

DCL：数据控制语言：grant revoke

实现类都叫驱动

测试方法要求：不能有返回值，不能有参数

assert.assertEquals（期望值，函数运行的结果值，允许的误差范围）；若期望和实际值相当，则运行成功，若不相等，则单元测试运行错误

```
//加载驱动
Class.forName("com.mysql.jdbc.Driver");

//建立连接
Connection connection = null;
Statement statement = null;
ResultSet resultSet = null;
try {
    connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/anqili","root","AQL271422");
    statement = connection.createStatement();
    resultSet = statement.executeQuery("select *from emp2");
}
```

一、JDBC概述

为什么要使用JDBC？

JDBC:java database connectivity SUN公司提供的一套操作数据库的标准规范。

JDBC与数据库驱动的关系：接口与实现的关系。

JDBC规范（掌握四个核心对象）：三个接口一个类

DriverManager:用于注册驱动

Connection: 表示与数据库创建的连接

Statement: 操作数据库sql语句的对象，相当于小货车，将sql语句执行去取数据库中的数据，若

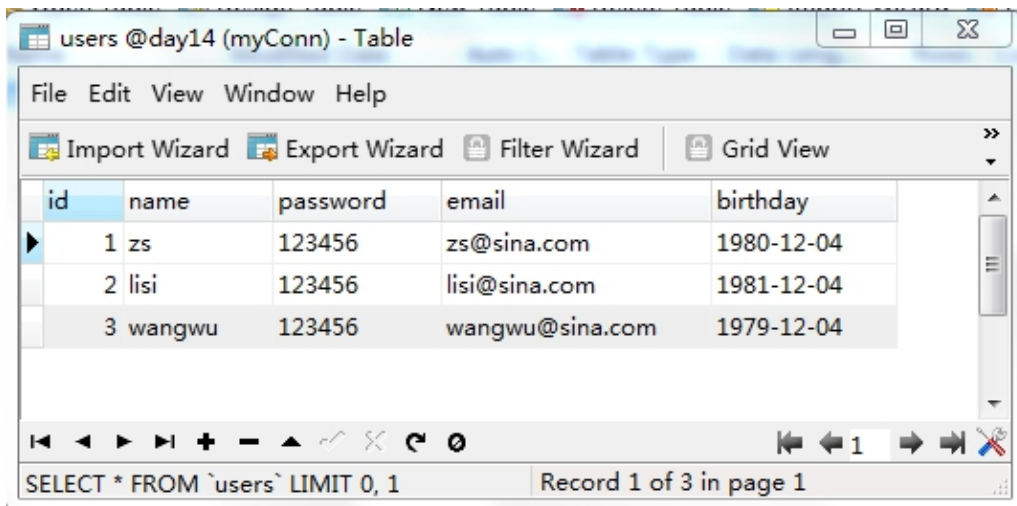
有返回的东西，则将其叫给resultSet中

ResultSet: 结果集或一张虚拟表。存在于客户端

- day14_demo1
 - src
 - JRE System Library [JavaSE-1.6]
 - Referenced Libraries
 - mysql-connector-java-5.0.8-bin.jar
 - lib

```
insert into users(name,password,email,birthday)
values('zs','123456','zs@sina.com','1980-12-04');
insert into users(name,password,email,birthday)
values('lisi','123456','lisi@sina.com','1981-12-04');
```

```
insert into users(name,password,email,birthday)
values('wangwu','123456','wangwu@sina.com','1979-12-04');
```



The screenshot shows a database application window titled "users @day14 (myConn) - Table". It features a menu bar (File, Edit, View, Window, Help) and a toolbar with buttons for Import Wizard, Export Wizard, Filter Wizard, and Grid View. Below the toolbar is a table with 5 columns: id, name, password, email, and birthday. The table contains 3 rows of data. At the bottom, there is a status bar showing the SQL query "SELECT * FROM `users` LIMIT 0, 1" and the record count "Record 1 of 3 in page 1".

id	name	password	email	birthday
1	zs	123456	zs@sina.com	1980-12-04
2	lisi	123456	lisi@sina.com	1981-12-04
3	wangwu	123456	wangwu@sina.com	1979-12-04

2、创建java project项目，添加数据库驱动 (*.jar)

3、实现JDBC操作

//1、注册驱动

//2、创建连接

//3、得到执行sql语句的Statement对象

//4、执行sql语句，并返回结果

//5、处理结果

//6关闭资源

```
//1. 注册驱动
DriverManager.registerDriver(new com.mysql.jdbc.Driver());
//2. 创建连接
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/day14", "root", "root");
//3. 得到操作数据库sql语句的对象 Statement
Statement stmt = conn.createStatement();
//4. 执行sql语句
String sql = "select id,name,password,email,birthday from users";
ResultSet rs = stmt.executeQuery(sql);
//5. 如何有返回结果集，处理结果
while(rs.next()){
    System.out.println(rs.getObject(1));
    System.out.println(rs.getObject(2));
    System.out.println(rs.getObject(3));
    System.out.println(rs.getObject(4));
    System.out.println(rs.getObject(5));
    System.out.println("-----");
}
//6关闭资源
rs.close();
stmt.close();
conn.close();
```

id	name	password	email	birthday
1	zs	123456	zs@sina.com	1980-12-04
2	lisi	123456	lisi@sina.com	1981-12-04
3	wangwu	123456	wangwu@sina.com	1979-12-04

ResultSet 对象中的数据如左图。此对象中有一个游标。默认是指向表的第一行数据之前（表头）。

调用些对象的 next()
boolean next()
方法调用一次，游标就向下移一行

三、JDBC常用的类和接口详解

1、java.sql.DriverManager类：创建连接

a、注册驱动

DriverManager.registerDriver(new com.mysql.jdbc.Driver()); 不建议使用
原因有2个：

- ＞ 导致驱动被注册2次。
- ＞ 强烈依赖数据库的驱动jar

解决办法：

```
Class.forName("com.mysql.jdbc.Driver");
```

b、与数据库建立连接

```
static Connection getConnection(String url, String user, String password)
```

试图建立到给定数据库 URL 的连接。

```
getConnection("jdbc:mysql://localhost:3306/day06", "root", "root");
```

URL: SUN公司与数据库厂商之间的一种协议。

```
jdbc:mysql://localhost:3306/day06
```

协议 子协议 IP : 端口号 数据库

mysql: jdbc:mysql://localhost:3306/day14 或者 jdbc:mysql:///day14（默认本机连接）

oracle: jdbc:oracle:thin:@localhost:1521:sid

```
Properties info = new Properties();//要参考数据库文档
```

```
info.setProperty("user", "root");
```

```
info.setProperty("password", "root");
```

```
getConnection(String url, Properties info)
```

```
getConnection(String url)
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/day14?  
user=root&password=root");
```

2、java.sql.Connection接口：一个连接

接口的实现在数据库驱动中。所有与数据库交互都是基于连接对象的。

```
Statement createStatement(); //创建操作sql语句的对象
```

3、java.sql.Statement接口：操作sql语句，并返回相应结果的对象（小货车）

接口的实现在数据库驱动中。用于执行静态 SQL 语句并返回它所生成结果的对象。

ResultSet executeQuery(String sql) 根据查询语句返回结果集。只能执行 select 语句。

int executeUpdate(String sql) 根据执行的DML（insert update delete）语句，返回受影响的行数。

boolean execute(String sql) 此方法可以执行任意sql语句。返回boolean值，表示是否返回ResultSet结果集。仅当执行select语句，且有返回结果时返回true, 其它语句都返回false;

4、java.sql.ResultSet接口：结果集（客户端存表数据对象）

a、封装结果集的。

提供一个游标，默认游标指向结果集第一行之前。

调用一次next()，游标向下移动一行。

提供一些get方法。

封装数据的方法

Object getObject(int columnIndex); 根据序号取值，索引从1开始

Object getObject(String ColomnName); 根据列名取值。

将结果集中的数据封装到javaBean中

java的数据类型与数据库中的类型的关系

byte	tinyint
short	smallint
int	int
long	bigint
float	float
double	double
String	char varchar
Date	date

boolean next() 将光标从当前位置向下移动一行

int getInt(int colIndex) 以int形式获取ResultSet结果集当前行指定列号值

int getInt(String colLabel) 以int形式获取ResultSet结果集当前行指定列名值

float getFloat(int colIndex) 以float形式获取ResultSet结果集当前行指定列号值

float getFloat(String colLabel) 以float形式获取ResultSet结果集当前行指定列名值

String getString(int colIndex) 以String 形式获取ResultSet结果集当前行指定列号值

String getString(String colLabel) 以String形式获取ResultSet结果集当前行指定列名值

Date getDate(int columnIndex);

Date getDate(String columnName);

void close() 关闭ResultSet 对象

b、可移动游标的方法

boolean next() 将光标从当前位置向前移一行。

boolean previous()

将光标移动到此 ResultSet 对象的上一行。

boolean absolute(int row) 参数是当前行的索引，从1开始

根据行的索引定位移动的指定索引行。

void afterLast()

将光标移动到末尾，正好位于最后一行之后。

void beforeFirst()

将光标移动到开头，正好位于第一行之前。

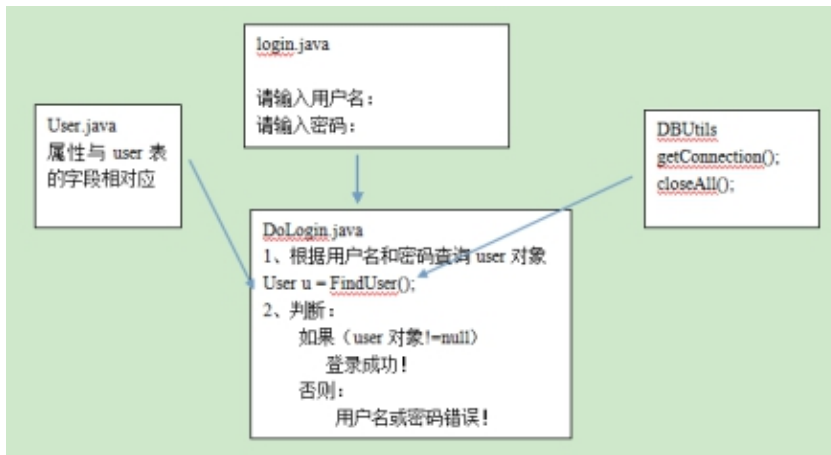
5、释放资源

资源有限，要正确关闭。

```
@Test
public void test3(){
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/day14", "root", "root");
        stmt = conn.createStatement();
        String sql = "select id,name,password,birthday,email from users";
        rs = stmt.executeQuery(sql);
        rs.afterLast();
        boolean b = rs.previous();
        if(b){
            User user = new User();
            user.setId(rs.getInt("id"));
            user.setName(rs.getString("name"));
            user.setPassword(rs.getString("password"));
            user.setBirthday(rs.getDate("birthday"));
            user.setEmail(rs.getString("email"));
            System.out.println(user);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        if(rs!=null){
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(stmt!=null){
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(conn!=null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

四、使用JDBC实现CRUD操作

五、实现一个用户登录的功能



六、SQL注入问题：preparedStatement

preparedStatement：预编译对象，是Statement对象的子类。

特点：

性能要高

会把sql语句先编译

sql语句中的参数会发生变化，过滤掉用户输入的关键字。

```

try {
    conn = DBUtils.getConnection();
    String sql = "select * from users where name=? and password=?";
    ps = conn.prepareStatement(sql); //注：创建此对象时，要把sql语句先放进去。
    //给预编译sql语句赋值
    ps.setString(1, name); //1代表sql语句中的第一个?
    ps.setString(2, password);
    rs = ps.executeQuery();
}
  
```

补充例子：

```

public void save(User user) {
    if(user==null)
        throw new IllegalArgumentException("The param user can not be null");

    Connection conn = null;
    PreparedStatement stmt = null;
    try{
        conn = JdbcUtil.getConnection();
        stmt = conn.prepareStatement("INSERT INTO user (username,password,email,birthday) VALUES (?, ?, ?, ?)");
        stmt.setString(1, user.getUsername());
        stmt.setString(2, user.getPassword());
        stmt.setString(3, user.getEmail());
        stmt.setDate(4, new java.sql.Date(user.getBirthday().getTime())); //必须明白
        stmt.executeUpdate();
    } catch (Exception e) {
        throw new RuntimeException(e);
    } finally {
        JdbcUtil.release(null, stmt, conn);
    }
}
  
```