**每日作业卷答案**

**Spring第4天**

极客营

# 关卡1

## 训练案例1

### 训练描述

使用Spring的JdbcTemplate模板技术完成数据的保存（要求使用IOC的方式，使用的是C3P0的数据源）。提供了如下的建表语句。  
创建数据库：

create database spring\_day04;

use spring\_day04;

创建表：

create table account(

id int primary key auto\_increment,

name varchar(40),

money double

)character set utf8 collate utf8\_general\_ci;

### 操作步骤描述

1. 创建数据库和表结构（拷贝上述的建表语句等执行）  
   创建数据库：

create database spring\_day04;

use spring\_day04;

创建表：

create table account(

id int primary key auto\_increment,

name varchar(40),

money double

)character set utf8 collate utf8\_general\_ci;

1. 在applicationContext.xml配置文件中配置C3P0的连接池  
   <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="com.mysql.jdbc.Driver"/>

<property name="jdbcUrl" value="jdbc:mysql:///spring\_day04"/>

<property name="user" value="root"/>

<property name="password" value="root"/>

</bean>

1. 在applicationContext.xml配置文件中配置JdbcTemplate模板  
   <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

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

</bean>

1. 在Demo1类中的注入JdbcTemplate对象  
   @Resource(name="jdbcTemplate")

private JdbcTemplate jdbcTemplate;

1. 在Demo1类的run1测试方法中编写代码，保存数据到数据库中  
   @Test

public void run1(){

jdbcTemplate.update("insert into account values (null,?,?)", "张三",1000d);

}

# 关卡2

## 训练案例

### 训练描述

在关卡一的基础上，把保存到数据库中的数据，使用Spring的JdbcTemplate模板技术查询出来，把查询结果打印到控制台上。要求查询所有的数据。  
创建数据库：

create database spring\_day04;

use spring\_day04;

创建表：

create table account(

id int primary key auto\_increment,

name varchar(40),

money double

)character set utf8 collate utf8\_general\_ci;

### 操作步骤答案

1. 创建数据库和表结构（拷贝上述的建表语句等执行）  
   创建数据库：

create database spring\_day04;

use spring\_day04;

创建表：

create table account(

id int primary key auto\_increment,

name varchar(40),

money double

)character set utf8 collate utf8\_general\_ci;

1. 在applicationContext.xml配置文件中配置C3P0的连接池  
   <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="com.mysql.jdbc.Driver"/>

<property name="jdbcUrl" value="jdbc:mysql:///spring\_day04"/>

<property name="user" value="root"/>

<property name="password" value="root"/>

</bean>

1. 在applicationContext.xml配置文件中配置JdbcTemplate模板  
   <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

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

</bean>

1. 在Demo1类中的注入JdbcTemplate对象  
   @Resource(name="jdbcTemplate")

private JdbcTemplate jdbcTemplate;

1. 在Demo1类的run1测试方法中编写代码，查询数据结果打印到控制台上。查询所有的数据。

@RunWith(value = SpringJUnit4ClassRunner.class)

@ContextConfiguration(value = "classpath:applicationContext.xml")

public class Demo1 {

private JdbcTemplate jdbcTemplate;

@Test

public void run1() {

List<Account> accounts = jdbcTemplate.query("select \* from account", new AccountRowMapper());

for (Account o : accounts) {

System.out.println(o);

}

}

}

class AccountRowMapper implements RowMapper<Account> {

public Account mapRow(ResultSet rs, int rowNum) throws SQLException {

Account account = new Account();

account.setId(rs.getInt("id"));

account.setName(rs.getString("name"));

account.setMoney(rs.getDouble("money"));

return account;

}

}

# 关卡3

## 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。

### 操作步骤描述

1. 先向account表中添加2条数据。执行如下sql语句即可。  
   INSERT INTO account VALUES (NULL,'赵四',1000);

INSERT INTO account VALUES (NULL,'王五',1000);

1. 在applicationContext.xml配置文件中配置C3P0的连接池  
   <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="com.mysql.jdbc.Driver"/>

<property name="jdbcUrl" value="jdbc:mysql:///spring\_day04"/>

<property name="user" value="root"/>

<property name="password" value="root"/>

</bean>

1. 在applicationContext.xml配置文件中，给accountDao注入dataSource对象  
   <bean id="accountDao" class="com.igeek.dao.AccountDaoImpl">

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

</bean>

4. 在AccountDaoImpl类中完成转账代码的编写  
/\*\*

\* 付款

\*/

public void fromMoney(String from, double money) {

this.getJdbcTemplate().update("update account set money = money - ? where name = ?", money,from);

}

/\*\*

\* 收款

\*/

public void toMoney(String to, double money) {

this.getJdbcTemplate().update("update account set money = money + ? where name = ?", money,to);

}

# 关卡4

## 训练案例

### 训练描述

现在提供了demo3的模板程序，程序中的业务逻辑是向account表中添加一条数据，同时需要修改王五的金额为1500元，程序已经编写完成。但是模拟了一个异常，运行程序会出现异常。结果是表中保存了数据，但是王五的金额没有进行修改。现在需要使用Spring的声明式事务管理的方式对AccountServiceImpl中的pay方法进行事务管理（使用配置文件的方式）。

### 操作步骤描述

1. 在applicationContext.xml配置文件中配置事务管理器  
   <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!-- 注入DataSource -->

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

</bean>

1. 在applicationContext.xml配置文件中配置事务通知  
   <tx:advice transaction-manager="transactionManager" id="txAdvice">

<tx:attributes>

<tx:method name="pay" isolation="DEFAULT" propagation="REQUIRED"/>

</tx:attributes>

</tx:advice>

1. 在applicationContext.xml配置文件中配置AOP的事务增强

<aop:config>

<aop:advisor advice-ref="txAdvice" pointcut="execution(public \* com.igeek.service.AccountServiceImpl.pay(..))"/>

</aop:config>

# 关卡5

## 训练案例

### 训练描述

现在提供了demo3的模板程序，程序中的业务逻辑是向account表中添加一条数据，同时需要修改王五的金额为1500元，程序已经编写完成。但是模拟了一个异常，运行程序会出现异常。结果是表中保存了数据，但是王五的金额没有进行修改。现在需要使用Spring的声明式事务管理的方式对AccountServiceImpl中的pay方法进行事务管理（使用 声明式事务注解加配置文件的方式）。

### 操作步骤答案

1. 在applicationContext.xml配置文件中配置事务管理器  
   <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!-- 注入DataSource -->

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

</bean>

1. 在applicationContext.xml配置文件中开启事务注解  
   <tx:annotation-driven transaction-manager="transactionManager"/>
2. 在AccountServiceImpl类中添加事务的注解  
   @Transactional

public class AccountServiceImpl implements AccountService