Day28

java课程-李彦伯

spring整合JDBC

> spring中提供了一个可以操作数据库的对象,这个对象封装了jdbc技术,这个对象叫做JDBCTemplate(JDBC模版对象)

演示JDBCTemplate如何进行数据库的操作

开发流程

- 导包4+2+1(aop)+1(test)+JDBC驱动+c3p0连接池+spring-jdbc(spring包中)+spring-tx(spring包中)
- 书写Dao层接口
- 书写Dao的实现类
- 填写增删改查方法

```java
@Repository("userDao")
public class UserDaoImpl implements UserDao {
    @Autowired
    private JdbcTemplate jt;

    @Override
    public void addUser(User u) {
        String sql = "insert into t_user values (null,?)";
        jt.update(sql, u.getName());
    }
    @Override
    public void updateUser(User u) {
        String sql = "update t_user set name=? where id = ?";
        jt.update(sql,u.getName(),u.getId());
    }
    @Override
    public void deleteUserById(int id) {
        String sql = "delete from t_user where id = ?";
        jt.update(sql,id);
    }

    @Override
    public List<User> findAllUser() {
        String sql = "select * from t_use
```

```
r";

        List<User> li = jt.query(sql, new R
owMapper<User>() {
            @Override
            public User mapRow(ResultSet
set, int arg) throws SQLException {
                    System.out.println(ar
g);
                    User u = new User();
                    u.setId(set.getInt("i
d"));
                    u.setName(set.getStri
ng("name"));
                return u;
            }
        });
         return li;
    }
    @Override
    public int findAllUserCount() {
        String sql = "select count(*) fro
m t_user";
         int count = jt.queryForObject(sq
l, Integer.class);
        return count;
    }
}
```

- 配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance"
xmlns="http://www.springframework.org/sch
ema/beans"
 xmlns:context="http://www.springframewor
k.org/schema/context"
 xsi:schemaLocation="http://www.springfra
mework.org/schema/beans
  http://www.springframework.org/schema/b
eans/spring-beans-4.2.xsd
  http://www.springframework.org/schema/c
ontext
  http://www.springframework.org/schema/c
ontext/spring-context-4.2.xsd ">
<bean name="datasource" class="com.mchang
e.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="c
om.mysql.jdbc.Driver"></property>
    <property name="jdbcUrl" value="jdb
c:mysql:///bigdata14"></property>
    <property name="user" value="root">
</property>
    <property name="password" value="roo
t"></property>
</bean>
<bean name="jdbcTemplate" class="org.spri
ngframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="data
```

```
source"></property>
</bean>
<context:component-scan base-package="co
m.zhiyou100.spring.dao"></context:compone
nt-scan>
</beans>
```

JdbcDaoSupport

> JdbcDaoSupport类内部封装了一个JDBCTemplate模版,在需要的时候调用this.getJdbcTemplate()即可,所以我们可以使我们的daoImpl继承JDBCDaoSupport,在需要的时候调用this.getJDBCTemplate

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance"
xmlns="http://www.springframework.org/sch
ema/beans"
 xmlns:context="http://www.springframewor
k.org/schema/context"
 xsi:schemaLocation="http://www.springfra
mework.org/schema/beans
  http://www.springframework.org/schema/b
eans/spring-beans-4.2.xsd
  http://www.springframework.org/schema/c
ontext
  http://www.springframework.org/schema/c
ontext/spring-context-4.2.xsd ">
<bean name="datasource" class="com.mchang
e.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="c
om.mysql.jdbc.Driver"></property>
    <property name="jdbcUrl" value="jdb
c:mysql:///bigdata14"></property>
    <property name="user" value="root">
</property>
    <property name="password" value="roo
t"></property>
</bean>
<bean name="userDao" class="com.zhiyou10
0.spring.dao.impl.UserDaoImpl">
    <property name="dataSource" ref="data
```

```
source"></property>
</bean>
</beans>
```

```
public class UserDaoImpl extends JdbcDaoS
upport implements UserDao {

    @Override
    public void addUser(User u) {
        String sql = "insert into t_user
values (null,?)";
        this.getJdbcTemplate().update(sq
l, u.getName());
    }

    @Override
    public void updateUser(User u) {
        String sql = "update t_user set n
ame=? where id = ?";
        this.getJdbcTemplate().update(sq
l,u.getName(),u.getId());
    }

    @Override
    public void deleteUserById(int id) {
        String sql = "delete from t_user
where id = ?";
        this.getJdbcTemplate().update(sq
l,id);
    }

    @Override
    public List<User> findAllUser() {
```

```
        String sql = "select * from t_use
r";

      List<User> li = this.getJdbcTemplat
e().query(sql, new RowMapper<User>() {
            @Override
            public User mapRow(ResultSet
set, int arg) throws SQLException {
                    System.out.println(ar
g);
                    User u = new User();
                    u.setId(set.getInt("i
d"));
                    u.setName(set.getStri
ng("name"));
                return u;
            }
        });
         return li;
    }
    @Override
    public int findAllUserCount() {
        String sql = "select count(*) fro
m t_user";
          int count = this.getJdbcTemplat
e().queryForObject(sql, Integer.class);
        return count;
    }
}
```

properyies配置jdbc连接信息

```
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.jdbcUrl=jdbc:mysql:///bigdata14
jdbc.user=root
jdbc.password=root
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance"
xmlns="http://www.springframework.org/sch
ema/beans"
 xmlns:context="http://www.springframewor
k.org/schema/context"
 xsi:schemaLocation="http://www.springfra
mework.org/schema/beans
  http://www.springframework.org/schema/b
eans/spring-beans-4.2.xsd
  http://www.springframework.org/schema/c
ontext
  http://www.springframework.org/schema/c
ontext/spring-context-4.2.xsd ">

<context:property-placeholder location="c
lasspath:db.properties"/>
<bean name="datasource" class="com.mchang
e.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" valu
e="${jdbc.driverClass}"></property>
    <property name="jdbcUrl" value="${jdb
c.jdbcUrl}"></property>
    <property name="user" value="${jdbc.u
ser}"></property>
    <property name="password" value="${jd
bc.password}"></property>
</bean>
```

```
<!-- <bean name="jdbcTemplate" class="or
g.springframework.jdbc.core.JdbcTemplat
e">
    <property name="dataSource" ref="data
source"></property>
</bean> -->
<context:component-scan base-package="co
m.zhiyou100.spring.dao"></context:compone
nt-scan>
<bean name="userDao" class="com.zhiyou10
0.spring.dao.impl.UserDaoImpl">
    <property name="dataSource" ref="data
source"></property>
</bean>
</beans>
```

事务

事务的特性

- 原子性（Atomicity）原子性是指事务是一个不可分割的工作单位，事务中的操作 要么都发生，要么都不发生。。
- 一致性（Consistency）一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态.拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A

和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性

- 隔离性（Isolation）隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离.即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行
- 持久性（Durability）持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

事务并发引起的问题

- 脏读 B事务读取到了A事务尚未提交的数据 ------ 要求B事务要读取A事 务提交的数据
- 不可重复读不可重复读是指在对于数据库中的某个数据，一个事务范围内多次查询却返回了不同的数据值，这是由于在查询间隔，被另一个事务修改并提交了
- 幻读(虚读)幻读是事务非独立执行时发生的一种现象。例如事务T1对一个表中所有的行的某个数据项做了从“1”修改为“2”的操作，这时事务T2又对这个表中插入了一行数据项，而这个数据项的数值还是为“1”并且提交给数据库。而操作事务T1的用户如果再查看刚刚修改的数据，会发现还有一行没有修改，其实这行是从事务T2中添加的，就好像产生幻觉一样，这就是发生了幻读

事务隔离级别

- read uncommitted：读取尚未提交的数据 ：哪个问题都不能解决
- read committed：读取已经提交的数据 ：可以解决脏读 —- oracle默认的
- repeatable read：重读读取：可以解决脏读 和 不可重复读 —mysql默认的
- serializable：串行化：可以解决 脏读 不可重复读 和 虚读—相当于锁表
- 查看mysql数据库默认的隔离级别：`select @@tx_isolation`
- 设置mysql的隔离级别：`set session transaction isolation level 设置事务隔离级别`
- mysql的事务控制：
  - 开启事务：start transaction;
  - 提交：commit；
  - 回滚：rollback；

- JDBC事务控制：
  - 开启事务：conn.setAutoCommit(false);
  - 提交：conn.commit()；
  - 回滚：conn.rollback():

- 隔离级别的性能：`read uncommitted>read committed>repeatable read>serialazable`
- 安全性:`read uncommitted<read committed<repeatable read<serialazable`

spring中的事务

> 不同平台操作事务的方式不同,如jdbc是使用connection,Hibernate操作事务是使用transition,mybatis操作事务是使用session操作,但是不管是哪种框架,都要实现spring的PlatformTransactionManager接口

spring管理事务的属性

- 隔离级别
  - 读未提交
  - 读已提交
  - 可重复读
  - 串行化

- 是否只读,表示操作事务的时候,当前事务是否是只读,比如查询操作就应该只读,其他方法牵扯到修改就是非只读
  - true表示只读(不允许做修改数据库的操作)
  - false表示非只读

- 事务的传播行为,在业务方法平行调用的时候,事务应该如何处理的问题
  - 保证同一事务中
    - PROPAGATION_REQUIRED 支持当前事

务，如果不存在 就新建一个(默认)
        - PROPAGATION_SUPPORTS 支持当前事务，如果不存在，就不使用事务
        - PROPAGATION_MANDATORY 支持当前事务，如果不存在，抛出异常

    - 保证不在同一事务中
        - PROPAGATION_REQUIRES_NEW 如果有事务存在，挂起当前事务，创建一个新的事务
            - PROPAGATION_NOT_SUPPORTED 以非事务方式运行，如果有事务存在，挂起当前事务
            - PROPAGATION_NEVER 以非事务方式运行，如果有事务存在，抛出异常
            - PROPAGATION_NESTED 如果当前事务存在，则嵌套事务执行

案例分析

转账业务

```
public class AccountDaoImpl extends JdbcD
aoSupport implements AccountDao {
    @Override
    public void addMoney(int id, double m
oney) {
        this.getJdbcTemplate().update("up
date  t_account set money = money + ? whe
re id = ?",money,id);
    }
    @Override
    public void reduceMoney(int id, doubl
e money) {
        this.getJdbcTemplate().update("up
date  t_account set money = money - ? whe
re id = ?",money,id);
    }
}
```

```
@Service("accountService")
public class AccountServiceImpl implement
s AccountService{
    @Autowired
    AccountDao ad;
    @Override
    public void transfer(int from, int t
o, double money) {
        ad.reduceMoney(from, money);
        int i = 1/0;
        ad.addMoney(to, money);
    }
}
```

添加事务管理

- 导包4+2+1(aop)+1(aspect)+1(aop联盟)+1(weaving)
- 配置核心事务管理器

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance"
    xmlns="http://www.springframework.or
g/schema/beans"
    xmlns:context="http://www.springframe
work.org/schema/context"
    xmlns:tx="http://www.springframewor
k.org/schema/tx"
    xmlns:aop="http://www.springframewor
k.org/schema/aop"
    xsi:schemaLocation="http://www.spring
framework.org/schema/beans http://www.spr
ingframework.org/schema/beans/spring-bean
s-4.2.xsd
        http://www.springframework.org/sc
hema/context http://www.springframework.o
rg/schema/context/spring-context-4.2.xsd
        http://www.springframework.org/sc
hema/aop http://www.springframework.org/s
chema/aop/spring-aop-4.2.xsd
        http://www.springframework.org/sc
hema/tx http://www.springframework.org/sc
hema/tx/spring-tx-4.2.xsd">

<context:property-placeholder location="c
lasspath:db.properties"/>
<bean name="datasource" class="com.mchang
e.v2.c3p0.ComboPooledDataSource">
```

```
    <property name="driverClass" valu
e="${jdbc.driverClass}"></property>
    <property name="jdbcUrl" value="${jdb
c.jdbcUrl}"></property>
    <property name="user" value="${jdbc.u
ser}"></property>
    <property name="password" value="${jd
bc.password}"></property>
</bean>

<bean name="userDao" class="com.zhiyou10
0.spring.dao.impl.UserDaoimpl">
<property name="dataSource" ref="datasour
ce"></property>
</bean>


<bean name="userService" class="com.zhiyo
u100.spring.service.impl.UserServiceImp
l">
    <property name="ud" ref="userDao"></p
roperty>
</bean>


<bean name="transactionManager" class="or
g.springframework.jdbc.datasource.DataSou
rceTransactionManager">
    <property name="dataSource" ref="data
source"></property>
```

```
</bean>

<tx:advice transaction-manager="transacti
onManager" id="txAdvice">
    <tx:attributes>
        <tx:method name="insert*" isolati
on="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="add*" isolatio
n="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="delete*" isolati
on="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="remove*" isolati
on="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="update*" isolati
on="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="modify*" isolati
on="REPEATABLE_READ" propagation="REQUIRE
D" read-only="false"/>
        <tx:method name="get*" isolatio
n="REPEATABLE_READ" propagation="REQUIRE
D" read-only="true"/>
        <tx:method name="find*" isolatio
n="REPEATABLE_READ" propagation="REQUIRE
D" read-only="true"/>
        <tx:method name="transfer" isolat
```

```
ion="REPEATABLE_READ" propagation="REQUIR
ED" read-only="false"/>
    </tx:attributes>
</tx:advice>
<aop:config>
    <aop:pointcut expression="execution(
* com.zhiyou100.spring.service.impl..*Ser
viceImpl.*(..))" id="pt"/>
    <aop:advisor advice-ref="txAdvice" po
intcut-ref="pt"/>
</aop:config>
</beans>
```