第 5 章
SSM三大框架整合

5.1 Spring 整合 MyBatis 开发

Spring 整合 MyBatis 可以实现让数据库连接、事务管理、实例化对象的创建与依赖关系等都统一由 Spring 负责，以及数据库的增、删、改、查操作由 spring-mybatis 整合包提供的 SQLSessionTemplate 对象来操作，或者利用整合包扫描接口，依据 Mapper 映射文件直接创建代理实现类，无须程序员手工实现接口，大大简化了开发流程。Spring MVC 本来就是 Spring 框架的一部分，这两者无须再做整合，因此，SSM 整合的关键是 Spring 对 MyBatis 的整合，三大框架整合完成后，将以 Spring 为核心，调用有关资源，高效运作。

5.1.1 Spring 整合 MyBatis 开发环境

除了需要 Spring 的 jar 包和 MyBatis 的 jar 包，还需要 Spring 与 MyBatis 整合的中间件 mybatis-spring-1.3.1.jar，此外还需要数据库驱动 jar 包 mySQL-connector-Java-5.1.37.jar。

5.1.2 DAO 接口实现类开发整合

项目案例：Spring＋MyBatis 实现 Prom 数据库的增、删、改、查。

实现步骤如下。

(1) 新建项目，导入 Spring 整合 MyBatis 开发所需的 jar 包，如图 5-1 所示。

(2) 创建实体类 Prom。

(3) 创建 DAO 接口 IPromDao。

(4) 创建 DAO 接口的实现类 PromDaoImpl。

(5) 创建业务层接口 IPromService。

(6) 创建业务层接口的实现类 PromServiceImpl。

(7) 创建 SQL 映射文件 PromMapper.xml。

(8) 创建 MyBatis 配置文件 mybatis-config.xml。

(9) 创建 Spring 配置文件 applicationContext.xml。

(10) 测试类。

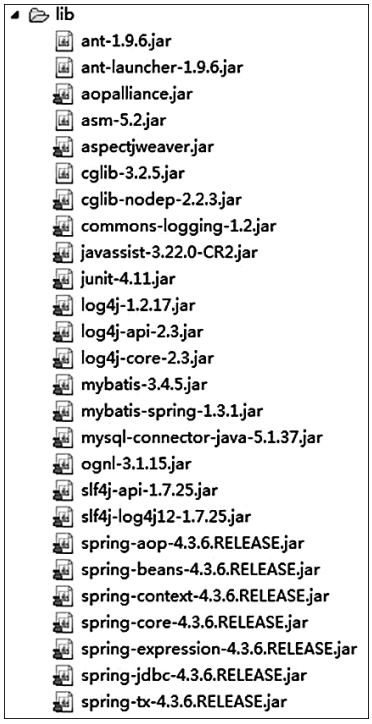

图 5-1　Spring 整合 MyBatis 开发所需的 jar 包

5.1.3　DAO 接口无实现类开发整合

由于 MyBatis 可以通过代理实现接口，无需 DAO 的实现类，但如果没有 DAO 的实现类，业务层 Service 类中的 DAO 属性该如何注入？通过下面这个案例可以得到答案。

项目案例：改造上述项目，去掉 DAO 的实现类，但能实现同样的功能。

复制项目 springmybatis1 为 springmybatis2，删除 PromDaoImpl 类。

修改 Spring 配置文件，删除 SQLSessionTemplate 的配置，修改 DAO 层的配置如下。

```
<bean id="PromDao" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.fqy.dao.IPromDao"/>
        <property name="SQLSessionFactory" ref="SQLSessionFactory"/>
</bean>
```

这里其实使用了 MapperFactoryBean 类来代理实现接口，假定项目还有其他 DAO 层，则依照这个模板再添加一条 bean。但如果一个项目有太多的 DAO 层，则需要添加太多的 bean，可以进行简化，配置批量扫描包，将包下的所有接口自动创建代理实现类，只需下面一条配置就可以，无须再一一创建 DAO 层的 bean。

```
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="SQLSessionFactory" ref="SQLSessionFactory"/>
```

```
        <property name="basePackage" value="com.fqy.dao"/>
    </bean>
```

MapperScannerConfigurer 将扫描 basePackage 属性指定的包下的所有接口类，如果这些接口都有对应的映射文件，则会将它们动态地定义一个 bean，这样就无须一个个定义 bean 了。

但问题在于 Service 层，之前有手工创建 DAO 层的 bean，Service 层再一个个进行手工注入这些 DAO 层的 bean，类似下面的配置。

```
<!-- 配置 DAO 层，注入 SQLSessionTemplate 属性值 -->
    <bean id="PromDao" class="com.fqy.dao.PromDaoImpl">
        <property name="SQLSessionTemplate" ref="SQLSessionTemplate"/>
    </bean>
    <!-- 配置 Service 层，注入 PromDao 属性值 -->
    <bean id="PromService" class="com.fqy.service.PromServiceImpl">
        <property name="PromDao" ref="PromDao"/>
    </bean>
```

之前这种配置，Service 层注入 PromDao 属性值，关系很清晰，每一个业务 bean 都能配置具体名称的 DAO 层的 bean。但现在 PromDao 这个 bean 配置已经没有了（bean 由代理动态的自动生成），这样运行的话，程序会报错：

```
Caused by: org.springframework.beans.factory.NoSuchBeanDefinitionException:
No bean named 'PromDao' available…
```

提示业务层找不到 PromDao 这个 bean。解决方案是让业务层按类型自动配置就行了，业务层的配置代码修改如下：

```
<!-- 配置 Service 层，注入 PromDao 属性值 -->
<bean id="PromService" class="com.fqy.service.PromServiceImpl" autowire=
"byType"/>
```

当然，在 Service 层用注解@Autowired 也可以，但要在 Spring 配置文件中开启扫描，具体如下：

```
<context:component-scan base-package="com.fqy.service"/>
```

其他不变，测试结果同前。本案例的最大特点就是没有 DAO 的实现类，但一样可以实现程序的所有功能。

5.2　SSM 整合案例

项目案例：SSM 整合 Web 项目，实现学生信息的基本管理。实现步骤如下。

新建项目 ssmfqy，在 5.1 节整合所需 jar 包的基础上，再把 Spring MVC 有关 jar 包导入项目即可，总的 jar 包如图 5-2 所示。

(1) 配置 web.xml 文件,指定 Spring 配置文件的位置。

(2) 注册 ServletContext 监听器。

(3) 注册字符集过滤器,解决请求参数的中文乱码问题。

(4) 配置 spring-mvc 核心控制器。

(5) 在 src 下创建 Spring 配置文件 applicationContext.xml。注册数据源 DataSource,注册事务,注册 MyBatis 配置文件,这样配置无须创建 DAO 层接口的实现类,项目会自动生成 DAO 层接口的代理实现类。

(6) 配置注解扫描 Service,以注解的形式创建 bean。

(7) 在 src 下创建 MyBatis 配置文件 mybatis-config.xml。

(8) 在 src 下创建 Spring 配置文件 Spring MVC.xml。

(9) 创建数据库表。

(10) 新建包 com.fqy.entity,创建实体类 Prom。

(11) 创建包 com.fqy.dao,创建接口 IPromDao。

(12) 在包 com.fqy.dao 下创建接口 IPromDao 对应的映射文件 PromMapper.xml。

(13) 创建包 com.fqy.service,创建接口 IPromService。

(14) 创建接口 IPromService 的实现类 PromServiceImpl,添加注解。

(15) 创建包 com.fqy.controller,创建控制器类。

在目录 WebContent/WEB-INF 下新建文件夹 jsp,创建以下 jsp 页面。

运行测试效果如图 5-3～图 5-5 所示。

图 5-2 Spring MVC 有关 jar 包

图 5-3 运行测试效果图-学生列表

图 5-4　运行测试效果图-添加学生

图 5-5　运行测试效果图-修改学生