# 新建工程，导入依赖

<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>

## Spring

*<!-- Spring容器 -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${springVersion}</version>  
</dependency>  
*<!-- Spring-Bean -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${springVersion}</version>  
</dependency>  
*<!-- Spring核心 -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${springVersion}</version>  
</dependency>  
*<!-- Spring表达式 -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-expression</artifactId>  
 <version>${springVersion}</version>  
</dependency>

## SpringMVC

*<!-- Spring web -->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${springVersion}</version>  
</dependency>  
  
*<!-- SpringMVC-->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${springVersion}</version>  
</dependency>

## MyBatis

*<!-- 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>  
  
*<!-- 与Spring整合时需要导入mybatis和mybatis-spring -->*<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.3</version>  
</dependency>  
<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>2.0.6</version>  
</dependency>  
*<!-- 事务-->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>${springVersion}</version>  
</dependency>  
*<!--Spring提供的JDBC-->*<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>${springVersion}</version>  
</dependency>

## 工具

*<!--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>  
  
  
*<!-- 日志：slf4j + Log4j -->*<dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.12</version>  
</dependency>  
<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.6.6</version>  
</dependency>  
<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.6.6</version>  
</dependency>

*<!-- 单元测试 -->*

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
</dependency>

## 静态资源过滤问题

**放在build目录：**

<resources>  
 <resource>  
 <directory>src/main/java</directory>  
 <includes>  
 <include>\*\*/\*.properties</include>  
 <include>\*\*/\*.xml</include>  
 </includes>  
 <filtering>false</filtering>  
 </resource>  
  
 <resource>  
 <directory>src/main/resources</directory>  
 <includes>  
 <include>\*\*/\*.properties</include>  
 <include>\*\*/\*.xml</include>  
 </includes>  
 <filtering>false</filtering>  
 </resource>  
</resources>

# Spring整合MyBatis

## 主配置文件

MyBatis主配置文件名：MyBatis-Config.xml

**特性：**

* 由于我们使用的是代理 Dao接口 的模式，Dao接口的 具体实现类由 MyBatis 使用代理方式创建，所以此时 mybatis 配置文件不能删。
* 当我们整合 spring 和 mybatis 时，mybatis 创建的 Mapper.xml 文件名必须和 Dao接口 文件 名一致
* 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

|  |
| --- |
| **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>-->* </configuration> |

## Mapper接口和Mapper.xml

**在mapper包中的接口方法与XML文件中SQL标签的关联：**

1. 接口的**方法名**必须等于相应XML配置文件中的**id名**
2. 接口的方法**返回值**必须与XML文件中的返回**类型一致**
3. 接口方法的**入参类型**与XML中的入参类型一致
4. XML中的**命名空间**必须绑定此接口

|  |
| --- |
| **PersonMapper.java接口**  package net.hackyle.mapper;  import net.hackyle.pojo.Person; import org.springframework.stereotype.Repository;  import java.util.List;  @Repository("personMapper") public interface PersonMapper {  *//查* List<Person> readAllPerson(); } |

|  |
| --- |
| **PersonMapper.xml**  *<?*xml version="1.0" encoding="UTF-8" *?>* <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>* <mapper namespace="net.hackyle.mapper.PersonMapper">  *<!--List<Person> readAllPerson();-->* <select id="readAllPerson" resultType="net.hackyle.pojo.Person">  select *\** from person  </select>  </mapper> |

## 与Spring整合的XML

### 在MyBatis主配置文件中导入MapperSQL

**在MyBatis-Config.xml中导入MapperSQL共有三种方式：**

* package-name
* mapper-class
* mapper-resource

|  |
| --- |
| **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，也可以将XML映射文件导入 -->   <!--<settings>  <setting name="" value=""/>  </settings>-->*  ***<!-- 在MyBatis主配置文件中引入Mapper-XML -->* <mappers>  *<!--方式一：强烈推荐使用这种方式, 因为：-->  <!--1.不管注解还是配置XML映射文件，都是通用的-->  <!--2.但是要把XML映射文件放在和接口在一起。如果放在resources同包的目录下，则不行-->* <package name="net.hackyle.mapper"/>   *<!--方式二：需要一个个地指定具体的XxxDao。映射文件和接口必须放在一起。-->* <mapper class="net.hackyle.mapper.PersonMapper"/>   *<!--方式三：指定为XML方式，不可使用注解-->  <!--既可以XML映射文件放在和接口在一起，也可以放在resources同包的目录下-->* <mapper resource="net.hackyle.mapper/PersonMapper.xml"/>  </mappers>   *<!--总结：  1.XML映射文件与接口放在一起的，则在导入时使用"."分割  2.XML映射文件放在resources目录的同包文件夹内的，则在导入时使用"/"分割  -->*** </configuration> |

|  |
| --- |
| **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" xmlns:tx="http://www.springframework.org/schema/tx"  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/tx http://www.springframework.org/schema/tx/spring-tx.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  -->   <!--  把连接池交给Spring管理  -->  <!--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"/>  </bean>   *<!--  把SqlSessionFactory交给Spring管理  -->* <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource"/>  ***<!--绑定MyBatis配置文件，不可缺少-->* <property name="configLocation" value="classpath:MyBatis-Config.xml"/>**</bean>   *<!--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.dao" />  </bean>  </beans> |

|  |
| --- |
| **测试：**  在test包下对应建立测试类  package net.hackyle.mapper;  import net.hackyle.pojo.Person; import org.junit.Test; import org.springframework.context.ApplicationContext; import org.springframework.context.support.ClassPathXmlApplicationContext;  import java.util.List;  public class PersonMapperTest {  @Test  public void test01() {  ApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext.xml");   PersonMapper personMapper = ac.getBean(PersonMapper.class);  List<Person> personList = personMapper.readAllPerson();   for (Person person : personList) {  System.*out*.println(person);  }  } }  输出： |

### 在Spring-Mapper.xml整合

在Spring-Mapper.xml中完成映射XML文件的导入：

* MapperScannerConfigurer-basePackage

|  |
| --- |
| **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-->* </configuration> |

|  |
| --- |
| **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"/>   *<!--只能将映射XML文件导入resources目录下的同包内-->  <!--注意分割方式：斜杠、点-->  <!--<property name="mapperLocations" value="classpath\*:net/hackyle/mapper/\*.xml"/>-->*  *<!--<property name="mapperLocations" value="classpath\*:net.hackyle.mapper/\*.xml"/>-->* </bean>   ***<!--自动扫描所有Mapper接口和文件，必须要有-->* <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  <property name="basePackage" value="net.hackyle.mapper" />  </bean>**  </beans> |

|  |
| --- |
| **测试**  package net.hackyle.mapper;  import net.hackyle.pojo.Person; import org.junit.Test; import org.springframework.context.ApplicationContext; import org.springframework.context.support.ClassPathXmlApplicationContext;  import java.util.List;  public class PersonMapperTest {  @Test  public void test01() {  ApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext.xml");   PersonMapper personMapper = ac.getBean(PersonMapper.class);  List<Person> personList = personMapper.readAllPerson();   for (Person person : personList) {  System.*out*.println(person);  }  } } |

## Mapper接口的实现

|  |
| --- |
| **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！

# Spring整合SpringMVC

## 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>

## 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>

## 测试

写一个Controller类：

@Controller

public class PersonController {

@ResponseBody

@GetMapping(“/hello”)

public String hello() {

return “hello, ssm!”;

}

}

# 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

# 整合测试

准备JSP页面

完成对一个实体的CRUD