# 新建工程，导入依赖

导入Spring的Jar包或者依赖

<properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>11</maven.compiler.source>  
 <maven.compiler.target>11</maven.compiler.target>  
 *<!-- Spring统一版本号 -->* <springVersion>5.2.6.RELEASE</springVersion>  
</properties>  
  
<dependencies>  
 *<!-- Spring表达式 -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-expression</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- Spring核心 -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- Spring实体 -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- Spring容器 -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- Spring web -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- spring-aop -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!-- spring-aspect -->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
 *<!-- SpringMVC-->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>${springVersion}</version>  
 </dependency>  
  
 *<!--与Spring整合时，需要导入mybatis和mybatis-spring两个依赖-->* <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>2.0.6</version>  
 </dependency>  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.3</version>  
 </dependency>  
  
 *<!-- mysql-connector-java -->* <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.47</version>  
 </dependency>  
  
 *<!-- Alibaba Druid Database datasource -->* <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.2.3</version>  
 </dependency>  
  
 *<!--Servlet依赖-->* <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>3.1.0</version>  
 <scope>provided</scope>  
 </dependency>  
  
 *<!--JSP依赖-->* <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>2.1</version>  
 <scope>provided</scope>  
 </dependency>  
 *<!--JSTL-->* <dependency>  
 <groupId>javax.servlet.jsp.jstl</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>taglibs</groupId>  
 <artifactId>standard</artifactId>  
 <version>1.1.2</version>  
 </dependency>  
  
 *<!--Junit4-->* <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>compile</scope>  
 </dependency>  
</dependencies>

# 整合mapper层与MyBatis

## mapper.xml

**在resources目录下编写MyBatis主配置文件：**

|  |
| --- |
| **MyBatis-Config.xml**  *<?*xml version="1.0" encoding="UTF-8" *?>* <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>* <configuration>   *<!--与Spring整合后，因为像连接池之类的类全部交给Spring管理了，所以在MyBatis的配置文件内容将会很少-->  <!--一般只会留下别名标签和设置标签-->   <!--<settings>  <setting name="" value=""/>  </settings>-->* <mappers>  <mapper resource="net/hackyle/mapper/PersonMapper.xml"/>  </mappers>  </configuration> |

**在resources目录下编写Spring与MyBatis整合的配置文件：**

|  |
| --- |
| **Spring-mapper.xml**  *<?*xml version="1.0" encoding="UTF-8"*?>* <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"   xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">   *<!--  Spring整合MyBatis专用配置文件  Spring管理mapper层的配置文件，即将mapper层的所有类自动交给Spring管理  -->   <!--  主要步骤：  1.配置连接池（关联数据库配置文件）  2.SqlSessionFactory  3.配置实体映射XML文件  -->   <!--引入外部文件：引入外部的数据库配置文件-->  <!--<context:property-placeholder location="classpath:database.properties"/>-->   <!--  配置连接池  -->  <!--这里可以使用Spring-JDBC提供的连接池实现与数据库的连接，当然也可以使用其他连接池，如C3P0，DBCP，Druid -->* <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  *<!--配置连接池属性-->* <property name="driverClassName" value="com.mysql.jdbc.Driver"/>  <property name="url" value="jdbc:mysql://localhost:3306/kdb?useSSL=true&amp;charsetEncoding=UTF-8"/>  <property name="username" value="root"/>  <property name="password" value="kyle"/>   *<!--从外部的配置文件中读取引入-->  <!--<property name="driverClassName" value="${driver}"/>  <property name="url" value="${url}"/>  <property name="username" value="${username}"/>  <property name="password" value="${password}" />-->* </bean>   *<!--  配置SqlSessionFactory  -->  <!--对于MyBatis来说，它就相当于原始JDBC中的连接池-->* <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource"/>  *<!--绑定MyBatis配置文件-->* <property name="configLocation" value="classpath:MyBatis-Config.xml"/>  *<!-- <property name="mapperLocations" value="classpath\*:net/hackyle/mapper/\*.xml"/>-->* </bean>    *<!--开启注解扫描-->* <context:component-scan base-package="net.hackyle.mapper" />   *<!--配置mapper接口也可以注入到IOC容器中-->* <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  <property name="basePackage" value="net.hackyle.mapper" />  </bean>  </beans> |

## Mapper接口和类

**在java代码包下建立pojo包及实体类和mapper包及实现包：**

|  |
| --- |
| **pojo.Person**  package net.hackyle.pojo;  public class Person {  private int id;  private String name;  private String sex;  private int age;  private String address;  private String tel;  private String email;   *//实体类必须提供无参构造，否则会有意想不到的异常抛出！* public Person() {   }   public Person(int id, String name, String sex, int age, String address, String tel, String email) {  this.id = id;  this.name = name;  this.sex = sex;  this.age = age;  this.address = address;  this.tel = tel;  this.email = email;  }  其他get、set方法  } |

|  |
| --- |
| **mapper.PersonMapper**  package net.hackyle.mapper;  import net.hackyle.pojo.Person;  import java.util.List;  public interface PersonMapper {  List<Person> readAll(); } |

|  |
| --- |
| **mapper.PersonMapperImpl**  package net.hackyle.mapper.impl;  import net.hackyle.mapper.PersonMapper; import net.hackyle.pojo.Person; import org.mybatis.spring.SqlSessionTemplate;  import java.util.List;  */\*\*  \* PersonMapper任然需要实现类，在这里实现数据库的真实操作  \*/* public class PersonMapperImpl implements PersonMapper {  private SqlSessionTemplate sqlSessionTemplate;   *//使用set方法是便于注入* public void setSqlSessionTemplate(SqlSessionTemplate sqlSessionTemplate) {  this.sqlSessionTemplate = sqlSessionTemplate;  }   @Override  public List<Person> readAll() {  PersonMapper personMapper = sqlSessionTemplate.getMapper(PersonMapper.class);   return personMapper.readAll();  } } |

注意：

* 根据官方文档：http://mybatis.org/spring/zh/sqlsession.html
* 除了把SqlSessionFactory交给Spring管理来获取SQLSession外，还可以使用SqlSessionDaoSupport
* 具体的用法是在XxxMapperImpl类上继承SqlSessionDaoSupport类，调用 getSqlSession() 方法你会得到一个 SqlSessionTemplate，之后可以用于执行 SQL 方法
* 通过这种方式就可以省略掉对XxxMapperImpl类的注入！

# 整合service层

**在resources目录下新建一个Spring-Service.xml，与service层进行整合：**

|  |
| --- |
| **Spring-Service.xml**  *<?*xml version="1.0" encoding="UTF-8"*?>* <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns: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 https://www.springframework.org/schema/context/spring-context.xsd">   *<!--  Spring管理service层的配置文件，将service层的所有类自动交给Spring管理  -->   <!--  主要步骤：  1.支持扫描注解（将Service层的类交给Spring管理）  2.事务、AOP切入  -->   <!--支持注解扫描-->* <context:component-scan base-package="net.hackyle.service" />   *<!--除了使用注解，也可以使用bean标签将类交给Spring管理，例如：-->  <!--<bean id="testService" class="net.hackyle.service.TestService">  <property name="xxxMapper" ref="xxxMapper"/>  </bean>-->    <!--申明式事务配置-->  <!--<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  &lt;!&ndash;注入数据源&ndash;&gt;  <property name="dataSource" value="dataSource"/>  </bean>-->   <!--AOP的事务支持-->* </beans> |

# 整合controller层与SpringMVC

**在resources目录下新建Spring-Controller.xml**

|  |
| --- |
| **Spring-Controller.xml**  *<?*xml version="1.0" encoding="UTF-8"*?>* <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns: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  https://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/mvc  https://www.springframework.org/schema/mvc/spring-mvc.xsd">   *<!--  Spring管理controller层的配置文件，将controller层的所有类自动交给Spring管理  -->    <!--支持注解扫描-->* <context:component-scan base-package="net.hackyle.controller" />   *<!--视图解析器-->* <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/jsp/" />  <property name="suffix" value=".jsp"/>  </bean> </beans> |

**在web.xml中添加DispatcherServlet：**

|  |
| --- |
| **web.xml**  *<?*xml version="1.0" encoding="UTF-8"*?>* <web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app\_5\_0.xsd"  version="5.0">  <welcome-file-list>  <welcome-file>/index.jsp</welcome-file>  </welcome-file-list>    *<!--全局乱码解决-->* <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  *<!--注意：这里要使用"/\*"，因为我们想要所有的文件都是以utf-8编码的-->* <url-pattern>/\*</url-pattern>  </filter-mapping>   *<!--前端控制分发器-->* <servlet>  <servlet-name>dispatcherServlet</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:ApplicationContext.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>dispatcherServlet</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

# Spring主配置文件

**编写Spring的主配置文件：**

|  |
| --- |
| **ApplicationContext.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"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">   *<!--Spring主配置文件-->   <!--整合mapper层，导入MyBatis和Spring整合的配置文件-->* <import resource="Spring-Mapper.xml"/>   *<!--整合service层-->* <import resource="Spring-Service.xml"/>   *<!--整合controller层，导入SpringMVC的配置文件-->* <import resource="Spring-Controller.xml"/>  </beans> |

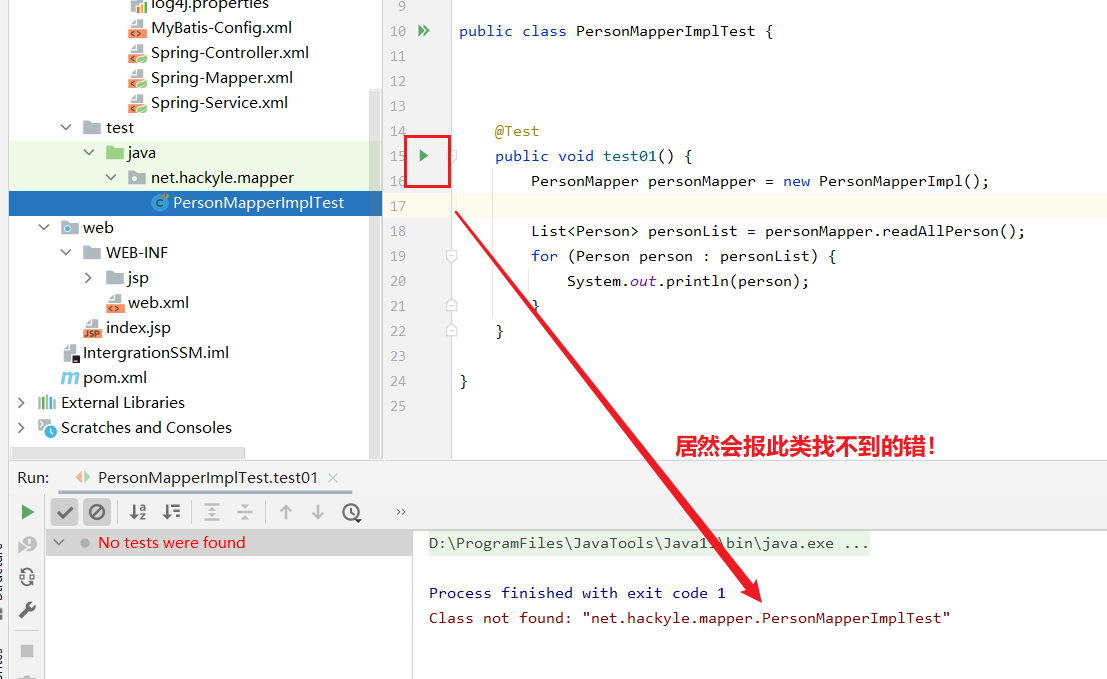
编写其他配置文件：日志等

# Q&A

在某些情况下不能使用@Test，但是依赖已经导入成功了的。这时就需要把依赖中的范围该为compile即可。

**maven中运行junit4测试类报错：class not found**

**现象：**

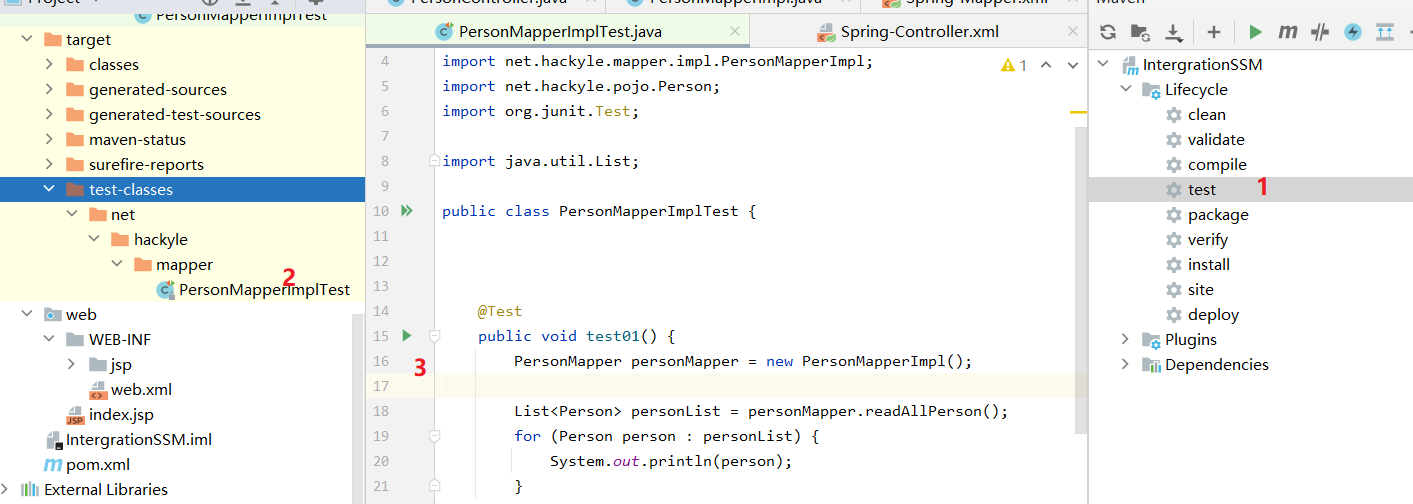


**原因：**

1. 明明编写了类，为什么会出现class not found错误呢，肯定是没有编译成功。
2. 在maven项目里要运行Junit，必须在 target/test-classes 找到对应的编译类。

**解决方案：**

1. 执行“Maven-Test”，就可以在target/test-classes能找到编译的测试类



1. 然后再运行刚刚的测试方法，就可以了

**报错现象：**

Property 'dataSource' threw exception; nested exception is java.lang.NoClassDefFoundError: org/springframework/jdbc/datasource/TransactionAwareDataSourceProxy

**问题原因：**

* 缺少Spring-jdbc依赖
* org.mybatis.spring.SqlSessionFactoryBean用到了spring-jdbc。

**解决：**导入该依赖。