JDBC

标签 空

新增属性

常规使用

连接 MySQL

在mysql应用服务器外部使用mysql，使用DriverManager类来管理连接。也就是说在你的java代码中连接并操作数据库。

```Java
public static void main(String[] args) {
    // 1.加载驱动 :根据类路径获取类并实例化(java反射)，驱动程序包需要额外引入
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    // 2.创建连接
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test?user=monty&password=greatsqldb","root","123456");

    // ======= 操作

    //关闭连接
    conn.close();
}
```

增 insert

```Java
//使用?,是通过PreparedStatement 对象来防止sql注入。
String sql = "insert into user1 values(id=?,name=?)";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setObject(1,1001);
ps.setObject(2,"zhangsan");
ps.executeUpdate();
ps.close();
```

删 delete

```Java
String sql="delete from user1 where id=? ";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setObject(1,1001);
ps.executeUpdate();
ps.close();
```

改 update

```Java
String sql = "update user1 set name=? where id=?";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setObject(1,1001);
ps.setObject(2,"lisi");
ps.executeUpdate();
ps.close();
```

查 select

```java
String sql = "select id,name from user1";
PreparedStatement ps = conn.prepareStatement(sql);
ResultSet res = ps.executeQuery();

// 结果不止一个，将结果放在list集合中
List<User> list = new ArrayList<>();
while(res.next()){
  User emp = new User();
  emp.setId(res.getInt("id"));
  emp.setName(res.getString("name"));
  list.add(emp);
}
ps.close();
```

由上面看，增删改都是一样的，查询略有不同，因为查询需要返回查询的记录信息，增删改返回影响的记录行数。

拓展知识

Statement 对象

Statement

正常执行sql语句，sql参数一般使用字符串拼接，但有注入风险。

```java
Statement  st = conn.createStatement("select * from user id = " + 1001);

boolean  bl = st.execute();
int num = st.executeUpdate()
ResultSet result = st.executeQuery();

st.close();
```

PreparedStatement

可以为sql设置参数，并可以自动对参数进行校验处理，支持Statement所有方法。

```java
PreparedStatement ps = conn.prepareStatement("select * from user id = ?");
ps.setObject(1,1001);
ps.executeUpdate();
ps.close();
```

CallableStatement

可用于调用存储过程，支持PreparedStatement 所有方法。

```java
CallableStatement cs = prepareCall("call getEmpName (?, ?)");
// INPUT参数
ps.setObject(1,1001);
// OUTPUT参数
ps.registerOutParameter(2,Types.INTEGER);
ps.executeUpdate();
cs.close();

```

事务管理

```java
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test","root","123456");
conn.setAutoCommit(false);    // 设置是否自动提交事务
conn.commit();                // 提交事务
conn.rollback();              // 回滚事务，一般在捕获异常后回滚

Savepoint point = conn.setSavepoint("point"); // 设置还原点
conn.rollback("point");       // 回滚事务到还原点

```

```java
conn.releaseSavepoint(point); // 删除还原点
```

批量处理

Statement

```java
conn.setAutoCommit(false);  // 关闭自动提交
Statement st = conn.createStatement();
st.addBatch("insert into user(name,seg) values('张三',20)");
st.addBatch("insert into user(name,seg) values('李四',21)");
st.addBatch("insert into user(name,seg) values('王五',22)");
int[] count = st.executeBatch();
conn.commit();              // 手动提交

```

PreparedStatement

```java
conn.setAutoCommit(false);  // 关闭自动提交
PreparedStatement st = conn.prepareStatement("insert into user(name,seg) values(?,?)");

st.setInt(1,"张三");
st.setString(2,20);
st.addBatch();

st.setInt(1,"李四");
st.setString(2,21);
st.addBatch();

st.setInt(1,"王五");
st.setString(2,22);
st.addBatch();

int[] count = st.executeBatch();
conn.commit();              // 手动提交

```

文件存储

即使不建议直接在数据库中存储文件，但mysql本身是可以存储文件的，因此jdbc支持了传递流类型的参数。

添加文件记录：

```java
File file = new File("test.xml");
FileInputStream input = new FileInputStream(f);

PreparedStatement ps = conn.prepareStatement("insert into tfile(file_data) values(?)");
ps.setAsciiStream(1,input,(int)file.length());
ps.execute();

input.close();
ps.close();
```

查询文件记录：

```java
Statement st = conn.createStatement("select file_data from tfile where id = 1 ");
ResultSet result = st.executeQuery();

if (rs.next ()){
   InputStream input = rs.getAsciiStream (1); // 从结果集获得文件流
}
```