

企业级框架 Spring

Unit02

Sping DAO

The background of the slide features a large, light-colored fan shape. Inside the fan is a faint, traditional East Asian ink wash landscape painting (suiboku-ga) depicting mountains, trees, and a body of water. A solid blue horizontal bar is positioned below the title text.

Spring DAO

- * DAO(Data Access Object)数据访问对象
- * Spring DAO为整合JDBC提供了封装, 简化了DAO组件的编写
- * Spring DAO提供了AOP模式的事务处理
- * Spring DAO提供了统一的异常处理层次, 它包装的异常类型DataAccessException继承自RuntimeException无须显式的捕获

Spring整合JDBC

- * Spring DAO为整合JDBC提供了JdbcTemplate和JdbcDaoSupport类。

Spring 对JDBC封装的核心是
JdbcTemplate

无须关心连接的
建立和关闭

无须关心Statement
建立和关闭

Spring整合JDBC

* JdbcTemplate提供了以下主要方法

- queryForInt()
- queryForObject()
- query()
- update()
- execute()

Spring整合JDBC

* JdbcTemplate使用方法如下：

```
public class UserDao extends JdbcDaoSupport  
    implements IUserDao{
```

```
    public void doThings(){  
        int n = super.getJdbcTemplate()  
            .queryForInt("select 8*8 from dual");  
        System.out.println("执行结果为："+n);  
    }  
}
```

Spring整合JDBC

* JdbcTemplate配置代码如下：

```
<bean id="dataSource"
      class="org.apache.commons.dbcp.BasicDataSource">
  <property name="url">
    <value>jdbc:oracle:thin:@127.0.0.1:1521:orcl
    </value>
  </property>
  ...
</bean>

<bean id="userdao" class="org.xdl.dao.UserDao">
  <property name="dataSource" ref="dataSource">
  </property>
</bean>
```

Spring整合JDBC

* 使用JdbcTemplate更新

```
String[] params = {"jdbcTemplate", "000001"};  
this.getJdbcTemplate().update(  
    "update userinfo set username=? where userid=?",  
    params);
```


Spring整合JDBC

* 使用JdbcTemplate查询

```
this.getJdbcTemplate()  
    .queryForInt("select 8*8 from dual");
```

```
Object obj = this.getJdbcTemplate()  
    .queryForObject("select 8*8 from dual",  
        Integer.class);
```

Spring整合JDBC

* 使用JdbcTemplate查询

```
List<Emp> list = this.getJdbcTemplate()  
    .queryForObject("select * from emp",  
        new EmpRowMapper());
```

Spring整合JDBC

- * EmpRowMapper负责将一行记录封装成Emp对象，代码如下：

```
public class EmpRowMapper implements RowMapper{  
    protected Object mapRow(ResultSet rs, int n)  
        throws SQLException {  
        Emp emp = new Emp();  
        emp.setId(rs.getString("ID"));  
        emp.setName(rs.getString("NAME"));  
        emp.setAge(rs.getString("age"));  
        return emp;  
    }  
}
```

Spring 事务

The background of the slide features a large, light-colored fan shape. Inside the fan, there is a faint, traditional Chinese ink wash painting of a landscape with mountains, trees, and a body of water. A solid blue horizontal bar is positioned below the title text.

Spring事务

- * Spring提供了两种事务管理方式
 - 编程式事务管理
 - 声明式事务管理

Spring事务

- * Spring编程式事务管理使用的API为TransactionTemplate，格式如下

```
transactionTemplate.execute(new TransactionCallback() {  
    public Object doInTransaction(TransactionStatus  
status) {  
        //TODO DB操作1  
        // TODO DB操作2  
        return xxx;//返回结果  
    }  
});
```

Spring事务

- * Spring声明式事务管理，是使用Spring的AOP方式实现的。
 - ✓ 通过Spring配置将操作纳入到事务管理中
 - ✓ 解除了事务管理和代码的耦合
 - ✓ 不需要事务管理时，可直接从Spring配置文件中移除

Spring事务

Spring事务注解配置的使用方法如下：

- * 步骤一：在applicationContext.xml中声明事务组件，开启事务注解扫描，示例代码如下：

```
<!-- 声明事务管理组件 -->
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionMa
nager">
    <property name="dataSource" ref="ds"/>
</bean>
<!-- 开启事务注解扫描 -->
<tx:annotation-driven
    transaction-manager="txManager" proxy-target-class="true"/>
```


Spring事务

- * 步骤二：使用@Transactional注解声明事务，使用方法如下：

```
@Transactional
public class SomeServiceImpl implements SomeService
{
    // @Transactional
    public void insertOperation(){...}
    public void updateOperation(){...}
}
```


Spring事务

- * @Transactional注解标记有以下属性，在使用时可以根据需要做特殊设定。
 - propagation：设置事务传播
 - isolation：设置事务隔离级别
 - readOnly：设置为只读，还是可读写
 - rollbackFor：设置遇到哪些异常必须回滚
 - noRollbackFor：设置遇到哪些异常不回滚

Spring事务

- * @Transactional注解属性默认设置如下：
 - 事务传播设置是 REQUIRED
 - 事务隔离级别是 DEFAULT, 根据数据库自动选择, Oracle默认READ_COMMITTED级别
 - 事务是 读/写 readOnly=false
 - 任何 RuntimeException 将触发事务回滚, 但是任何 Checked Exception 将不触发事务回滚

总结和答疑

The background of the slide features a large, light-colored fan shape. Inside the fan, there is a faint, traditional Chinese ink wash painting of a landscape with mountains, trees, and a body of water. The fan is positioned centrally, with its handle pointing towards the bottom center of the slide.