Spring中的数据库操作模板类

JdbcTemplate：用于直接通过JDBC访问数据用的

HibernateTemplate：用于通过Hibernate框架访问数据库用的

使用：JdbcTemplate必备JAR包：

Spring的IOC基础6个：

commons-logging-1.2.jar

log4j-1.2.16.jar

spring-beans-4.2.4.RELEASE.jar

spring-context-4.2.4.RELEASE.jar

spring-core-4.2.4.RELEASE.jar

spring-expression-4.2.4.RELEASE.jar

Spring的事务3个：

spring-jdbc-4.2.4.RELEASE.jar

spring-orm-4.2.4.RELEASE.jar

spring-tx-4.2.4.RELEASE.jar

数据库连接的驱动1个：

mysql-connector-java-5.1.7-bin.jar

JdbcDaoSupport是**Spring**提供的一个**JdbcTemplate**的模板，

这个模板里已经定以好了**JdbcTemplate**的**get/set**方法以及**dataSource**数据源对象

直接使用**JdbcTemplate**和继承**JdbcDaoSupport**模板的区别：

1、继承**JdbcDaoSupport**模板，**XML**的**IOC**配置更简洁

2、直接使用JdbcTemplate，可以通过注解IOC配置

使用JdbcTemplate

```
/**
 * 账户操作的持久层实现类，直接使用JdbcTemplate模板进行JDBC操作
 */
public class AccountDaoImpl implements IAccountDao {

    private JdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    @Override
    public void saveAccount(Account account) {
        jdbcTemplate.update("insert into account(name,money)values(?,?)",account.getName(),account.getMoney());
    }
```

直接使用JdbcTemplate操作JDBC

```
<!-- 配置dao -->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <!-- 注入jdbcTemplate -->
    <property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

<!-- 配置一个jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <!-- 注入数据源 -->
    <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>

<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!-- 注入数据 -->
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

使用JdbcTemplate的IOC配置

继承JdbcDaoSupport模板

```
/**
 * 账户操作的持久层实现类
 * JdbcDaoSupport是Spring提供的一个JdbcTemplate的模板，
 * 这个模板里已经定以好了JdbcTemplate的get/set方法以及dataSource数据源对象
 */
public class AccountDaoImpl extends JdbcDaoSupport implements IAccountDao {

    @Override
    public void saveAccount(Account account) {
        getJdbcTemplate().update("insert into account(name,money)values(?,?)",account.getName(),account.getMoney());
    }
```

继承JdbcDaoSupport模板类后不再手动去创建JdbcTemplate对象，直接通过getJdbcTemplate方式获取JdbcTemplate对象来操作数据库

```xml
<!-- 配置dao -->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>

<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!-- 注入数据 -->
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

直接注入数据源

JdbcTemplate的基本操作：

创建数据库连接源

创建JdbcTemplate对象设置连接源

DriverManagerDataSource是Spring内置的数据源（相当于C3P0）

执行SQL语句

```java
//创建一个数据源（连接池） c3p0 dbcp  spring内置的DriverManagerDataSource
DriverManagerDataSource ds = new DriverManagerDataSource();
//设置连接数据库的信息
ds.setDriverClassName("com.mysql.jdbc.Driver");
ds.setUrl("jdbc:mysql://localhost:3306/day36_szee03_spring");
ds.setUsername("root");
ds.setPassword("1234");
```

基本操作实现

```java
//1.创建JdbcTemplate对象,并通过构造设置数据源
JdbcTemplate jt = new JdbcTemplate(ds);
//给jdbcTemplate设置数据源
jt.setDataSource(ds);
//2.执行方法：保存一个账户
jt.execute("insert into account(name,money)values('ddd',1000)");
```

通过Spring的IOC配置JdbcTemplate对象和DriverManagerDataSource数据源对象实现

```xml
<!-- 配置一个jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <!-- 注入数据源 -->
    <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>

<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!-- 注入数据 -->
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

```java
//1.获取Spring容器
ApplicationContext ac = new ClassPathXmlApplicationContext("bean.xml");
//2.根据id获取bean对象
JdbcTemplate jt = (JdbcTemplate) ac.getBean("jdbcTemplate");
//3.执行操作
jt.execute("insert into account(name,money)values('eee',2000)");
```

Spring整合junit

使用Spring注解的方式将@Test测试整合到一起：

junit的@Test只所以能运行是因为junit内有main方法，通过反射判断方法有没有@Test注解，如果有就将方法放入main方法里执行。

整合junit后，junit的main方法不知道spring容器的存在所以会出现无法执行测试方法

解决办法：

把不知道spring容器的main方法给替换了，换成知道有spring容器存在的。

* 步骤：

1、拷贝spring-test-4.2.4.jar到lib目录中

2、使用

@RunWith注解替换运行器main方法

@ContextConfiguration指定spring配置文件的位置

要用到的Jar包：除了Spring的IOC及用于Spring注解包以外

spring-test-4.2.4.jar　　整合junit的核心JAR包

junit-4.9.jar　　单元测试包

```
 * spring整合junit:
 *    思路：把不知道spring容器的main方法给替换了，换成知道有spring容器存在的。
 *   步骤：
 *      1、拷贝spring-test-4.2.4.jar到lib目录中
 *      2、使用
 *          @RunWith注解替换运行器main方法
 *          @ContextConfiguration指定spring配置文件的位置
 */
@RunWith(SpringJUnit4ClassRunner.class)//指定知道spring容器的main方法
@ContextConfiguration(locations={"classpath:bean.xml"})//指定spring配置文件的位置
public class AccountDaoTest {

    @Autowired   //IOC注解自动匹配注入
    private IAccountDao accountDao = null;

    @Test
    public void testFindById(){
//      ApplicationContext ac = new ClassPathXmlApplicationContext("bean.xml");
//      IAccountDao accountDao = (IAccountDao) ac.getBean("accountDao");
        Account account = accountDao.findAccountById(1);
        System.out.println(account);
    }
```

有注解注入就不需要下面的获取容器了