->  <dubbo:application name=*"sun-web"* />  <!--配置Dubbo的注册中心地址 -->  <dubbo:registry address=*"zookeeper://192.168.46.246:2181"* />  <!--配置Dubbo扫描类，将这个类作为服务进行发布 -->  <dubbo:annotation package=*"controller"* />  </beans> |

## 4启动测试

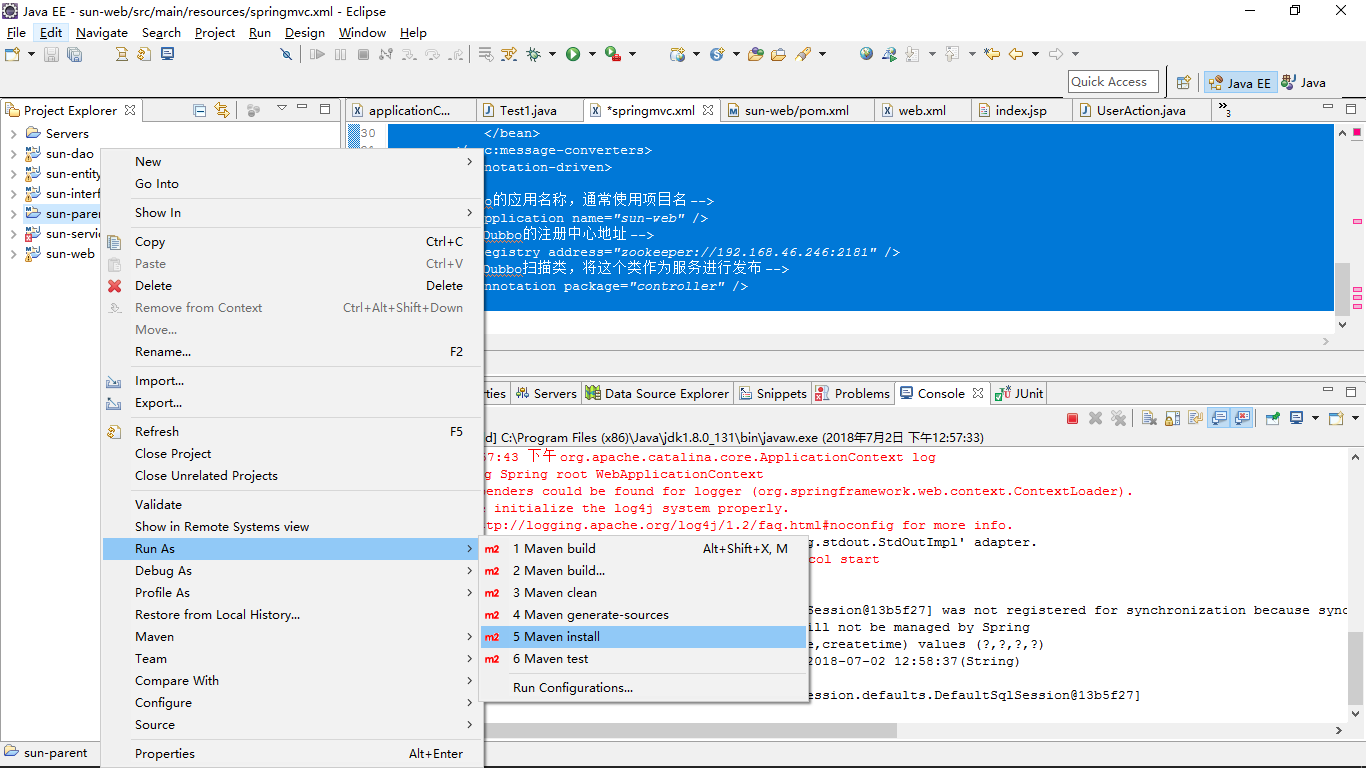
### 更新项目

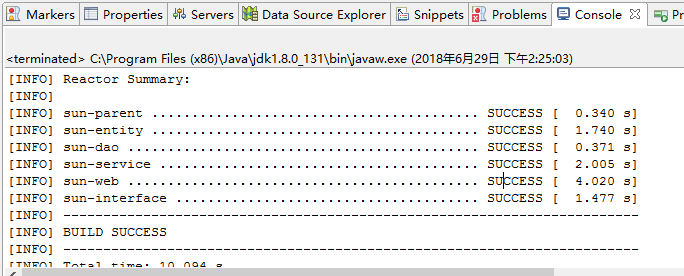




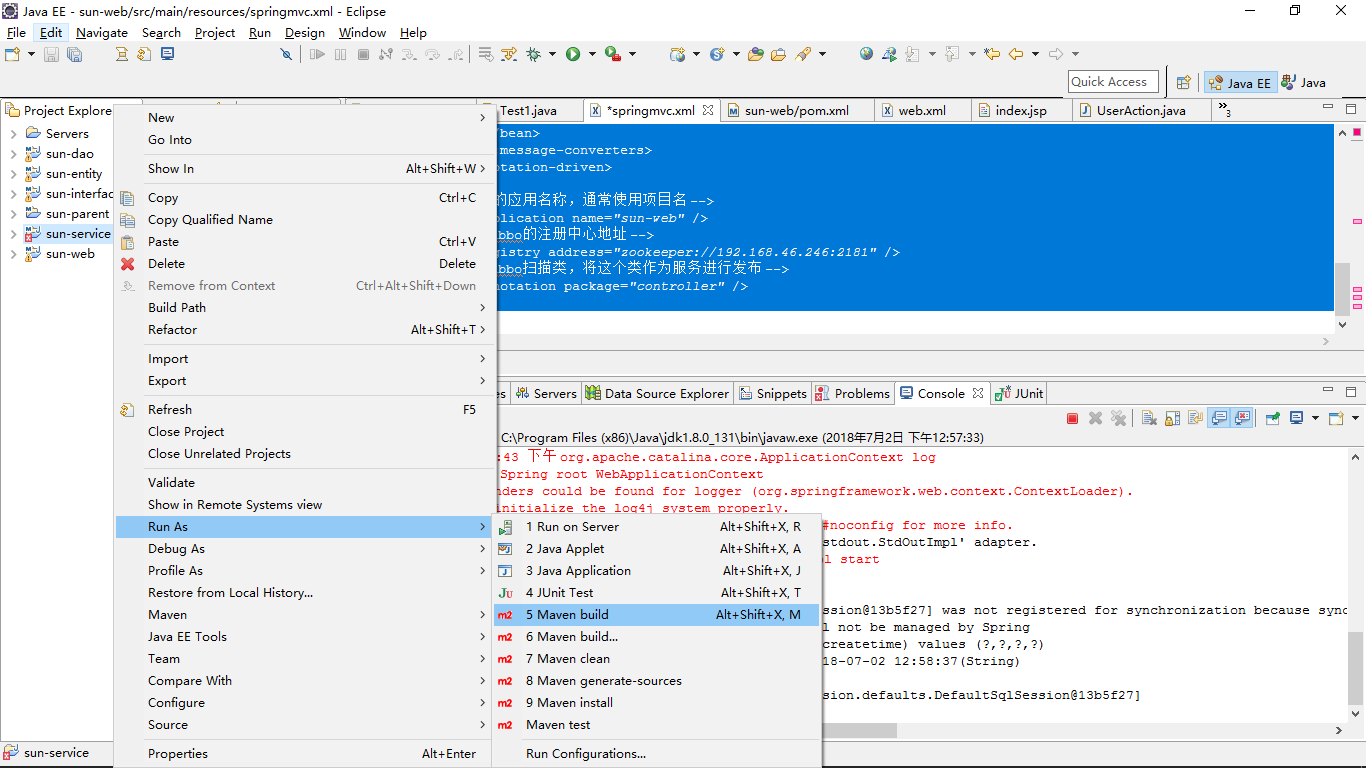
### 安装项目

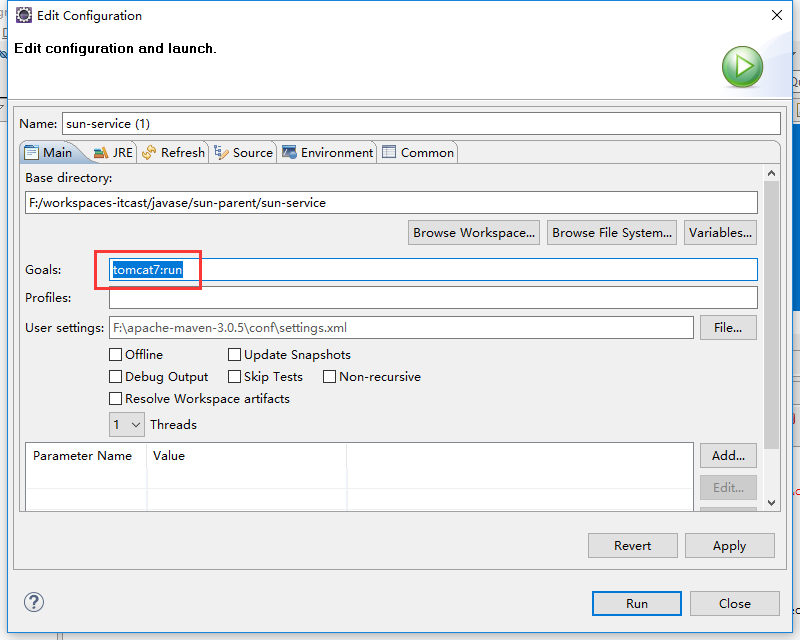
先选择父项目（聚合工程）进行全员安装成jar

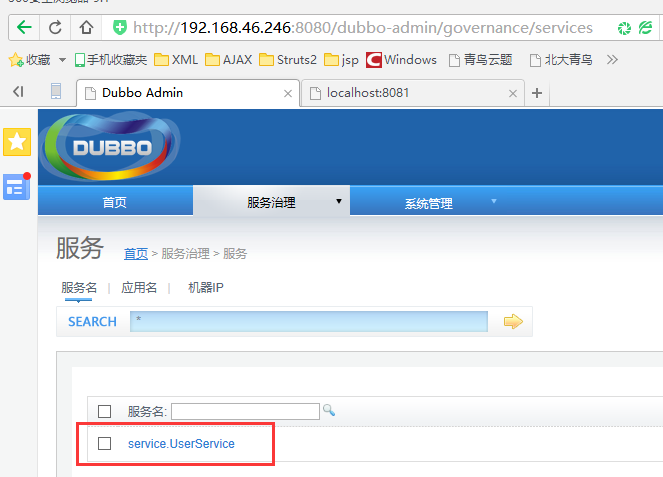


ok

### 启动服务service





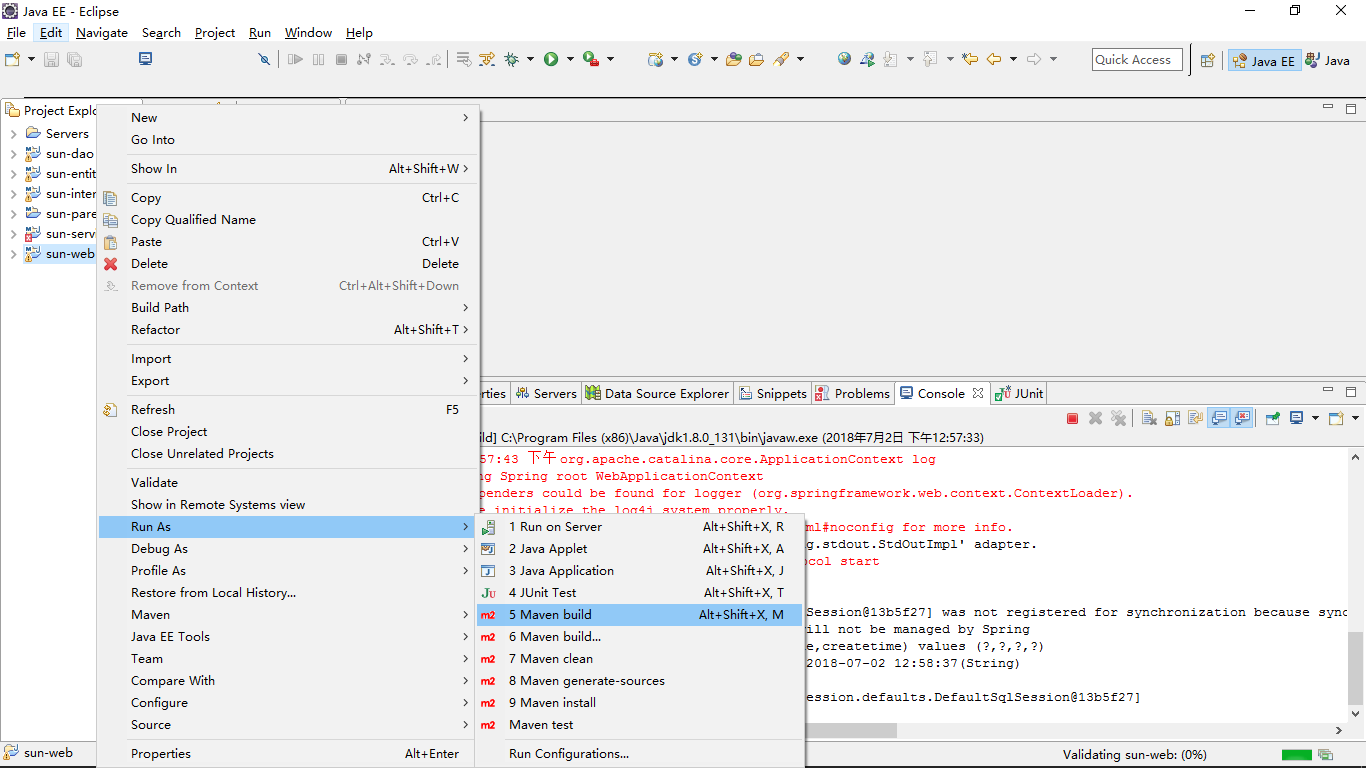


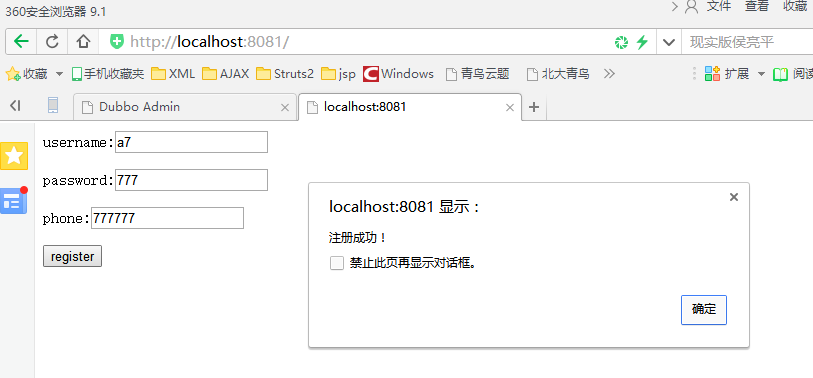
提供者：



### 启动调用者web

同上

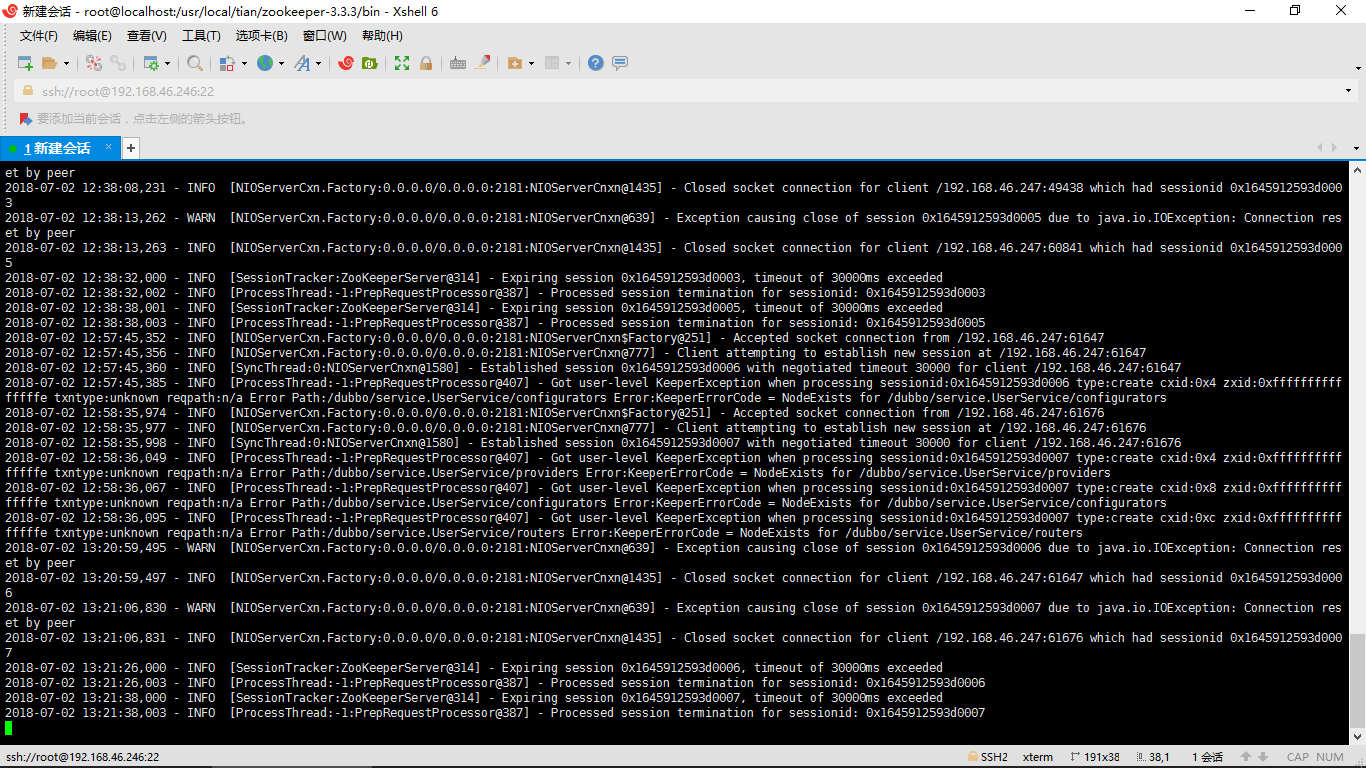




消费者：



linux服务器：



项目代码：百度云：

