**每日作业卷答案**

**Spring第5天**

极客营

# 关卡1

## 训练案例1

### 训练描述

把demo1项目发布到tomcat服务器中，启动服务器。访问demo1.jsp的页面，填入数据，提交表单。在CustomerAction中接收表单数据，打印到控制台上。（使用Spring框架整合Struts2框架）。

### 操作步骤描述

1. 把demo1.jsp中，给表单的action属性填入值，访问后台的CustomerAction的save方法  
     
   <form ction="${ pageContext.request.contextPath }/customerAction\_save.action" method="post">
2. 在applicationContext.xml配置文件中配置CustomerAction，交给IOC进行管理。  
     
   <bean d="customerAction" class="com.igeek.customer.action.CustomerAction" scope="prototype"></bean>
3. 在struts.xml配置文件中配置CustomerAction

<package name="crm" extends="struts-default" namespace="/">

<action name="customerAction\_\*" class="customerAction" method="{1}"></action>

</package>

1. stomerAction使用模型驱动的方式接收表单提交的值，在save方法中输出结果  
   public class CustomerAction extends ActionSupport implements ModelDriven<Customer>{

private Customer customer = new Customer();

public Customer getModel() {

return customer;

}

public String save(){

System.out.println("整合Struts2框架...");

System.out.println(customer.getCust\_name());

return NONE;

}

}

# 关卡2

## 训练案例1

### 训练描述

使用Spring的JdbcTemplate完成转账的代码编写。提供的demo2模板，完成转账的代码的编写, 创建数据库spring\_day05。

### 操作步骤描述

1. 在applicationContext.xml配置文件整合hibernate框架，创建SessionFactory对象。  
  
<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="configLocation" value="classpath:hibernate.cfg.xml"/>

</bean>

2. 在applicationContext.xml配置文件中给CustomerDao注入SessionFactory对象。  
  
<bean id="customerDao" class="com.igeek.customer.dao.CustomerDaoImpl">

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

3. 在CustomerDaoImpl类的save方法中完成保存客户的代码  
  
public void save(Customer customer) {

this.getHibernateTemplate().save(customer);

}

4. 在applicationContext.xml配置文件中配置声明式事务管理  
<bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

<tx:advice id="myAdvice" transaction-manager="transactionManager">

<tx:attributes>

<tx:method name="save\*"/>

</tx:attributes>

</tx:advice>

<aop:config>

<aop:advisor advice-ref="myAdvice" pointcut="execution(public \* com.igeek.customer.service.CustomerServiceImpl.save(..))"/>

</aop:config>

# 关卡3

## 训练案例

### 训练描述

把demo3项目发布到tomact服务器中，启动服务器。访问demo1.jsp的页面，点击查询按钮，需要查询出所有的客户数据，然后转发回demo1.jsp的页面，使用struts的迭代标签把数据显示到jsp的页面上。

### 操作步骤描述

1. 在demo1.jsp页面上，编写超链接的href属性，访问CustomerAction类中的list方法  
     
   <a href="${ pageContext.request.contextPath }/customerAction\_list">查询</a>

2. 在CustomerAction类中的list方法中调用customerService的list方法，查询所有的客户，把查询到的客户集合压入到值栈中，转发回demo1.jsp的页面  
public String list(){

List<Customer> list = customerService.list();

ActionContext.getContext().put("list", list);

return "list";

}

1. 在struts.xml配置文件中配置list方法的跳转页面  
     
   <action name="customerAction\_\*" class="customerAction" method="{1}">

<result name="list">/demo1.jsp</result>

</action>

1. 编写CustomerDaoImpl的list方法，查询所有的客户  
     
   public List<Customer> list() {

return (List<Customer>) this.getHibernateTemplate().find("from Customer");

}

5. 在demo1.jsp页面上，使用struts2的迭代标签回显数据（显示主键和名称即可）。  
<s:iterator value="#list">

<s:property value="cust\_id"/>

<s:property value="cust\_name"/>

</s:iterator>

# 关卡4

## 训练案例

### 训练描述

现在给出了关卡4的模板，要求使用配置文件和注解的方式完成SSH三大框架的整合。把demo3项目发布到tomact服务器中，启动服务器。访问demo1.jsp的页面，在表单中输入数据，点击保存按钮，把数据保存到数据库中。

### 操作步骤答案

1. 在applicationContext.xml的配置文件开启注解扫描  
     
   <context:component-scan base-package="com.igeek"/>
2. 在CustomerAction类上添加Controller注解  
   @Controller(value="customerAction")
3. 在CustomerAction类上添加Scope注解  
   @Scope(value="prototype")
4. 在CustomerServiceImpl上添加Service注解  
   @Service(value="customerService")
5. 在CustomerDaoImpl类上添加Repository注解

@Repository(value="customerDao")

1. 在CustomerAction类中注入CustomerServiceImpl对象  
   @Resource(name="customerService")

private CustomerService customerService;

1. 在CustomerServiceImpl中注入CustomerDaoImpl对象  
   @Resource(name="customerDao")

private CustomerDao customerDao;

1. 在applicationContext.xml配置文件配置HibernateTemplate  
   <bean id="hibernateTemplate" class="org.springframework.orm.hibernate5.HibernateTemplate">

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

1. 在CustomerDaoImpl中注入HibernateTemplate对象  
   @Resource(name="hibernateTemplate")

private HibernateTemplate hibernateTemplate;

1. 配置平台事务管理器对象  
   <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!-- 注入DataSource -->

<property name="dataSource" ref="dataSource"></property>

</bean>

1. 开启事务注解  
   <tx:annotation-driven transaction-manager="transactionManager"/>
2. 在CustomerServiceImpl上添加事务注解  
   @Transactional
3. 在demo1.jsp页面上，添加数据，点击保存按钮，完成数据的保存。