* 1. 微服务基础理论

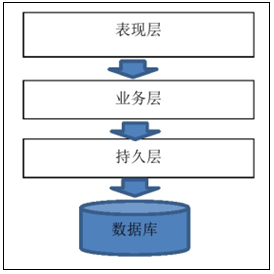
### 背景

2008年以后，国内互联网行业飞速发展，我们对软件系统的需求已经不再是过去”能用就行”这种很low的档次了，像**抢红包、双十一**这样的活动不断逼迫我们去突破软件系统的性能上限，传统的IT企业”能用就行”的开发思想已经不能满足互联网**高并发、大流量**的性能要求。系统架构走向**分布式**已经是服务器开发领域解决该问题唯一的出路，然而分布式系统由于天生的复杂度，并不像开发单体应用一样把框架一堆就能搞定，因此各大互联网公司都在投入技术力量研发自己的基础设施。这里面比较有名的如阿里的开源项目dubbo, Netflix开发的一系列服务框架。

### 系统架构的演变

#### 1.1.2.2 单体架构

单体架构也称之为单体系统或者是单体应用。就是一种把系统中所有的功能、模块耦合在一个应用中的架构方式。

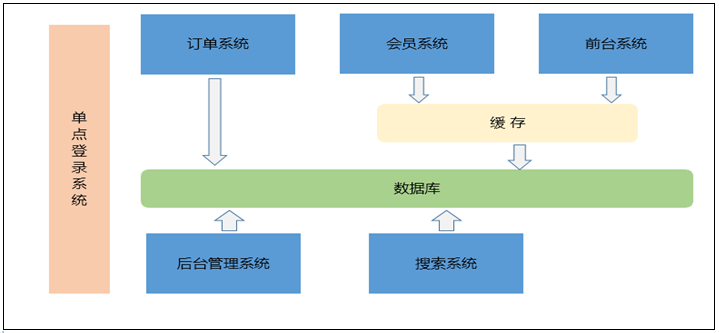


存在的问题：

* 代码耦合：模块的边界模糊、依赖关系不清晰，整个项目非常复杂，每次修改代码都心惊胆战
* 迭代困难：每次功能的变更或bug的修复都会导致重新部署整个应用，随着代码的增多，构建、测试和部署的时间也会增加
* 扩展受限：单体应用只能作为一个整体进行扩展，无法根据业务模块的需要进行伸缩
* 技术债务：随着时间推移、需求变更和人员更迭，会逐渐形成应用程序的技术债务，并且越积越多不坏不修
* 阻碍创新：单体应用往往使用统一的技术平台或方案解决所有的问题，要想引入新技术平台会非常困难

#### 1.1.2.3 分布式架构

分布式：需要按照功能点把系统拆分，拆分成独立的功能，单独为某一个节点添加服务器，需要系统之间配合才能完成整个业务逻辑。



分布式架构优点：

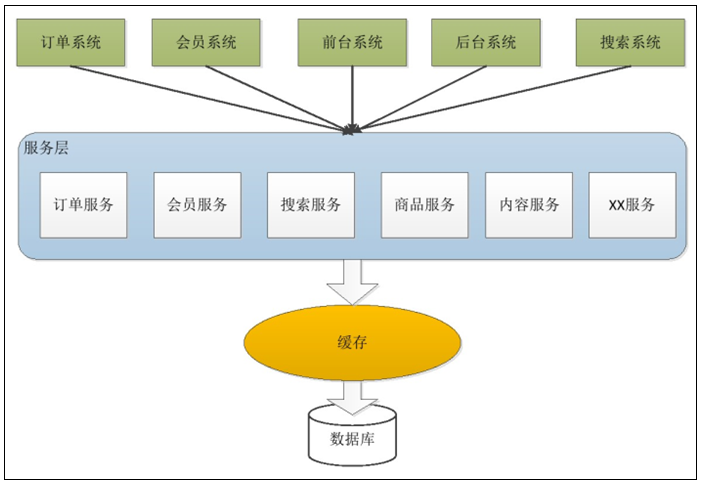
* 不同的团队负责不同的子项目
* 可以灵活的进行分布式部署
* 可以为某一模块单独加集群

分布式架构缺点：

* 模块之间有一些通用的业务逻辑无法共用。

#### 1.1.2.4 soa架构

SOA：Service Oriented Architecture（面向服务的架构）。也就是把工程拆分成**服务层，表现层**两个工程。服务层中包含业务逻辑，只需要对外提供服务即可。表现层只需要处理和页面的交互，业务逻辑都是调用服务层的服务来实现，使用ESB（Enterparise Servce Bus企业服务总线，代表技术：Mule、WSO2）提供表现层和服务层之间的交互。



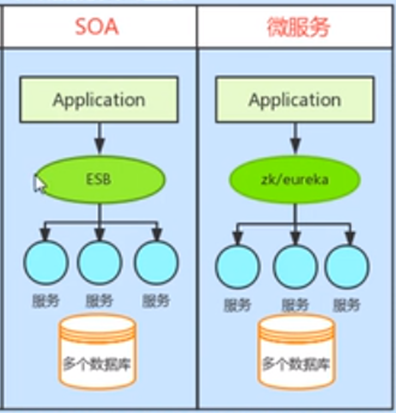
存在的问题：

* 不支持集群、臃肿

#### 1.1.2.5 微服务架构

微服务就是一个轻量级的服务治理方案。

代表技术：Eureka、zookeeper 等等



## 1.2 dubbox框架

### 1.2.1 dubbox简介

Dubbo(读音[ˈdʌbəʊ])是阿里巴巴公司开源的一个基于Java的高性能**RPC**（Remote Procedure Call）框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，可以和 Spring框架无缝集成。后期阿里巴巴停止了该项目的维护，于是当当网在这之上推出了自己的Dubbox。

### 1.2.2 dubboX架构



节点角色说明：

**Provider**: 暴露服务的服务提供方。

**Container**: 服务运行容器。

**Registry**: 服务注册与发现的注册中心。

**Consumer**: 调用远程服务的服务消费方。

**Monitor**: 统计服务的调用次调和调用时间的监控中心。

调用关系说明：

0. 服务容器负责启动，加载，运行服务提供者。

1. 服务提供者在启动时，向注册中心注册自己提供的服务。

2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

3. 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。

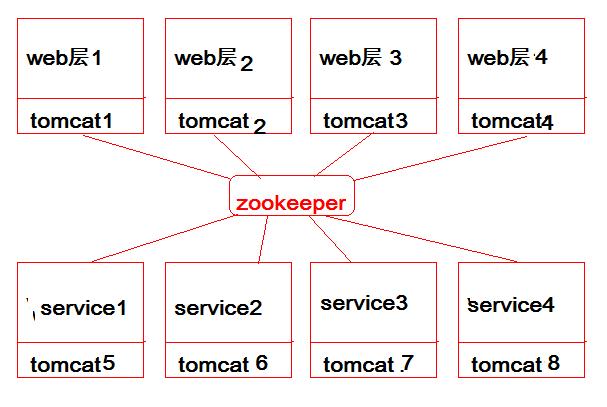
4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

5. 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。

## 1.3注册中心 zookeeper

### 1.3.1 zookeeper介绍

Zookeeper是Apacahe Hadoop的子项目，可以为分布式应用程序协调服务，适合作为Dubbo服务的注册中心，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。



### 1.3.2 zookeeper的安装

1、安装jdk

2、上传并解压缩zookeeper压缩包

|  |
| --- |
| tar -zxvf zookeeper-3.4.11.tar.gz -C /usr/java |

3、将conf文件夹下zoo\_sample.cfg复制一份，改名为zoo.cfg

|  |
| --- |
| cd /usr/java/zookeeper-3.4.11/conf  cp zoo\_sample.cfg zoo.cfg |

4、修改配置dataDir属性，指定一个真实目录

|  |
| --- |
| cd /usr/java/zookeeper-3.4.11  mkdir data |

打开 zoo.cfg , 修改 data 属性：dataDir=/usr/java/zookeeper-3.4.11/data

### 1.3.3 启动zookeeper

进入 bin 目录，启动服务输入命令

|  |
| --- |
| ./zkServer.sh star |

输出以下内容表示启动成功

|  |
| --- |
| JMX enabled by default  Using config: /root/zookeeper-3.4.6/bin/../conf/zoo.cfg  Starting zookeeper ... STARTED |

关闭服务输入命令

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

查看服务状态

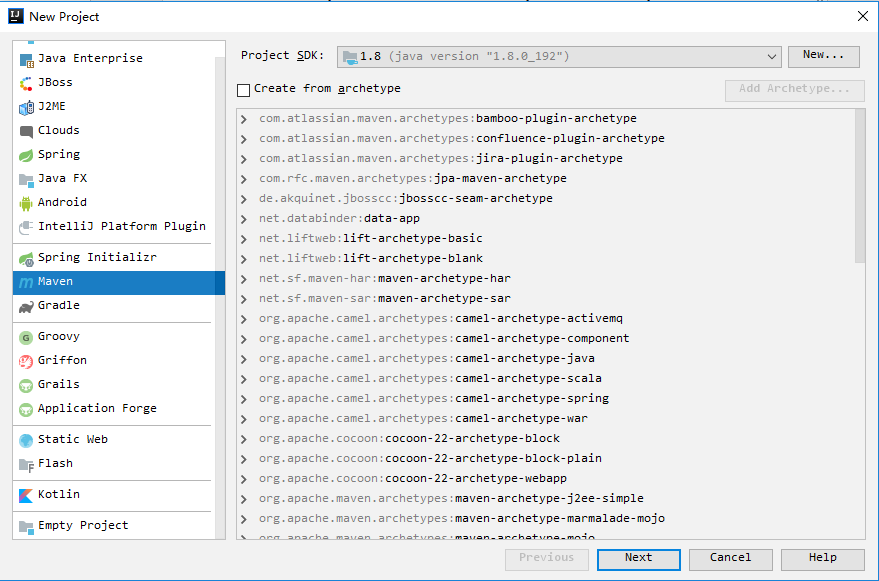
|  |
| --- |
| ./zkServer.sh status |

如果是启动状态则是以下提示

|  |
| --- |
| JMX enabled by default  Using config: /root/zookeeper-3.4.6/bin/../conf/zoo.cfg  Mode: standalone |

## 1.4 dubbox入门案例

### 1.4.1 创建父工程



点击Next

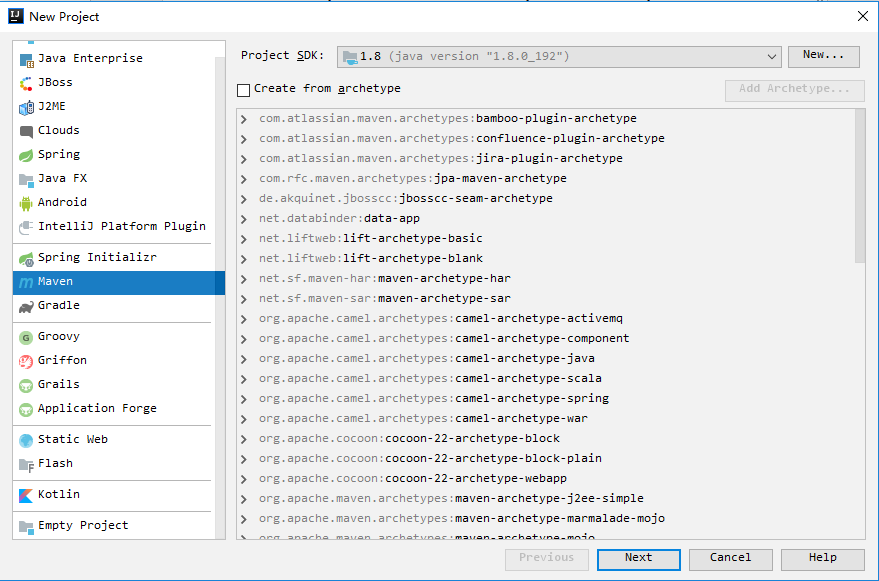


在父工程的pom.xml中添加依赖

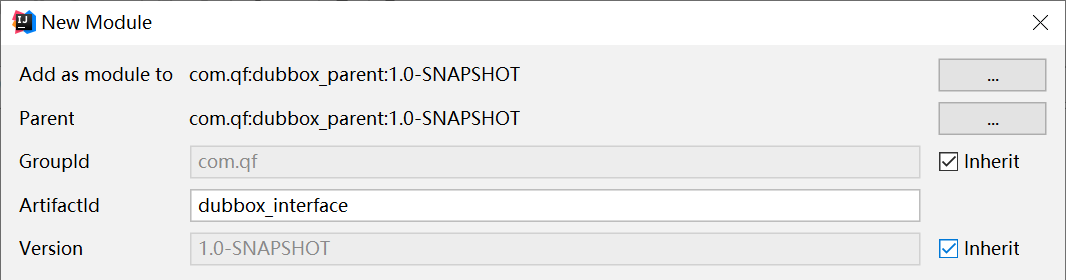
|  |
| --- |
| <**properties**>  <**spring.version**>5.0.5.RELEASE</**spring.version**> </**properties**>  <**dependencies**>  *<!-- Spring -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>${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**>  *<!-- dubbo依赖 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.8.4</**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**> |

### 1.4.2 定义公共接口

#### 1.4.2.1 创建公共接口模块



点击Next

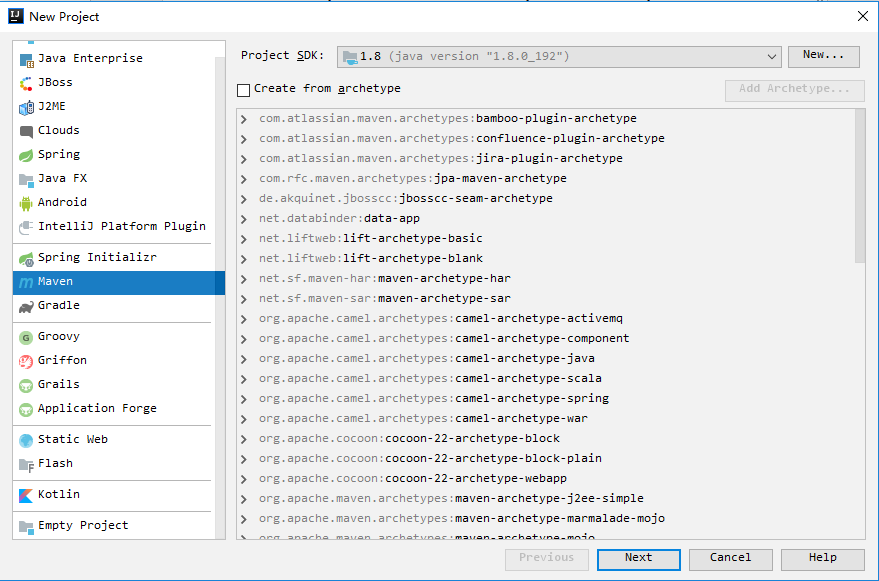


#### 1.4.2.2 创建公共接口

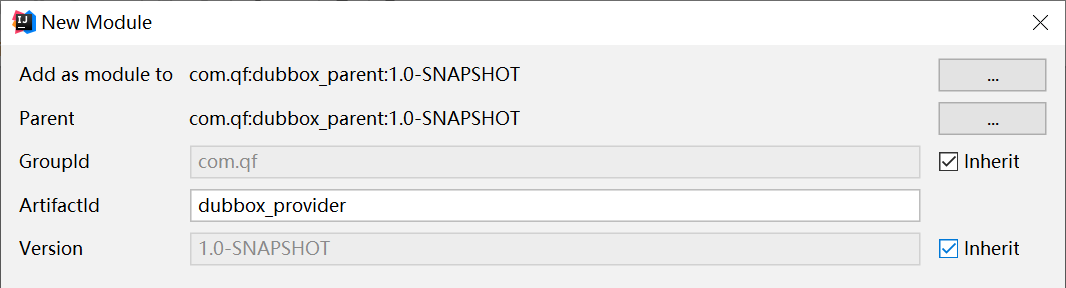
|  |
| --- |
| **public interface** UserService {  **public** String getName(); } |

### 1.4.3定义服务提供方

#### 1.4.3.1 创建服务提供方模块



点击Next



点击next

#### 1.4.3.2.创建pom.xml

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.qf</**groupId**>  <**artifactId**>dubbox\_interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  <**configuration**>  <**port**>8081</**port**>  <**path**>/</**path**>  </**configuration**>  </**plugin**>  </**plugins**> </**build**> |

#### 1.4.3.3 创建service

|  |
| --- |
| **import com.alibaba.dubbo.config.annotation.Service;**  //import org.springframework.stereotype.Service; @Service **public class** UserServiceImpl **implements** UserService {  @Override  **public** String getName() {  **return "hello,Dubbox......."**;  } } |

#### 1.4.2.4 创建配置文件

创建applicationContext-service.xml

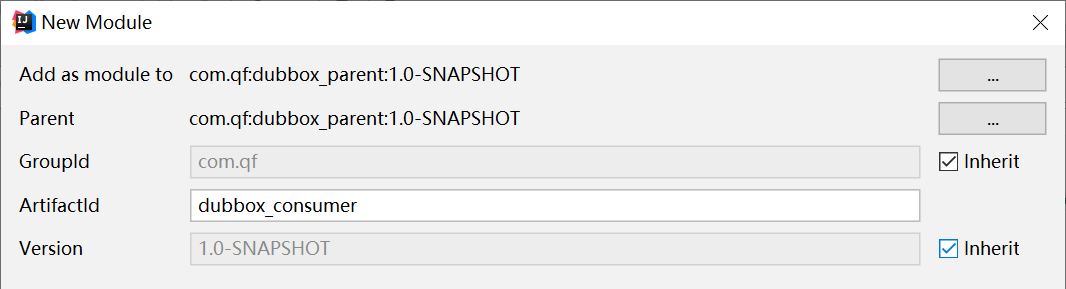
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:p="http://www.springframework.org/schema/p"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.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  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  <!--<context:component-scan base-package="com.qf.service"></context:component-scan>-->  <!-- 用于配置当前应用名，不管该应用是提供者还是消费者 -->  <**dubbo:application name="dubbo-service"**/>  <!--注册中心地址，与zookeeper端口保持一致 -->  <**dubbo:registry address="zookeeper://192.168.204.130:2181"**/>  <!-- 用dubbo协议在20880端口暴露服务  name：传输协议，Dubbo支持的协议有Dubbo、RMI、http、WebService  port：端口，其他应用可以通过这个端口调用服务  -->  <**dubbo:protocol name="dubbo" port="28080"**/>  <!--发布服务  interface：暴露了UserService接口  ref：引用了 Spring 中名为 userServiceImpl 的类  -->  <!--<dubbo:service interface="com.qf.service.UserService" ref="userServiceImpl"></dubbo:service>-->  <!--批量扫描，发布服务-->  <**dubbo:annotation package="com.qf.service"**/> </**beans**> |

#### 1.4.3.5 配置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"  version="2.5"**>  *<!-- 加载spring容器 -->* <**context-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext-\*.xml</**param-value**>  </**context-param**>  <**listener**>  <**listener-class**>org.springframework.web.context.ContextLoaderListener</**listener-class**>  </**listener**> </**web-app**> |

### 1.4.4 定义服务消费方

#### 1.4.4.1 创建服务消费方模块



#### 1.4.4.2.创建pom.xml

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.qf</**groupId**>  <**artifactId**>dubbox\_interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  <**configuration**>  <**port**>8080</**port**>  <**path**>/</**path**>  </**configuration**>  </**plugin**>  </**plugins**> </**build**> |

#### 1.4.4.3 定义controller

|  |
| --- |
| @Controller @RequestMapping(**"/user"**) **public class** UserController {   @Reference  //@Autowired  **private** UserService **userService**;   @RequestMapping(**"/getName"**)  @ResponseBody  **public** String getName() {  **return userService**.getName();  } } |

注意：Controller中注入HelloService使用的是Dubbo提供的@Reference注解

#### 1.4.3.4 配置springmvc.xml

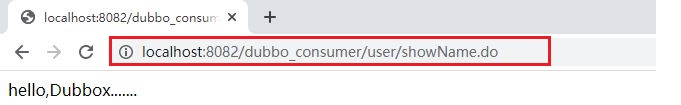
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.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  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"**>  <!--<context:component-scan base-package="com.qf.controller" />-->  <**mvc:annotation-driven** />  <**dubbo:application name="dubbo-consumer"**/>  <**dubbo:registry address="zookeeper://192.168.204.130:2181"**/>  <!--引用服务  interface：和服务提供者的暴露的服务类型保持一致  id：消费方可以直接通过该id注入接口实例到controller  -->  <!--<dubbo:reference interface="com.qf.service.UserService" id="userService" />-->  <!--spring支持@Reference-->  <**dubbo:annotation package="com.qf.controller"**></**dubbo:annotation**> </**beans**> |

#### 1.4.3.5 定义web.xml

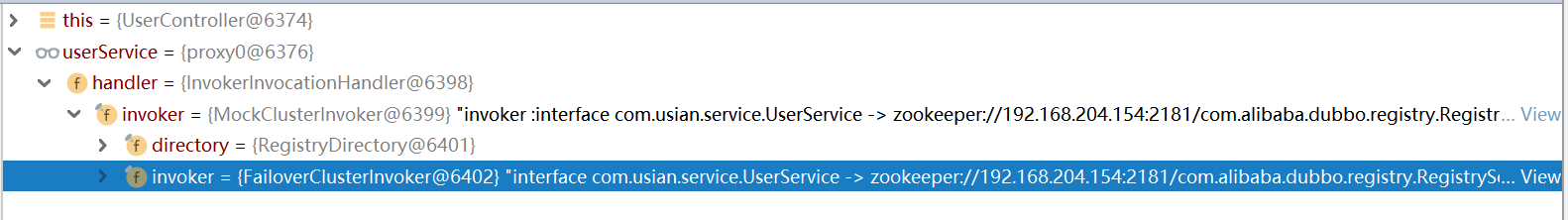
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"  version="2.5"**>  *<!-- 解决post乱码 -->* <**filter**>  <**filter-name**>CharacterEncodingFilter</**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**>CharacterEncodingFilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**>  </**filter-mapping**>   <**servlet**>  <**servlet-name**>springmvc</**servlet-name**>  <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  *<!-- 指定加载的配置文件 ，通过参数contextConfigLocation加载-->* <**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**>\*.do</**url-pattern**>  </**servlet-mapping**> </**web-app**> |

#### 1.4.4 启动测试

1. 注意：先install父工程(大war包要去父工程找依赖的版本号)
2. 先启动服务提供方再启动服务消费方
3. 访问: [http://localhost:8080/user/getName.do](http://localhost:8082/dubbo_consumer/user/showName.do)

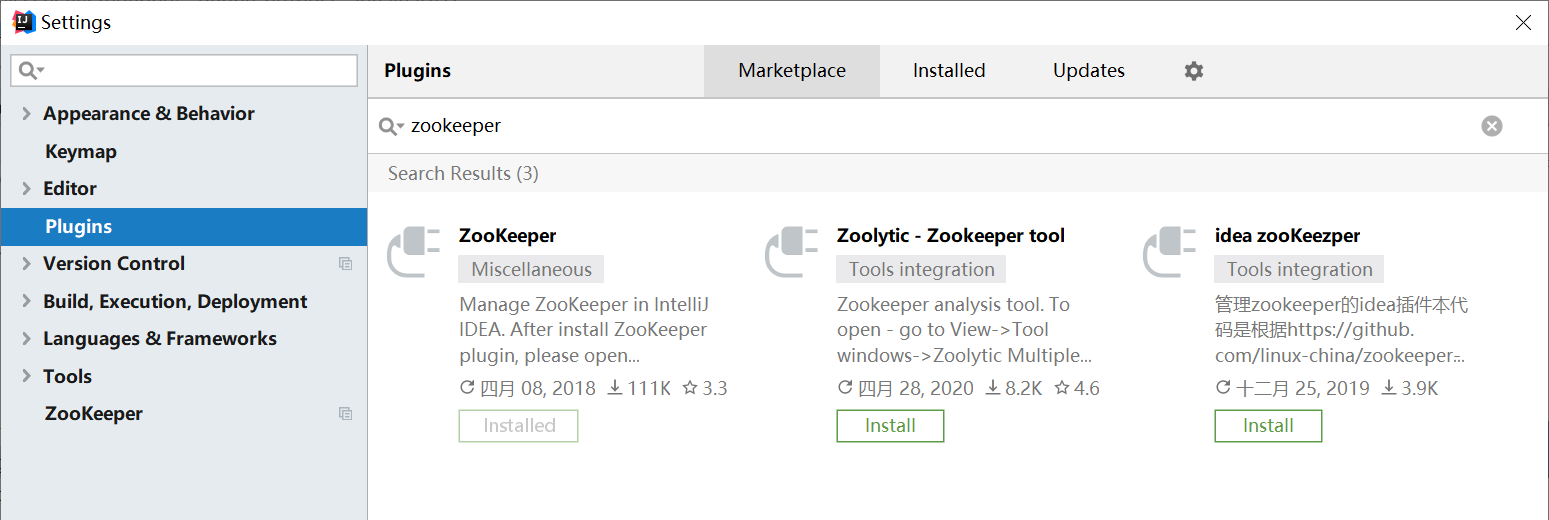


3. Debug查看UserService

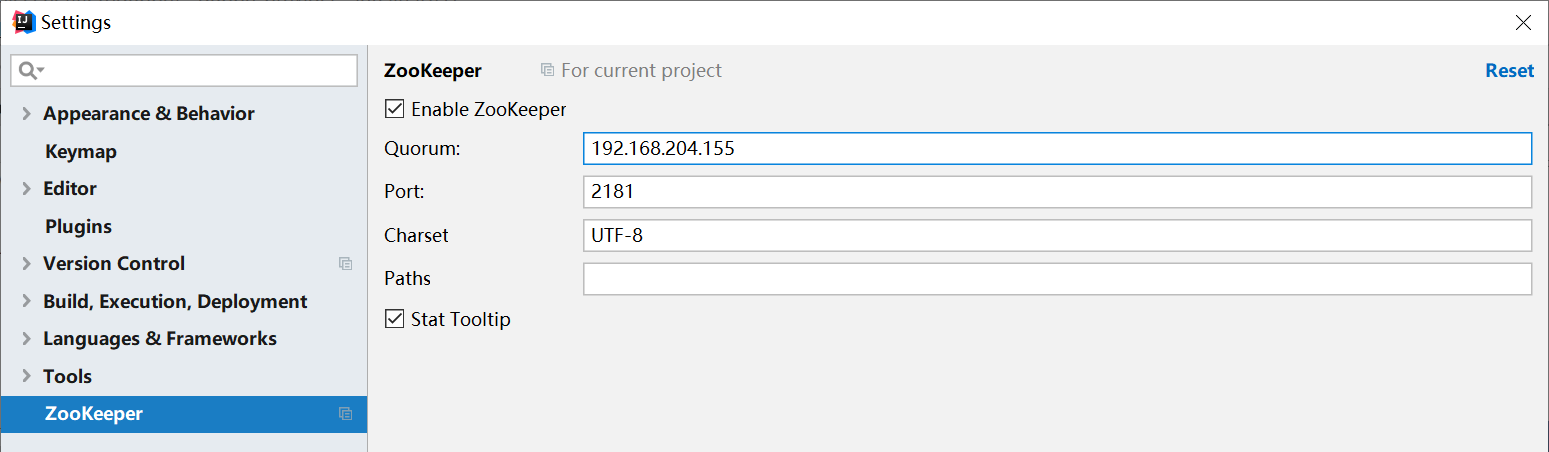


### 1.4.5 idea安装zookeeper插件

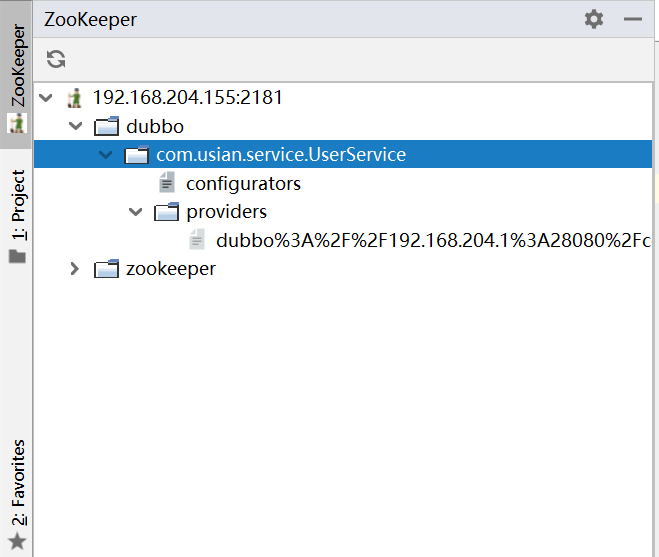
#### 1.4.5.1搜索zookeeper插件



#### 1.4.5.2.配置zookeeper插件



#### 1.4.5.3.使用zookeeper插件

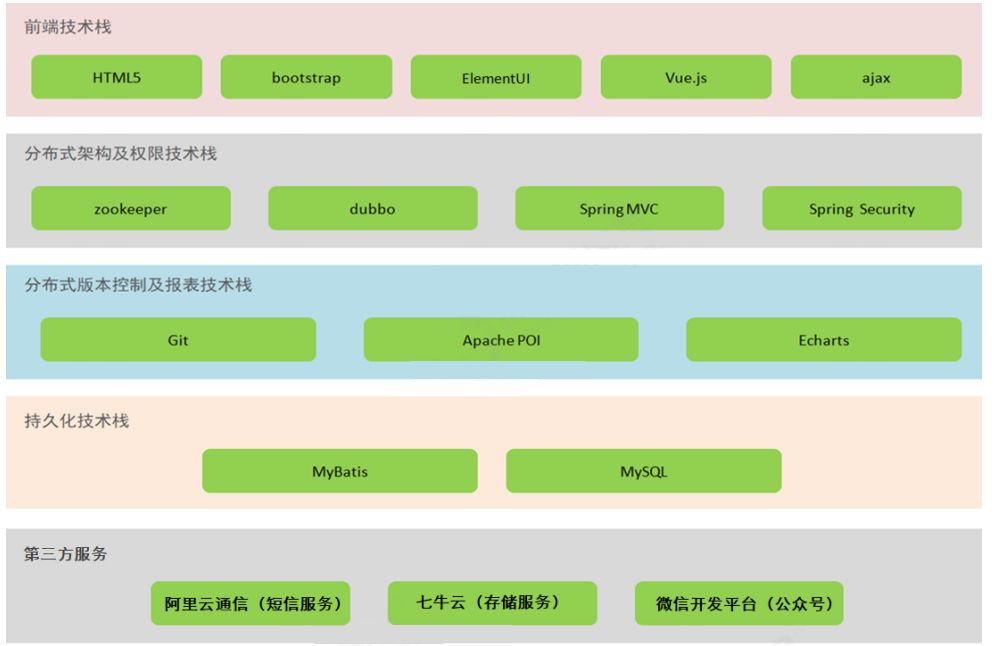


## 2.1 项目概述和环境搭建

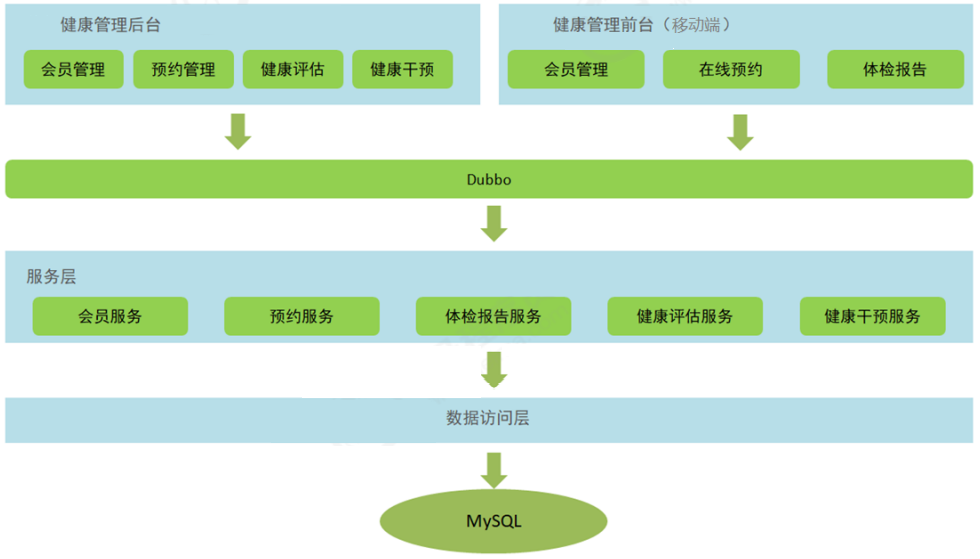
### 2.1.1 项目介绍

编易健康管理系统是一款应用于健康管理机构的业务系统，实现健康管理机构工作内容 可视化、会员管理专业化、健康评估数字化、健康干预流程化、知识库集成化，从而提 高健康管理师的工作效率，加强与会员间的互动，增强管理者对健康管理机构运营情况 的了解。

### 2.1.2 技术架构



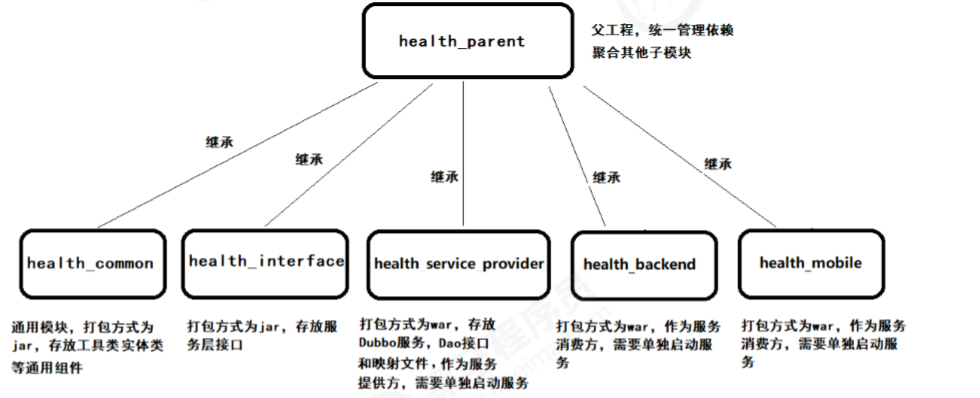
### 2.1.3 功能架构



### 2.1.4 环境搭建

#### 2.1.4.1 项目结构

本项目采用maven分模块开发方式，即对整个项目拆分为几个maven工程，每个maven 工程存放特定的一类代码，具体如下：



各模块职责定位：

health\_parent：父工程，打包方式为pom，统一锁定依赖的版本，同时聚合其他子模块 便于统一执行maven命令 health\_common：通用模块，打包方式为jar，存放项目中使用到的一些工具类、实体 类、返回结果和常量类

health\_interface：打包方式为jar，存放服务接口

health\_provider：Dubbo服务模块，打包方式为war，存放服务实现类、Dao接 口、Mapper映射文件等，作为服务提供方，需要部署到tomcat运行

health\_backend：编易健康管理后台，打包方式为war，作为Dubbo服务消费方，存放 Controller、HTML页面、js、css、spring配置文件等，需要部署到tomcat运行

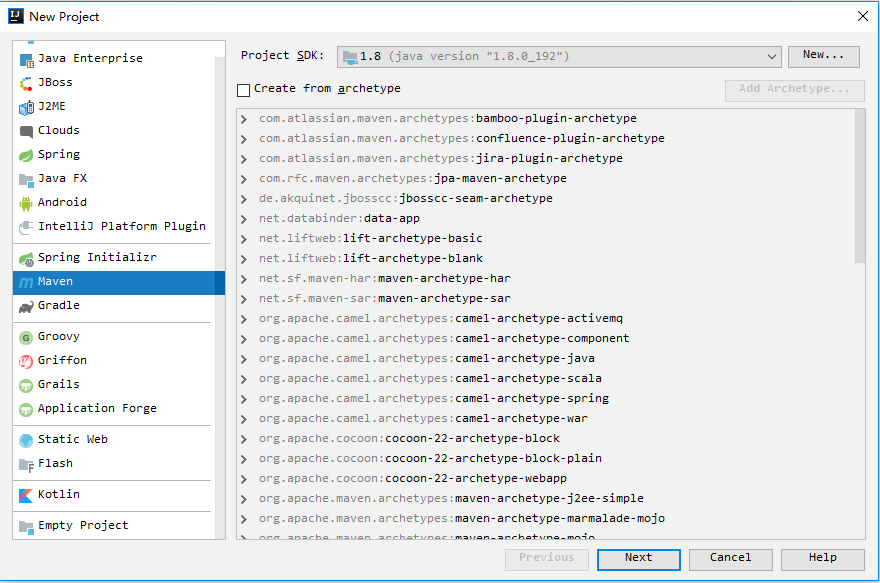
health\_mobile：移动端前台，打包方式为war，作为Dubbo服务消费方，存放 Controller、HTML页面、js、css、spring配置文件等，需要部署到tomcat运行。

#### 2.1.4.2 创建maven项目

通过前面的项目功能架构图可以知道本项目分为编易健康管理后台和编易健康前台（微信端）

#### 2.1.4.3 创建父工程 health\_parent

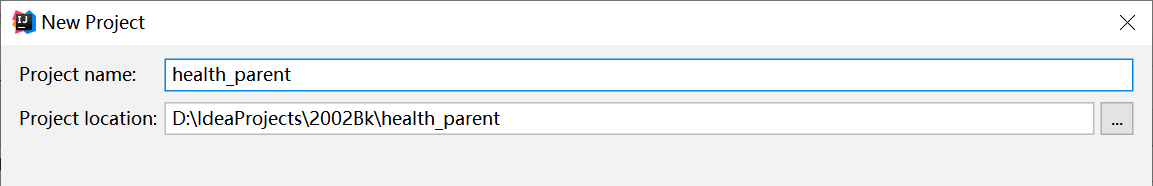
创建health\_parent，父工程，打包方式为pom，用于统一管理依赖版本



点击Next。



点击Next。



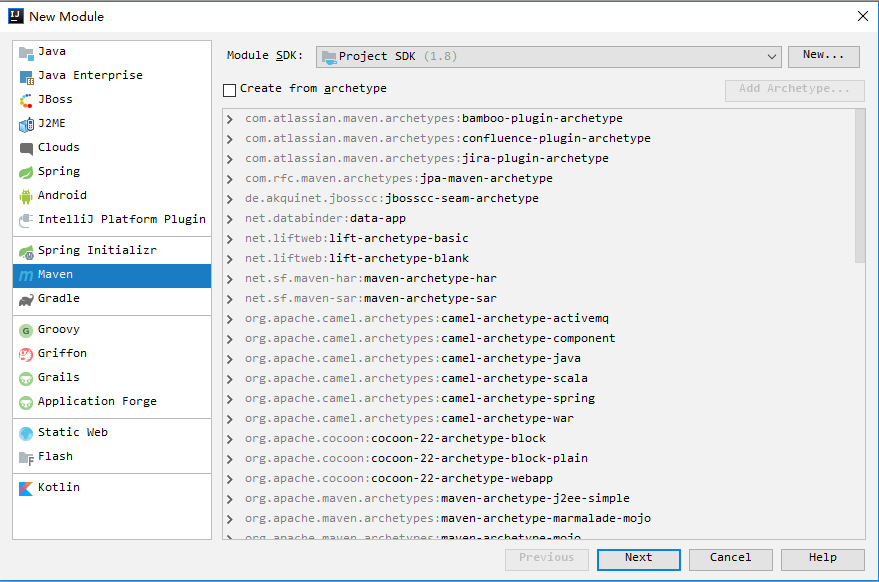
pom.xml

|  |
| --- |
| <**packaging**>pom</**packaging**> *<!-- 集中定义依赖版本号 -->* <**properties**>  <**junit.version**>4.12</**junit.version**>  <**spring.version**>5.0.5.RELEASE</**spring.version**>  <**pagehelper.version**>4.1.4</**pagehelper.version**>  <**servlet-api.version**>2.5</**servlet-api.version**>  <**dubbo.version**>2.6.0</**dubbo.version**>  <**zookeeper.version**>3.4.7</**zookeeper.version**>  <**zkclient.version**>0.1</**zkclient.version**>  <**mybatis.version**>3.4.5</**mybatis.version**>  <**mybatis.spring.version**>1.3.1</**mybatis.spring.version**>  <**mybatis.paginator.version**>1.2.15</**mybatis.paginator.version**>  <**mysql.version**>5.1.32</**mysql.version**>  <**druid.version**>1.0.9</**druid.version**>  <**commons-fileupload.version**>1.3.1</**commons-fileupload.version**>  <**spring.security.version**>5.0.5.RELEASE</**spring.security.version**>  <**poi.version**>3.14</**poi.version**>  <**jedis.version**>2.9.0</**jedis.version**>  <**quartz.version**>2.2.1</**quartz.version**> </**properties**> *<!-- 依赖管理标签 必须加 -->* <**dependencyManagement**>  <**dependencies**>  *<!-- 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-web</**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**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jms</**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-test</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- dubbo相关 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>${dubbo.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**>  <**version**>${zookeeper.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**version**>${zkclient.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.12</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  <**version**>1.2.47</**version**>  </**dependency**>  <**dependency**>  <**groupId**>javassist</**groupId**>  <**artifactId**>javassist</**artifactId**>  <**version**>3.12.1.GA</**version**>  </**dependency**>  <**dependency**>  <**groupId**>commons-codec</**groupId**>  <**artifactId**>commons-codec</**artifactId**>  <**version**>1.10</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**>  <**version**>${pagehelper.version}</**version**>  </**dependency**>  *<!-- Mybatis -->* <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis</**artifactId**>  <**version**>${mybatis.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis-spring</**artifactId**>  <**version**>${mybatis.spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.miemiedev</**groupId**>  <**artifactId**>mybatis-paginator</**artifactId**>  <**version**>${mybatis.paginator.version}</**version**>  </**dependency**>  *<!-- MySql -->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>${mysql.version}</**version**>  </**dependency**>  *<!-- 连接池 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>druid</**artifactId**>  <**version**>${druid.version}</**version**>  </**dependency**>  *<!-- 文件上传组件 -->* <**dependency**>  <**groupId**>commons-fileupload</**groupId**>  <**artifactId**>commons-fileupload</**artifactId**>  <**version**>${commons-fileupload.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.quartz-scheduler</**groupId**>  <**artifactId**>quartz</**artifactId**>  <**version**>${quartz.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.quartz-scheduler</**groupId**>  <**artifactId**>quartz-jobs</**artifactId**>  <**version**>${quartz.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.sun.jersey</**groupId**>  <**artifactId**>jersey-client</**artifactId**>  <**version**>1.18.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.qiniu</**groupId**>  <**artifactId**>qiniu-java-sdk</**artifactId**>  <**version**>7.2.0</**version**>  </**dependency**>  *<!--POI报表-->* <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi</**artifactId**>  <**version**>${poi.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi-ooxml</**artifactId**>  <**version**>${poi.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>redis.clients</**groupId**>  <**artifactId**>jedis</**artifactId**>  <**version**>${jedis.version}</**version**>  </**dependency**>  *<!-- 安全框架 -->* <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-web</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-config</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-taglibs</**artifactId**>  <**version**>${spring.security.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.penggle</**groupId**>  <**artifactId**>kaptcha</**artifactId**>  <**version**>2.3.2</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>javax.servlet-api</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>dom4j</**groupId**>  <**artifactId**>dom4j</**artifactId**>  <**version**>1.6.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>xml-apis</**groupId**>  <**artifactId**>xml-apis</**artifactId**>  <**version**>1.4.01</**version**>  </**dependency**>  <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>servlet-api</**artifactId**>  <**version**>${servlet-api.version}</**version**>  <**scope**>provided</**scope**>  </**dependency**>  </**dependencies**> </**dependencyManagement**> <**build**>  <**pluginManagement**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  </**plugin**>  </**plugins**>  </**pluginManagement**> </**build**> |

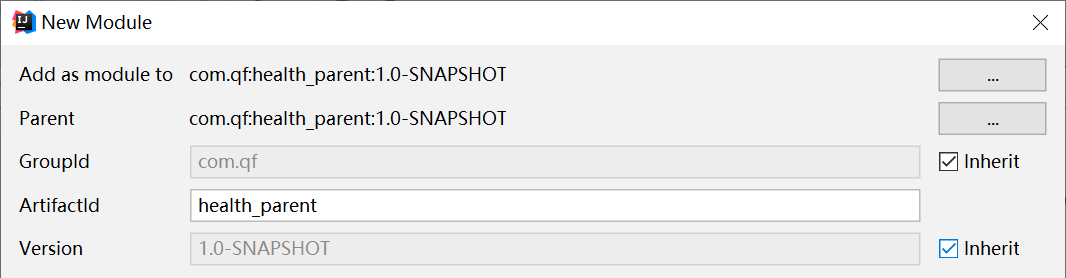
问题：为什么只在父工程定义版本号？1、避免继承父工程2T的依赖 2、方便升级版本号

#### 2.1.4.4 创建health\_common

创建health\_common，子工程，打包方式为jar，存放通用组件，例如工具类、实体类等



点击Next.

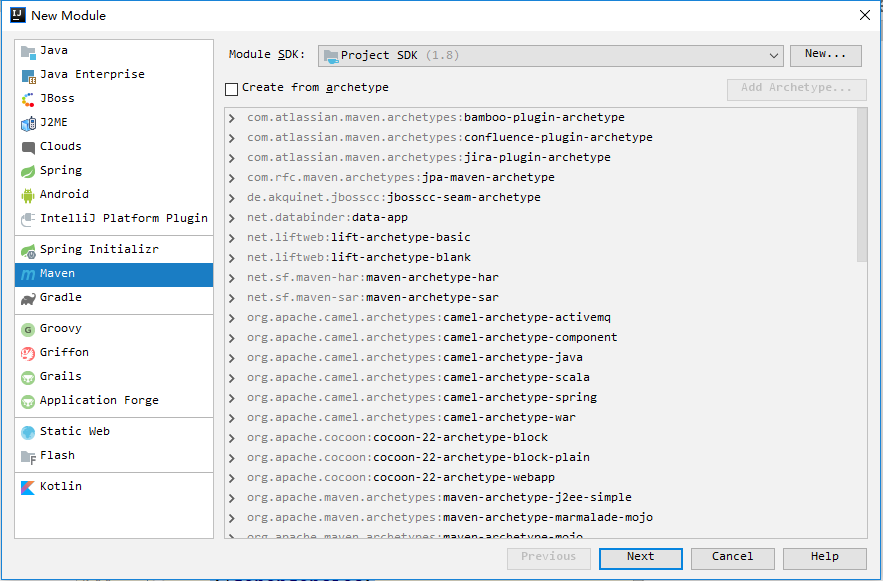


pom.xml

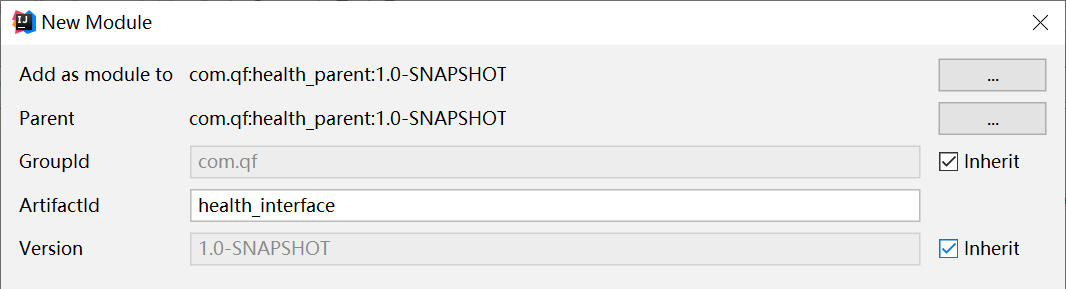
|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**>  </**dependency**>  *<!-- Mybatis -->* <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.mybatis</**groupId**>  <**artifactId**>mybatis-spring</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.miemiedev</**groupId**>  <**artifactId**>mybatis-paginator</**artifactId**>  </**dependency**>  *<!-- MySql -->* <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  </**dependency**>  *<!-- 连接池 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>druid</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>commons-fileupload</**groupId**>  <**artifactId**>commons-fileupload</**artifactId**>  </**dependency**>  *<!-- Spring -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-beans</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aspects</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jms</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  </**dependency**>  *<!-- dubbo相关 -->* <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.zookeeper</**groupId**>  <**artifactId**>zookeeper</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>javassist</**groupId**>  <**artifactId**>javassist</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>commons-codec</**groupId**>  <**artifactId**>commons-codec</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>redis.clients</**groupId**>  <**artifactId**>jedis</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.qiniu</**groupId**>  <**artifactId**>qiniu-java-sdk</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.sun.jersey</**groupId**>  <**artifactId**>jersey-client</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi-ooxml</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-web</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-config</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-taglibs</**artifactId**>  </**dependency**> </**dependencies**> |

#### 2.1.4.5 创建health\_interface

创建health\_interface，子工程，打包方式为jar，存放服务接口



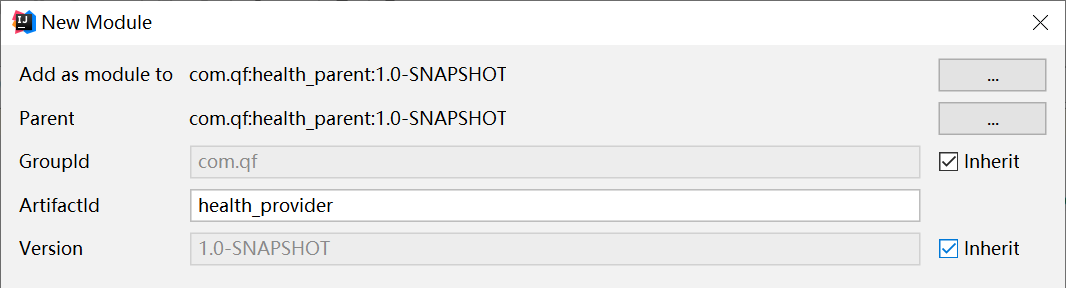
点击Next



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**>health\_parent</**artifactId**>  <**groupId**>com.qf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.qf</**groupId**>  <**artifactId**>health\_interface</**artifactId**>  *<!--直接依赖health\_common即可-->* <**dependencies**>  <**dependency**>  <**groupId**>com.qf</**groupId**>  <**artifactId**>health\_common</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**> </**project**> |

#### 2.1.4.6 创建health\_provider

创建health\_provider子工程，打包方式为war，存放服务实现类、Dao接 口、Mapper映射文件等，作为服务提供方

pom.xml

|  |
| --- |
| <**packaging**>war</**packaging**>  <**dependencies**>  <!--依赖health\_interface-->  <**dependency**>  <**groupId**>com.qf</**groupId**>  <**artifactId**>health\_interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**>  <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**configuration**>  <!-- 指定端口 -->  <**port**>8081</**port**>  <!-- 请求路径 -->  <**path**>/</**path**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

引入日志文件(log4j.proerties)

|  |
| --- |
| *### direct log messages to stdout ###* **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.err log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**%d{ABSOLUTE} %5p %c{1}:%L - %m%n** *### direct messages to file mylog.log ###* **log4j.appender.file**=**org.apache.log4j.FileAppender log4j.appender.file.File**=**c:\\mylog.log log4j.appender.file.layout**=**org.apache.log4j.PatternLayout log4j.appender.file.layout.ConversionPattern**=**%d{ABSOLUTE} %5p %c{1}:%L - %m%n** *### set log levels - for more verbose logging change 'info' to 'debug' ###* **log4j.rootLogger**=**debug, stdout** |

SqlMapConfig.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**>  <**plugins**>  *<!-- com.github.pagehelper 为 PageHelper 类所在包名 -->* <**plugin interceptor="com.github.pagehelper.PageHelper"**>  *<!-- 设置数据库类型 Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL 六种数据库-->* <**property name="dialect" value="mysql"**/>  </**plugin**>  </**plugins**> </**configuration**> |

applicationContext-dao.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**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"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd  http://www.springframework.org/schema/util  http://www.springframework.org/schema/util/spring-util.xsd"**>   <!--数据源-->  <**bean id="dataSource"  class="com.alibaba.druid.pool.DruidDataSource" destroy-method="close"**>  <**property name="username" value="root"** />  <**property name="password" value="1111"** />  <**property name="driverClassName" value="com.mysql.jdbc.Driver"** />  <**property name="url"**  **value="jdbc:mysql://localhost:3306/health?characterEncoding=utf-8"**/>  </**bean**>  <!--spring和mybatis整合的工厂bean-->  <**bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>  <**property name="dataSource" ref="dataSource"** />  <**property name="configLocation" value="classpath:SqlMapConfig.xml"** />  </**bean**>  <!--批量扫描接口生成代理对象-->  <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  <!--指定接口所在的包-->  <**property name="basePackage" value="com.qf.mapper"** />  <**property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"**></**property**>  </**bean**> </**beans**> |

applicationContext-trans.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:context="http://www.springframework.org/schema/context"  xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  *<!-- 事务管理器 -->* <**bean id="transactionManager"  class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>  <**property name="dataSource" ref="dataSource"**/>  </**bean**>  *<!--  开启事务控制的注解支持  注意：此处必须加入proxy-target-class="true"，  需要进行事务控制，会由Spring框架产生代理对象，  Dubbo需要将Service发布为服务，要求必须使用cglib创建代理对象。  -->* <**tx:annotation-driven transaction-manager="transactionManager"  proxy-target-class="true"**/> </**beans**> |

applicationContext-services.xml

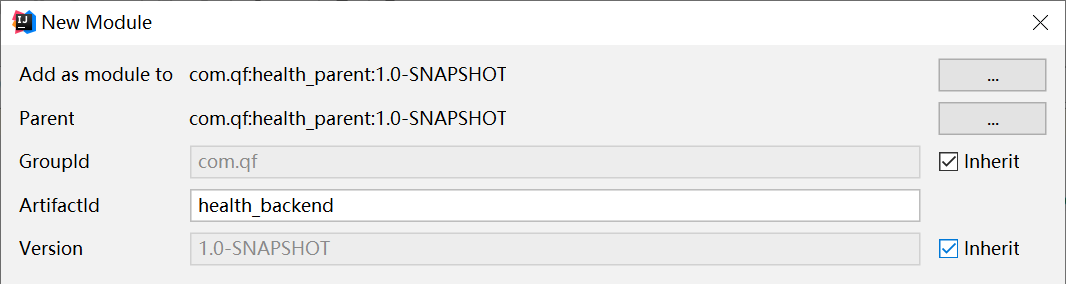
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.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  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  <!-- 指定应用名称 -->  <**dubbo:application name="health\_provider"**/>  <!--指定暴露服务的端口，如果不指定默认为20880-->  <**dubbo:protocol name="dubbo" port="20887"**/>  <!--指定服务注册中心地址-->  <**dubbo:registry address="zookeeper://192.168.204.154:2181"**/>  <!--批量扫描，发布服务-->  <**dubbo:annotation package="com.qf.service"**/> </**beans**> |

Web.xml

|  |
| --- |
| **<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  "http://java.sun.com/dtd/web-app\_2\_3.dtd" *>*** <**web-app**>  <**display-name**>Archetype Created Web Application</**display-name**>  *<!-- 加载spring容器 -->* <**context-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext-\*.xml</**param-value**>  </**context-param**>  <**listener**>  <**listener-class**>org.springframework.web.context.ContextLoaderListener</**listener-class**>  </**listener**> </**web-app**> |

#### 2.1.4.7创建health\_backend

创建health\_backend，子工程，打包方式为war，单独部署，存放Controller、页面等.使用骨架创建。



pom.xml配置

|  |
| --- |
| <**packaging**>war</**packaging**>  <**dependencies**>  <**dependency**>  <**groupId**>com.qf</**groupId**>  <**artifactId**>health\_interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**>  <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**configuration**>  <!-- 指定端口 -->  <**port**>80</**port**>  <!-- 请求路径 -->  <**path**>/</**path**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

log4j.properties

|  |
| --- |
| ### direct log messages to stdout ### **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.err log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**%d{ABSOLUTE} %5p %c{1}:%L - %m%n** ### direct messages to file mylog.log ### #log4j.appender.file=org.apache.log4j.FileAppender #log4j.appender.file.File=c:\\mylog.log #log4j.appender.file.layout=org.apache.log4j.PatternLayout #log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n ### set log levels - for more verbose logging change 'info' to 'debug' ### **log4j.rootLogger**=**debug, stdout** |

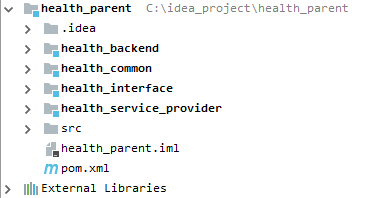
springmvc.xml

|  |
| --- |
| *<?***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:context="http://www.springframework.org/schema/context"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.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  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  <**mvc:annotation-driven**></**mvc:annotation-driven**>  *<!-- 指定应用名称 -->* <**dubbo:application name="health\_backend"** />  *<!--指定服务注册中心地址-->* <**dubbo:registry address="zookeeper://127.0.0.1:2181"**/>  *<!--批量扫描-->* <**dubbo:annotation package="com.qf.controller"** />  *<!--文件上传组件-->* <**bean id="multipartResolver"  class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>  <**property name="maxUploadSize" value="104857600"** />  <**property name="maxInMemorySize" value="4096"** />  <**property name="defaultEncoding" value="UTF-8"**/>  </**bean**> </**beans**> |

Web.xml

|  |
| --- |
| **<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  "http://java.sun.com/dtd/web-app\_2\_3.dtd" *>*** <**web-app**>  <**display-name**>Archetype Created Web Application</**display-name**>  *<!-- 解决post乱码 -->* <**filter**>  <**filter-name**>CharacterEncodingFilter</**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**>CharacterEncodingFilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**>  </**filter-mapping**>  <**servlet**>  <**servlet-name**>springmvc</**servlet-name**>  <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  *<!-- 指定加载的配置文件 ，通过参数contextConfigLocation加载 -->* <**init-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:springmvc.xml</**param-value**>  </**init-param**>  <**load-on-startup**>1</**load-on-startup**>  </**servlet**>  <**servlet-mapping**>  <**servlet-name**>springmvc</**servlet-name**>  <**url-pattern**>\*.do</**url-pattern**>  </**servlet-mapping**> </**web-app**> |

最后目录结构如下:



## 3.1 Power Designer

### 3.1.1 Power Designer介绍

PowerDesigner是Sybase公司的一款软件，使用它可以方便地对系统进行分析设计，他 几乎包括了数据库模型设计的全过程。利用PowerDesigner可以制作数据流程图、概念 数据模型、物理数据模型、面向对象模型。

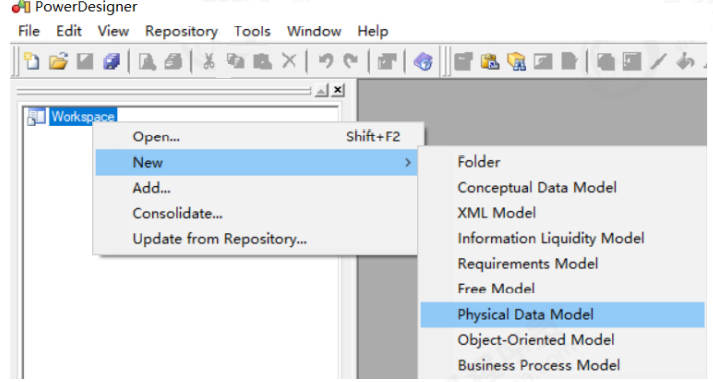
在项目设计阶段通常会使用PowerDesigner进行数据库设计。使用PowerDesigner可以 更加直观的表现出数据库中表之间的关系，并且可以直接导出相应的建表语句。

### 3.1.2 Power Designer的使用

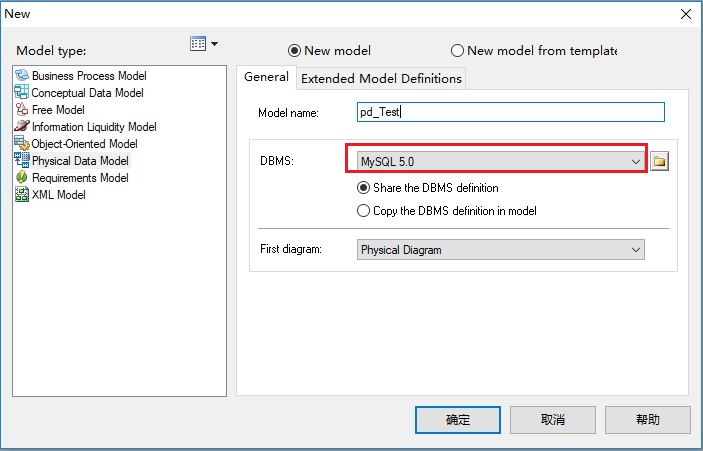
#### 3.1.2.1 创建物理数据模型

操作步骤：

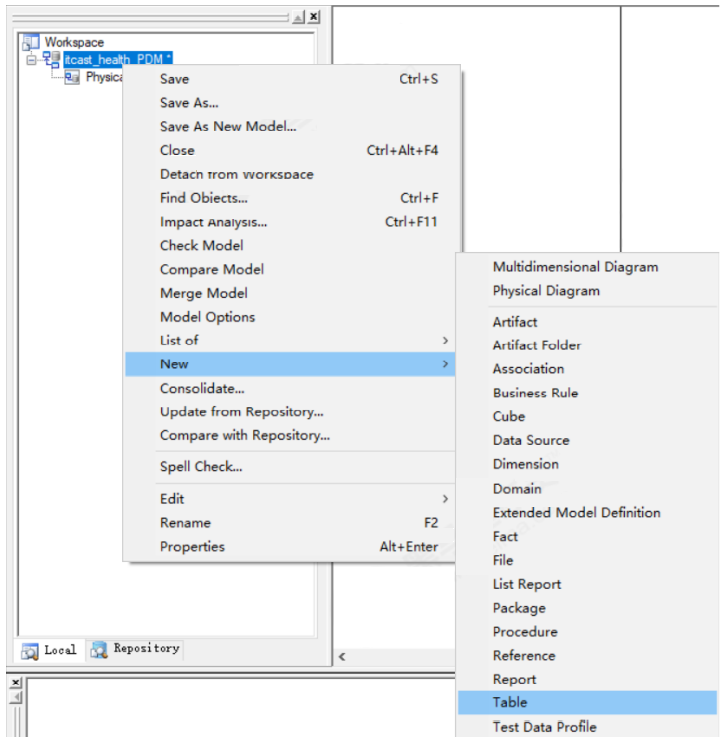
1. 创建数据模型PDM



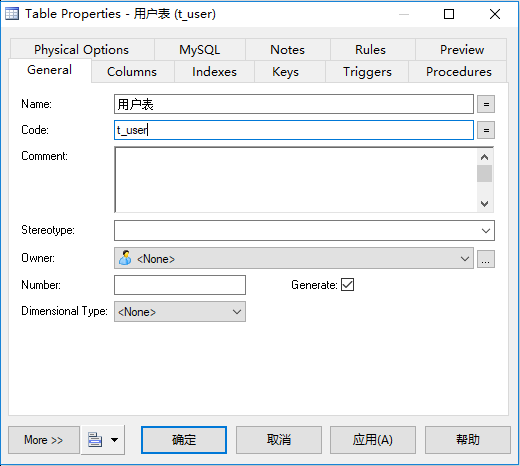
1. 选择数据库类型



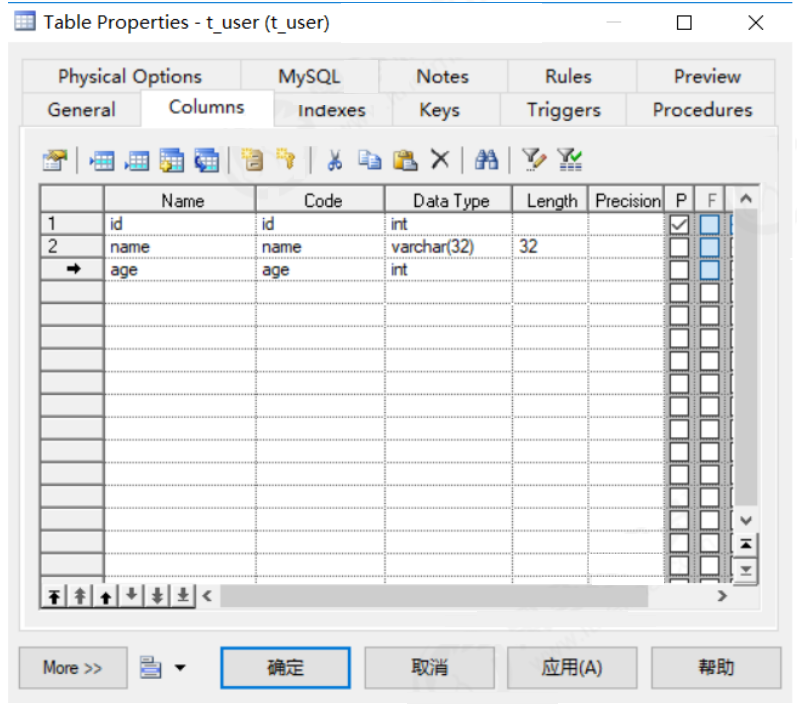
1. 创建表和字段



指定表名

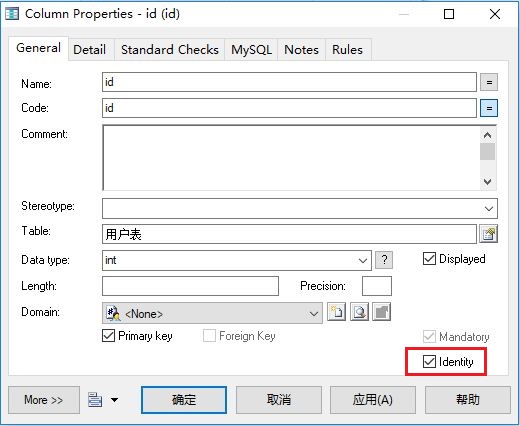


创建字段

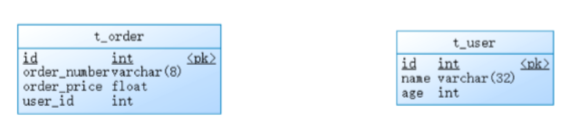


主键自增长：

选中id字段右键:



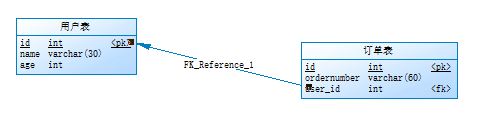
1. 添加外键字段



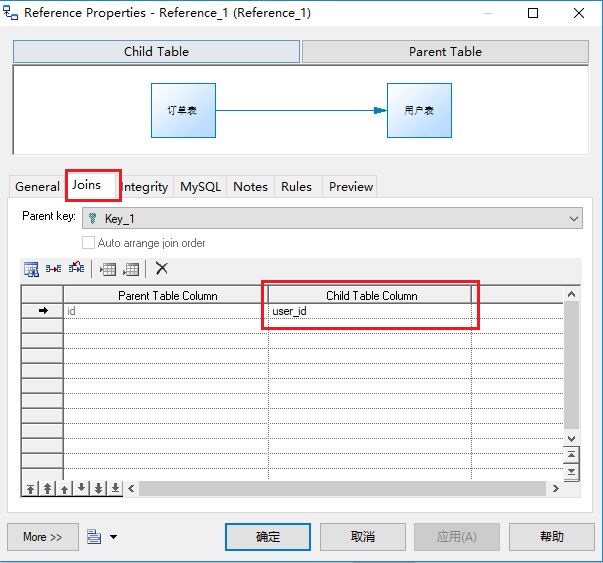
给当前两张表添加主外键关系

点击

然后将光标从订单表向用户表拖拽

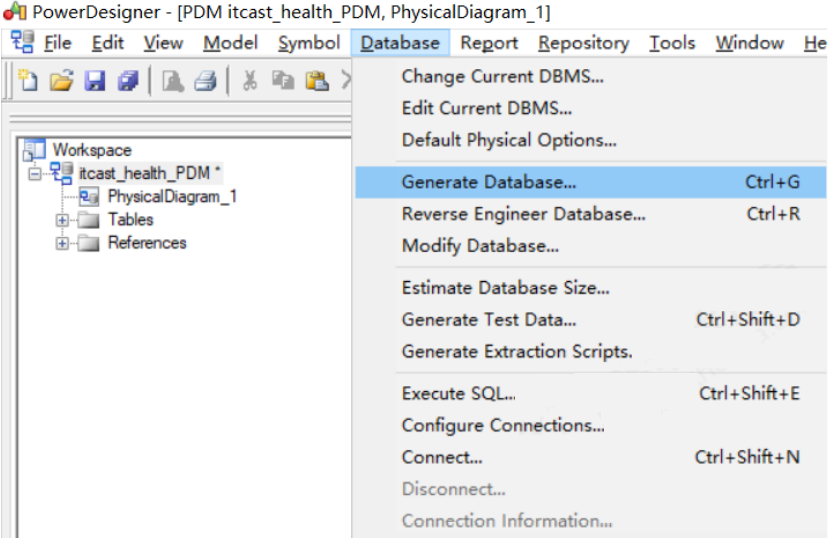


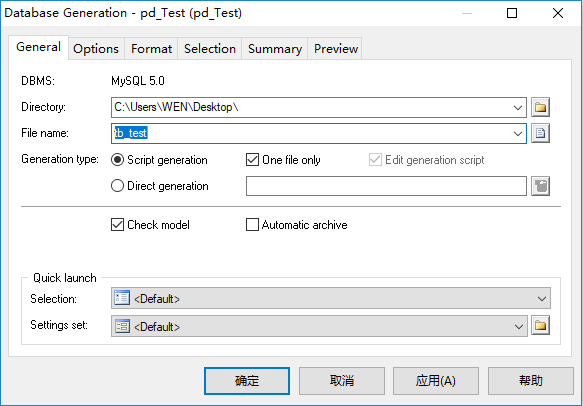
双击箭头



#### 3.1.2.3 从PDM导出SQL脚本

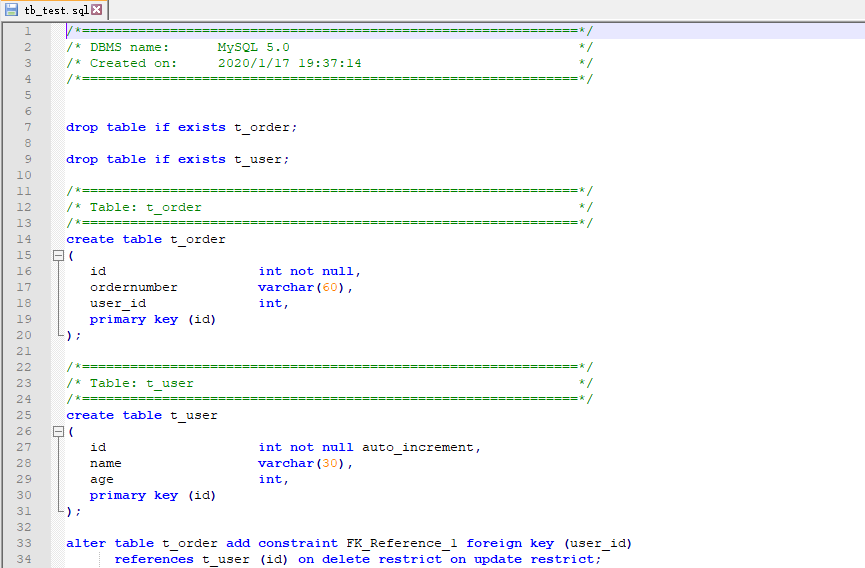
可以通过PowerDesigner设计的PDM模型导出为SQL脚本，如下：





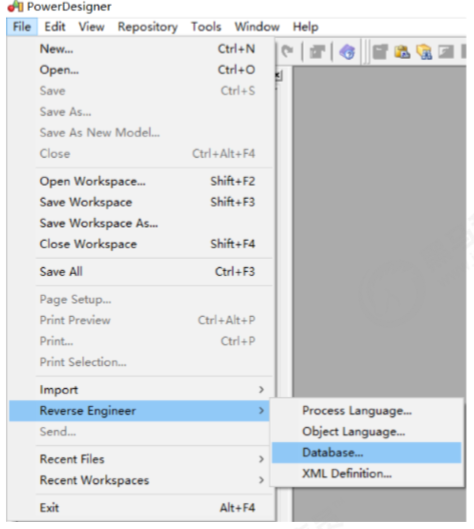
点击确定。

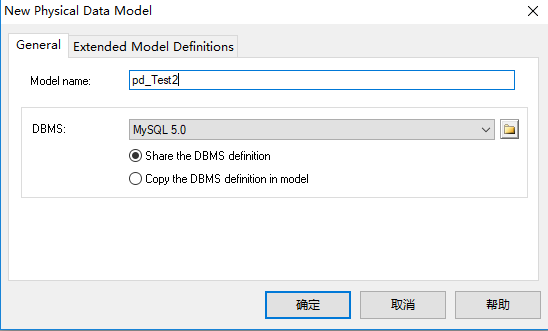
打开目录效果:



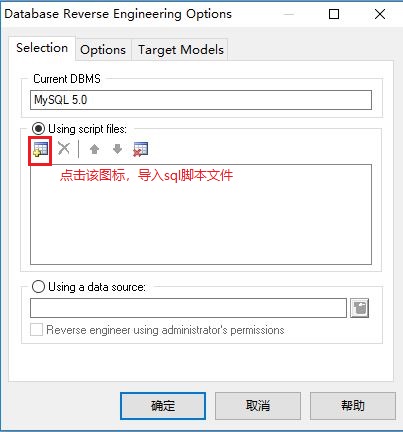
#### 3.1.2.4 逆向工程

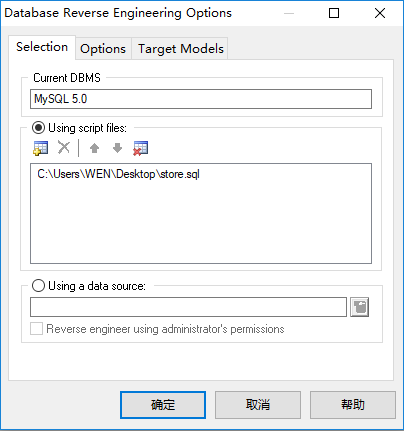
上面我们是首先创建PDM模型，然后通过PowerDesigner提供的功能导出SQL脚本。实 际上这个过程也可以反过来，也就是我们可以通过SQL脚本逆向生成PDM模型，这称为 逆向工程，操作如下：





点击确定。





点击确定即可。

