链滴

最新 Springboot +Dubbo 分布式服务搭建

摘要:

本文中涉及的代码在 github 均可找到 https://github.com/G-

little/priest，本人从最早的dubbo项目开源，便一直是dubbo的忠实粉丝,后来dubbo项目被apach
作为顶级开源项目引入也是欢欣鼓舞，作为忠实粉丝也希望为dubbo的推广尽一份绵薄之力。

<br/>

dubbo 作为分布式系统的教科书式开源项目，独立使用起来是非常简单的，但在与新版的springboot
及其他开源项目的整合使用过程中，因为不同版本开源项目的兼容问题，还是会产生各种匪夷所思的
题。作为开发界的老司机有时也被这些问题折磨的焦头烂额。本文将最新版dubbo与springboot ,my
atis 构建分布式项目的过程详细记录，希望能将大家从项目构建过程的版本坑中解救出来。

<br/>

另外也把自己开源的良心项目诚心推荐给大家 priest，

整个项目采用最新版springboot+dubbo+mybatis3+springdata-redis 架构，将传统的rest接口开
，用户token认证，及管理后台项目打包开源，所有代码均可通过插件自动生成，将开发人员从996
加班节奏中解救出来。开源不易，感兴趣的同学不妨留下你的小心心 heart 。

概述

写程序我想最讲究的就是知其然，知其所以然，提起项目整合，首先不得不说的就是dubbo的设计架
。

<center>
![image.png](https://b3logfile.com/file/2019/04/image-de29c20e.png)
</center>

首先我们看一下dubbo 的调用流程这里主要涉及4个模块:

- Registry: 注册中心，dubbo的服务注册，服务发现均通过该服务作为桥梁。

- Provider: 服务提供者，开发者实现的具体接口逻辑
- Consumer: 消费者，Provider 接口的订阅方
- Monitor: dubbo服务的调用次数，调用时间监控中心。

从图中我们可以了解到整个RPC服务调用过程为：

1. 服务提供者将服务注册到服务中心
2. 消费者在注册中心中订阅服务
3. 消费者调用已经在注册中心注册的服务

项目构建

一 基础服务

透过dubbo的架构我们可以梳理出整个项目依赖的基础服务和技术

- dubbo 分布式服务
- spring-boot 基于spring的依赖管理
- jdk 1.8
- zookeeper dubbo 依赖的注册中心
- mybatis mysql orm 框架
- mysql   数据库

二 项目结构规划

一个好的项目规划，一定要考虑到项目的层次划分明确，未来扩展，开发敏捷等等方面。 对于spring
oot+dubbo 的项目特点我们不难想到，web 项目

需要远程调用dubbo 分布式服务 ，dubbo 项目需对外提供 api 服务，因此我们得出如下项目结构：

- 将dubbo 项目分为 dubbo-service，和dubbo-api 部分，方便项目依赖和调用。
- dao 层逻辑的独立性，我们将dao 层单独拆分项目（清晰而已，不必要）
- web 层负责对代码参数的校验及dubbo 层业务逻辑的对外输出单独一个项目
- dubbo 项目代码的扩展项目
- 项目间公共代码项目

最终得到项目结构如下：

```
├── dubbo                  // dubbo 打包部署
│   ├── assembly           // dubbo assembly 打包配置
│   └── bin              // dubbo 启动相关脚本
├── dubbo-extend           // dubbo 项目扩展，可基于dubbo spi机制，对dubbo 进行扩展
│   └── src
├── plugin-test            // 插件测试项目，用于代码生成插件的测试(作者开源项目，普通项目构建
以忽略)
│   └── src
├── priest-common          // 项目共用代码
```

```
│   └── src
├── priest-common-web      // web 项目共用代码
│   └── src
├── priest-demo           // demo 项目
│   ├── priest-demo-api    // demo api 项目
│   ├── priest-demo-dao    // demo dao 项目
│   ├── priest-demo-http   // demo http 项目
│   └── priest-demo-service //demo service 项目
└── priest-generator       // 代码生成插件 (作者开源项目，普通项目构建可以忽略)
```

三 父项目pom配置说明

maven 的父子项目与java 中的继承关系非常的相似，子项目继承父项目的属性，依赖版本，插件配等，又可以根据子项目的特点和需求，重写项目配置。总结来说父项目的职责如下:

● **全局版本控制**，对于经常出现版本冲突问题的童鞋这一点至关重要，将所有项目版本的控制全部交父项目控制，当项目出现版本冲突问题时，父级项目便可通过依赖的全局版本控制，轻松解决依赖冲。

● **项目公共properties配置** 例如jdk 最低版本,项目源代码编码格式等。

● **多环境 profile 切换** ，例如开发环境，线上环境不同的注册中心配置，redis 配置

● **repositories 私有仓库配置** ，多人协同开发，api 打包发布私有仓库

● **插件管理** 类似版本依赖

下面粘贴顶级项目的pom 配置(含注释):

```
<?xml version="1.0"?>
<project
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd"
      xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLS
hema-instance">

   <modelVersion>4.0.0</modelVersion>
   <groupId>com.little.g</groupId>
   <artifactId>priest</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>pom</packaging>
   <name>priest</name>
   <url>http://maven.apache.org</url>
   <modules>
      <!-- 项目模块管理 -->
      <module>dubbo-extend</module>
      <module>priest-demo</module>
      <module>priest-common</module>
      <module>priest-common-web</module>
      <module>priest-generator</module>
      <module>plugin-test</module>
```

```
</modules>

    <!-- 全局属性配置 -->
<properties>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <resource.delimiter>${}</resource.delimiter>
    <java.version>1.8</java.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <mysql-connector-java.version>8.0.13</mysql-connector-java.version>
    <tk.mapper.version>4.1.5</tk.mapper.version>
    <dubbo.version>2.7.0</dubbo.version>
</properties>
<!-- 打包配置信息 -->
<profiles>
    <profile>
        <!-- 开发环境 -->
        <id>develop</id>
        <!-- 默认 -->
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
        <properties>
            <!-- 日志 -->
            <priest.log.level>DEBUG</priest.log.level>
            <priest.log.path>/data/logs</priest.log.path>
            <!--打包编码 -->
            <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
            <priest.dubbo.zk.url>zookeeper://127.0.0.1:2181</priest.dubbo.zk.url>
            <priest.redis.nodes>127.0.0.1:6379</priest.redis.nodes>
            <priest.redis.password>123456</priest.redis.password>
            <priest.online>false</priest.online>
        </properties>
    </profile>



    <profile>
        <!-- 线上环境 -->
        <id>online</id>
        <activation>
            <activeByDefault>false</activeByDefault>
        </activation>

        <properties>

            <!-- 日志 -->
            <priest.log.level>DEBUG</priest.log.level>
            <priest.log.path>/data/logs</priest.log.path>
            <!--打包编码 -->
            <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
            <priest.dubbo.zk.url>zookeeper://192.168.0.195:2181</priest.dubbo.zk.url>
        </properties>
    </profile>
```

```
</profiles>

<!--开发者信息-->
<developers>
    <developer>
        <name>llg.java</name>
        <id>ligang</id>
        <email>llg.java@gmail.com</email>
        <organization>xiaogang.org.cn</organization>
        <roles>
            <role>Java Developer</role>
        </roles>
    </developer>

</developers>

<!-- <distributionManagement>

    <repository>
        <id>nexus-releases</id>
        <name>Nexus Release Repository</name>
        <url>http://nexus.shuzijiayuan.com/content/repositories/releases/</url>
    </repository>
    <snapshotRepository>
        <id>nexus-snapshots</id>
        <name>Nexus Snapshot Repository</name>
        <url>http://nexus.shuzijiayuan.com/content/repositories/snapshots/</url>
    </snapshotRepository>
</distributionManagement> -->

<!-- 仓库配置 -->
<repositories>
    <repository>
        <id>apache.snapshots.https</id>
        <name>Apache Development Snapshot Repository</name>
        <url>https://repository.apache.org/content/repositories/snapshots</url>
        <releases>
            <enabled>false</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
</repositories>

<!-- 依赖管理 -->
<dependencyManagement>
    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.1.3.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>



        <!--本系统依赖管理 start-->

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-admin-common</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-admin-api</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-admin-dao</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-user-token</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-common</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-user-api</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-user-dao</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>dubbo-extend</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
```

```xml
<dependency>
   <groupId>com.little.g</groupId>
   <artifactId>priest-common-web</artifactId>
   <version>0.0.1-SNAPSHOT</version>
</dependency>

<dependency>
   <groupId>com.little.g</groupId>
   <artifactId>priest-demo-api</artifactId>
   <version>0.0.1-SNAPSHOT</version>
</dependency>

<dependency>
   <groupId>com.little.g</groupId>
   <artifactId>priest-demo-dao</artifactId>
   <version>0.0.1-SNAPSHOT</version>
</dependency>
<!--本系统依赖管理 end-->


<dependency>
   <groupId>com.github.qcloudsms</groupId>
   <artifactId>qcloudsms</artifactId>
   <version>1.0.5</version>
</dependency>


<dependency>
   <groupId>commons-fileupload</groupId>
   <artifactId>commons-fileupload</artifactId>
   <version>[1.3.3)</version>
</dependency>

<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>servlet-api</artifactId>
   <version>2.5</version>
   <scope>provided</scope>
</dependency>


<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-databind</artifactId>
   <version>[2.9.8,)</version>
</dependency>

<dependency>
   <groupId>com.fasterxml.jackson.module</groupId>
   <artifactId>jackson-module-afterburner</artifactId>
   <version>2.9.6</version>
</dependency>
```

```xml
<dependency>
    <groupId>org.reflections</groupId>
    <artifactId>reflections</artifactId>
    <version>0.9.10</version>
</dependency>

<dependency>
    <groupId>org.jodd</groupId>
    <artifactId>jodd-props</artifactId>
    <version>3.6.1</version>
</dependency>

<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>RELEASE</version>
</dependency>

<dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>RELEASE</version>
</dependency>


<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
    <version>2.1.3.RELEASE</version>
</dependency>

<!-- Apache Dubbo  -->

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-framework-bom</artifactId>
    <version>5.1.5.RELEASE</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>



<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-dependencies-bom</artifactId>
    <version>${dubbo.version}</version>
    <type>pom</type>
    <scope>import</scope>

</dependency>

<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>${dubbo.version}</version>
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </exclusion>
        <exclusion>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
        </exclusion>
        <exclusion>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
        </exclusion>
    </exclusions>
</dependency>


<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>20.0</version>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.8.1</version>
</dependency>

<dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
</dependency>

<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>

<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.10.Final</version>
</dependency>

<dependency>
```

```xml
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>2.0.0</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql-connector-java.version}</version>
        </dependency>

        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper-spring-boot-starter</artifactId>
            <version>2.1.5</version>
        </dependency>

        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper</artifactId>
            <version>${tk.mapper.version}</version>
        </dependency>

        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper-generator</artifactId>
            <version>1.1.5</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.1.10</version>
        </dependency>

        <dependency>
            <groupId>com.googlecode.libphonenumber</groupId>
            <artifactId>libphonenumber</artifactId>
            <version>7.0.6</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>[1.2.31,)</version>
        </dependency>
    </dependencies>
</dependencyManagement>


<!-- 公共构建属性及插件配置 -->
<build>
    <resources>
        <resource>
```

```xml
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
        </resource>
        <resource>
            <directory>src/main/conf</directory>
            <filtering>true</filtering>
            <targetPath>conf</targetPath>
        </resource>
    </resources>

    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-source-plugin</artifactId>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
        </plugin>
    </plugins>

    <!-- 插件版本管理 -->

    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5</version>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                    <encoding>${project.build.sourceEncoding}</encoding>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
    <version>2.3.2</version>
    <configuration>
        <archive>
            <index>true</index>
        </archive>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-source-plugin</artifactId>
    <version>2.4</version>
    <executions>
        <execution>
            <id>attach-sources</id>
            <goals>
                <goal>jar-no-fork</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifactId>
    <version>2.10</version>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.19.1</version>
    <configuration>
        <skipTests>true</skipTests>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-release-plugin</artifactId>
    <version>2.5.3</version>
    <configuration>
        <autoVersionSubmodules>true</autoVersionSubmodules>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>2.6</version>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
```

```
                <artifactId>maven-deploy-plugin</artifactId>
                <version>2.8.2</version>
            </plugin>


            <plugin>
                <groupId>org.mybatis.generator</groupId>
                <artifactId>mybatis-generator-maven-plugin</artifactId>
                <version>1.3.7</version>
                <configuration>
                    <verbose>true</verbose>
                    <overwrite>true</overwrite>
                    <configurationFile>${project.basedir}/src/test/resources/generatorConfig.xml
/configurationFile>
                </configuration>
                <dependencies>
                    <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>${mysql-connector-java.version}</version>
                    </dependency>
                    <dependency>
                        <groupId>tk.mybatis</groupId>
                        <artifactId>mapper</artifactId>
                        <version>${tk.mapper.version}</version>
                    </dependency>

                </dependencies>
            </plugin>

            <plugin>
                <groupId>com.little.g</groupId>
                <artifactId>generator-maven-plugin</artifactId>
                <version>0.0.1-SNAPSHOT</version>

                <configuration>
                    <configurationFile>${project.basedir}/src/main/conf/GenerateConfig.xml</con
igurationFile>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.1.3.RELEASE</version>
            </plugin>
        </plugins>
    </pluginManagement>
</build>

</project>
```

四 dao 项目配置

mybatis3 + mybatis-generator 组合可以称得上是 orm 界的瑞士军刀，mybatis 本身轻量，灵活性能高但不具备敏捷开发的能力，mybatis-generator 插件则完美填补了 mybatis 本身的缺陷，下让我们一睹他们的风采。

pom 依赖配置

没什么可说的直接贴带注释的pom配置:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <parent>
        <groupId>com.little.g</groupId>
        <artifactId>priest-demo</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>priest-demo-dao</artifactId>
    <packaging>jar</packaging>

    <name>priest-demo-dao</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.basedir>${project.basedir}</project.basedir>
    </properties>

    <profiles>
        <profile>
            <!-- 开发环境 -->
            <id>develop</id>
            <!-- 默认 -->
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
                <!-- 数据库配置 -->
                <priest.jdbc.driver>com.mysql.jdbc.Driver</priest.jdbc.driver>
                <priest.jdbc.url>jdbc:mysql://192.168.2.101:3306/little_g?useUnicode=true&amp;ch
racterEncoding=UTF-8&amp;autoReconnect=true</priest.jdbc.url>
                <priest.jdbc.username>priest</priest.jdbc.username>
                <priest.jdbc.password>priest</priest.jdbc.password>
            </properties>
        </profile>


        <profile>
            <!-- 线上环境  TODO -->
            <id>online</id>
            <activation>
                <activeByDefault>false</activeByDefault>
            </activation>
            <properties>
                <!-- 数据库 -->
                <priest.jdbc.driver>com.mysql.jdbc.Driver</priest.jdbc.driver>
                <priest.jdbc.url>jdbc:mysql://192.168.2.101:3306/little_g?useUnicode=true&amp;ch
racterEncoding=UTF-8&amp;autoReconnect=true</priest.jdbc.url>
                <priest.jdbc.username>priest</priest.jdbc.username>
                <priest.jdbc.password>priest</priest.jdbc.password>
            </properties>
        </profile>
    </profiles>

    <dependencies>

        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper-spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>


        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- 带监控的阿里数据库连接池 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.mybatis.generator</groupId>
                <artifactId>mybatis-generator-maven-plugin</artifactId>
                <configuration>
                        <verbose>true</verbose>
                        <overwrite>true</overwrite>
                        <configurationFile>${project.basedir}/src/test/resources/generatorConfig.xml
/configurationFile>
                    </configuration>
                    <dependencies>
                <dependency>
                    <groupId>mysql</groupId>
                    <artifactId>mysql-connector-java</artifactId>
                </dependency>
                <dependency>
                    <groupId>tk.mybatis</groupId>
                    <artifactId>mapper</artifactId>
                </dependency>

            </dependencies>
        </plugin>
    </plugins>
  </build>
</project>
```

spring 及 mybatis-generator 配置

为了让mybatis-generator 和 spring 的数据库连接配置能够共享，我们对数据库配置配置配置信息进行单独文件抽取 jdbc.properties

```
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.url=${priest.jdbc.url}
jdbc.user=${priest.jdbc.username}
jdbc.password=${priest.jdbc.password}
```

遵从dubbo 的默认配置规范，我们将所有的spring 配置文件放在

META-INF/spring 目录下，下面是 applicationContext-dao.xml 的配置:

```
<?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:tx="http://www.springframework.org/schema/tx"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/context http://www.sprin
framework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema
beans/spring-beans-3.0.xsd

    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/
pring-tx-3.0.xsd">

   <context:property-placeholder order="21" location="classpath*:jdbc.properties" file-encod
ng="UTF-8" ignore-unresolvable="true"/>

   <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init
 destroy-method="close">
       <property name="url" value="${jdbc.url}" />
       <property name="username" value="${jdbc.user}" />
       <property name="password" value="${jdbc.password}" />
       <property name="filters" value="stat" />
```

```
        <property name="maxActive" value="20" />
        <property name="initialSize" value="1" />
        <property name="maxWait" value="60000" />
        <property name="minIdle" value="1" />
        <property name="timeBetweenEvictionRunsMillis" value="60000" />
        <property name="minEvictableIdleTimeMillis" value="300000" />
        <property name="testWhileIdle" value="true" />
        <property name="testOnBorrow" value="false" />
        <property name="testOnReturn" value="false" />
        <property name="poolPreparedStatements" value="true" />
        <property name="maxOpenPreparedStatements" value="20" />

        <property name="asyncInit" value="true" />
    </bean>

    <!-- transaction manager, use JtaTransactionManager for global tx -->
    <bean id="transactionManager"
          class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!--事务模板 -->
    <bean id="transactionTemplate"
          class="org.springframework.transaction.support.TransactionTemplate">
        <property name="transactionManager" ref="transactionManager" />
        <!--ISOLATION_DEFAULT 表示由使用的数据库决定  -->
        <property name="isolationLevelName" value="ISOLATION_READ_COMMITTED"/>
        <property name="propagationBehaviorName" value="PROPAGATION_REQUIRED"/>
        <!-- <property name="timeout" value="30"/> -->
    </bean>

    <!-- enable autowire -->
    <context:annotation-config/>
    <!-- enable transaction demarcation with annotations -->
    <tx:annotation-driven mode="aspectj" transaction-manager="transactionManager" />

    <!-- define the SqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <!--要映射类的包路径，即POJO对象的路径-->
        <property name="typeAliasesPackage" value="com.little.g.*.model" />
        <!--扫描mapper文件，否则如果Mapper接口文件改名的话，就会出现找不到对应的Mapper
方法的错误-->
        <property name="mapperLocations" value="classpath*:com/little/g/*/mapper/*.xml"/>
    </bean>

    <bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg index="0" ref="sqlSessionFactory"></constructor-arg>
    </bean>

    <!-- scan for mappers and let them be autowired -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!--扫描Mapper类并使它们自动装载-->
        <property name="basePackage" value="com.little.g.**.mapper" />
    </bean>

</beans>
```

mybatis-generator 代码生成配置generatorConfig.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
        PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
        "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>

    <properties url="file://${project.basedir}/target/classes/jdbc.properties"/>

    <!--<context id="Mysql" targetRuntime="MyBatis3Simple" defaultModelType="flat">-->

    <context id="DB2Tables" targetRuntime="MyBatis3">

        <!-- 自动识别数据库关键字，默认false -->
        <property name="autoDelimitKeywords" value="true" />
        <!--可以使用``包括字段名，避免字段名与sql保留字冲突报错 -->
        <property name="beginningDelimiter" value="`" />
        <property name="endingDelimiter" value="`" />


        <plugin type="org.mybatis.generator.plugins.SerializablePlugin"></plugin>
        <plugin type="org.mybatis.generator.plugins.CaseInsensitiveLikePlugin"></plugin>
        <!-- caseSensitive默认false，当数据库表名区分大小写时，可以将该属性设置为true -->
        <!--  支持该模式
        <plugin type="tk.mybatis.mapper.generator.MapperPlugin">
            <property name="mappers" value="tk.mybatis.mapper.common.Mapper"/>
            <property name="caseSensitive" value="true"/>
        </plugin>
        -->
        <commentGenerator>
            <property name="suppressDate" value="true" />
            <property name="suppressAllComments" value="true" />
        </commentGenerator>
        <jdbcConnection driverClass="${jdbc.driverClass}"
                        connectionURL="${jdbc.url}"
                        userId="${jdbc.user}"
                        password="${jdbc.password}">
            <property name="nullCatalogMeansCurrent" value="true" />
        </jdbcConnection>

        <javaTypeResolver>
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>

        <javaModelGenerator targetPackage="com.little.g.demo.model"
                            targetProject="${project.basedir}/src/main/java">
            <property name="enableSubPackages" value="true" />
            <property name="trimStrings" value="true" />
        </javaModelGenerator>

        <sqlMapGenerator targetPackage="com.little.g.demo.mapper"
                        targetProject="${project.basedir}/src/main/resources">
            <property name="enableSubPackages" value="true" />
        </sqlMapGenerator>

        <javaClientGenerator type="XMLMAPPER"
                          targetPackage="com.little.g.demo.mapper" targetProject="${project.basedir
/src/main/java">
            <property name="enableSubPackages" value="true" />
        </javaClientGenerator>


        <table tableName="user">
            <generatedKey column="id"  sqlStatement="JDBC"/>
        </table>

    </context>
</generatorConfiguration>
```

数据库建表

demo 测试 User 表建表sql:

```
CREATE TABLE `user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT '唯一标识',
`my_name` varchar(30) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT '名字',
`age` int(11) DEFAULT NULL COMMENT '年龄',
`mobile` varchar(20) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT '手机号',
`create_time` bigint(20) DEFAULT NULL COMMENT '创建时间',
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_
nicode_ci;
```

代码生成

命令行执行:

```
mvn clean install
cd priest/priest-demo/priest-demo-dao/
mvn  mybatis-generator:generate
```

输出生成文件名即可运行代码测试了

dao 测试

创建测试代码，运行junit测试

```
package com.little.g.common.web;

import com.little.g.demo.mapper.UserMapper;
import com.little.g.demo.model.User;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import javax.annotation.Resource;

/**
 * Created by lengligang on 2019/3/9.
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath*:/META-INF/spring/*.xml")
public class UserMapperTest {
    @Resource
    private UserMapper userMapper;

    @Test
    public void testAdd(){
        User  user = new User();
        user.setMyName("张三");

        int r=userMapper.insert(user);

        Assert.assertTrue(r>0);
    }
}
```

五 api 项目配置

api 项目为web层 调用 dubbo 的接口项目即接口规范,依照项目分层的逻辑，dao 层的pojo 不可以被 api 依赖，api 层需要有自己的 pojo 即 dto:

创建dto

```
package com.little.g.demo.dto;

import java.io.Serializable;

public class UserDTO implements Serializable {
    private Integer id;
    private String myName;

    private Integer age;

    private String mobile;
    private Long createTime;

    private static final long serialVersionUID = 1L;
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getMyName() {
        return myName;
    }

    public void setMyName(String myName) {
        this.myName = myName == null ? null : myName.trim();
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getMobile() {
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile == null ? null : mobile.trim();
    }

    public Long getCreateTime() {
        return createTime;
    }

    public void setCreateTime(Long createTime) {
        this.createTime = createTime;
    }


}
```

创建 UserService

```
package com.little.g.demo.api;

import com.little.g.common.dto.ListResultDTO;
import com.little.g.common.params.TimeQueryParam;
import com.little.g.demo.dto.UserDTO;

/**
```

```
 * Created by lengligang on 2019/3/9.
 */
public interface UserService {
    /**
     * 添加
     * @param entity
     * @return
     */
    boolean add(UserDTO entity);

    /**
     * 根据id获取
     * @param id
     * @return
     */
    UserDTO get(Integer id);

    /**
     * 更新
     * @param entity
     * @return
     */
    boolean update(UserDTO entity);

    /**
     * 删除
     * @param id
     * @return
     */
    boolean delete(Integer id);

    /**
     * 增量查询
     * @param param
     * @return
     */
    ListResultDTO<UserDTO> list(TimeQueryParam param);

}
```

六 service 项目配置

配置依赖

将 priest-demo-api ,priest-demo-dao 加入service 项目依赖最终pom.xml 配置如下 :

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.ap
che.org/POM/4.0.0"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
n-4.0.0.xsd">
<parent>
```

```
<groupId>com.little.g</groupId>
<artifactId>priest-demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>priest-demo-service</artifactId>
<packaging>jar</packaging>

<name>priest-demo-service</name>
<url>http://maven.apache.org</url>



<properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<profiles>
   <profile>
      <!-- 开发环境 -->
      <id>develop</id>
      <!-- 默认 -->
      <activation>
         <activeByDefault>true</activeByDefault>
      </activation>
      <properties>

      </properties>
   </profile>

   <profile>
      <!-- 线上环境  -->
      <id>online</id>
      <activation>
         <activeByDefault>false</activeByDefault>
      </activation>
      <properties>

      </properties>
   </profile>
</profiles>

<dependencies>

   <dependency>
      <groupId>org.mybatis.spring.boot</groupId>
      <artifactId>mybatis-spring-boot-starter</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
```

```xml
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <!-- dubbo 依赖 -->

    <dependency>
        <groupId>com.little.g</groupId>
        <artifactId>dubbo-extend</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
    </dependency>

    <dependency>
        <groupId>org.apache.dubbo</groupId>
        <artifactId>dubbo</artifactId>
    </dependency>
    <dependency>
        <groupId>io.netty</groupId>
        <artifactId>netty-all</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.curator</groupId>
        <artifactId>curator-framework</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.curator</groupId>
        <artifactId>curator-recipes</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.zookeeper</groupId>
        <artifactId>zookeeper</artifactId>
    </dependency>
    <!-- dubbo 依赖 -->


    <dependency>
        <groupId>com.little.g</groupId>
        <artifactId>priest-demo-api</artifactId>
    </dependency>

    <dependency>
        <groupId>com.little.g</groupId>
        <artifactId>priest-demo-dao</artifactId>
    </dependency>
```

```
</dependencies>

<build>
    <plugins>

        <!--  assembly 打包 maven dubbo 部署包配置 -->
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <configuration>
                <finalName>${project.name}</finalName>
                <descriptor>${project.parent.parent.basedir}/dubbo/assembly/assembly.xml</descr
ptor>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

    </plugins>
</build>

</project>
```

spring 配置

config.properties 配置:

```
# 读取 parent zookeeper 配置
zookeeper.url=${priest.dubbo.zk.url}
```

META-INF/spring/spring-config.xml 配置:

```
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns="http://www.springframework.org/schema/beans" xmlns:tx="http://www.springfr
mework.org/schema/tx"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf
amework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/sch
ma/context/spring-context.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/u
il/spring-util.xsd">

    <context:property-placeholder location="classpath:config.properties" ignore-unresolvable
```

```
"true"
                                file-encoding="UTF-8" />
    <!--  service 自动扫描-->
    <context:component-scan base-package="com.little.g.**.service"/>

</beans>
```

userService 代码实现

```
package com.little.g.demo.service;

import com.little.g.common.dto.ListResultDTO;
import com.little.g.common.params.TimeQueryParam;
import com.little.g.demo.api.UserService;
import com.little.g.demo.dto.UserDTO;
import com.little.g.demo.mapper.UserMapper;
import com.little.g.demo.model.User;
import com.little.g.demo.model.UserExample;
import org.springframework.beans.BeanUtils;
import org.springframework.stereotype.Service;
import org.springframework.util.CollectionUtils;

import javax.annotation.Resource;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Created by lengligang on 2019/3/9.
 */
@Service("userService")
public class UserServiceImpl implements UserService {
    @Resource
    private UserMapper userMapper;

    @Override
    public boolean add(UserDTO entity) {
        User user=new User();
        BeanUtils.copyProperties(entity,user);
        return userMapper.insertSelective(user)>0;
    }

    @Override
    public UserDTO get(Integer id) {
        User user=userMapper.selectByPrimaryKey(id);
        if(user == null){
            return null;
        }
        UserDTO dto=new UserDTO();
        BeanUtils.copyProperties(user,dto);
        return dto;
    }

    @Override
    public boolean update(UserDTO entity) {
        if(entity.getId() == null) return false;
        User user=new User();
        BeanUtils.copyProperties(entity,user);
        return userMapper.updateByPrimaryKeySelective(user)>0;
    }

    @Override
    public boolean delete(Integer id) {
        return userMapper.deleteByPrimaryKey(id)>0;
    }

    @Override
    public ListResultDTO<UserDTO> list(TimeQueryParam param) {
        ListResultDTO<UserDTO> result=param.getResult(ListResultDTO.class);

        UserExample example = new UserExample();
        example.or().andCreateTimeLessThan(param.getLast());
        example.setOrderByClause(String.format("create_time desc limit %d",result.getLimit()));
        List<User> list= userMapper.selectByExample(example);
        if(CollectionUtils.isEmpty(list)){
            return result;
        }
        result.setLast(list.get(list.size()-1).getCreateTime());
        result.setList(list.stream().map(entity->{
            UserDTO dto=new UserDTO();
            BeanUtils.copyProperties(entity,dto);
            return dto;
        }).collect(Collectors.toList()));

        return result;
    }


}
```

dubbo 服务配置

META-INF/spring/dubbo-config.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf
amework.org/schema/beans/spring-beans-2.5.xsd
       http://dubbo.apache.org/schema/dubbo  http://dubbo.apache.org/schema/dubbo/dubb
.xsd">

    <dubbo:application name="com.little.g.demo" logger="slf4j"/>
    <dubbo:protocol name="dubbo"/>
    <dubbo:registry address="${zookeeper.url}"/>
    <dubbo:service interface="com.little.g.demo.api.UserService" ref="userService"/>

</beans>
```

dubbo 启动测试

```
package com.little.g.demo;

import org.apache.dubbo.container.Main;

/**
 * Created by lengligang on 2019/3/9.
 */
public class TestDubbo {
    public static void main(String[] args) {
        Main.main(args);
    }
}
```

dubbo junit 测试

```
package com.little.g.demo.service;

import com.little.g.demo.api.UserService;
import com.little.g.demo.dto.UserDTO;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import javax.annotation.Resource;

/**
 * Created by lengligang on 2019/3/9.
 */

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath*:/META-INF/spring/*.xml")
public class UserServiceTest{
    @Resource
    private UserService userService;
    @Test
    public void testAdd(){
        UserDTO dto= new UserDTO();
        dto.setCreateTime(System.currentTimeMillis());
        Assert.assertTrue(userService.add(dto));
    }
}
```

七 web 项目配置

项目依赖配置

web 层增加 priest-demo-api 及dubbo依赖 pom.xml 配置如下

```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/
MLSchema-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.little.g</groupId>
        <artifactId>priest-demo</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <groupId>com.little.g.demo</groupId>
    <artifactId>priest-demo-http</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>priest-demo-http</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.apache.dubbo</groupId>
            <artifactId>dubbo</artifactId>
        </dependency>

        <dependency>
```

```
            <groupId>io.netty</groupId>
            <artifactId>netty-all</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.curator</groupId>
            <artifactId>curator-framework</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.curator</groupId>
            <artifactId>curator-recipes</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
        </dependency>

        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-common-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.little.g</groupId>
            <artifactId>priest-demo-api</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

        </plugins>
    </build>

</project>
```

demo springboot 启动类编写

```
package com.little.g.demo.web;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
```

```
@SpringBootApplication
public class PriestDemoHttpApplication {

    public static void main(String[] args) {
        SpringApplication.run(PriestDemoHttpApplication.class, args);
    }

}
```

springboot 配置

```
package com.little.g.common.web.config;


import com.little.g.common.web.exception.GlobalExceptionHandler;
import com.little.g.common.web.utils.ReloadableResourceBundleMessageSource;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.web.servlet.error.ErrorAttributes;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.ImportResource;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import java.util.Locale;

/**
 * Created by lengligang on 2019/3/12.
 */
@ImportResource(locations = {"classpath:META-INF/spring/dubbo-consume.xml"})   //dubbo
配置引用
@Configuration
public class AppConfig {
     /**
      *  国际化配置文件
     */
    @Value("${spring.messages.basename}")
    private String baseName;

    /**
     *  异常统一处理
     * @return
     */
    @Bean
    GlobalExceptionHandler exceptionHandler(){
        return new GlobalExceptionHandler();
    }
    /**
     * 自定义 rest 协议格式
     * @return
     */
    @Bean
    public ErrorAttributes errorAttributes() {
        return new PriestErrorAttributes();
    }

    /**
     * 默认解析器 其中locale表示默认语言
     */
    @Bean
    public LocaleResolver localeResolver() {
        SessionLocaleResolver localeResolver = new SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.CHINA);
        return localeResolver;
    }

    /**
     * 默认拦截器 其中lang表示切换语言的参数名
     */
    @Bean
    public WebMvcConfigurer localeInterceptor() {
        return new WebMvcConfigurer() {
            @Override
            public void addInterceptors(InterceptorRegistry registry) {
                LocaleChangeInterceptor localeInterceptor = new LocaleChangeInterceptor();
                localeInterceptor.setParamName("lang");
                registry.addInterceptor(localeInterceptor);
            }
        };
    }

    @Bean
    public MessageSource messageSource(){
        ReloadableResourceBundleMessageSource messageSource=new ReloadableResourceBu
dleMessageSource();
        messageSource.setBasename(baseName);
        messageSource.setCacheSeconds(3600);
        return messageSource;
    }

    /**
     * 国际化配置
     * @return
     */
    @Bean
    public LocalValidatorFactoryBean getValidator() {
        LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
        bean.setValidationMessageSource(messageSource());
        return bean;
    }
    @Bean
    public MessageSourceUtil messageSourceUtil(){
        return new MessageSourceUtil();
    }



}
```

application.properties 配置:

```
#启动端口
server.port=8888
#国际化
spring.messages.basename=classpath*:i18n/messages
# dubbo 注册中心
zookeeper.url=${priest.dubbo.zk.url}
```

dubbo 服务引用

META-INF/spring/dubbo-consume.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf
amework.org/schema/beans/spring-beans-2.5.xsd
       http://dubbo.apache.org/schema/dubbo  http://dubbo.apache.org/schema/dubbo/dubb
.xsd">

    <dubbo:application name="com.little.g.demo.web" logger="slf4j"/>

    <dubbo:protocol name="dubbo"/>
    <dubbo:registry address="${zookeeper.url}"/>


    <dubbo:reference id="userService" interface="com.little.g.demo.api.UserService" />


</beans>
```

UserController.java 编写

```
package com.little.g.demo.web;

import com.little.g.common.ResultJson;
import com.little.g.common.dto.ListResultDTO;
import com.little.g.common.params.TimeQueryParam;
import com.little.g.demo.api.UserService;
import com.little.g.demo.dto.UserDTO;
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;

import javax.annotation.Resource;
import javax.validation.Valid;

/**
 * Created by lengligang on 2019/3/12.
 */
@RequestMapping("/user")
@RestController
public class UserController {
    @Resource
    private UserService userService;

    @RequestMapping(value = "/add",method = RequestMethod.POST)
    public ResultJson add(@Valid UserDTO params){
        ResultJson r=new ResultJson();

        if(userService.add(params)){
            return r;
        }
        r.setC(ResultJson.SYSTEM_UNKNOWN_EXCEPTION);
        return r;
    }

    @RequestMapping(value = "/del",method = RequestMethod.GET)
    public ResultJson del(@RequestParam Integer id){
        ResultJson r=new ResultJson();
        if(userService.delete(id)){
            return r;
        }
        r.setC(ResultJson.SYSTEM_UNKNOWN_EXCEPTION);
        return r;
    }

    @RequestMapping(value = "/update",method = RequestMethod.POST)
    public ResultJson update(@Valid UserDTO params){
        ResultJson r=new ResultJson();
        if(userService.update(params)){
            return r;
        }
        r.setC(ResultJson.SYSTEM_UNKNOWN_EXCEPTION);
        return r;
    }


    @RequestMapping(value = "/list",method = RequestMethod.GET)
    public ResultJson list(@Valid TimeQueryParam params){
        ResultJson r=new ResultJson();
        ListResultDTO<UserDTO> list= userService.list(params);
        r.setData(list);
        return r;
    }
}
```

八 测试

启动dubbo

TestDubbo main 方法启动 dubbo

spring-boot插件运行 spring-boot:run 启动web 项目

```
cd priest/priest-demo/priest-demo-web
mvn spring-boot:run
```

相关链接

项目源代码 https://github.com/G-little/priest

[springboot] (https://docs.spring.io/spring-boot/docs/2.1.4.RELEASE/reference/htmlsingle/)

dubbo

mybatis3

mybatis-generator