1.前言

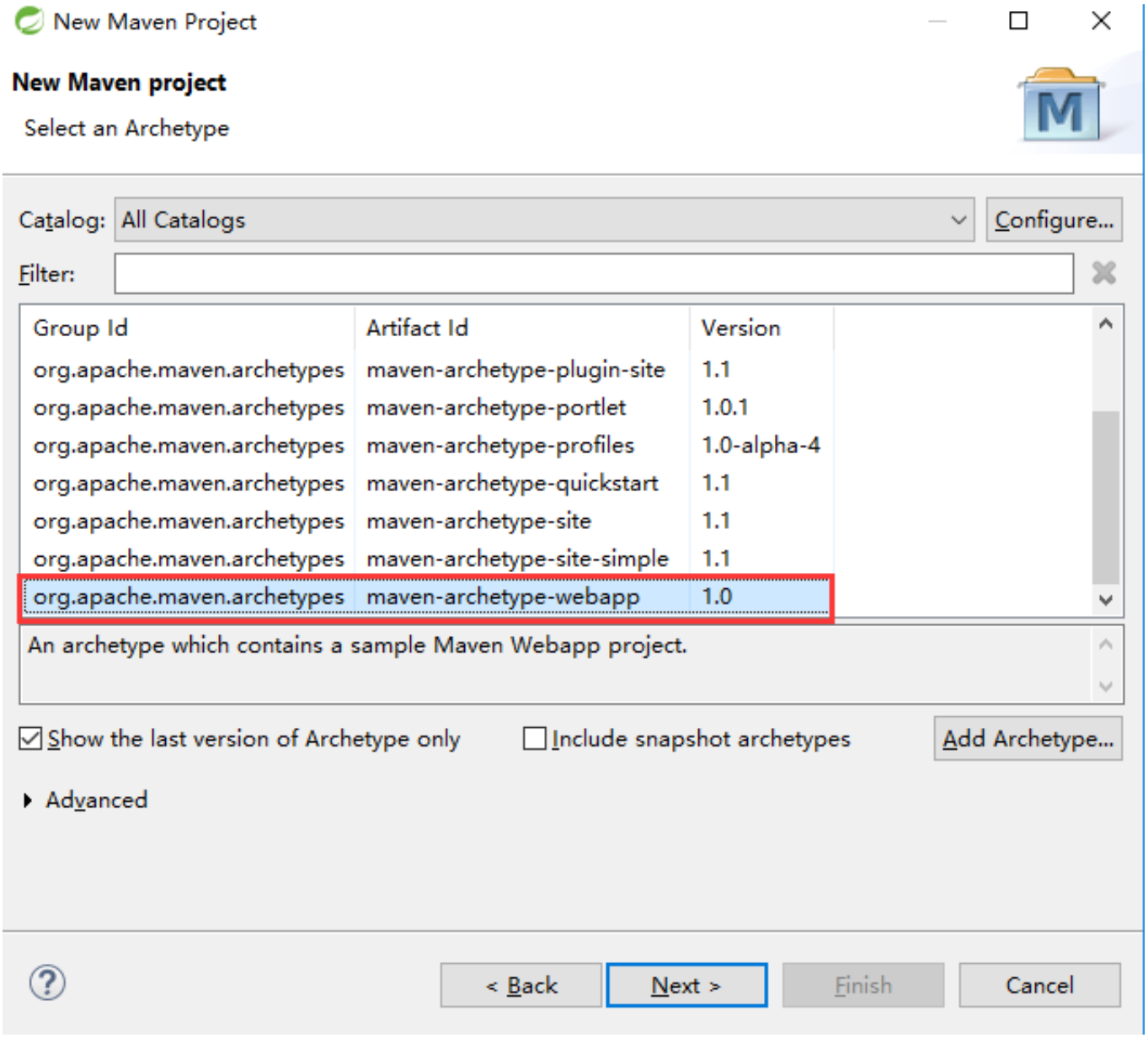
Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Boot致力于在蓬勃发展的快速应用开发领域（rapid application development）成为领导者。本文基于Spring Boot，介绍如何搭建一个Restful风格的web项目。

2.环境

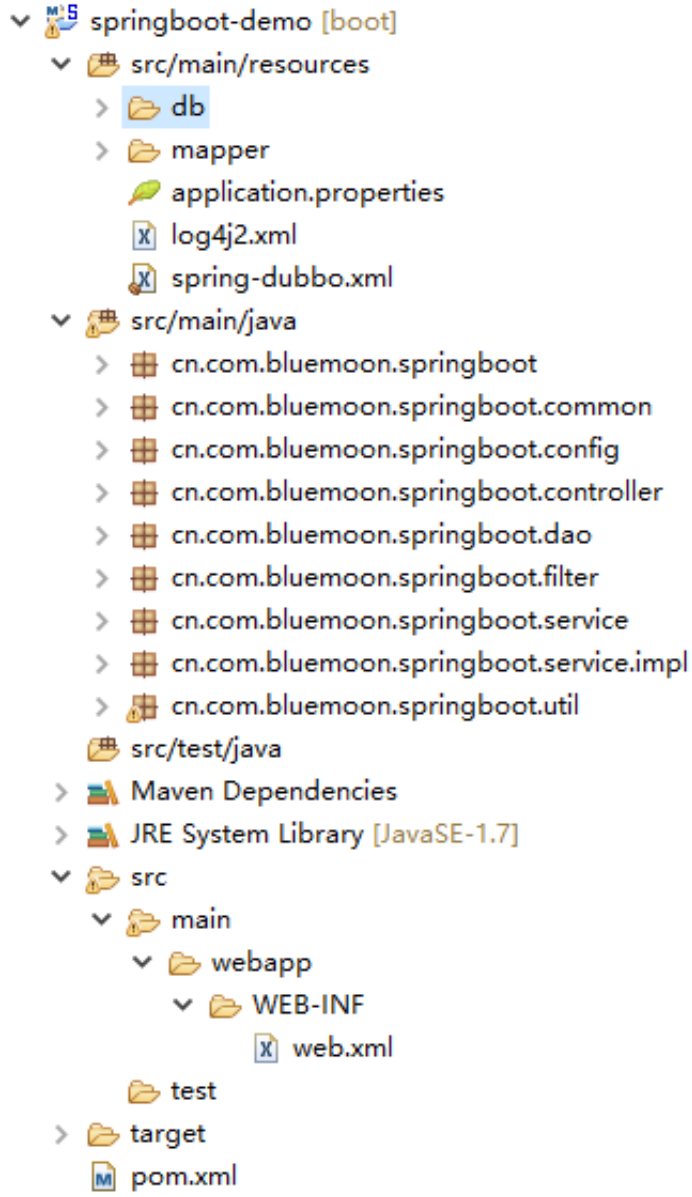
JDK 7
Eclipse
Tomcat 7
Spring Boot 1.2.3
Maven 3.2.3
Mybatis 3.2.7
Dubbo 2.5.3
Mariadb 10.0.14

3.创建项目

3.1 创建Maven项目，选择maven-archetype-webapp类型。


New Maven Project
New Maven project
Select an Archetype
Catalog: All Catalogs
Configure...
Filter:
Group Id
Artifact Id
Version
org.apache.maven.archetypes maven-archetype-plugin-site 1.1
org.apache.maven.archetypes maven-archetype-portlet 1.0.1
org.apache.maven.archetypes maven-archetype-profiles 1.0-alpha-4
org.apache.maven.archetypes maven-archetype-quickstart 1.1
org.apache.maven.archetypes maven-archetype-site 1.1
org.apache.maven.archetypes maven-archetype-site-simple 1.1
org.apache.maven.archetypes maven-archetype-webapp 1.0
An archetype which contains a sample Maven Webapp project.
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Finish
Cancel

项目结构如下：



*资源说明：

db：本示例项目用到的建表SQL；

mapper：Mybatis SQL映射文件；

application.propertis：本项目配置文件；

log4j2.xml：日志配置文件；

spring-dubbo.xml：dubbo服务提供者配置文件；

3.2 编写POM文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/200
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/m
    <modelVersion>4.0.0</modelVersion>
    <groupId>cn.com.bluemoon</groupId>
    <artifactId>spring-demo</artifactId>
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-demo</name>
    <url>http://maven.apache.org</url>

    <properties>
```

```
        <!-- 字符编码 -->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <!-- mybatis版本号 -->
        <mybatis.version>3.2.7</mybatis.version>
        <!-- jdk版本号 -->
        <jdk.version>1.7</jdk.version>
        <!-- tomcat版本号 -->
        <tomcat.version>7.0.55</tomcat.version>
        <!-- jstl版本号 -->
        <jstl.version>1.2</jstl.version>
        <!-- spring-boot版本号 -->
        <spring.boot.version>1.2.3.RELEASE</spring.boot.version>
        <!-- jsp-api版本号 -->
        <jsp-api.version>2.2</jsp-api.version>
        <!-- dubbo版本号 -->
        <dubbo.version>2.5.3</dubbo.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <!-- spring-boot依赖 -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>${spring.boot.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!-- 替换spring-boot内嵌tomcat -->
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-core</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-el</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-logging-juli</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-jasper</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-websocket</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
```

```
            <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-jdbc</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-jsp-api</artifactId>
                <version>${tomcat.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>

        <!-- dubbo（必须去除spring与log相关的包，避免包冲突） -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>dubbo</artifactId>
            <version>${dubbo.version}</version>
            <exclusions>
                <exclusion>
                    <artifactId>spring</artifactId>
                    <groupId>org.springframework</groupId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-log4j12</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-api</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>log4j</groupId>
                    <artifactId>log4j</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
            <version>3.4.6</version>
            <exclusions>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-log4j12</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-api</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>log4j</groupId>
```

```
                    <artifactId>log4j</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>com.101tec</groupId>
            <artifactId>zkclient</artifactId>
            <version>0.7</version>
            <exclusions>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-log4j12</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-api</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <!-- Servlet API and JSTL -->
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>${jstl.version}</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>${jsp-api.version}</version>
        </dependency>

        <!-- 单元测试 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>1.5.0</version>
        </dependency>

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

```
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-aop</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j2</artifactId>
        </dependency>
        <!-- provided 保证页面正常显示 begin -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
            <scope>provided</scope>
        </dependency>
        <!-- provided 保证页面正常显示 end -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jms</artifactId>
        </dependency>

        <!-- druib&dbcp连接池 -->
        <dependency>
            <groupId>com.alibaba.druid</groupId>
            <artifactId>druid-wrapper</artifactId>
            <version>0.2.9</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
        </dependency>
```

```
228.
229.         <!-- Mybatis -->
230.         <dependency>
231.             <groupId>org.mybatis</groupId>
232.             <artifactId>mybatis</artifactId>
233.             <version>${mybatis.version}</version>
234.         </dependency>
235.         <dependency>
236.             <groupId>org.mybatis</groupId>
237.             <artifactId>mybatis-spring</artifactId>
238.             <version>1.2.0</version>
239.         </dependency>
240.
241.         <!-- mysql -->
242.         <dependency>
243.             <groupId>mysql</groupId>
244.             <artifactId>mysql-connector-java</artifactId>
245.         </dependency>
246.
247.         <!-- json -->
248.         <dependency>
249.             <groupId>org.codehaus.jackson</groupId>
250.             <artifactId>jackson-mapper-asl</artifactId>
251.             <version>1.9.8</version>
252.             <type>jar</type>
253.             <scope>compile</scope>
254.         </dependency>
255.         <dependency>
256.             <groupId>net.sf.json-lib</groupId>
257.             <artifactId>json-lib</artifactId>
258.             <version>2.4</version>
259.             <classifier>jdk15</classifier>
260.         </dependency>
261.
262.         <!-- commons -->
263.         <dependency>
264.             <groupId>commons-io</groupId>
265.             <artifactId>commons-io</artifactId>
266.             <version>1.3.2</version>
267.         </dependency>
268.
269.         <dependency>
270.             <groupId>org.springframework.boot</groupId>
271.             <artifactId>spring-boot-starter-batch</artifactId>
272.         </dependency>
273.     </dependencies>
274.
275.     <build>
276.         <finalName>spring-demo</finalName>
277.         <!-- 打包时包含资源文件 -->
278.         <resources>
279.             <resource>
280.                 <directory>src/main/resources</directory>
281.                 <filtering>true</filtering>
```

```
            </resource>
        </resources>
        <plugins>
            <!-- 设定maven所用jre版本避免maven update project时项目jre变为其它版本 -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>${jdk.version}</source>
                    <target>${jdk.version}</target>
                    <encoding>${project.build.sourceEncoding}</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

*注意：spring-boot中内嵌了tomcat，本项目替换了内嵌的tomcat，是为了日后方便更改tomcat版本。引入dubbo时必须去除spring与log相关的包，避免包冲突。其他注意事项请看pom文件的注解。

3.3 编写配置文件和配置类

一般而言，spring-boot只需一个配置文件，所有项目中相关的配置都配置在此配置文件中，application.properties：

```
# 项目配置，用于本地测试
server.context-path=/spring-boot
server.port=8085

# DEV TEST PROD
env=TEST

# dubbo
dubbo_registry_address<DEV>=zookeeper://192.168.243.4:9009?backup=192.168.243.4:901
dubbo_registry_address<TEST>=zookeeper://192.168.243.4:9009?backup=192.168.243.4:90
dubbo_registry_address<PROD>=zookeeper://192.168.63.11:9011?backup=192.168.63.11:90
dubbo_port<DEV>=9050
dubbo_port<TEST>=9050
dubbo_port<PROD>=9050
dubbo_register<DEV>=false
dubbo_register<TEST>=true
dubbo_register<PROD>=true

# 数据库配置
spring.mysql.datasource.driverClassName=com.mysql.jdbc.Driver
spring.mysql.datasource.url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&charac
spring.mysql.datasource.username=root
spring.mysql.datasource.password=root
# 连接池配置
spring.mysql.datasource.filters=stat
spring.mysql.datasource.maxActive=5
spring.mysql.datasource.initialSize=1
```

```
spring.mysql.datasource.maxWait=60000
spring.mysql.datasource.minIdle=1
spring.mysql.datasource.maxIdle=3
spring.mysql.datasource.timeBetweenEvictionRunsMillis=60000
spring.mysql.datasource.minEvictableIdleTimeMillis=300000
spring.mysql.datasource.validationQuery=SELECT 'x'
spring.mysql.datasource.testWhileIdle=true
spring.mysql.datasource.testOnBorrow=false
spring.mysql.datasource.testOnReturn=false
spring.mysql.datasource.maxOpenPreparedStatements=10
spring.mysql.datasource.removeAbandoned=true
spring.mysql.datasource.removeAbandonedTimeout=1800
spring.mysql.datasource.logAbandoned=true
# druid监控页面登陆
druid.username=admin
druid.password=123
```

日志配置，log4j2.xml：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Properties>
        <Property name="FILE_PATH">E:</Property>
        <Property name="APP_NAME">springboot</Property>
        <Property name="LOG_PATTERN">[%p][%d{yyyy-MM-dd HH:mm:ss}][%C{1}:%M] %m%n
        </Property>
    </Properties>
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT" follow="true">
            <PatternLayout pattern="${LOG_PATTERN}" />
        </Console>
        <RollingFile name="RollingFile" fileName="${FILE_PATH}/${APP_NAME}.log"
            filePattern="${FILE_PATH}/${APP_NAME}_${date:yyyy-MM}/${APP_NAME}_%d{MM
            <PatternLayout pattern="${LOG_PATTERN}" />
            <SizeBasedTriggeringPolicy size="10MB" />
        </RollingFile>
    </Appenders>
    <Loggers>
        <Root level="info">
            <AppenderRef ref="Console" />
            <AppenderRef ref="RollingFile"/>
        </Root>
    </Loggers>
</Configuration>
```

dubbo服务提供者配置，spring-dubbo.xml：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://code.alibabatech.com/schema/dubbo
        http://code.alibabatech.com/schema/dubbo/dubbo.xsd
        ">

    <!-- 加载properties -->
    <bean id="placeholder"
        class="cn.com.bluemoon.springboot.common.MutilPropertyPlaceholderConfigurer
        <property name="locations">
            <list>
                <value>classpath:application.properties</value>
            </list>
        </property>
    </bean>

    <!-- 具体的实现bean -->
    <bean id="userService" class="cn.com.bluemoon.springboot.service.impl.UserServi

    <!-- 提供方应用信息，用于计算依赖关系 -->
    <dubbo:application name="spring-boot-demo" />

    <!-- 使用zookeeper注册中心暴露服务地址 -->
    <dubbo:registry protocol="zookeeper" address="${dubbo_registry_address}"
        register="${dubbo_register}" />

    <!-- 用dubbo协议在20880端口暴露服务 -->
    <dubbo:protocol name="dubbo" port="${dubbo_port}" />

    <!-- 监控中心配置，protocol="registry"，表示从注册中心发现监控中心地址 -->
    <dubbo:monitor protocol="registry" />

    <!-- 当ProtocolConfig和ServiceConfig某属性没有配置时,采用此缺省值 -->
    <dubbo:provider timeout="10000" threadpool="fixed"
        threads="100" accepts="1000" />

    <!-- 声明需要暴露的服务接口 -->
    <dubbo:service interface="cn.com.bluemoon.springboot.service.UserService"
        ref="userService" />

</beans>
```

spring-boot提倡代码式配置，所以除了配置文件外，还要相关的配置类。

应用入口类，通常用于定义bean，此类可以直接运行，启动内嵌web容器，MainApplication.java：

```
package cn.com.bluemoon.springboot;

import org.mybatis.spring.mapper.MapperScannerConfigurer;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```java
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.context.embedded.ConfigurableEmbeddedServletContain
import org.springframework.boot.context.embedded.EmbeddedServletContainerCustomizer
import org.springframework.boot.context.embedded.ErrorPage;
import org.springframework.boot.context.embedded.FilterRegistrationBean;
import org.springframework.boot.context.web.SpringBootServletInitializer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ImportResource;
import org.springframework.http.HttpStatus;
import org.springframework.web.filter.CharacterEncodingFilter;

import cn.com.bluemoon.springboot.filter.CorsFilter;
import cn.com.bluemoon.springboot.util.BeanTool;

/**
 *
 * spring-boot启动类<br>
 * 不包含数据库配置
 *
 * @author cipher
 */
@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })
@ImportResource("classpath:spring-dubbo.xml")
public class MainApplication extends SpringBootServletInitializer {

    private static final String basePackage = "cn.com.bluemoon.springboot.dao";

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder applicati
        return application.sources(MainApplication.class);
    }

    /**
     *
     * 功能描述: <br>
     * 启动主函数。正式发布时应注释掉。
     */
    public static void main(String[] args) {
        SpringApplication.run(MainApplication.class, args);
    }

    /**
     * bean工具类，可以在普通类中获取spring创建的bean
     */
    @Bean
    public BeanTool beanTool() {
        return new BeanTool();
    }

    /**
     * HTTP编码
     */
    @Bean
```

```
60.     public FilterRegistrationBean filterRegistrationBean() {
61.         FilterRegistrationBean registrationBean = new FilterRegistrationBean();
62.         CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilt
63.         registrationBean.setFilter(characterEncodingFilter);
64.         characterEncodingFilter.setEncoding("UTF-8");
65.         characterEncodingFilter.setForceEncoding(true);
66.         registrationBean.setOrder(Integer.MIN_VALUE);
67.         registrationBean.addUrlPatterns("/*");
68.         return registrationBean;
69.     }
70. 
71.     /**
72.      * 跨域请求过滤
73.      */
74.     @Bean
75.     public FilterRegistrationBean corsRegistrationBean() {
76.         FilterRegistrationBean registrationBean = new FilterRegistrationBean();
77.         CorsFilter corsFilter = new CorsFilter();
78.         registrationBean.setFilter(corsFilter);
79.         registrationBean.setOrder(Integer.MIN_VALUE);
80.         registrationBean.addUrlPatterns("/*");
81.         return registrationBean;
82.     }
83. 
84.     /**
85.      * 扫描指定包下的dao
86.      */
87.     @Bean
88.     public MapperScannerConfigurer mapperScannerConfigurer() {
89.         MapperScannerConfigurer bean = new MapperScannerConfigurer();
90.         bean.setBasePackage(basePackage);
91.         return bean;
92.     }
93. 
94.     /**
95.      * 异常处理
96.      */
97.     @Bean
98.     public EmbeddedServletContainerCustomizer containerCustomizer() {
99.         /**
100.          * 自定义ServletContainer，集中处理异常
101.          *
102.          * @author cipher
103.          */
104.         class MyCustomizer implements EmbeddedServletContainerCustomizer {
105.             @Override
106.             public void customize(ConfigurableEmbeddedServletContainer container) {
107.                 container.addErrorPages(new ErrorPage(HttpStatus.BAD_REQUEST, "/400
108.                 container.addErrorPages(new ErrorPage(HttpStatus.NOT_FOUND, "/404")
109.             }
110.         }
111.         return new MyCustomizer();
112.     }
113. 
```

```
}
```

对于本项目而言，需要数据库配置类，配置类中的成员变量均读取application.properties中对应的key，DataSourceConfig.java：

```
package cn.com.bluemoon.springboot.config;

import java.io.IOException;
import java.sql.SQLException;
import javax.sql.DataSource;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.embedded.FilterRegistrationBean;
import org.springframework.boot.context.embedded.ServletRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.springframework.core.io.support.ResourcePatternResolver;
import com.alibaba.druid.pool.DruidDataSource;
import com.alibaba.druid.support.http.StatViewServlet;
import com.alibaba.druid.support.http.WebStatFilter;

/**
 * 数据库配置类
 *
 * @author cipher
 *
 */
@Configuration
public class DataSourceConfig {

    @Value("${spring.mysql.datasource.driverClassName}")
    private String driverClassName;
    @Value("${spring.mysql.datasource.url}")
    private String url;
    @Value("${spring.mysql.datasource.username}")
    private String username;
    @Value("${spring.mysql.datasource.password}")
    private String password;
    @Value("${spring.mysql.datasource.filters}")
    private String filters;
    @Value("${spring.mysql.datasource.maxActive}")
    private int maxActive;
    @Value("${spring.mysql.datasource.initialSize}")
    private int initialSize;
    @Value("${spring.mysql.datasource.maxWait}")
    private long maxWait;
    @Value("${spring.mysql.datasource.minIdle}")
```

```
47.     private int minIdle;
48.     @Value("${spring.mysql.datasource.timeBetweenEvictionRunsMillis}")
49.     private long timeBetweenEvictionRunsMillis;
50.     @Value("${spring.mysql.datasource.minEvictableIdleTimeMillis}")
51.     private long minEvictableIdleTimeMillis;
52.     @Value("${spring.mysql.datasource.validationQuery}")
53.     private String validationQuery;
54.     @Value("${spring.mysql.datasource.testWhileIdle}")
55.     private boolean testWhileIdle;
56.     @Value("${spring.mysql.datasource.testOnBorrow}")
57.     private boolean testOnBorrow;
58.     @Value("${spring.mysql.datasource.testOnReturn}")
59.     private boolean testOnReturn;
60.     @Value("${spring.mysql.datasource.removeAbandoned}")
61.     private boolean removeAbandoned;
62.     @Value("${spring.mysql.datasource.logAbandoned}")
63.     private boolean logAbandoned;
64.     @Value("${spring.mysql.datasource.maxOpenPreparedStatements}")
65.     private int maxOpenPreparedStatements;
66.     @Value("${spring.mysql.datasource.removeAbandonedTimeout}")
67.     private int removeAbandonedTimeout;
68.     @Value("${druid.username}")
69.     private String druidUser;
70.     @Value("${druid.password}")
71.     private String druidPassword;
72. 
73.     /**
74.      * druid 数据库连接池
75.      *
76.      * @return
77.      */
78.     @Bean(name = "mysqlDS")
79.     @Qualifier("mysqlDS")
80.     @Primary
81.     public DataSource dataSource() {
82.         DruidDataSource dataSource = new DruidDataSource();
83.         dataSource.setUrl(url);
84.         dataSource.setUsername(username);
85.         dataSource.setPassword(password);
86.         dataSource.setDriverClassName(driverClassName);
87.         try {
88.             dataSource.setFilters(filters);
89.         } catch (SQLException e) {
90.             e.printStackTrace();
91.             return dataSource;
92.         }
93.         dataSource.setMaxActive(maxActive);
94.         dataSource.setInitialSize(initialSize);
95.         dataSource.setMaxWait(maxWait);
96.         dataSource.setMinIdle(minIdle);
97.         dataSource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);
98.         dataSource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);
99.         dataSource.setValidationQuery(validationQuery);
100.         dataSource.setTestWhileIdle(testWhileIdle);
```

```
101.            dataSource.setTestOnBorrow(testOnBorrow);
102.            dataSource.setTestOnReturn(testOnReturn);
103.            dataSource.setMaxOpenPreparedStatements(maxOpenPreparedStatements);
104.            dataSource.setRemoveAbandoned(removeAbandoned);
105.            dataSource.setRemoveAbandonedTimeout(removeAbandonedTimeout);
106.            dataSource.setLogAbandoned(logAbandoned);
107.            dataSource.setPoolPreparedStatements(false);
108.            return dataSource;
109.        }
110.
111.        /**
112.         * Mybatis SqlSessionFactory 与mapper关联
113.         *
114.         * @return
115.         */
116.        @Bean
117.        public SqlSessionFactoryBean sqlSessionFactory() {
118.            SqlSessionFactoryBean bean = new SqlSessionFactoryBean();
119.            ResourcePatternResolver resolver = new PathMatchingResourcePatternResolver(
120.            try {
121.                Resource[] resources = resolver.getResources("classpath*:mapper/*.xml")
122.                bean.setDataSource(this.dataSource());
123.                bean.setMapperLocations(resources);
124.            } catch (IOException e) {
125.                e.printStackTrace();
126.            }
127.            return bean;
128.        }
129.
130.        /**
131.         * druid 监控页面
132.         *
133.         * @return
134.         */
135.        @Bean
136.        public ServletRegistrationBean druidServletBean() {
137.            ServletRegistrationBean registrationBean = new ServletRegistrationBean();
138.            StatViewServlet statViewServlet = new StatViewServlet();
139.            registrationBean.addInitParameter("loginUsername", druidUser);
140.            registrationBean.addInitParameter("loginPassword", druidPassword);
141.            registrationBean.addInitParameter("resetEnable", "true");
142.            registrationBean.addUrlMappings("/druid/*");
143.            registrationBean.setServlet(statViewServlet);
144.            return registrationBean;
145.        }
146.
147.        /**
148.         * druid 资源监控过滤
149.         *
150.         * @return
151.         */
152.        @Bean
153.        public FilterRegistrationBean druidWebStatFilter() {
154.            FilterRegistrationBean registrationBean = new FilterRegistrationBean();
```

```
        WebStatFilter webStatFilter = new WebStatFilter();
        registrationBean.addInitParameter("sessionStatMaxCount", "2000");
        registrationBean.addInitParameter("sessionStatEnable", "true");
        registrationBean.addInitParameter("principalSessionName", "session_user_key
        registrationBean.addInitParameter("profileEnable", "true");
        registrationBean.addInitParameter("exclusions", "*.js,*.gif,*.jpg,*.png,*.c
        registrationBean.setFilter(webStatFilter);
        registrationBean.addUrlPatterns("/*");
        return registrationBean;
    }

}
```

3.4 编写DAO与Mapper文件

UserDao.java：

```
package cn.com.bluemoon.springboot.dao;

import java.util.List;
import java.util.Map;

public interface UserDao {

    List<Map<String, Object>> getData(Map<String, Object> map);

}
```

user.xml：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dt

<mapper namespace="cn.com.bluemoon.springboot.dao.UserDao">

    <select id="getData" resultType="java.util.HashMap" parameterType="java.util.Ha
        SELECT
            u.`name`,
            u.age,
            u.sex,
            u.create_time,
            u.update_time
        FROM user u
        WHERE u.`name` = #{name}
    </select>

</mapper>
```

3.5 编写Service与Impl

UserService.java：

```java
package cn.com.bluemoon.springboot.service;

import java.util.HashMap;

import cn.com.bluemoon.springboot.common.RespBean;

public interface UserService {

    RespBean getData(HashMap<String, Object> param);

}
```

UserServiceImpl.java：

```java
package cn.com.bluemoon.springboot.service.impl;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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

import cn.com.bluemoon.springboot.common.RespBean;
import cn.com.bluemoon.springboot.dao.UserDao;
import cn.com.bluemoon.springboot.service.UserService;

public class UserServiceImpl implements UserService {

    @Autowired
    private UserDao userDao;

    @Override
    public RespBean getData(HashMap<String, Object> param) {
        RespBean result = new RespBean();
        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
        list = userDao.getData(param);
        result.setRespData(list);
        return result;
    }

}
```

3.6 编写Controller

异常Controller：

```java
package cn.com.bluemoon.springboot.controller;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.com.bluemoon.springboot.common.RespBean;

/**
 *
 * 异常controller<br>
 * 集中处理异常
 *
 * @author cipher
 */
@Controller
@ControllerAdvice
public class ErrorController {

    private static final Logger LOGGER = LoggerFactory.getLogger(ErrorController.cl

    @RequestMapping(value = { "/404" })
    @ResponseBody
    public RespBean notFound(HttpServletRequest request) {
        RespBean result = new RespBean();
        result.setRespCode("404");
        result.setRespDesc("找不到页面");
        return result;
    }

    @RequestMapping(value = { "/400" })
    @ResponseBody
    public RespBean badRequest(HttpServletRequest request) {
        RespBean result = new RespBean();
        result.setRespCode("400");
        result.setRespDesc("数据处理校验失败");
        return result;
    }

    @ExceptionHandler(Exception.class)
    @ResponseBody
    public RespBean handleException(Exception e, HttpServletRequest request) {
        RespBean result = new RespBean();
        result.setRespCode("500");
        result.setRespDesc("处理失败");
        LOGGER.error("" + e);
        return result;
```

```
    }
}
```

UserController.java：

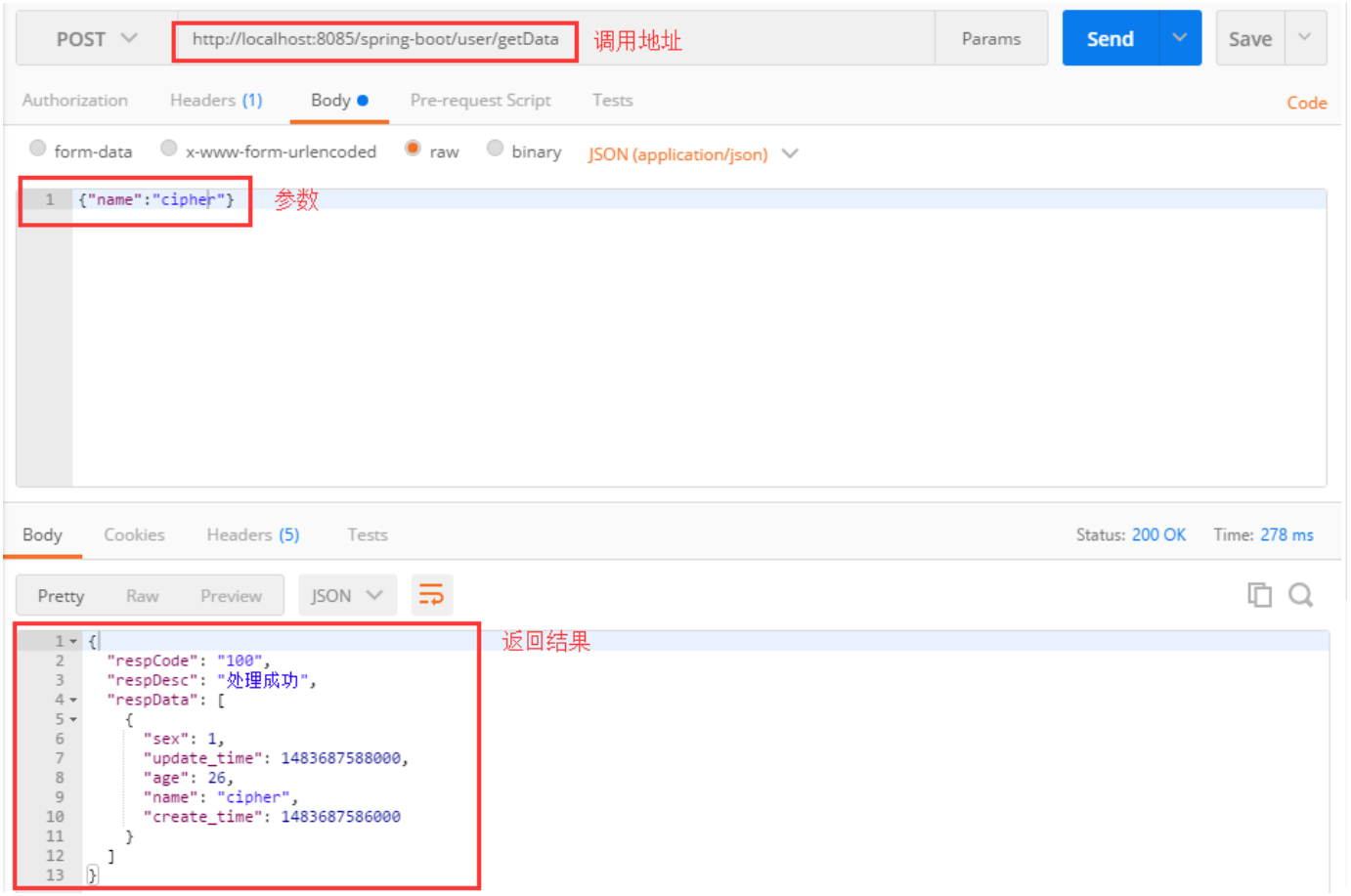
```
package cn.com.bluemoon.springboot.controller;

import java.util.HashMap;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.com.bluemoon.springboot.common.RespBean;
import cn.com.bluemoon.springboot.service.UserService;

@Controller
@RequestMapping("/user")
public class UserController {

    @Autowired
    private UserService userService;

    @RequestMapping(value = "getData", method = RequestMethod.POST, produces = "app
    @ResponseBody
    public RespBean getData(@RequestBody HashMap<String, Object> jsonParam) {
        return userService.getData(jsonParam);
    }

}
```

3.7 测试服务

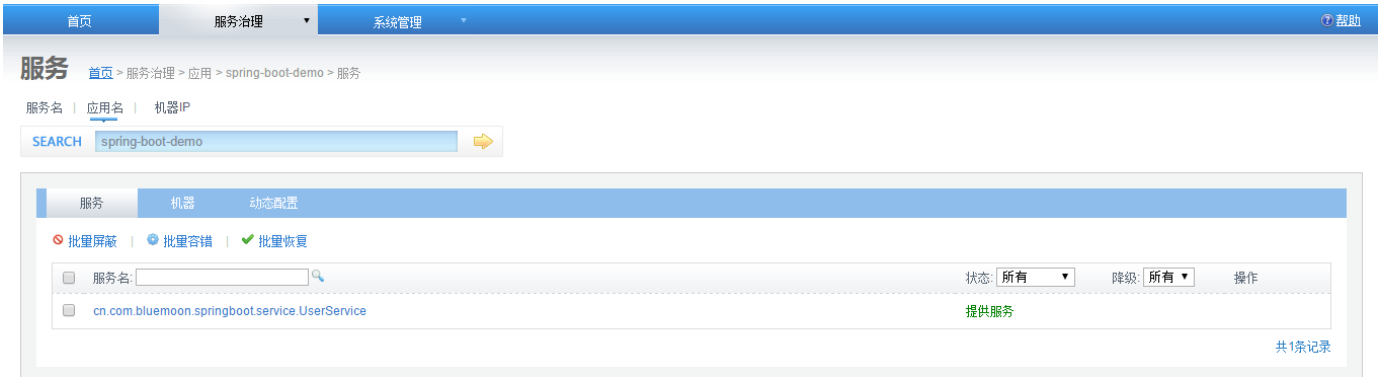
运行入口类，MainApplication，

```
[INFO][2017-01-09 11:13:02][StartupInfoLogger:logStarted] Started MainApplication i
```

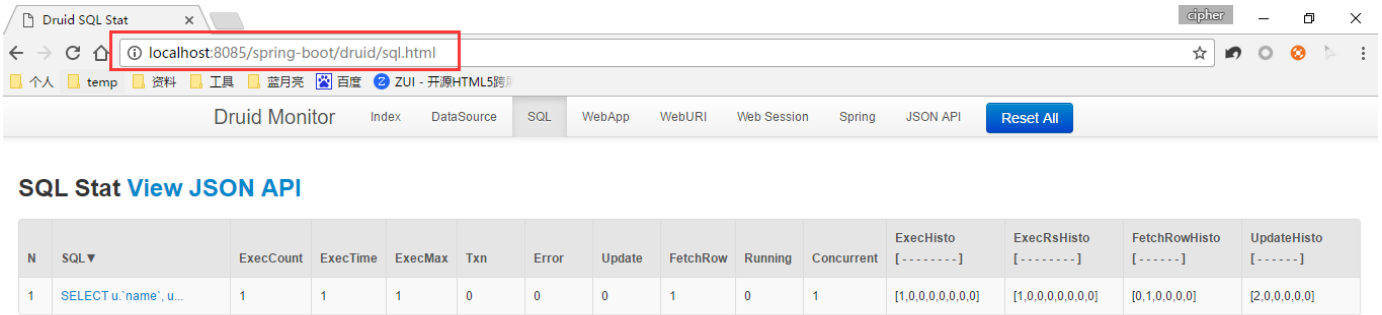
看到上述日志信息就代表应用已启动成功，使用postman进行接口测试，测试POST方法：



可以在dubbo注册中心看到注册的服务信息：



本项目使用druid连接池，可登陆监控页面查看：



4.总结

以上就是Spring Boot项目搭建的主要步骤，看起来步骤好像很多，但是很多地方只需配置一次，

而且结构清晰，一目了然。Spring Boot已经默认集成了许多第三方组件，使用时只需在application配置文件中添加即可，后续的业务开发只需重点关注DAO、Service、Controller层，而其内嵌了Web容器，使得本地测试也变得简单，甚至可以集成Swagger，方便接口测试。

5.资料

a.源码，有相当多的例子：

https://github.com/spring-projects/spring-boot

b.官方文档，基本所有问题都能在这找到答案：

http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/

c.配置文件示例，配置其他组件的例子：

http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

附件：