```
* 学习目标
 * 能够自定义JDBC的框架
   * JDBCUtils.update
      * sql，params
      * stmt.getParamterMetaData()-->getParamterCount();
      * stmt.setOject(i+1,params[i]);
   * JDBCUtils.query
      * sql,params,ResultSetHandler
      * rs---
>obj(Bean,List<Bean>,Map,MapList,ScalarHandler,ColunmListHander,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).tolowercace();
                     * ResultSet rs.next(),.rs.getObject(i+1);
            * 反射基础代码，内省，BeanUtils
     * MapHandler
         * map
             * ResultSetMetadata
```

    * getColumnCount
      * getColumnName(i+1).tolowercace();
      * 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-->PrepareStatement(?,?,?)

  * 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PrepareStatement,executeUpdate,处理异常，释放资源

* 能够自定义JDBC的框架

```
* Student
  public class Student {
    private String sid;
    private String sname;
    private int age;
    private String gender;
 }
* 观察StudentDaoImpl
public class StudentDaoImpl implements StudentDao {

    @Override
    public boolean add(Student student) {
        if (student == null) {
            throw new IllegalArgumentException("student is null");
        }
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        int result = 0;
        try {
            String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)
            st = con.prepareStatement(sql);
            st.setString(1, student.getSid());
            st.setString(2, student.getSname());
```

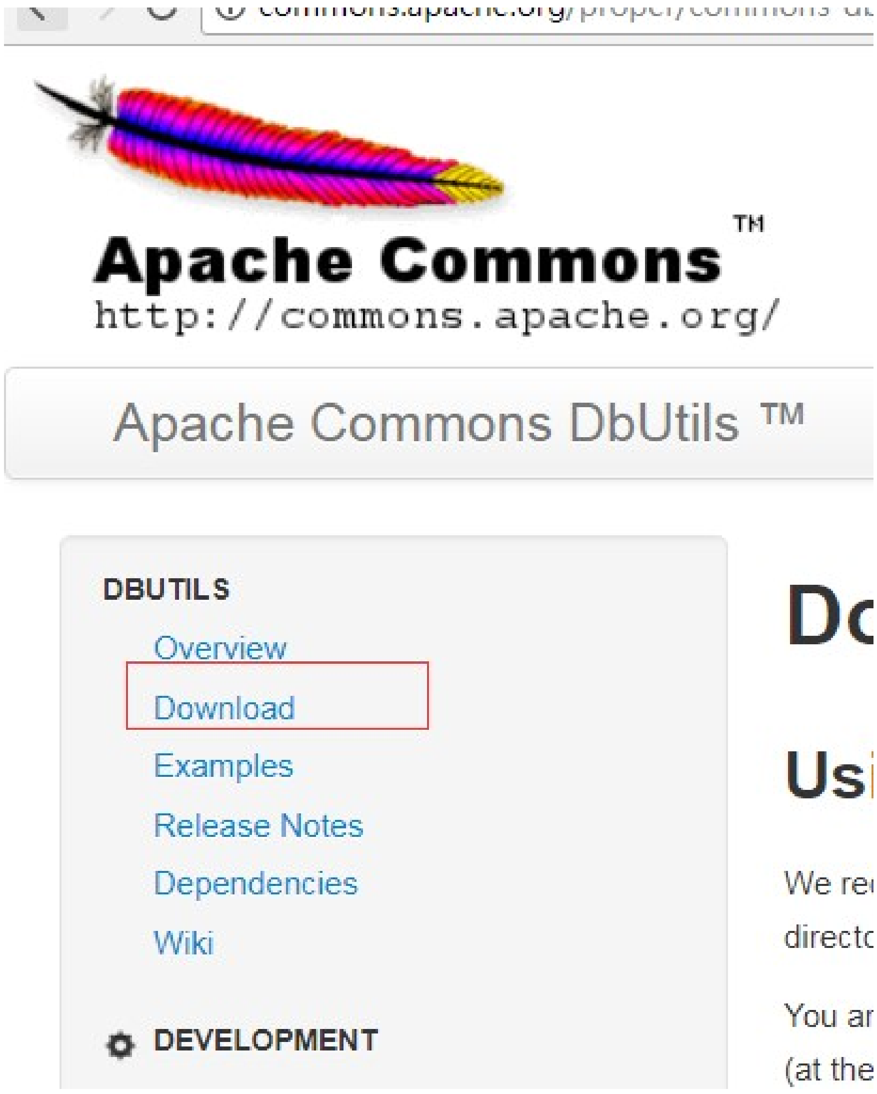
```java
            st.setInt(3, student.getAge());
            st.setString(4, student.getGender());
            result = st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, null);
        }
        return result > 0;
    }

    @Override
    public boolean update(String sid, String name) {
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        int result = 0;
        try {
            String sql = "update stu set sname=? where sid=?";
            st = con.prepareStatement(sql);
            st.setString(1, name);
            st.setString(2, sid);
            result = st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, null);
        }
        return result > 0;
    }

    @Override
    public boolean delete(String sid) {
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        int result = 0;
        try {
            String sql = "delete from stu where sid=?";
            st = con.prepareStatement(sql);
            st.setString(1, sid);
            result = st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, null);
        }
        return result > 0;
    }

    @Override
    public Student queryById(String id) {
        String sql = "select sid,sname,age,gender from stu where sid=?";
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        ResultSet rs = null;
        Student student = null;
        try {
            st = con.prepareStatement(sql);
            st.setString(1, id);
            rs = st.executeQuery();
            if (rs.next()) {
                String sid = rs.getString("sid");
                String sname = rs.getString("sname");
                String gender = rs.getString("gender");
                try {
                    int age = Integer.parseInt(rs.getString("age"));
                    student = new Student(sid, sname, age, gender);
                } catch (NumberFormatException e) {
                    student = new Student(sid, sname, 0, gender);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, rs);
        }

        return student;
    }

    @Override
    public List<Student> queryAllStudents() {
        String sql = "select sid,sname,age,gender from stu ";
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        ResultSet rs = null;
        List<Student> students = new ArrayList<Student>();
        try {
            st = con.prepareStatement(sql);
            rs = st.executeQuery();
            while (rs.next()) {
                String sid = rs.getString("sid");
                String sname = rs.getString("sname");
                String gender = rs.getString("gender");
                try {
                    int age = Integer.parseInt(rs.getString("age"));
                    students.add(new Student(sid, sname, age, gender));
                } catch (NumberFormatException e) {
                    students.add(new Student(sid, sname, 0, gender));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, rs);
        }
        return students;
    }

    // offset: row:多少行
    @Override
    public List<Student> queryStudents(int offset, int row) {
        String sql = "select sid,sname,age,gender from (select rownum rn,sid,sr
        Connection con = ConnectionUtils.getConn();
        PreparedStatement st = null;
        ResultSet rs = null;
        List<Student> students = new ArrayList<Student>();
        try {
            st = con.prepareStatement(sql);
            st.setInt(1, row);
            st.setInt(2, offset);
            rs = st.executeQuery();
            while (rs.next()) {
                String sid = rs.getString("sid");
                String sname = rs.getString("sname");
                String gender = rs.getString("gender");
                try {
                    int age = Integer.parseInt(rs.getString("age"));
                    students.add(new Student(sid, sname, age, gender));
                } catch (NumberFormatException e) {
                    students.add(new Student(sid, sname, 0, gender));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(con, st, rs);
        }
        return students;
    }

}

* 总结增删改
   * 一样地方获取连接，释放资源，获得PreparedStatement，执行executeUpdate
   * 不一样的地方，sql语句不一样，设置参数也不一样
* 总结查询单个
   * 一样地方获取连接，释放资源，获得PreparedStatement，executeQuery
   * 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
* 总结查询列表
    * 一样地方获取连接，释放资源，获得PreparedStatement，executeQuery
    * 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

* JdbcUtils的编写
public class JdbcUtils {

    /**
     * @param sql
     * @param args
     * @return
     * CUD
```

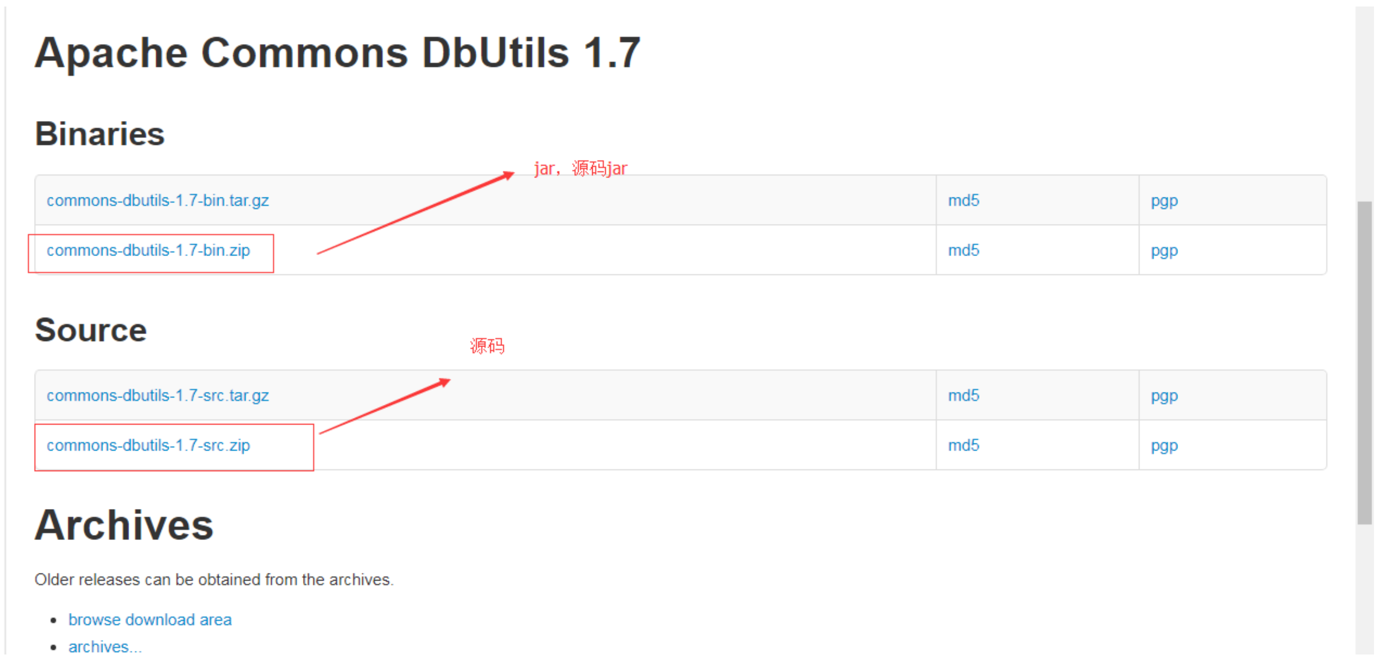
```java
     */
    public static boolean update(String sql,Object... args) {
        Connection conn=ConnectionUtils.getConn();
        PreparedStatement psmt=null;
        int result=-1;
        try {
            psmt=conn.prepareStatement(sql);
            int paramCount=psmt.getParameterMetaData().getParameterCount();
            if(paramCount!=args.length) {
                throw new IllegalArgumentException("expected is "+paramCount+",
            }
            for (int i = 0; i < args.length; i++) {
                psmt.setObject(i+1, args[i]);
            }
            result=psmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            ConnectionUtils.close(conn, psmt, null);
        }
        return result>0;
    }

    public static Object query(String sql,ResultSetHandler handler, Object... a
        Connection conn=ConnectionUtils.getConn();
        PreparedStatement psmt=null;
        ResultSet rs=null;
        try {
            psmt=conn.prepareStatement(sql);
            int paramCount=psmt.getParameterMetaData().getParameterCount();
            if(paramCount!=args.length) {
                throw new IllegalArgumentException("expected is "+paramCount+",
            }
            for (int i = 0; i < args.length; i++) {
                psmt.setObject(i+1, args[i]);
            }
            rs = psmt.executeQuery();
            // 如何处理这个结果：有可能是Bean，List<Bean>,Map,List<Map>,1个值(cour
            // 不能在处理这些结果，而是交给用户自己决定，
            // 框架决定不了，此时可以定接口，让调用者实现
            if(handler==null) {
                throw new IllegalArgumentException("handler is null");
            }
            Object result=handler.handle(rs);
            return result;
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            ConnectionUtils.close(conn, psmt, rs);
        }
        return null;
    }
}
* 定义ResultSetHandler
public interface ResultSetHandler {
    Object handle(ResultSet rs);
}
* BeanHandler编写
public class BeanHandler implements ResultSetHandler {
    private Class<?> clazz;

    public BeanHandler(Class<?> clazz) {
        this.clazz = clazz;
    }

    @Override
    public Object handle(ResultSet rs) {
        Object obj = null;
        try {
            if (rs.next()) {
                obj = this.clazz.newInstance();
                Map<String, Object> properties = new HashMap<String, Object>();
                ResultSetMetaData metaData = rs.getMetaData();
                int columnCount = metaData.getColumnCount();
                for (int i = 0; i < columnCount; i++) {
                    String columnName = metaData.getColumnName(i + 1);
                    Object param = rs.getObject(i + 1);
                    properties.put(columnName.toLowerCase(), param);
                }
                BeanUtils.populate(obj, properties);
```

```java
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return obj;
    }

}
* ListBeanHandler 编写
public class ListBeanHandler implements ResultSetHandler {
    private Class<?> clazz;

    public ListBeanHandler(Class<?> clazz) {
        this.clazz = clazz;
    }

    @Override
    public Object handle(ResultSet rs) {
        List<Object> objs = new ArrayList<Object>();
        try {
            while (rs.next()) {
                Object obj = this.clazz.newInstance();
                Map<String, Object> properties = new HashMap<String, Object>();
                ResultSetMetaData metaData = rs.getMetaData();
                int columnCount = metaData.getColumnCount();
                for (int i = 0; i < columnCount; i++) {
                    String columnName = metaData.getColumnName(i + 1);
                    Object param = rs.getObject(i + 1);
                    properties.put(columnName.toLowerCase(), param);
                }
                BeanUtils.populate(obj, properties);
                objs.add(obj);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return objs;
    }
}

```

```java
* copy StudentDaoImpl修改成StudentDaoImpl2
public class StudentDaoImpl2 implements StudentDao {

    @Override
    public boolean add(Student student) {
        if (student == null) {
            throw new IllegalArgumentException("student is null");
        }
        String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)";
        Object[] args = { student.getSid(), student.getSname(), student.getAge(
        boolean result = JdbcUtils.update(sql, args);
        return result;
    }

    @Override
    public boolean update(String sid, String name) {
        String sql = "update stu set sname=? where sid=?";
        Object[] args = { name, sid };
        boolean result = JdbcUtils.update(sql, args);
        return result;
    }

    @Override
    public boolean delete(String sid) {
        String sql = "delete from stu where sid=?";
        Object[] args = { sid };
        boolean result = JdbcUtils.update(sql, args);
        return result;
    }

    @Override
    public Student queryById(String id) {
        String sql = "select sid,sname,age,gender from stu where sid=?";
        Object[] args = { id };
        Student student=(Student) JdbcUtils.query(sql, new BeanHandler(Student.
        return student;
    }

    @Override
    public List<Student> queryAllStudents() {
        String sql = "select sid,sname,age,gender from stu ";
        @SuppressWarnings("unchecked")
        List<Student> students=(List<Student>) JdbcUtils.query(sql, new ListBea
        return students;
    }

    // offset: row:多少行
    @Override
    public List<Student> queryStudents(int offset, int row) {
        String sql = "select sid,sname,age,gender from (select rownum rn,sid,sn
        Object[] args = {row,offset };
        @SuppressWarnings("unchecked")
        List<Student> students=(List<Student>) JdbcUtils.query(sql, new ListBea
        return students;
    }
}

* 测试
  * studentDao=new StudentDaoImpl2();

```

* 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 DbUtils 1.7

Binaries

commons-dbutils-1.7-bin.tar.gz	md5	pgp
commons-dbutils-1.7-bin.zip	md5	pgp

Source

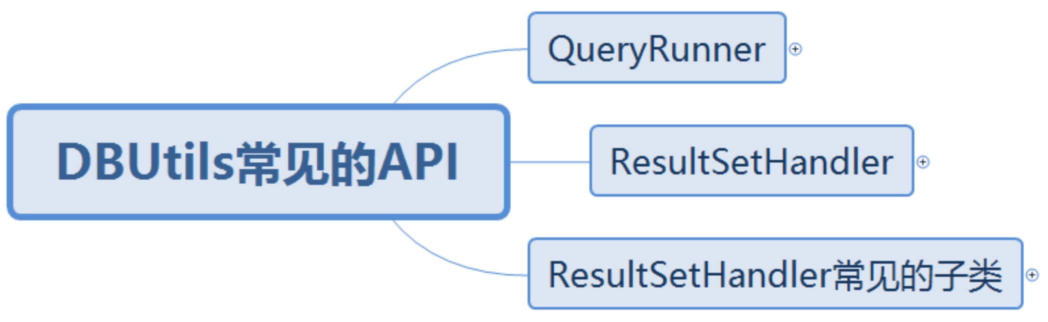
commons-dbutils-1.7-src.tar.gz	md5	pgp
commons-dbutils-1.7-src.zip	md5	pgp

Archives

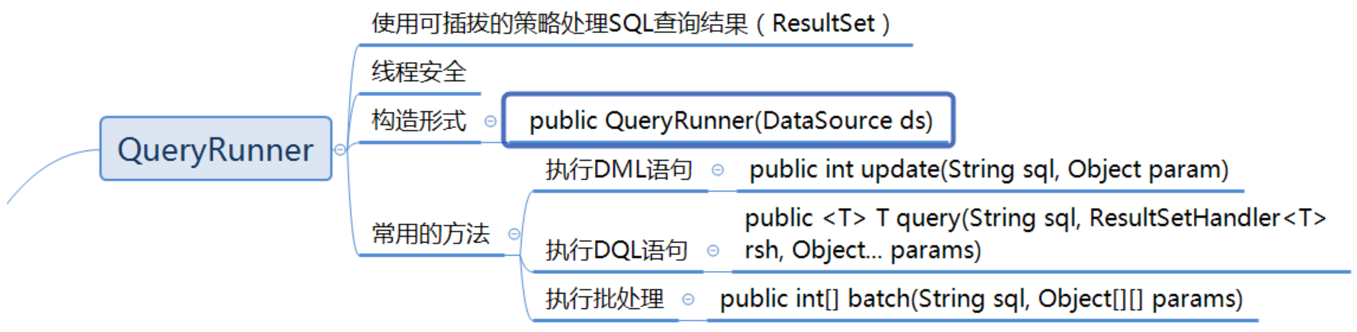
Older releases can be obtained from the archives.

- browse download area
- archives...

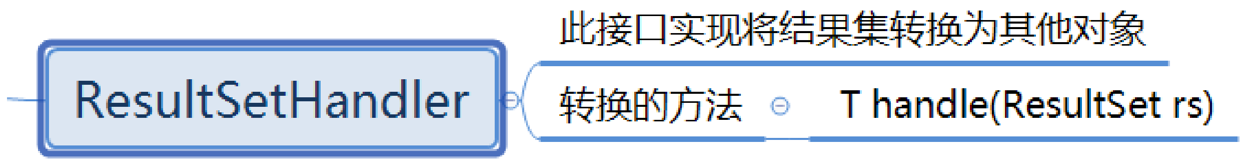
* DBUtils常见的API



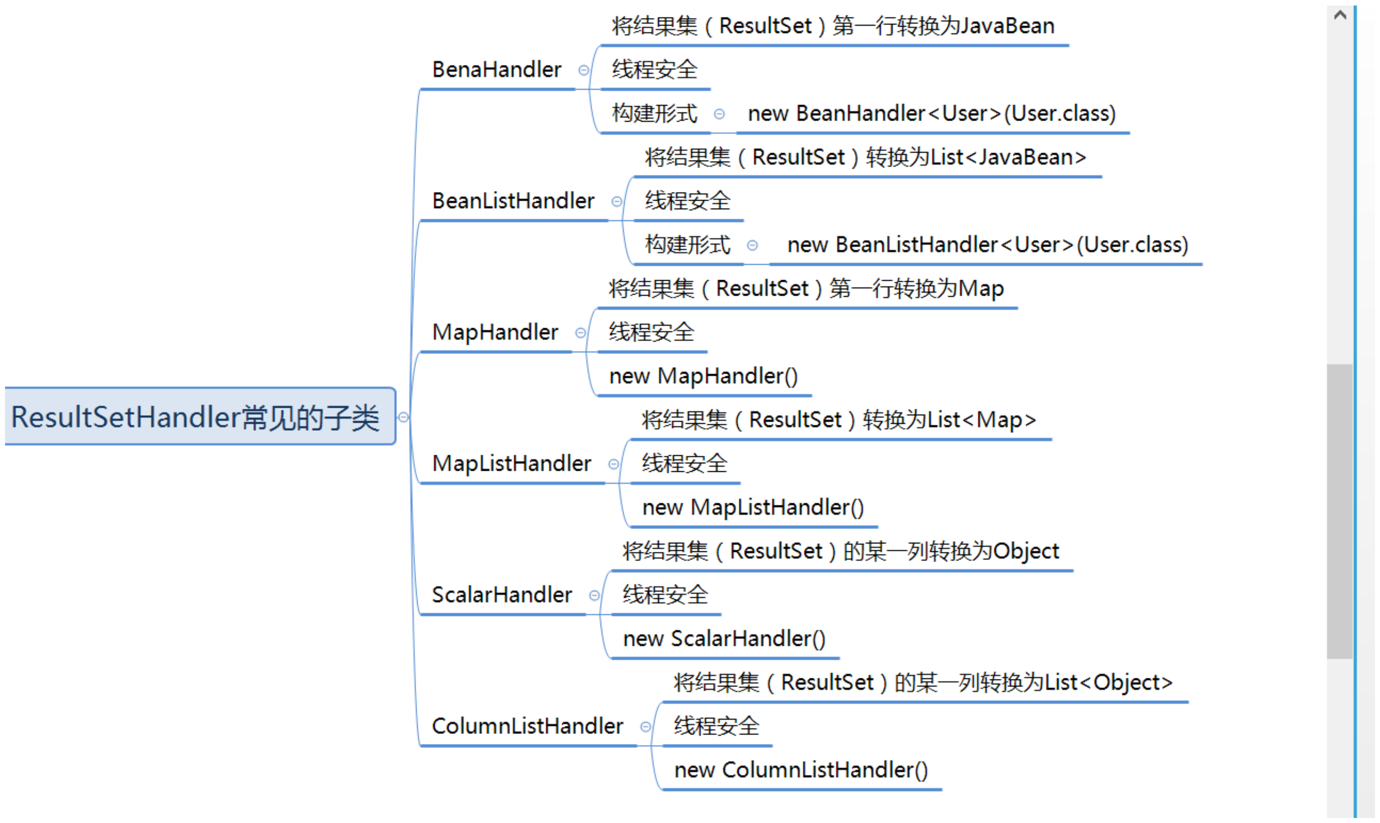
* QueryRunner


QueryRunner
使用可插拔的策略处理SQL查询结果（ResultSet）
线程安全
构造形式
public QueryRunner(DataSource ds)
常用的方法
执行DML语句
public int update(String sql, Object param)
执行DQL语句
public <T> T query(String sql, ResultSetHandler<T> rsh, Object... params)
执行批处理
public int[] batch(String sql, Object[][] params)

* ResultSetHandler



* ResultSetHandler常见的子类


ResultSetHandler常见的子类
BenaHandler
将结果集（ResultSet）第一行转换为JavaBean
线程安全
构建形式
new BeanHandler<User>(User.class)
BeanListHandler
将结果集（ResultSet）转换为List<JavaBean>
线程安全
构建形式
new BeanListHandler<User>(User.class)
MapHandler
将结果集（ResultSet）第一行转换为Map
线程安全
new MapHandler()
MapListHandler
将结果集（ResultSet）转换为List<Map>
线程安全
new MapListHandler()
ScalarHandler
将结果集（ResultSet）的某一列转换为Object
线程安全
new ScalarHandler()
ColumnListHandler
将结果集（ResultSet）的某一列转换为List<Object>
线程安全
new ColumnListHandler()

```java
* StudentDaoImpl 使用DBUtils
 * StudentDaoImpl3
public class StudentDaoImpl3 implements StudentDao {

    @Override
    public boolean add(Student student) {
        if (student == null) {
            throw new IllegalArgumentException("student is null");
        }
        String sql = "insert into stu(sid,sname,age,gender) values(?,?,?,?)";
        Object[] args = { student.getSid(), student.getSname(), student.getAge(
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        int result=-1;
        try {
            result = queryRunner.update(sql, args);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result>0;
    }

    @Override
    public boolean update(String sid, String name) {
        String sql = "update stu set sname=? where sid=?";
        Object[] args = { name, sid };
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        int result=-1;
        try {
            result = queryRunner.update(sql, args);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result>0;
    }

    @Override
    public boolean delete(String sid) {
        String sql = "delete from stu where sid=?";
        Object[] args = { sid };
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        int result=-1;
        try {
            result = queryRunner.update(sql, args);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result>0;
    }

    @Override
    public Student queryById(String id) {
        String sql = "select sid,sname,age,gender from stu where sid=?";
        Object[] args = { id };
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        Student student=null;
        try {
            student = queryRunner.query(sql, new BeanHandler<Student>(Student.
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return student;
    }

    @Override
    public List<Student> queryAllStudents() {
        String sql = "select sid,sname,age,gender from stu ";
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        List<Student> students=null;
        try {
            students = queryRunner.query(sql, new BeanListHandler<Student>(Stu
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return students;
    }

    // offset: row:多少行
    @Override
    public List<Student> queryStudents(int offset, int row) {
        String sql = "select sid,sname,age,gender from (select rownum rn,sid,sn
        Object[] args = {row,offset };
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        List<Student> students=null;
        try {
            students = queryRunner.query(sql, new BeanListHandler<Student>(Stu
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return students;
    }

}

* DataSourceUtil
public class DataSourceUtil {
    private static DataSource dataSource;
    static {
        Properties props=new Properties();
        try {
            props.load(ConnectionUtils.class.getClassLoader().getResourceAsStre
            dataSource=DruidDataSourceFactory.createDataSource(props);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        };

    }

    public static DataSource getDataSource() {
        return dataSource;
    }
}

* 测试
    studentDao=new StudentDaoImpl3();

* 测试其他子类
  * MapHandler, MapListHandler, ScalarHandler, ColumnListHandler
@Test
    public void test1() throws SQLException {
        String sql = "select sid,sname,age,gender from stu where sid=?";
        Object[] params = { "S_1006" };
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        Map<String, Object> map = queryRunner.query(sql, new MapHandler(), para
        System.out.println(map);
    }

    @Test
    public void test2() throws SQLException {
        String sql = "select sid,sname,age,gender from stu";
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        List<Map<String, Object>> maps = queryRunner.query(sql, new MapListHanc
        for(Map<String,Object> map:maps) {
            System.out.println(map);
        }
    }

    @Test
    public void test3() throws SQLException {
        String sql = "select count(*) from stu";
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        BigDecimal bd = queryRunner.query(sql, new ScalarHandler<BigDecimal>()
        System.out.println("count:"+bd.intValue());
    }

    @Test
    public void test4() throws SQLException {
        String sql = "select sname from stu";
        QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource()
        List<String> names = queryRunner.query(sql, new ColumnListHandler<Strin
        System.out.println(names);
    }


* 测试批处理
 create table testbatch(
    name varchar2(50)
 );
 @Test
 public void test5() throws SQLException {
    long start = System.currentTimeMillis();
    // 1 获得连接
    String sql = "insert into testbatch(name) values(?)";
    QueryRunner queryRunner=new QueryRunner(DataSourceUtil.getDataSource());
    Object[][] params=new Object[1000][];
    for (int i = 1; i <= 10000; i++) {
            params[i % 1000]=new Object[] {"xiaobai" + i};
            // 每次发一千条
            if (i % 1000 == 0) {
                queryRunner.batch(sql, params);
            }
        }
            // 剩余再执行一次
    params=new Object[8][];
    for (int i = 10001; i <= 10008; i++) {
            params[i % 8]=new Object[] {"xiaobai" + i};
            }
            queryRunner.batch(sql, params);
```

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

```

* 能够阅读DBUtils的核心源码
    * 参考09-DBUtils源码分析