第 5 章

当当书城 Spring 整合 JDBC

5.1 当当书城基本功能

本节为当当书城项目介绍。它分为前台和后台两部分，前台是普通用户和游客访问，后台为系统管理员访问。

前台基本功能有主页图书推荐、图书详情显示、购物车管理、图书付款、我的订单等。

后台基本功能有用户订单查询、图书上架、图书下架、图书促销管理等。

用户的角色共分为四种：游客、注册用户、会员（会员为预留用户）、管理员。游客为非注册用户，可以浏览主页和图书详情；注册用户登录后可以添加商品到购物车，并付款结算；会员也可称为 VIP 用户，可以进入会员专区；管理员进入系统后台，可以浏览用户订单、上架图书、下架图书、修改图书价格等。

本章项目所有代码，参见附件中的 DangDang（基础版）和 DangSpring 源码包。

5.1.1 项目开发环境

项目环境是以 Jakarta EE 9+为基础，具体为 JDK17、Tomcat 10.1、Sevlet 5.0、Spring 6.0。数据库采用 MySQL 8 或 Oracle 11。IDE 为 Eclipse，选择 2022-12-eclipse-inst-jre-win64 或更高版本均可。数据库的设计工具使用 Power Designer 15。逻辑设计工具使用 Rational Rose。

5.1.2 表结构设计

当当书城项目同时支持两套数据库，即 MySQL 和 Oracle，还可以扩展支持其他数据库。

书城项目的主要目的是在实际项目中强化前面的知识点，应该尽量覆盖更多知识点，而不是使功能更完善，因此设计上不能过于复杂，重复功能都尽量省略了，这与大型的实际企业级项目是有很大区别的。

当当书城的 MySQL 表结构见图 5-1。

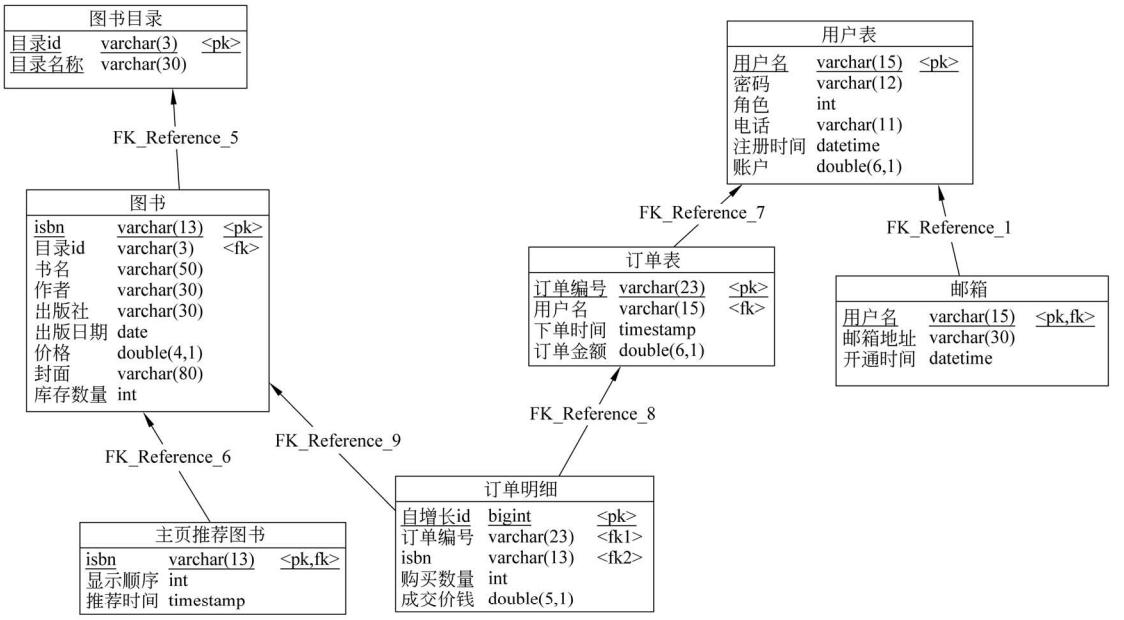

图 5-1　当当书城表结构

表设计需注意以下几点。

（1）Oracle 中使用 varchar2 类型，MySQL 中使用 varchar 类型。

（2）Oracle 中的整型和浮点型推荐使用 number，MySQL 中使用 int 和 double。

（3）Oracle 与 MySQL 库的所有字段名称、字段数量相同。

（4）“订单”表的主键为字符型，需要保证订单编号在高并发环境的唯一性。

（5）“订单明细”表的主键为长整型，Oracle 采用 sequence，MySQL 使用 auto_increment。一个用户可以有多个订单，每个订单有多个订单明细。

（6）“图书”表用 isbn 作为主键，即每个 isbn 只存储一条记录，不是每本书一条记录。这与大型设备的管理模式不同，大型设备如汽车，必须是一辆车存一条记录。

（7）“主页推荐图书”与“图书”表为一对一关系，即从所有图书中挑选部分图书，作为主页推荐，而且可以设置推荐图书的显示顺序。

（8）在 MySQL 中创建数据库 bk，然后创建表。

```
create database bk;        //创建数据库，名字为 bk
use bk;                    //打开库
                           //其他创建表的 SQL 语句，参见本书配套资源
```

5.1.3　当当书城原型

当当书城部分功能的原型样式见图 5-2~图 5-9。

图 5-2　书城主页

图 5-3　图书详情页

图 5-4　用户登录页

图 5-5　购物车页

图 5-6　商品结算页

图 5-7　付款页

图 5-8　我的订单页

图 5-9　图书上架页

5.2　Spring 整合 JDBC 实战

本节基于当当书城基础版（使用 Servlet+JSP+JDBC+Spring 实现，参见本书配套资源），使用 Spring 框架的 IoC、AOP 和持久层整合技术，实现当当书城逻辑层和持久层的代码管理。

5.2.1　导包

配置 pom.xml 文件，导入相应的包。主要依赖包的配置如下：

```
<dependency>
    <groupId>jakarta.servlet</groupId>
    <artifactId>jakarta.servlet-api</artifactId>
    <version>5.0.0</version>
</dependency>
<dependency>
```

```
        <groupId>org.glassfish.web</groupId>
        <artifactId>jakarta.servlet.jsp</artifactId>
        <version>3.0.0</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.web</groupId>
        <artifactId>jakarta.servlet.jsp.jstl</artifactId>
        <version>3.0.1</version>
    </dependency>
    <dependency>
        <groupId>jakarta.servlet.jsp.jstl</groupId>
        <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
        <version>3.0.0</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>6.0.3</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>6.0.3</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.27</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-slf4j-impl</artifactId>
        <version>2.13.3</version>
    </dependency>
```

5.2.2 Spring 配置文件

在当当书城项目的 src 下，新建 beans.xml 文件，此为 Spring 的核心配置文件。配置信息操作如下。

（1）配置 schema。

```
<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
       https://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/tx
       https://www.springframework.org/schema/tx/spring-tx.xsd
       http://www.springframework.org/schema/context
       https://www.springframework.org/schema/context/spring-context.xsd">
</beans>
```

（2）配置数据源。

```
<bean id="dataSource"
       class="org.springframework.jdbc.datasource.DriverManagerDataSource">
       <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
```

```
    <property name="url"
        value="jdbc:mysql://localhost:3306/bk?useSSL=false&amp;
        serverTimezone=Asia/Shanghai&amp;allowPublicKeyRetrieval=true" />
    <property name="username" value="root" />
    <property name="password" value="123456" />
</bean>
```

（3）配置事务管理器。

```
<bean id="txManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
 </bean>
<tx:annotation-driven transaction-manager="txManager" />
```

（4）配置 Spring Bean 的扫描位置。

```
<context:component-scan base-package="com.icss.dao"/>
<context:component-scan base-package="com.icss.biz"/>
```

5.2.3　封装 BaseDao

BaseDao 是所有持久层类的抽象父类，在此类中注入数据源，同时封装获取数据库连接和关闭连接的操作。下面讲述操作步骤。

（1）BaseDao 继承 JdbcDaoSupport 工具类。在 JdbcDaoSupport 中封装 JdbcTemplate 和 DataSourceUtils（参见 4.5.3 节）。

```
public abstract class BaseDao extends JdbcDaoSupport{    }
```

（2）注入数据源。

```
public abstract class BaseDao extends JdbcDaoSupport{
    @Autowired
    private void setDataSource(DriverManagerDataSource ds) {
        super.setDataSource(ds);
    }
}
```

（3）从 Spring 事务环境中获取数据库连接。

```
public Connection openConnection() throws Exception{
    return this.getConnection();
}
```

（4）关闭数据库连接，如果当前为 Spring 事务环境则不关闭。

```
public void closeConnection(Connection con) {
    boolean isTransaction = DataSourceUtils.isConnectionTransactional
                        (con, this.getDataSource());
    if(isTransaction) {
        Log.logger.info("事务环境，重用数据库连接");
    }else {
        Log.logger.info("非事务环境，关闭数据库连接");
        try {
           con.close();
        } catch (SQLException e) {
           Log.logger.error(e.getMessage(),e);
        }
    }
}
```

5.2.4 封装 SpringFactory

在 com.icss.util 包下，自定义工厂类，封装 Spring 的 IoC 容器，确保 IoC 容器为单例模式，同时封装 getBean()方法。

```
public class SpringFactory {
     private static ApplicationContext ctx;
     static {
          ctx = new ClassPathXmlApplicationContext("beans.xml");
     }
     public static Object getBean(String name) throws BeansException{
          return ctx.getBean(name);
     }
     public static <T> T getBean(Class<T> requiredType) throws BeansException{
           return ctx.getBean(requiredType);
     }
}
```

5.2.5 定义 Spring Bean 和依赖关系

使用@Service 定义逻辑层的 Bean，使用@Repository 定义持久层的 Bean，同时使用 Autowire 注入模式。

（1）定义逻辑层的 Bean。

```
@Service
public class BookBiz implements IBook{}
@Service
public class UserBiz implements IUser{}
```

（2）定义持久层的 Bean。

```
@Repository
public class BookDao extends BaseDao implements IBookDao{}
@Repository
public class UserDao extends BaseDao implements IUserDao{}
```

（3）Autowire 注入 Bean 的依赖。

```
@Service
public class BookBiz implements IBook{
       @Autowired
       private IBookDao bookDao;
}
@Service
public class UserBiz implements IUser{
       @Autowired
       private IUserDao userDao;
}
```

5.2.6 配置声明性事务

在当当书城项目中，用户在商品结算页进行付款时，需要创建订单、创建订单明细、账户扣款、减少图书库存等很多步操作，这些操作具有 ACID 特性，因此应该使用事务管理。在当当书城基础版中使用本地事务进行付款管理，本节使用 Spring 的声明性事务管理。

下面使用@Transactional 注解，对用户付款行为使用声明性事务进行管理。

```
@Transactional(rollbackFor=Throwable.class)
public void buyBooks(String uname, double allMoney,
       Map<String, Integer>shopCar) throws Exception {
   if(uname== null || uname.equals("")) {
      throw new Exception("用户名不能为空");
   }
   if(allMoney<=0) {
      throw new Exception("金额错误");
   }
   if(shopCar==null || shopCar.size()==0) {
      throw new Exception("购物车为空");
   }
   userDao.updateUserAccount(uname, -allMoney);
   userDao.addBuyRecord(uname, allMoney, shopCar);
}
```

5.2.7　控制器调用 Bean

在当当书城的 Servlet 中，调用业务逻辑对象时，不能再使用传统的 new 方式创建对象，而是应该从 IoC 容器中获取 Bean 对象。下面以当当书城主页的 MainAction 为例，其他项目代码参见本书配套资源。

```
protected void service(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
   IBook ibook = SpringFactory.getBean(IBook.class);
      try {
         List<MainBook> bks = ibook.getMainBook();
         request.setAttribute("bks", bks);
         request.getRequestDispatcher("/jsp/main.jsp").forward(req, res);
      } catch (Exception e) {
         Log.logger.error(e.getMessage(),e);
         request.getRequestDispatcher("/error/err.jsp").forward(req, res);
      } }
```

5.2.8　项目部署

参见本书配套资源中的 DangSpring 项目，这是一个 Maven Web 项目，项目部署到 Tomcat 10.1 中。注意：项目部署后，由于 Tomcat 10.1 中已存在 Servlet 和 JSP 环境，这会导致环境冲突报错。因此项目部署后，在启动 Tomcat 前，需要进入 Tomcat 10.1 的 webapps 目录，找到 DangSpring 的 WEB-INF\lib 目录，删除该目录下 jakarta.servlet.jsp 和 jakarta.servlet-api 等相关包。