Day15

java课程-李彦伯

JDBC

> JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。是Java访问数据库的标准规范

开发步骤

- 在工程中导入jar包

- 注册驱动`Class.forName("com.mysql.jdbc.Driver");`
- 获得连接`Connection con = DriverManager.getConnection(“jdbc:mysql://localhost:3306/mydb”,”root”,”root”);`，如果是本机路径可以写为`Connection con = DriverManager.getConnection(“jdbc:mysql:///mydb”,”root”,”root”);`
- 获取执行平台`Statement stmt = con.createStatement();`
- 执行sql语句`int num = stmt.executeUpdate(String sql);` 或者 `ResultSet rs= stmt.executeQuery(String sql);`
- 处理结果集

```
while( rs.next() ){
//方法参数为数据库表中的列名
String sid = rs.getString("pid");
//获取当前行的分类名称
String sname = rs.getString("pname");
}
```

- 释放资源

```
rs.close();
stmt.close();
con.close();
```

sql注入问题

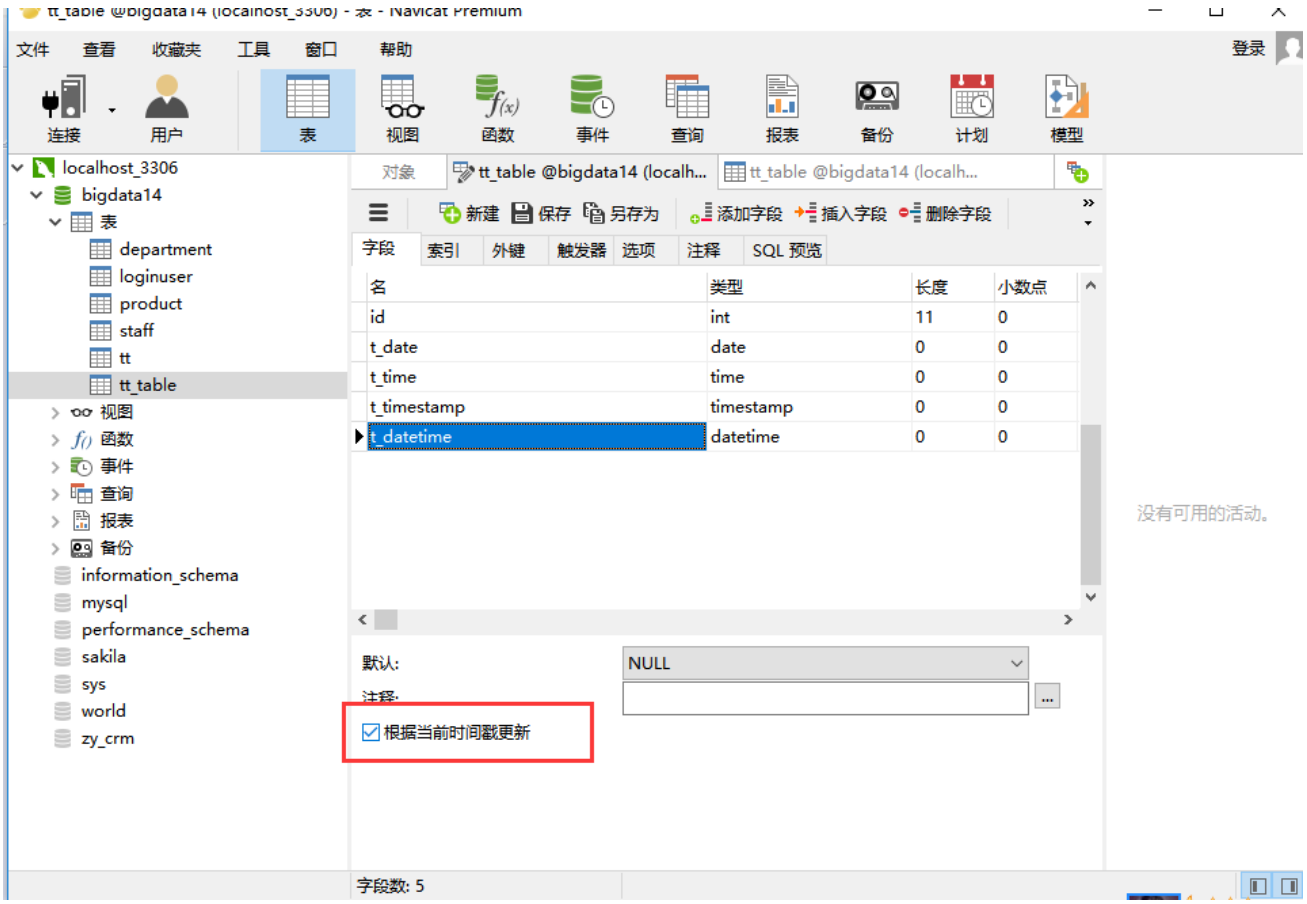
```
select * from product where pname ='电风扇' and price = '10' or '1=1'
```

预处理对象

```
// 1注册驱动
Class.forName("com.mysql.jdbc.Driver");
// 2获取连接
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3
306/mydb", "root", "root");
// 3获得预处理对象
String sql = "insert into product(pname) values(?)";
PreparedStatement stat = conn.prepareStatement(sql);
// 4 SQL语句占位符设置实际参数
stat.setString(1, "洗衣机");
// 5执行SQL语句
int line = stat.executeUpdate();
System.out.println("新添加记录数: " + line);
// 6释放资源
stat.close();
conn.close()
```

日期处理

- 对于timestamp和datetime类型,如果在创建表的时候勾选了根据当前时间更新,那么在修改当前数据的时候,就会自动把当前时间进行更新

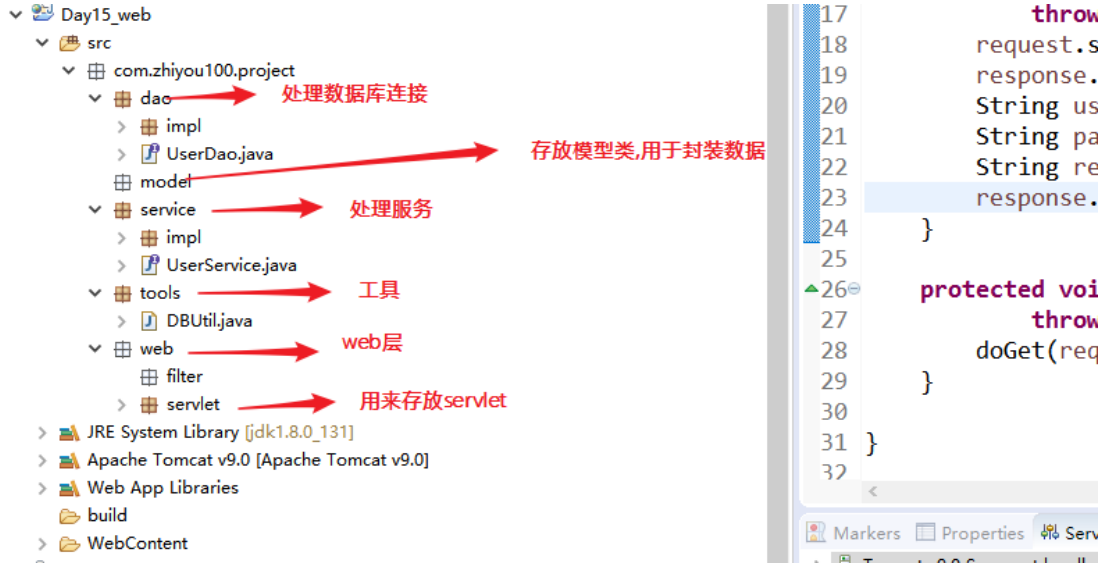


```
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql:///bigdata
14", "root", "root");
    String sql = "insert into tt_table (t_datetime) values(?)";
    String sql2 = "update tt_table set t_time = ? where id = 4";
    PreparedStatement stmt = con.prepareStatement(sql2);
    /*
    Date date = new Date(System.currentTimeMillis());
    stmt.setDate(1, date);
    */
    //对于数据库是date类型
    //stmt.setDate(1, new Date(System.currentTimeMillis()));
    //对于数据库中是time类型
    //stmt.setTime(1, new Time(System.currentTimeMillis()));
    //对于数据库中是timestamp类型
    //stmt.setTimestamp(1, new Timestamp(System.currentTimeMillis()));
    //对于数据库中是datetime类型
    //stmt.setTimestamp(1,  new Timestamp(System.currentTimeMillis()));
    //当我们进行了修改后,如果勾选了根据当前时间戳进行更新,那么对当前记录进行修改的时
候,时间戳会自动更新
    stmt.setTime(1, new Time(System.currentTimeMillis()));
    stmt.executeUpdate();
    stmt.close();
    con.close();
}catch(Exception e){
    e.printStackTrace();
}
```

JDBC工具类

```java
/*
 * JDBC工具类
 */
public class JDBCUtils {
public static final  String DRIVERNAME = "com.mysql.jdbc.Driver";
public static final  String URL = "jdbc:mysql://localhost:3306/bigdata1
4";
public static final  String USER = "root";
public static final  String PASSWORD = "root";
static {
    try {
        Class.forName(DRIVERNAME);
    } catch (ClassNotFoundException e) {
        System.out.println("数据库驱动注册失败！");
    }
}
//提供获取连接的方法
public static Connection getConn() throws Exception {
    // 2. 获得连接
    Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
    // 返回连接
    return conn;
    }
}
```

项目分层



C3P0连接池

> C3P0是目前开源免费的连接池,我们以后学习中会遇到的Spring和Hibernate都使用他,C3P0使用的时候需要添加配置文件c3p0-config.xml

操作步骤

- 导入jar包
- 将c3p0-config.xml的配置文件添加到src下进行配置
- 创建DBUtils工具类

```
public class DBUtils {
    public static DataSource db = new ComboPooledDataSource();
        public static DataSource getDataSource(){
        return db;
    }
    public static Connection getConnection(){
        Connection con = null;
        try {
            con =  db.getConnection();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return con;
    }
}
```

DBUtils工具类

> DBUtils就是JDBC的简化开发工具包。需要项目导入第三方jar包才能够正常使用DBUtils工具。

增删改操作

- 创建QueryRunner类的对象,创建对象的时候可以直接指定数据库的连接池
- 调用`update(String sql,Object...params)`方法执行

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
int a = qr.update("insert into staff values (null,?,?)", "kkk",1);
System.out.println(a);
```

查询操作

- 创建QueryRunner类的对象,创建对象的时候可以直接指定数据库的连接池对象
- 调用`query(String sql, ResultSetHandler<T> rsh, Object... params)`方法执行

ResultSetHandler

> ResultSetHandler是一个接口,当我们使用的时候需要他的实现类.

名称	用法
ArrayHandler	将结果集中的第一条记录封装到一个Object[]数组中，数组中的每一个元素就是这条记录中的每一个字段的值
ArrayListHandler	将结果集中的每一条记录都封装到一个Object[]数组中，将这些数组在封装到List集合中。
BeanHandler	将结果集中第一条记录封装到一个指定的javaBean中。
BeanListHandler	将结果集中每一条记录封装到指定的javaBean中，将这些javaBean在封装到List集合中
ColumnListHandler	将结果集中指定的列的字段值，封装到一个List集合中
ScalarHandler	它是用于单数据。例如select count(*) from 表操作。
MapHandler	将结果集第一行封装到Map集合中,Key 列名, Value 该列数据
MapListHandler	将结果集每一行封装到Map集合中,Key 列名, Value 该列数据,Map集合存储到List集合

JavaJBean

> JavaBean就是一个类，在开发中常用封装数据。具有如下特性

- 需要实现接口：java.io.Serializable ，通常实现接口这步骤省略了，不会影响程序.(可以暂时不用处理)
- 提供私有字段：private 类型 字段名;
- 提供getter/setter方法：
- 提供无参构造

ArrayHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
Object[] objects = qr.query(theSql, new ArrayHandler());
System.out.println(Arrays.toString(objects));
```

ArrayListHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
List<Object[]> objectList = qr.query(theSql, new ArrayListHandler());
System.out.println(objectList);
```

BeanHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
Staff staff = qr.query(theSql, new BeanHandler<>(Staff.class));
System.out.println(staff);
```

BeanListHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
List<Staff> list = qr.query(theSql, new BeanListHandler<>(Staff.clas
s));
System.out.println(list);
```

ColumnListHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
List<Object> query = qr.query(theSql, new ColumnListHandler<>("name"));
System.out.println(query);
```

ScalarHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
Long num = qr.query("select count(*) from staff", new ScalarHandler<Lon
g>());
System.out.println("----"+num);
```

MapHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
Map<String, Object> map = qr.query(theSql, new MapHandler());
System.out.println(map);
```

MapListHandler

```
QueryRunner qr = new QueryRunner(DBUtils.getDataSource());
String theSql = "select * from staff";
List<Map<String, Object>> query2 = qr.query(theSql, new MapListHandle
r());
System.out.println(query2);
```