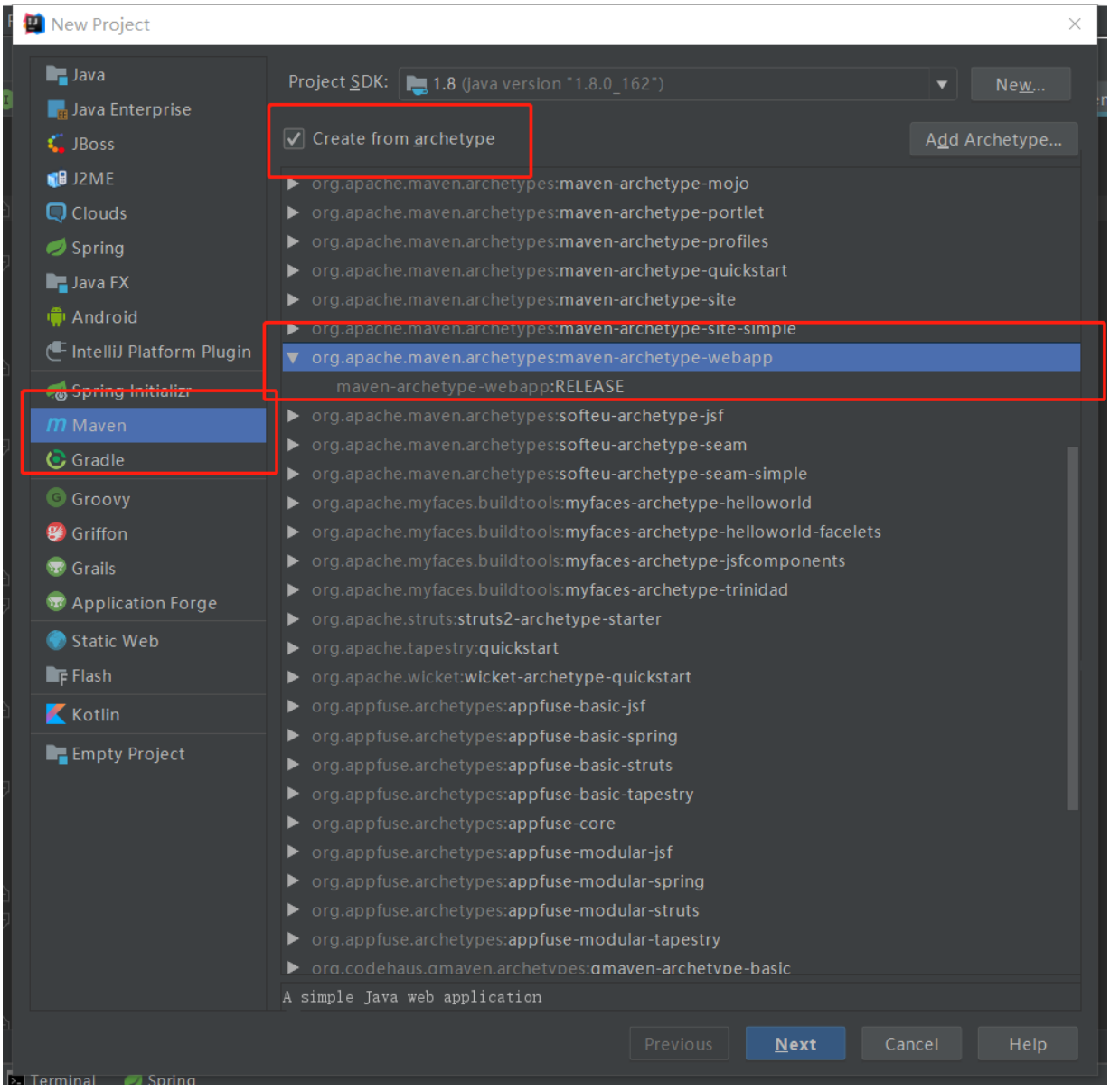
# IDEA+Maven+SSM 框架整合步骤

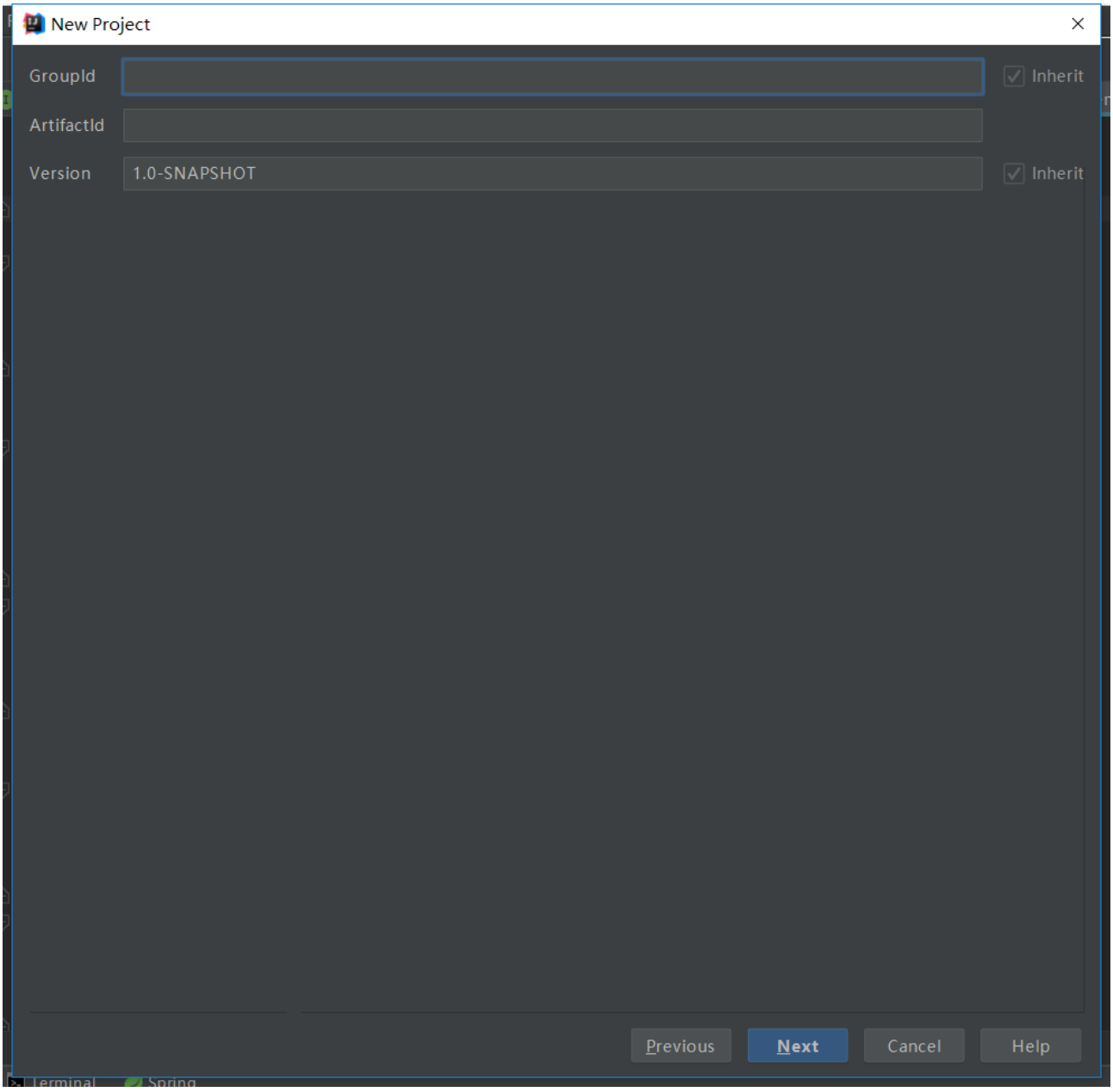
因为前段时间自己想写个 SSM 的 demo，然而不知怎么回事，配置完之后出现错误，怎么都调不好。最后从朋友那里拷了一个 SSM 的 demo 过来搭建成功，写这篇东西也是为了以后如果还有需要可以方便的查阅，并且也可能帮到一些别的和我一样的新程序员。

首先说一下版本，我采用的是 IDEA 2018，maven 版本是 3.5.3，需要额外说明的是，maven 和 jdk 有版本对应 k 要求，可以去官网查看，http://maven.apache.org/docs/history.html。SSM 的版本可以在之后的 pom.xml 里看到。好了，接下来开始搭建 SSM 工程。

1. 第一步用 IDEA 创建工程。在左上角依次点击 New File -> new -> Project... 然后左侧列表中，选中 Maven 来新建一个 Maven 工程。然后勾上 Create from archetype 来创建默认的目录，一般 web 项目都是选择 maven-archetype-webapp，不要选错成上面的 cocoon-22-archetype-webapp。最后点击 next，如下图

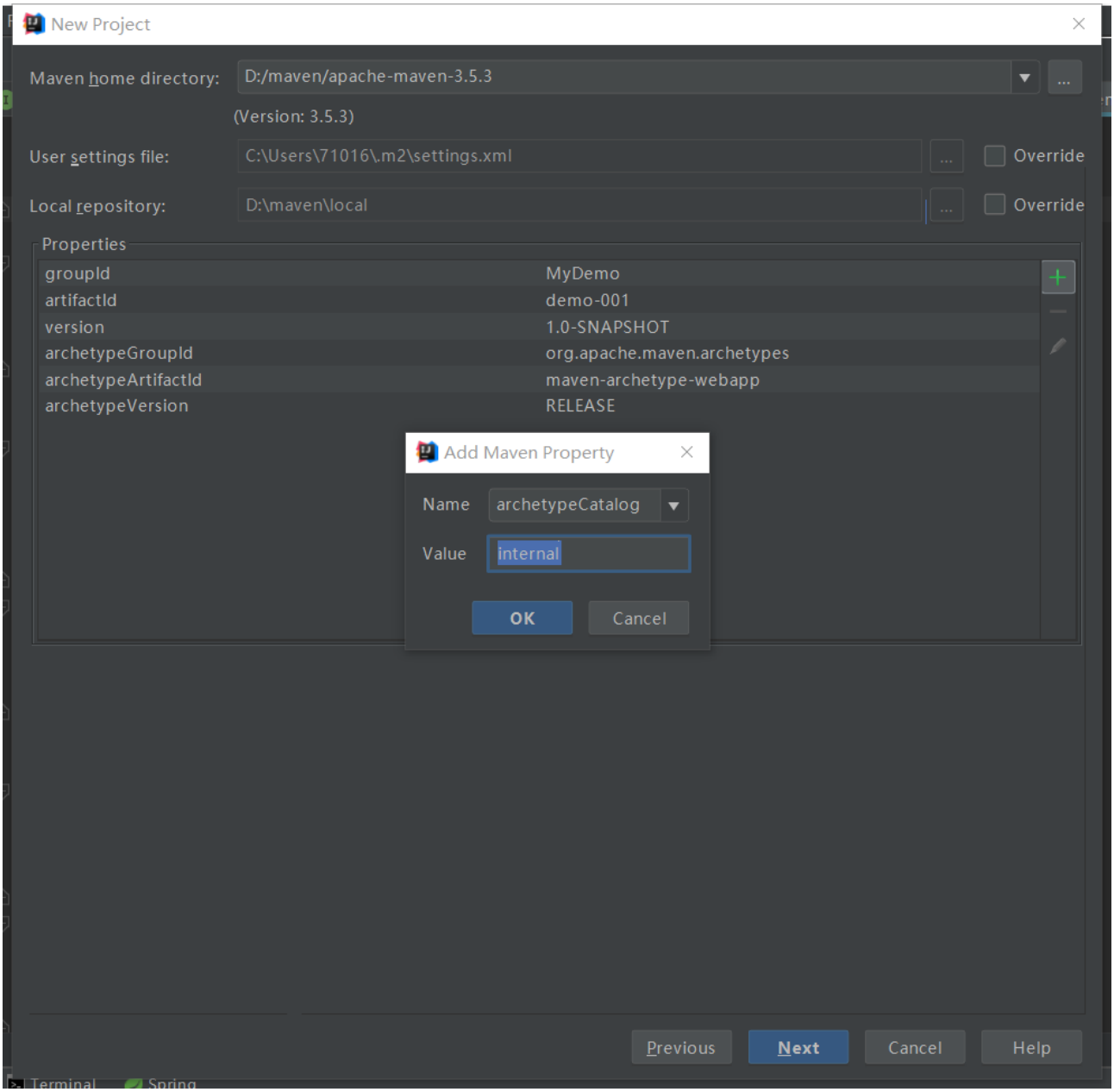


2. 然后填写自己的 GroupID 和 ArtifactID。前者是指项目组织的唯一标识名，一般是域名+公司名+项目名，我这里就随意填了，叫 MyDemo。后者是指项目唯一标识符，一般就是项目名，我这里填 demo-001。




3. 然后到了选择 maven 版本，这里就选择自己安装的 maven 版本就好，然后 Local repository 是配置 Maven 时选择的本地 jar 包仓库文件夹。然后理论上就可以继续下一步了，但一般来说国内 maven 有时候下载一些东西会很慢，导致工程的搭建速度会很慢。这里可以点击右边绿色的加号，然后填写 Name：archetypeCatalog，Value：internal。因为 maven 在构建时会下载一个配置文件，有几 M 大小，容易卡住所以用这个方式使用内置的 archetype-catalog.xml 文件。还有别的方式可以解决，这里就不表了，可以直接去搜索 archetypeCatalog internal 就可

以找到其他的方法。接着继续点击 next。




4. 然后就是一些本地的设置，比如项目名以及项目本地的存放路径。设置完点击 finish 来打开工程即可。
5. 然后我们能看到工程已经被搭建好了，然后我们要做的事还有引入依赖的 jar 包，完善默认的项目目录。先来引入工程依赖的 jar 包，可以看到有一个蓝色 m 图标的文件叫 pom.xml，就是在这个文件中输入依赖的 jar 包信息就可以导入了。这里给出一个依赖的清单，把这段 xml 代码直接复制到 dependencies 标签中，然后再点击右小角 IDEA 弹出的 Import changes，也可以点击 Enable Auto-Import 来使用自动引入。pom.xml 的 dependencies 标签内的配置代码如下：

```
        <!-- 单元测试 -->
        <dependency>
            <groupId>junit</groupId>
```

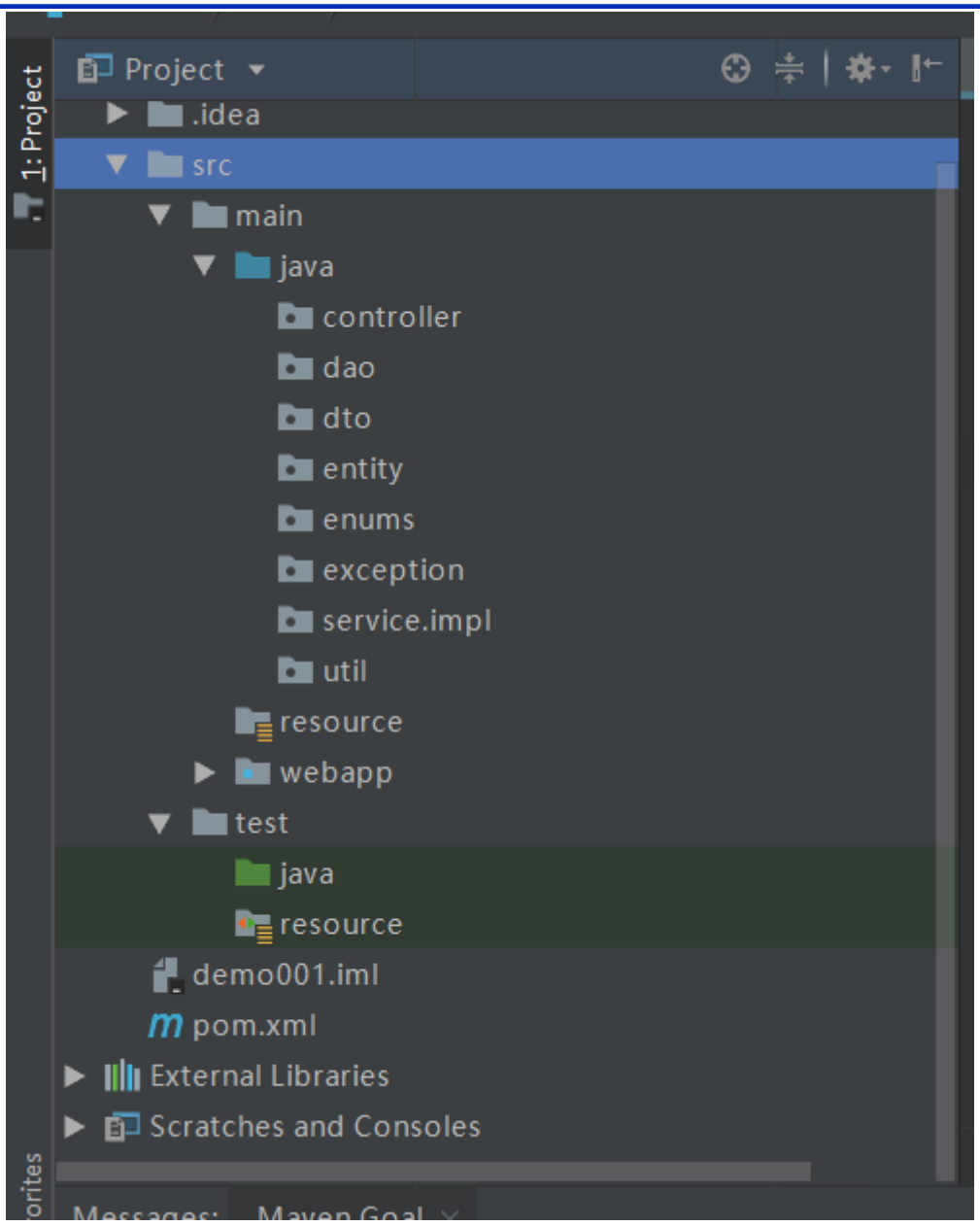
```
        <artifactId>junit</artifactId>
        <version>4.11</version>
    </dependency>

    <!-- 1.日志 -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.6</version>
    </dependency>
    <!-- 实现 slf4j 接口并整合 -->
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.1.1</version>
    </dependency>

    <!-- 2.数据库 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.37</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>
    </dependency>

    <!-- DAO: MyBatis -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.3.0</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>1.2.3</version>
    </dependency>

    <!-- 3.Servlet web -->
    <dependency>
```

```
        <groupId>taglibs</groupId>
        <artifactId>standard</artifactId>
        <version>1.1.2</version>
    </dependency>
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.5.4</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
    </dependency>

    <!-- 4.Spring -->
    <!-- 1)Spring 核心 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <!-- 2)Spring DAO 层 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <dependency>
```

```
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <!-- 3)Spring web -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>
    <!-- 4)Spring test -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>4.1.7.RELEASE</version>
    </dependency>

    <!-- wx-tools 依赖包 -->
    <!-- HttpClient -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.6</version>
    </dependency>
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2</version>
    </dependency>
    <!-- http -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpmime</artifactId>
        <version>4.3.6</version>
    </dependency>
    <!-- XML -->
    <dependency>
        <groupId>com.thoughtworks.xstream</groupId>
        <artifactId>xstream</artifactId>
```

```
            <version>1.4.7</version>
        </dependency>
        <dependency>
            <groupId>org.dom4j</groupId>
            <artifactId>dom4j</artifactId>
            <version>2.0.0</version>
        </dependency>
        <!-- IO -->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.4</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/commons-fileupload/commons-
fileupload -->
        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
            <version>1.3.1</version>
        </dependency>
        <!-- Json -->
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180130</version>
        </dependency>
        <!-- 分页 PageHelper -->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>4.1.6</version>
        </dependency>
```

6. 如果没有问题的话，那么已经成功导入了一般项目需要用的 jar 包了。接下来把默认的目录补全。这一步可做也可不做，比如有时只是做一个很小的项目，可能不需要那么清晰的分层。首先在 src 下的 main 文件夹下新建一个 java 和 resource 文件夹，然后右键文件夹名，在菜单的下面找到 Mark Directory as，将 java 文件夹设置为 Sources Root，resource 文件夹设置为 Resources Root。然后在 src 下建立一个和 main 同级的 test 文件夹，然后在 test 下再建立一个 java 和 resource 文件夹，分别标记为 Test Sources Root 和 Test Resources Root。然后我们在 main 下的 java 文件夹根据需要来新建几个包，dao、dto、controller、entity、enums、exception、service、util 等。其中 service 下再建一个子目录叫 impl。建完的目录大致如下：




7. 首先需要在 **webapp/WEB-INF** 下新建一个 jsp 目录，用来存放 jsp 页面。之所以建在 WEB-INF 文件夹下，是因为 WEB-INF 下的资源无法通过浏览器直接获取，有较高安全性。然后接下来就是配置 web.xml 文件，代码如下（红名路径不用管，等配置结束后即可）：

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0"
         metadata-complete="true">
  <!-- 修改 servlet 版本为 3.1 -->
  <!-- 配置 DispatcherServlet -->
  <servlet>
```

```
    <servlet-name>demo-dispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 配置 springMVC 需要加载的配置文件
        spring-dao.xml,spring-service.xml,spring-web.xml
        Mybatis -> spring -> springMVC
     -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring/spring-*.xml</param-value>
    </init-param>
  </servlet>
  <filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter
</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>utf8</param-value>
    </init-param>
  </filter>

  <filter-mapping>
    <filter-name>encodingFilter</filter-name >
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <servlet-mapping>
    <servlet-name>demo-dispatcher</servlet-name>
    <!-- 默认匹配所有的请求 -->
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <!-- 这里配置默认进入的主页，根据需要填写 -->
  <welcome-file-list>
    <welcome-file>home.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

8. 接下来就是添加 Spring 的配置文件了，这里将 Spring 的配置文件分成三部分，dao、service 和 web。首先先在 **main/resource** 下建一个目录叫 spring。然后分别新建三个文件：**spring-dao.xml**、**spring-service.xml**、**spring-web.xml**。**spring-dao.xml** 代码如

下（其中${ jdbc.xxx }是自己的数据库连接属性，也可以在 **main/resource** 下新建一个 **jdbc.properties** 来存放连接属性，在 properties 中使用的是键值对保存，举例：key 是 jdbc.url = 数据库 URL）

```
<?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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd" >
    <!-- 配置整合 mybatis 过程 -->
    <!-- 1.配置数据库相关参数 properties 的属性：${url} -->
    <context:property-placeholder location="classpath:jdbc.properties"/>
    <!-- 2.数据库连接池 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <!-- 3.配置连接池属性 -->
        <property name="driverClass" value="${jdbc.driver}"/>
        <property name="jdbcUrl" value="${jdbc.url}"/>
        <property name="user" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>

        <!-- c3p0 连接池的私有属性 -->
        <property name="maxPoolSize" value="30"/>
        <property name="minPoolSize" value="10"/>
        <!-- 关闭链接后不自动 commit -->
        <property name="autoCommitOnClose" value="false"/>
        <!-- 获取连接超时时间 -->
        <property name="checkoutTimeout" value="1000"/>
        <!-- 当前获取连接失败重试次数 -->
        <property name="acquireRetryAttempts" value="2"/>
    </bean>

    <!-- 3.配置 SqlSessionFactory 对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 注入数据库连接池 -->
        <property name="dataSource" ref="dataSource"/>
        <!-- 配置 MyBatis 全局配置文件：mybatis-config.xml -->
        <property name="configLocation" value="classpath:mybatis-
config.xml"/>
        <!-- 扫描 entity 包 使用别名 -->
```

```
        <property name="typeAliasesPackage" value="entity"/>
        <!-- 扫描 sql 配置文件：mapper 需要的配置 xml 文件， -->
        <property name="mapperLocations" value="classpath:mapper/*/*.xml"/>
    </bean>

    <!-- 4.配置扫描 Dao 接口包，动态实现 Dao 接口，注入到 spring 容器中 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 注入 sqlSessionFactory -->
        <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory"/>
        <!-- 给出需要扫描 Dao 接口包 -->
        <property name="basePackage" value="dao"/>
    </bean>

</beans>
```

spring-service.xml 代码如下：

```
<?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/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd"
        default-autowire="byName">

    <!-- 扫描 service 包下所有使用注解的类型 -->
    <context:component-scan base-package="service" />

    <!-- 配置事务管理器 -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 注入数据库连接池 -->
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!-- 配置基于注解的声明式事务     默认使用注解来管理事务行为 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>
```

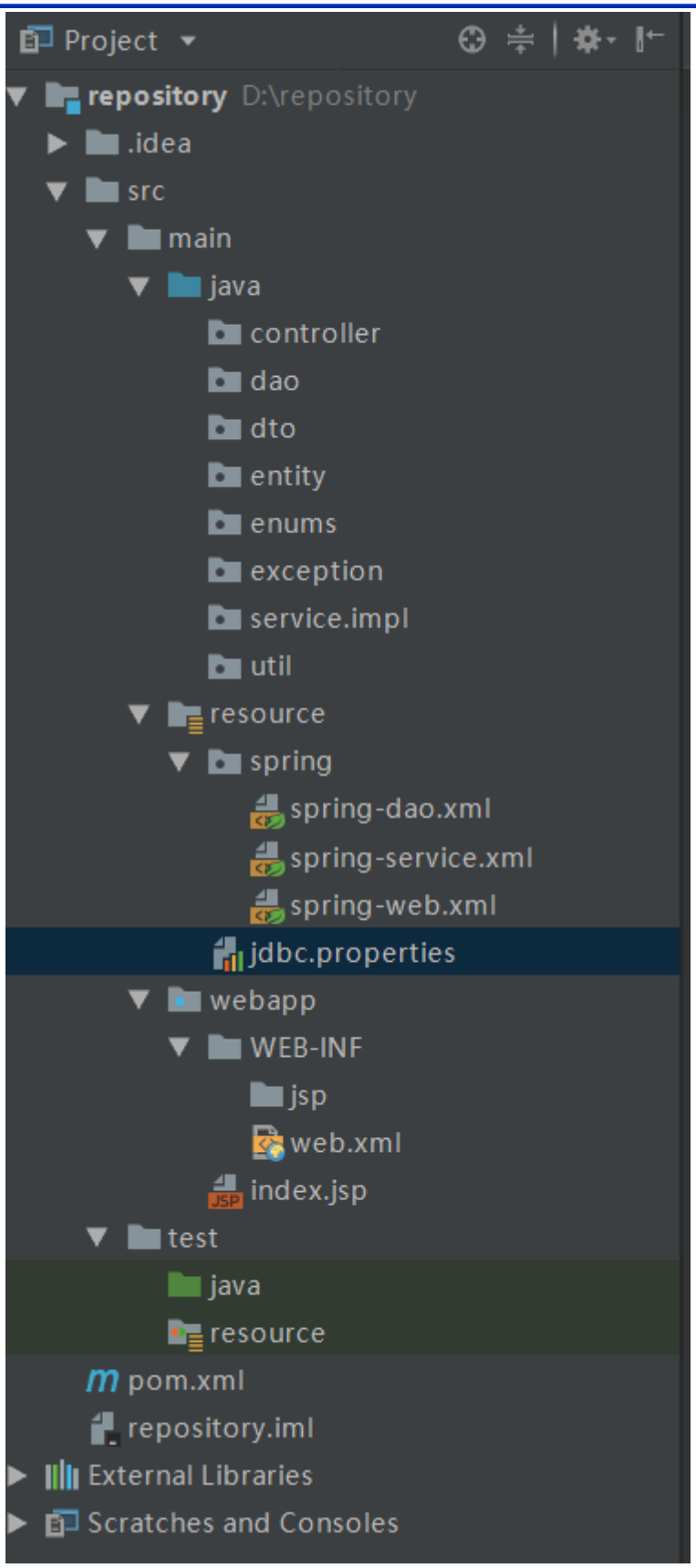
```
</beans>
```

spring-web.xml 代码如下：

```
<?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: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/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

    <!-- 配置 SpringMVC -->
    <!-- 1:开启 SpringMVC 注解模式 -->
    <!-- 简化配置:
        (1)自动注册
DefaultAnnotationHandlerMapping,AnnotationMethodHandlerAdapter
        (2)提供一系列:数据绑定,数字和日期的 format
@NumberFormat,@DataTimeFormat,
            xml,json 默认读写支持.
     -->
    <mvc:annotation-driven/>

    <!--
        2:静态资源默认 servlet 配置
        1:加入对静态资源的处理:js,gif,png
        2:允许使用"/"做整体映射
     -->
    <mvc:default-servlet-handler/>

    <!--3:配置 jsp 显示 ViewResolver -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
```

```
    <!--4:扫描 web 相关的 bean -->
    <context:component-scan base-package="controller"/>
</beans>
```

这一步结束后，目录的样子应该像下图这样：


Project
repository D:\repository
.idea
src
main
java
controller
dao
dto
entity
enums
exception
service.impl
util
resource
spring
spring-dao.xml
spring-service.xml
spring-web.xml
jdbc.properties
webapp
WEB-INF
jsp
web.xml
index.jsp
test
java
resource
pom.xml
repository.iml
External Libraries
Scratches and Consoles

9. 配置好 spring 的文件后，接下来配置 Mybatis 的配置文件。先在 **main/resource** 下新建一个 mapper 目录用于后续存放 mybatis 的映射文件。然后还是在 main/resource 下新建一个文件，叫做 **mybatis-config.xml**。代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
    <settings>
        <setting name="useGeneratedKeys" value="true"/>
        <setting name="useColumnLabel" value="true"/>
        <setting name="mapUnderscoreToCamelCase" value="true"/>
    </settings>
</configuration>
```

10. 接下来还不能运行，因为此时 spring-dao 下有红色报错代码，如果想检验框架是否搭建完成，可以将 **spring-dao.xml 的红色代码注释掉**，部署 tomcat 之后运行，就能看到默认生成的 index.jsp 里的 hello world！了。当 mybatis 的映射文件补齐后，就可以取消掉之前的注释了。希望大家注意这个文件里的问题，曾经的我因为没察觉到这个地方的错误，在没有写代码的时候就运行工程，发现怎么都报 sqlsession 的错误，现在才明白原来是缺少映射文件。希望大家不要走这段弯路。