（可选）实验六补充实验——轻型框架整合（以 SSH 为例）

一、基础实验——Spring 与 Hibernate 的整合

（一）实验目的

1、进一步熟悉 Spring 基础环境搭建的方法，以及在 Eclipse 中开发 Spring 应用的主要步骤；
2、掌握 Spring 框架与 Hibernate 框架整合的基本步骤，理解 Spring 容器对 DataSource 实例和 SessionFactory 实例的管理；
3、进一步理解 Spring 中控制反转 IoC 的核心机制；
4、进一步熟悉 Spring 配置文件，掌握 Spring 配置文件中对 DataSource 和 SessionFactory 的配置方法和属性注入的方式。

（二）基本知识与原理

1、Spring 框架与 Hibernate 框架整合，将生成 DataSource 对象和 SessionFactory 对象的过程交给 Spring 容器实现，而不在代码中实现，即由 IoC 容器控制对象的生成和属性的注入；
2、使用 IoC 装配对象（如配置 SessionFactory 对象），就必须在 Spring 配置文件（默认为 applicationContext.xml）中进行配置；
3、Spring 框架中的 IoC 容器管理的对象都被称为 bean，bean 都需要再配置文件的<beans>元素下使用<bean>元素配置；
4、Spring IoC 容器的代表者是 API 中的 BeanFactory 接口，IoC 容器装配成功的对象，都将通过 BeanFactory 获得，进而在应用中使用。

（三）实验内容及步骤

1、在 MySQL 中创建一个名称为 hibernatedb 的数据库，并在该数据库中创建一个名称为 customer 的数据表，表结构如表 6-1 所示：

表 6-1 customer 数据表

字段名称	类型	中文含义
customerID	INTEGER(11), Primary key, Not Null	用户编号

account	VARCHAR(20)	登录用户名
password	VARCHAR(20)	登录密码
name	VARCHAR(20)	真实姓名
sex	BOOLEAN(1)	性别
birthday	DATE	出生日期
phone	VARCHAR(20)	联系电话
email	VARCHAR(100)	电子邮箱
address	VARCHAR(200)	联系地址
zipcode	VARCHAR(10)	邮政编码
fax	VARCHAR(20)	传真号码

2、在 Eclipse 中新建 Java 工程 spring-prj1，并添加 common-logging-1.2.jar、MySQL 驱动程序库文件和 Hibernate 核心包、Spring 核心包，以及 Spring 框架中与数据库操作相关的三个 JAR 包到该工程中：spring-jdbc-*.RELEASE.jar、spring-orm-*.RELEASE.jar、spring-tx-*.RELEASE.jar；

3、在 spring-prj1 中新建 cn.edu.zjut.po 包，并在其中创建持久化类 Customer.java 以及 Hibernate 映射文件 Customer.hbm.xml；

4、在 spring-prj1 中新建 cn.edu.zjut.dao 包，并在其中创建 ICustomerDAO 接口定义数据持久层的操作，以及实现类 CustomerDAO 实现数据持久层的操作，具体代码如下：

```
package cn.edu.zjut.dao;
import cn.edu.zjut.po.Customer;

public interface ICustomerDAO {
    void save(Customer transientInstance);
}
```

```
package cn.edu.zjut.dao;
import org.hibernate.Session;
import cn.edu.zjut.po.Customer;

public class CustomerDAO  implements ICustomerDAO{
    public void setSession(Session session) {
        this.session=session;   }

    public void save(Customer transientInstance) {
        Session session = getSession();
```

```
        session.save(transientInstance);
    }
}
```

5、在 spring-prj1 中创建 Spring 配置文件 applicationContext.xml，并在其中配置 CustomerDAO 实例，具体代码如下：

```
<?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:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

    <bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" />

</beans>
```

6、在工程 spring-prj1 中新建 cn.edu.zjut.service 包，在其中创建数据库操作基础类 BaseHibernate.java，具体代码如下：

```
package cn.edu.zjut.dao;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

public class BaseHibernateDAO{
    private SessionFactory sessionFactory;

    public Session getSession(){
        return sessionFactory.openSession();
    }

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
}
```

7、在 cn.edu.zjut.dao 包中创建类 UserService 实现注册逻辑，具体代码如下：

```
package cn.edu.zjut.service;
import cn.edu.zjut.dao.ICustomerDAO;
import org.hibernate.Session;
import org.hibernate.Transaction;
import cn.edu.zjut.po.Customer;

public class UserService  extends BaseHibernate {
     private ICustomerDAO customerDAO = null;

     public void setCustomerDAO(ICustomerDAO customerDAO) {
```

```
       this.customerDAO = customerDAO;
    }

    public void saveUser(Customer c) {
       Transaction tran = null;
       Session session = null;
       try {
            session = getSession();
            tran = session.beginTransaction();
            customerDAO.setSession(session);
            customerDAO.insert(c);
            tran.commit();
       } catch (RuntimeException re) {
           if(tran != null) tran.rollback();
           throw re;
       } finally {
           session.close();
       }
    }
}
```

8、修改 Spring 配置文件 applicationContext.xml，修改 beans 元素的属性，并在其中增加数据源的配置，代码片段如下：

```
<?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:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-5.0.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-5.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-5.0.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-5.0.xs
d">

    <bean id="dataSource" class="org.springframework.jdbc.
                              datasource.DriverManagerDataSource">
        <property name="driverClassName"
                  value="com.mysql.jdbc.Driver"/>
        <property name="url"
                  value="jdbc:mysql://localhost:3306/hibernatedb"/>
```

```
        <property name="username" value="root"/>
        <property name="password" value=""/>
    </bean>
……
</beans>
```

9、修改 Spring 配置文件 applicationContext.xml，在其中增加 SessionFactory 实例的配置，代码片段如下：

```
<bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.MySQLDialect
            </prop>
        </props>
    </property>
    <property name="mappingResources">
        <list>
            <value>cn/edu/zjut/po/Customer.hbm.xml</value>
        </list>
    </property>
</bean>
```

10、修改 Spring 配置文件 applicationContext.xml，在其中增加 BaseHibernate 实例的配置并在其中注入 sessionFactory，代码片段如下：

```
<bean id="base" class="cn.edu.zjut.dao.BaseHibernate">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
```

11、修改 Spring 配置文件 applicationContext.xml，在其中增加 UserService 实例的配置并在其中注入 userDAO，代码片段如下：

```
<bean id="userService" class="cn.edu.zjut.service.UserService"
          parent="base">
    <property name="customerDAO" ref="userDAO" />
</bean>
```

12、在 spring-prj1 中新建 cn.edu.zjut.app 包，并在其中创建测试类 SpringEnvTest，注册一个新用户，具体代码如下：

```
package cn.edu.zjut.app;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
import cn.edu.zjut.po.Customer;
import cn.edu.zjut.service.UserService;
```

```
public class SpringEnvTest {
    public static void main(String[] args) {
        ApplicationContext ctx = new ClassPathXmlApplicationContext(
                "applicationContext.xml");
        UserService userService =
                (UserService) ctx.getBean("userService");
        Customer cust = new Customer();
        cust.setAccount("SPRING");
        cust.setPassword("SPRING");
        userService.register(cust);
    }
}
```

13、运行测试类 SpringEnvTest，观察控制台的输出，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，总结 DataSource、SessionFactory、CustomerDAO、UserService 对象之间的依赖关系，并记录下来；

（3）根据实验步骤，查找相关资料，总结 Spring 配置文件中对 DataSource、SessionFactory、CustomerDAO、UserService 的配置方法，以及属性注入的方式；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

二、提高实验——Spring、Struts 与 Hibernate 的整合

（一）实验目的

1、掌握 Spring 框架与 Hibernate 框架、Struts 框架整合的基本步骤，理解 Spring 容器对 Bean 实例的管理；

2、理解 Spring 容器对 Struts2 核心控制器 Action 的管理，并理解 Struts 配置文件和 web.xml 中相应产生的变化；

3、进一步理解 Spring 中控制反转 IoC 的核心机制。

（二）基本知识与原理

1、使用 Spring 整合 Struts2 框架，其核心思想是将 Struts2 的 Action 实例交给 Spring 框架的 IoC 容器装配管理，因此在 Action 类中应提供必要的 setters 方法以注入所需的属性，同时 struts.xml 文件中的<action>元素的 class 属性将不再是该 Action 对应的实际类型，而是与 applicationContext.xml 中 Action 的 bean 的 id 对应；
2、使用 Spring 整合 Struts2 框架，还需要在 web.xml 文件中配置一个 listener 来完成加载 Spring 配置文件的功能。

（三）实验内容及步骤

1、在 Eclipse 中新建 **Web 工程** spring-prj1，并将 common-logging-1.2.jar、MySQL 驱动程序库文、Hibernate 核心包和 Spring 的 7 个 JAR 包（其中包括 Spring 的 4 个基础 JAR 包以及与数据库操作相关的三个 JAR 包）添加到工程中；
2、将 Struts2 中的 8 个核心包（参考实验二 Struts2 基础应用中的基础实验步骤），以及 Struts2 对 Spring 进行支持的 JAR 包 struts2-spring-plugin-2.3.15.1.jar 添加到工程中；
3、将 Spring 支持 web 开发的 JAR 包 spring-web-*.RELEASE.jar 添加到工程中；
4、在 spring-prj1 中新建 cn.edu.zjut.po 包，并在其中创建持久化类 Customer.java 以及 Hibernate 映射文件 Customer.hbm.xml，代码略；
5、在 spring-prj1 中新建 cn.edu.zjut.dao 包，并在其中创建 ICustomerDAO 接口和 CustomerDAO 实现类，代码略；
6、在 spring-prj1 中新建 cn.edu.zjut.service 包，并在其中创建数据库操作基础类 BaseHibernate.java 和 UserService 实现类，并提取 IUserService 接口，代码略；
7、在 spring-prj1 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，用于调用用户注册逻辑，代码片段如下：

```
package cn.edu.zjut.action;
import cn.edu.zjut.po.Customer;
import cn.edu.zjut.service.IUserService;

public class UserAction {
    private Customer loginUser;
    private IUserService userService = null;

    //省略 loginUser 的 getters/setters 方法
```

```
    public void setUserService(IUserService userService) {
        this.userService = userService;
    }

    public String execute() {
        userService.register(loginUser);
        return "success";
    }
}
```

8、在项目的 WebRoot/WEB-INF/路径下创建 Spring 配置文件 applicationContext.xml，并参考基础实验中的内容进行配置；

9、修改 Spring 配置文件 applicationContext.xml，增加对 UserAction 实例的配置，代码片段如下：

```
<bean id="userAction" class="cn.edu.zjut.action.UserAction"
       scope="prototype">
    <property name="userService" ref="userService" />
</bean>
```

10、在项目的 src/路径下创建 Struts2 配置文件 struts.xml，用于配置 Action 并设置页面导航，具体代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 2.1//EN"
"http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
    <package name="strutsBean" extends="struts-default"
             namespace="/">
        <action name="register" class="userAction">
            <result name="success">/regSuccess.jsp</result>
            <result name="fail">/regFail.jsp</result>
        </action>
    </package>
</struts>
```

11、编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置，并添加对 Spring 监听器的配置，代码片段如下：

```
<web-app>
  <listener>
     <listener-class>
           org.springframework.web.context.ContextLoaderListener
     </listener-class>
  </listener>
……
<web-app>
```

12、在 spring-prj1 中新建 register.jsp 页面，作为用户注册视图，新建 regiSuccess.jsp 和 regFail.jsp 页面，分别作为注册成功和失败的视图（代码略）；
13、将 spring-prj1 部署在 Tomcat 服务器上；
14、通过浏览器访问 register.jsp 页面，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：
（1）运行结果截图；
（2）结合实验过程，总结 Spring 整合 Struts2 框架的关键步骤，并记录下来；
（3）根据实验步骤 7，总结本实验中的 UserAction 与以往实验中的写法关键区别，并记录下来；
（4）根据实验步骤 9，查找相关资料，总结配置文件 applicationContext.xml 中 bean 元素的 prototype 属性及其取值的含义，并记录下来；
（5）根据实验步骤 11，查找相关资料，总结 web.xml 文件中添加监听器的目的，并记录下来；
（6）碰到的问题及解决方案或思考；
（7）实验收获及总结。
2、上交程序源代码，代码中应有相关注释。

三、扩展实验——Spring AOP 实现事务管理

（一）实验目的

1、进一步理解 AOP 的基本概念与作用；
2、理解 Spring 框架基于 AOP 实现声明式事务管理的基本机制，掌握进行声明式事务管理的基本方法；
3、掌握 Spring 配置文件为实现声明式事务管理所涉及的主要元素及属性，理解其作用，并能进行正确的配置；
4、了解 Spring 事务管理中的七种事务传播行为（propagation），并理解不同的事务传播行为起到的不同作用。

（二）基本知识与原理

1、事务管理是企业应用中非常重要的部分，Spring 框架对事务管理进行了高层

次的抽象，定义了各种类型的事务管理器，用来实现事务管理功能；

2、Spring 框架支持编程式事务管理，也就是可以通过编写代码实现事务管理；Spring 同时也支持声明式事务管理，声明式事务管理基于 Spring AOP 实现，不在源文件中编写代码管理事务，而是使用 AOP 框架，在 IoC 容器中装配；

3、Spring 配置文件中提供了<tx:advice.../>元素来配置事务增强处理，并使用<aop:advisor.../>为 IoC 容器中的 Bean 配置自动事务代理。

（三）实验内容及步骤

1、修改 Spring 配置文件 applicationContext.xml，在头文件中添加“xmlns:tx”的命名申明，在“xsi:schemaLocation”中指定 tx 配置的 schema 的地址，代码片段如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-5.0.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-5.0.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-5.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-5.0.xsd
        ">
     ……
</beans>
```

2、修改 Spring 配置文件，配置事务管理器，代码片段如下：

```
<!-- 配置事务管理器 -->
<bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">
    <property name="sessionFactory">
        <ref bean="sessionFactory" />
    </property>
</bean>
```

3、修改 Spring 配置文件，通过 AOP 的方式为工程 spring-prj1 添加事务管理，代码片段如下：

```
<!-- 定义增强处理拦截方法 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="add*" propagation="REQUIRED" />
        <tx:method name="update*" propagation="REQUIRED" />
        <tx:method name="delete*" propagation="REQUIRED" />
        <tx:method name="del*" propagation="REQUIRED" />
        <tx:method name="*" read-only="true" />
    </tx:attributes>
</tx:advice>
```

4、修改 Spring 配置文件，定义事务拦截切面，通过 AOP 的方式添加事务管理，代码片段如下：

```
<!-- 定义事务拦截切面 -->
<aop:config>
    <aop:pointcut id="allServiceMethod"
         expression="execution(* cn.edu.zjut.service.*.*(..))" />
    <aop:advisor pointcut-ref="allServiceMethod"
                 advice-ref="txAdvice" />
</aop:config>
```

5、运行测试类 SpringEnvTest，观察数据库中是否新增加了 SPRING 用户，思考导致该结果的原因并记录下来。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验步骤，查找相关资料，总结 Spring 配置文件中与实现声明式事务管理相关的主要元素及其属性的作用，并记录下来；

（3）查找相关资料，总结 Spring 事务管理中的七种事务传播行为（propagation）所起到的不同作用，并记录下来；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。