

## \* 学习目标

### \* 能够自定义JDBC的框架

#### \* JDBCUtils.update

- \* sql , params

- \* stmt.getParamterMetaData()-->getParamterCount();

- \* stmt.setObject(i+1,params[i]);

#### \* JDBCUtils.query

- \* sql,params,ResultSetHandler

- \* rs---

>obj(Been,List<Bean>,Map,MapList,ScalarHandler,ColumnListHander,ArrayHandler,Array  
ListHandler,....)

#### \* ResultSetHandler

- \* Object handle(ResultSet rs);

- \* T handle ( ResultSet rs);

#### \* BeanHandler implements ResultSetHandler

- \* Class<?> clazz

- \* handle

- \* clazz.newInstance();

- \* map

- \* ResultSetMetadata

- \* getColumnCount

- \* getColumnName(i+1).tolowercase();

- \* ResultSet rs.next(),.rs.getObject(i+1);

- \* 反射基础代码，内省，BeanUtils

#### \* MapHandler

- \* map

- \* ResultSetMetadata

- \* getColumnCount

- \* getColumnName(i+1).toLowerCase();

- \* ResultSet rs.next(),rs.getObject(i+1);

- \* ArrayHandler

- \* BeanListHandler

- \* ArrayListHandler

- \* MapListHandler

- \* ScalarHandler

- \* ColumnListHandler

- \* 能够掌握DBUtils的框架

- \* apache

- \* QueryRunner

- \* update

- \* query

- \* ResultSetHandler

- \* batch

- \* 能够阅读DBUtils的核心源码

- \* QueryRunner--update

- \* 健壮性比我好，处理细节， ....

- \* query

- \* batch

---

--

- \* 回顾

- \* 反射：在运行时，可以动态创建对象，调用方法，给属性设置值， ....

- \* Class,Constructor,Method,Field

- \* 内省：JavaBean：set，get

- \* Instrospector,BeanInfo,PropertyDescripto

- \* BeanUtils.populate(obj,map);
- \* 数据库的元数据
  - \* DatabaseMetaData
  - \* ParameterMetaData-->PreparedStatement(?,?,?)
  - \* ResultSetMetaData-->ResultSet
- \* JDBCUtils
  - \* CUD
    - \* 不同点：参数不一样，sql不一样
    - \* 共同点：获取链接，通过sql获得PreparedStatement,executeUpdate,处理异常，释放资源
  - \* 能够自定义JDBC的框架

```

1 * Student
2 public class Student {
3     private String sid;
4     private String sname;
5     private int age;
6     private String gender;
7 }
8 * 观察StudentDaoImpl
9 public class StudentDaoImpl implements StudentDao {
10
11     @Override
12     public boolean add(Student student) {
13         if (student == null) {
14             throw new IllegalArgumentException("student is null");
15         }
16         Connection con = ConnectionUtils.getConnection();
17         PreparedStatement st = null;
18         int result = 0;
19         try {
20             String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)";
21             st = con.prepareStatement(sql);
22             st.setString(1, student.getSid());
23             st.setString(2, student.getSname());

```

```

24         st.setInt(3, student.getAge());
25         st.setString(4, student.getGender());
26         result = st.executeUpdate();
27     } catch (SQLException e) {
28         e.printStackTrace();
29     } finally {
30         ConnectionUtils.close(con, st, null);
31     }
32     return result > 0;
33 }
34
35 @Override
36 public boolean update(String sid, String name) {
37     Connection con = ConnectionUtils.getConn();
38     PreparedStatement st = null;
39     int result = 0;
40     try {
41         String sql = "update stu set sname=? where sid=?";
42         st = con.prepareStatement(sql);
43         st.setString(1, name);
44         st.setString(2, sid);
45         result = st.executeUpdate();
46     } catch (SQLException e) {
47         e.printStackTrace();
48     } finally {
49         ConnectionUtils.close(con, st, null);
50     }
51     return result > 0;
52 }
53
54 @Override
55 public boolean delete(String sid) {
56     Connection con = ConnectionUtils.getConn();
57     PreparedStatement st = null;
58     int result = 0;
59     try {
60         String sql = "delete from stu where sid=?";
61         st = con.prepareStatement(sql);
62         st.setString(1, sid);
63         result = st.executeUpdate();

```

```

64         } catch (SQLException e) {
65             e.printStackTrace();
66         } finally {
67             ConnectionUtils.close(con, st, null);
68         }
69         return result > 0;
70     }
71
72     @Override
73     public Student queryById(String id) {
74         String sql = "select sid,sname,age,gender from stu where sid=?";
75         Connection con = ConnectionUtils.getConn();
76         PreparedStatement st = null;
77         ResultSet rs = null;
78         Student student = null;
79         try {
80             st = con.prepareStatement(sql);
81             st.setString(1, id);
82             rs = st.executeQuery();
83             if (rs.next()) {
84                 String sid = rs.getString("sid");
85                 String sname = rs.getString("sname");
86                 String gender = rs.getString("gender");
87                 try {
88                     int age = Integer.parseInt(rs.getString("age"));
89                     student = new Student(sid, sname, age, gender);
90                 } catch (NumberFormatException e) {
91                     student = new Student(sid, sname, 0, gender);
92                 }
93             }
94         } catch (SQLException e) {
95             e.printStackTrace();
96         } finally {
97             ConnectionUtils.close(con, st, rs);
98         }
99
100         return student;
101     }
102
103     @Override

```

```

104 public List<Student> queryAllStudents() {
105     String sql = "select sid,sname,age,gender from stu ";
106     Connection con = ConnectionUtils.getConnection();
107     PreparedStatement st = null;
108     ResultSet rs = null;
109     List<Student> students = new ArrayList<Student>();
110     try {
111         st = con.prepareStatement(sql);
112         rs = st.executeQuery();
113         while (rs.next()) {
114             String sid = rs.getString("sid");
115             String sname = rs.getString("sname");
116             String gender = rs.getString("gender");
117             try {
118                 int age = Integer.parseInt(rs.getString("age"));
119                 students.add(new Student(sid, sname, age, gender));
120             } catch (NumberFormatException e) {
121                 students.add(new Student(sid, sname, 0, gender));
122             }
123         }
124     } catch (SQLException e) {
125         e.printStackTrace();
126     } finally {
127         ConnectionUtils.close(con, st, rs);
128     }
129     return students;
130 }
131
132 // offset: row:多少行
133 @Override
134 public List<Student> queryStudents(int offset, int row) {
135     String sql = "select sid,sname,age,gender from (select rownum rn,sid,sr";
136     Connection con = ConnectionUtils.getConnection();
137     PreparedStatement st = null;
138     ResultSet rs = null;
139     List<Student> students = new ArrayList<Student>();
140     try {
141         st = con.prepareStatement(sql);
142         st.setInt(1, row);
143         st.setInt(2, offset);

```

```

144         rs = st.executeQuery();
145         while (rs.next()) {
146             String sid = rs.getString("sid");
147             String sname = rs.getString("sname");
148             String gender = rs.getString("gender");
149             try {
150                 int age = Integer.parseInt(rs.getString("age"));
151                 students.add(new Student(sid, sname, age, gender));
152             } catch (NumberFormatException e) {
153                 students.add(new Student(sid, sname, 0, gender));
154             }
155         }
156     } catch (SQLException e) {
157         e.printStackTrace();
158     } finally {
159         ConnectionUtils.close(con, st, rs);
160     }
161     return students;
162 }

```

164 }

165

166 \* 总结增删改

167 \* 一样地方获取连接, 释放资源, 获得PreparedStatement, 执行executeUpdate

168 \* 不一样的地方, sql语句不一样, 设置参数也不一样

169 \* 总结查询单个

170 \* 一样地方获取连接, 释放资源, 获得PreparedStatement, executeQuery

171 \* 不一样的地方, sql语句不一样, 设置参数也不一样, 返回值不一样

172 \* 总结查询列表

173 \* 一样地方获取连接, 释放资源, 获得PreparedStatement, executeQuery

174 \* 不一样的地方, sql语句不一样, 设置参数也不一样, 返回值列表不一样

175

176 \* JdbcUtils的编写

177 public class JdbcUtils {

178

179 /\*\*

180 \* @param sql

181 \* @param args

182 \* @return

183 \* CUD

```

184     */
185     public static boolean update(String sql, Object... args) {
186         Connection conn = ConnectionUtils.getConnection();
187         PreparedStatement psmt = null;
188         int result = -1;
189         try {
190             psmt = conn.prepareStatement(sql);
191             int paramCount = psmt.getParameterMetaData().getParameterCount();
192             if (paramCount != args.length) {
193                 throw new IllegalArgumentException("expected is " + paramCount + ", " + args.length);
194             }
195             for (int i = 0; i < args.length; i++) {
196                 psmt.setObject(i + 1, args[i]);
197             }
198             result = psmt.executeUpdate();
199         } catch (SQLException e) {
200             e.printStackTrace();
201         } finally {
202             ConnectionUtils.close(conn, psmt, null);
203         }
204         return result > 0;
205     }
206
207     public static Object query(String sql, ResultSetHandler handler, Object... args) {
208         Connection conn = ConnectionUtils.getConnection();
209         PreparedStatement psmt = null;
210         ResultSet rs = null;
211         try {
212             psmt = conn.prepareStatement(sql);
213             int paramCount = psmt.getParameterMetaData().getParameterCount();
214             if (paramCount != args.length) {
215                 throw new IllegalArgumentException("expected is " + paramCount + ", " + args.length);
216             }
217             for (int i = 0; i < args.length; i++) {
218                 psmt.setObject(i + 1, args[i]);
219             }
220             rs = psmt.executeQuery();
221             // 如何处理这个结果：有可能是Bean, List<Bean>, Map, List<Map>, 1个值(construct)
222             // 不能在处理这些结果，而是交给用户自己决定，
223             // 框架决定不了，此时可以定接口，让调用者实现

```



```

224         if(handler==null) {
225             throw new IllegalArgumentException("handler is null");
226         }
227         Object result=handler.handle(rs);
228         return result;
229     } catch (SQLException e) {
230         e.printStackTrace();
231     }finally {
232         ConnectionUtils.close(conn, psmt, rs);
233     }
234     return null;
235 }
236 }
237 * 定义ResultSetHandler
238 public interface ResultSetHandler {
239     Object handle(ResultSet rs);
240 }
241 * BeanHandler编写
242 public class BeanHandler implements ResultSetHandler {
243     private Class<?> clazz;
244
245     public BeanHandler(Class<?> clazz) {
246         this.clazz = clazz;
247     }
248
249     @Override
250     public Object handle(ResultSet rs) {
251         Object obj = null;
252         try {
253             if (rs.next()) {
254                 obj = this.clazz.newInstance();
255                 Map<String, Object> properties = new HashMap<String, Object>();
256                 ResultSetMetaData metaData = rs.getMetaData();
257                 int columnCount = metaData.getColumnCount();
258                 for (int i = 0; i < columnCount; i++) {
259                     String columnName = metaData.getColumnName(i + 1);
260                     Object param = rs.getObject(i + 1);
261                     properties.put(columnName.toLowerCase(), param);
262                 }
263                 BeanUtils.populate(obj, properties);

```

```

264     }
265     } catch (Exception e) {
266         e.printStackTrace();
267     }
268     return obj;
269 }
270
271 }
272 * ListBeanHandler 编写
273 public class ListBeanHandler implements ResultSetHandler {
274     private Class<?> clazz;
275
276     public ListBeanHandler(Class<?> clazz) {
277         this.clazz = clazz;
278     }
279
280     @Override
281     public Object handle(ResultSet rs) {
282         List<Object> objs = new ArrayList<Object>();
283         try {
284             while (rs.next()) {
285                 Object obj = this.clazz.newInstance();
286                 Map<String, Object> properties = new HashMap<String, Object>();
287                 ResultSetMetaData metaData = rs.getMetaData();
288                 int columnCount = metaData.getColumnCount();
289                 for (int i = 0; i < columnCount; i++) {
290                     String columnName = metaData洗getColumnName(i + 1);
291                     Object param = rs.getObject(i + 1);
292                     properties.put(columnName.toLowerCase(), param);
293                 }
294                 BeanUtils.populate(obj, properties);
295                 objs.add(obj);
296             }
297         } catch (Exception e) {
298             e.printStackTrace();
299         }
300         return objs;
301     }
302 }
303

```

```

304 * copy StudentDaoImpl修改成StudentDaoImpl2
305 public class StudentDaoImpl2 implements StudentDao {
306
307     @Override
308     public boolean add(Student student) {
309         if (student == null) {
310             throw new IllegalArgumentException("student is null");
311         }
312         String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)";
313         Object[] args = { student.getSid(), student.getSname(), student.getAge(), student.getGender() };
314         boolean result = JdbcUtils.update(sql, args);
315         return result;
316     }
317
318     @Override
319     public boolean update(String sid, String name) {
320         String sql = "update stu set sname=? where sid=?";
321         Object[] args = { name, sid };
322         boolean result = JdbcUtils.update(sql, args);
323         return result;
324     }
325
326     @Override
327     public boolean delete(String sid) {
328         String sql = "delete from stu where sid=?";
329         Object[] args = { sid };
330         boolean result = JdbcUtils.update(sql, args);
331         return result;
332     }
333
334     @Override
335     public Student queryById(String id) {
336         String sql = "select sid,sname,age,gender from stu where sid=?";
337         Object[] args = { id };
338         Student student = (Student) JdbcUtils.query(sql, new BeanHandler<Student>(Student.class));
339         return student;
340     }
341
342     @Override
343     public List<Student> queryAllStudents() {

```

```

344     String sql = "select sid,sname,age,gender from stu ";
345     @SuppressWarnings("unchecked")
346     List<Student> students=(List<Student>) JdbcUtils.query(sql, new ListBea
347     return students;
348 }
349
350 // offset: row:多少行
351 @Override
352 public List<Student> queryStudents(int offset, int row) {
353     String sql = "select sid,sname,age,gender from (select rownum rn,sid,sr
354     Object[] args = {row,offset };
355     @SuppressWarnings("unchecked")
356     List<Student> students=(List<Student>) JdbcUtils.query(sql, new ListBea
357     return students;
358 }
359 }
360
361 * 测试
362 * studentDao=new StudentDaoImpl2();
363

```

## \* 能够掌握DBUtils的框架

### \* DBUtils的概述

- \* DBUtils库是一个小类集，目标是使jdbc更容易使用

### \* DBUtils的优点

- \* 不可能发生资源泄漏
- \* 更干净，更清晰的持久性代码
- \* 从结果集自动填充JavaBean属性

### \* 下载jar和源码

- \* <https://commons.apache.org/proper/commons-dbutils/>



**Apache Commons**<sup>TM</sup>  
<http://commons.apache.org/>

Apache Commons DbUtils<sup>TM</sup>

**DBUTILS**

[Overview](#)

[Download](#)

[Examples](#)

[Release Notes](#)

[Dependencies](#)

[Wiki](#)



**DEVELOPMENT**

**Do**

**Us**

We re  
direct

You ar  
(at the

# Apache Commons DbUtils 1.7

## Binaries

<a href="#">commons-dbutils-1.7-bin.tar.gz</a>	<a href="#">md5</a>	<a href="#">pgp</a>
<a href="#">commons-dbutils-1.7-bin.zip</a>	<a href="#">md5</a>	<a href="#">pgp</a>

jar, 源码jar

## Source

<a href="#">commons-dbutils-1.7-src.tar.gz</a>	<a href="#">md5</a>	<a href="#">pgp</a>
<a href="#">commons-dbutils-1.7-src.zip</a>	<a href="#">md5</a>	<a href="#">pgp</a>

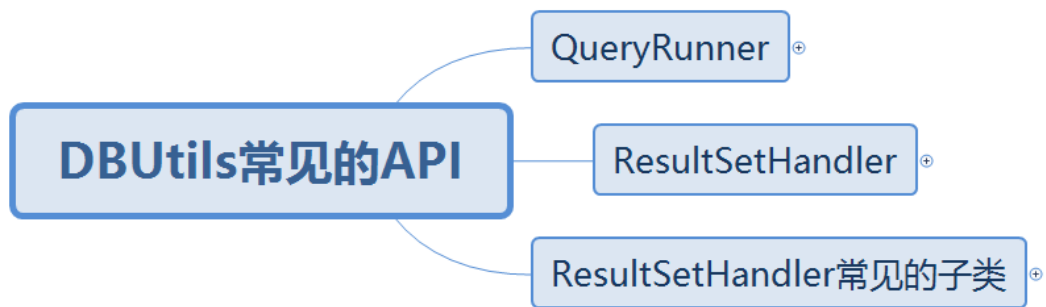
源码

## Archives

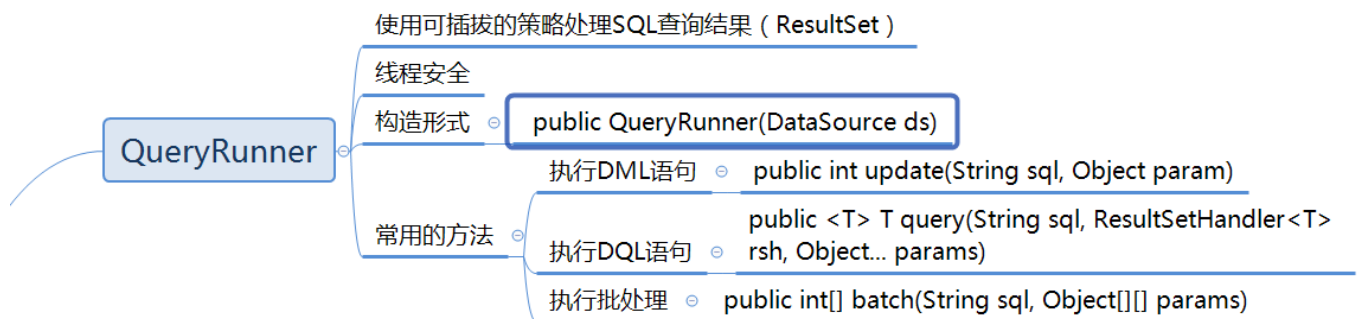
Older releases can be obtained from the archives.

- [browse download area](#)
- [archives...](#)

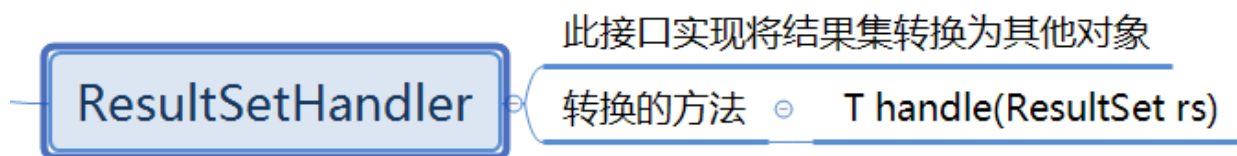
\* DbUtils常见的API



### \* QueryRunner

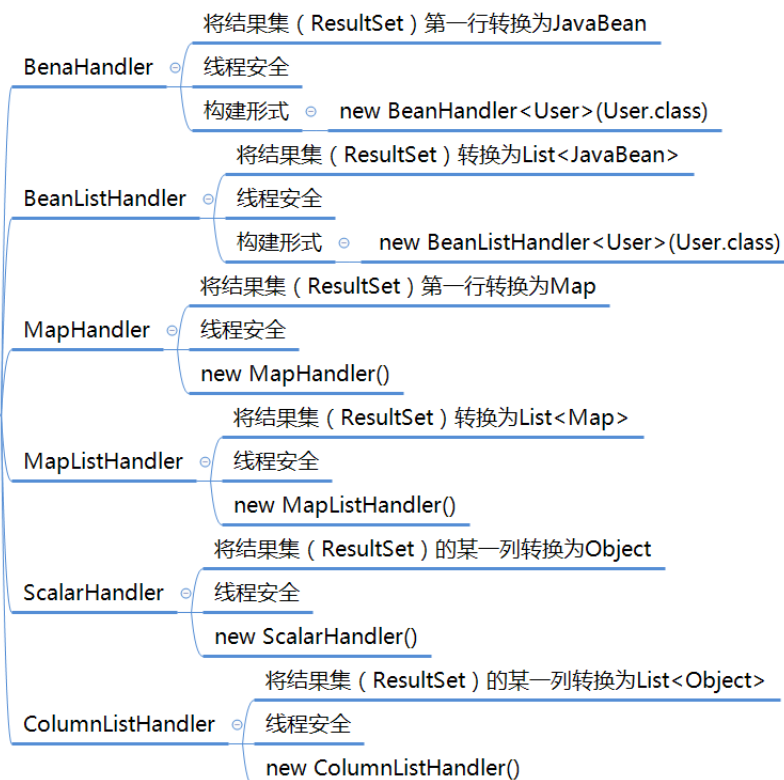


### \* ResultSetHandler



### \* ResultSetHandler常见的子类

## ResultSetHandler常见的子类



```

1 * StudentDaoImpl 使用DBUtils
2 * StudentDaoImpl3
3 public class StudentDaoImpl3 implements StudentDao {
4
5     @Override
6     public boolean add(Student student) {
7         if (student == null) {
8             throw new IllegalArgumentException("student is null");
9         }
10        String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)";
11        Object[] args = { student.getSid(), student.getSname(), student.getAge(), student.getGender() };
12        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
13        int result=-1;
14        try {
15            result = queryRunner.update(sql, args);
16        } catch (SQLException e) {
17            e.printStackTrace();
18        }
19        return result>0;
  
```

```

20     }
21
22     @Override
23     public boolean update(String sid, String name) {
24         String sql = "update stu set sname=? where sid=?";
25         Object[] args = { name, sid };
26         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
27         int result=-1;
28         try {
29             result = queryRunner.update(sql, args);
30         } catch (SQLException e) {
31             e.printStackTrace();
32         }
33         return result>0;
34     }
35
36     @Override
37     public boolean delete(String sid) {
38         String sql = "delete from stu where sid=?";
39         Object[] args = { sid };
40         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
41         int result=-1;
42         try {
43             result = queryRunner.update(sql, args);
44         } catch (SQLException e) {
45             e.printStackTrace();
46         }
47         return result>0;
48     }
49
50     @Override
51     public Student queryById(String id) {
52         String sql = "select sid,sname,age,gender from stu where sid=?";
53         Object[] args = { id };
54         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
55         Student student=null;
56         try {
57             student = queryRunner.query(sql, new BeanHandler<Student>(Student.class));
58         } catch (SQLException e) {
59             e.printStackTrace();

```



```

60     }
61     return student;
62 }
63
64 @Override
65 public List<Student> queryAllStudents() {
66     String sql = "select sid,sname,age,gender from stu ";
67     QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
68     List<Student> students=null;
69     try {
70         students = queryRunner.query(sql, new BeanListHandler<Student>(Student.class));
71     } catch (SQLException e) {
72         e.printStackTrace();
73     }
74     return students;
75 }
76
77 // offset: row:多少行
78 @Override
79 public List<Student> queryStudents(int offset, int row) {
80     String sql = "select sid,sname,age,gender from (select rownum rn,sid,sname,age,gender from stu)";
81     Object[] args = {row,offset };
82     QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
83     List<Student> students=null;
84     try {
85         students = queryRunner.query(sql, new BeanListHandler<Student>(Student.class), args);
86     } catch (SQLException e) {
87         e.printStackTrace();
88     }
89     return students;
90 }
91
92 }
93
94 * DataSourceUtil
95 public class DataSourceUtil {
96     private static DataSource dataSource;
97     static {
98         Properties props=new Properties();
99         try {

```

```

100         props.load(ConnectionUtils.class.getClassLoader().getResourceAsStream(
101             "druid.properties"));
102         dataSource=DruidDataSourceFactory.createDataSource(props);
103     } catch (IOException e) {
104         e.printStackTrace();
105     } catch (Exception e) {
106         e.printStackTrace();
107     }
108 }
109
110 public static DataSource getDataSource() {
111     return dataSource;
112 }
113 }
114
115 * 测试
116     studentDao=new StudentDaoImpl3();
117
118 * 测试其他子类
119     * MapHandler, MapListHandler, ScalarHandler, ColumnListHandler
120 @Test
121     public void test1() throws SQLException {
122         String sql = "select sid,sname,age,gender from stu where sid=?";
123         Object[] params = { "S_1006" };
124         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
125         Map<String, Object> map = queryRunner.query(sql, new MapHandler(), params);
126         System.out.println(map);
127     }
128
129 @Test
130     public void test2() throws SQLException {
131         String sql = "select sid,sname,age,gender from stu";
132         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
133         List<Map<String, Object>> maps = queryRunner.query(sql, new MapListHandler());
134         for(Map<String, Object> map:maps) {
135             System.out.println(map);
136         }
137     }
138
139 @Test

```

```

140     public void test3() throws SQLException {
141         String sql = "select count(*) from stu";
142         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
143         BigDecimal bd = queryRunner.query(sql, new ScalarHandler<BigDecimal>());
144         System.out.println("count:"+bd.intValue());
145     }
146
147     @Test
148     public void test4() throws SQLException {
149         String sql = "select sname from stu";
150         QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
151         List<String> names = queryRunner.query(sql, new ColumnListHandler<String>());
152         System.out.println(names);
153     }
154
155
156 * 测试批处理
157 create table testbatch(
158     name varchar2(50)
159 );
160 @Test
161 public void test5() throws SQLException {
162     long start = System.currentTimeMillis();
163     // 1 获得连接
164     String sql = "insert into testbatch(name) values(?)";
165     QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
166     Object[][] params=new Object[1000][];
167     for (int i = 1; i <= 10000; i++) {
168         params[i % 1000]=new Object[] {"xiaobai" + i};
169         // 每次发一千条
170         if (i % 1000 == 0) {
171             queryRunner.batch(sql, params);
172         }
173     }
174     // 剩余再执行一次
175     params=new Object[8][];
176     for (int i = 10001; i <= 10008; i++) {
177         params[i % 8]=new Object[] {"xiaobai" + i};
178     }
179     queryRunner.batch(sql, params);

```

```
180         long end = System.currentTimeMillis();
181         //使用批处理花的时间:1079
182         System.out.println("使用批处理花的时间:" + (end - start));
183     }
184
```

\* 能够阅读DBUtils的核心源码

\* 参考[09-DBUtils源码分析](#)