今日内容

1. 数据库连接池
2. Spring JDBC : JDBC Template

数据库连接池

1. 概念：其实就是一个容器(集合)，存放数据库连接的容器。
   当系统初始化好后，容器被创建，容器中会申请一些连接对象，当用户来访问数据库时，从容器中获取连接对象，用户访问完之后，会将连接对象归还给容器。

2. 好处：

3. 节约资源

4. 用户访问高效

5. 实现：

6. 标准接口：DataSource javax.sql包下的
   方法：

- 获取连接：getConnection()
- 归还连接：Connection.close()。
  如果连接对象Connection是从连接池中获取的，那么调用Connection.close()方法，则不会关闭连接，而是归还连接。

1. 一般我们不去实现它，有数据库厂商来实现
   a. C3P0：数据库连接池技术
   b. Druid[ˈdruːɪd]：数据库连接池实现技术，由阿里巴巴提供的

2. C3P0：数据库连接池技术

- 步骤：
  a. 导入jar包 (两个) c3p0-0.9.5.2.jar mchange-commons-java-0.2.12.jar

- 不要忘记导入数据库驱动jar包
  b. 定义配置文件：
- 名称： c3p0.properties 或者 c3p0-config.xml
- 路径：直接将文件放在src目录下即可

c. 创建核心对象 数据库连接池对象 ComboPooledDataSource
d. 获取连接： getConnection

- 代码：
  //1.创建数据库连接池对象
  DataSource ds = new ComboPooledDataSource();
  //2. 获取连接对象
  Connection conn = ds.getConnection();

1. Druid：数据库连接池实现技术，由阿里巴巴提供的
2. 步骤：
   a. 导入jar包 druid-1.0.9.jar
   b. 定义配置文件：

- 是properties形式的

- 可以叫任意名称，可以放在任意目录下
  c. 加载配置文件。Properties
  d. 获取数据库连接池对象：通过工厂来来获取 DruidDataSourceFactory
  e. 获取连接：getConnection

- 代码：
  //3.加载配置文件
  Properties pro = new Properties();
  InputStream is =
  DruidDemo.class.getClassLoader().getResourceAsStream("druid.properties");
  pro.load(is);
  //4.获取连接池对象
  DataSource ds = DruidDataSourceFactory.createDataSource(pro);
  //5.获取连接
  Connection conn = ds.getConnection();

1. 定义工具类
2. 定义一个类 JDBCUtils
3. 提供静态代码块加载配置文件，初始化连接池对象

4. 提供方法
   a. 获取连接方法：通过数据库连接池获取连接
   b. 释放资源
   c. 获取连接池的方法

- 代码：

```
public class JDBCUtils {
//1.定义成员变量 DataSource
private static DataSource ds ;

static{
try {
//1.加载配置文件
Properties pro = new Properties();
pro.load(JDBCUtils.class.getClassLoader().getResourceAsStream("druid.properties"));
//2.获取DataSource
ds = DruidDataSourceFactory.createDataSource(pro);
} catch (IOException e) {
e.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
}
}

/**
```

- 获取连接

```
*/
public static Connection getConnection() throws SQLException {
return ds.getConnection();
}

/**
```

- 释放资源

```
*/
public static void close(Statement stmt,Connection conn){
close(null,stmt,conn);
}

/**
```

- 获取连接池方法
  */
  public static DataSource getDataSource(){
  return ds;
  }
  }

Spring JDBC

- Spring框架对JDBC的简单封装。提供了一个JDBCTemplate对象简化JDBC的开发
- 步骤：

1. 导入jar包
2. 创建JdbcTemplate对象。依赖于数据源DataSource

- JdbcTemplate template = new JdbcTemplate(ds);

1. 调用JdbcTemplate的方法来完成CRUD的操作

- update(): 执行DML语句 增、删、改语句
- queryForMap()
  查询结果将结果集封装为map集合，将列名作为key，将值作为value 将这条记录封装为一个map集合
- 注意：这个方法查询的结果集长度只能是1
- queryForList():查询结果将结果集封装为list集合
- 注意：将每一条记录封装为一个Map集合，再将Map集合装载到List集合中
- query():查询结果，将结果封装为JavaBean对象
- query的参数：RowMapper
- 一般我们使用BeanPropertyRowMapper实现类。可以完成数据到JavaBean的自动封装
- new BeanPropertyRowMapper<类型>(类型.class)
- queryForObject：查询结果，将结果封装为对象
- 一般用于聚合函数的查询

1. 练习：

- 需求：

1. 修改s_emp表中 25号数据的salary为 2300
2. 添加一条记录

3. 删除刚才添加的记录
4. 查询id为1的记录，将其封装为Map集合
5. 查询所有记录，将其封装为List
6. 查询所有记录，将其封装为Emp对象的List集合
7. 查询总记录数

- 代码：

```
/*

- 查询所有记录，将其封装为Emp对象的List集合
  */
  public static void test_queryEmpList2() {
  //String sql = "select id,last_name,start_date,dept_id from s_emp";
  String sql = "select * from s_emp";
  //该方法会查询得到map对象，然后将map对象添加到list集合中
  List list = temp.query(sql,new BeanPropertyRowMapper(Emp.class));

for (Emp e : list) {
System.out.println(e);
}
}

/*

- 7. 查询总记录数
  */
  public static void test_queryCount(){
  String sql = "select count(id) from s_emp";
  Long total = temp.queryForObject(sql, Long.class);
  System.out.println(total);
  }
```