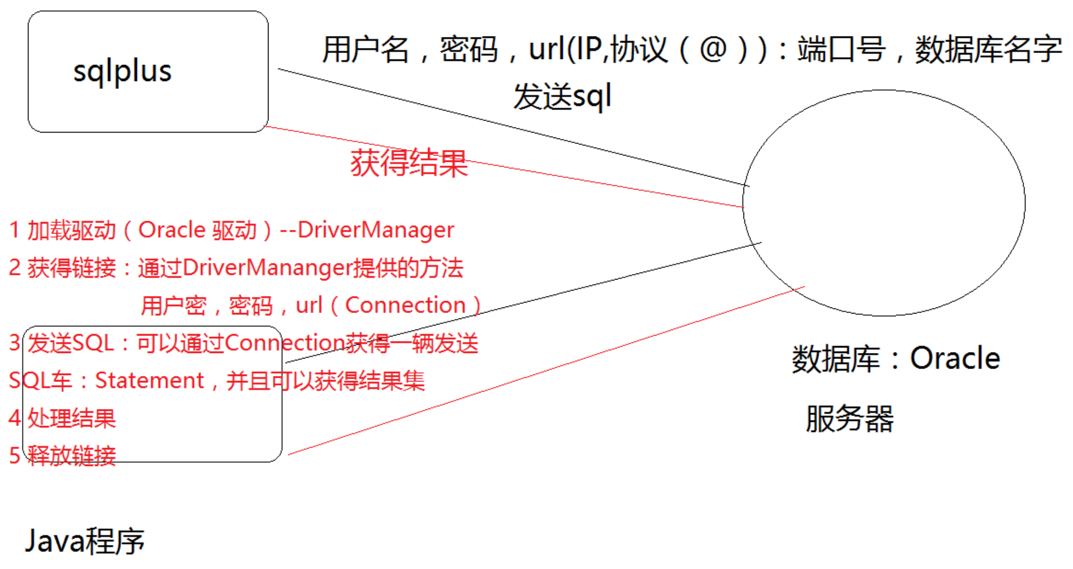
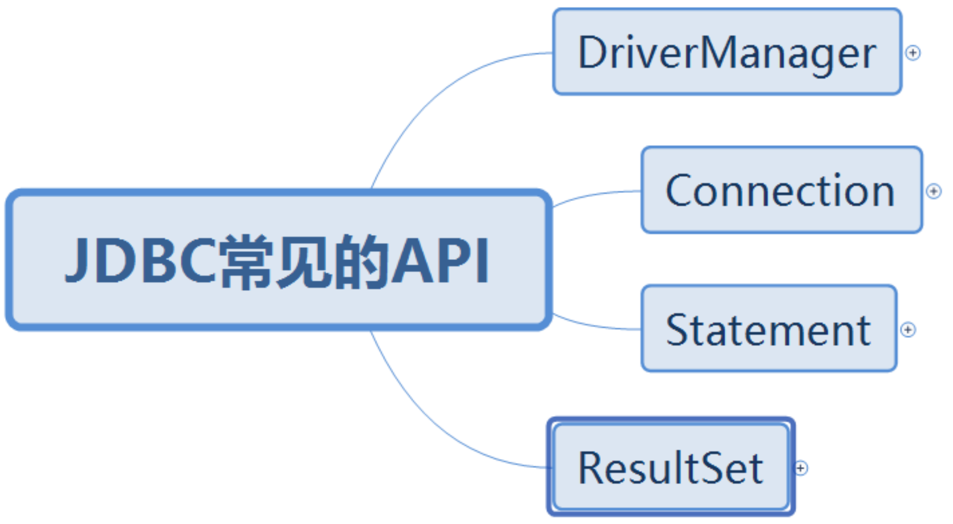
* 学习目标
 * 能够理解JDBC的概述
 * 能够掌握JDBC常见的API
 * 能够掌握JDBC的HelloWorld的开发
 * 能够掌握JDBC工具类的编写
 * 能够掌握JDBC的CRUD的编写
 * 能够使用PreparedStatement防止SQL注入
 * 能够掌握JDBC的批处理

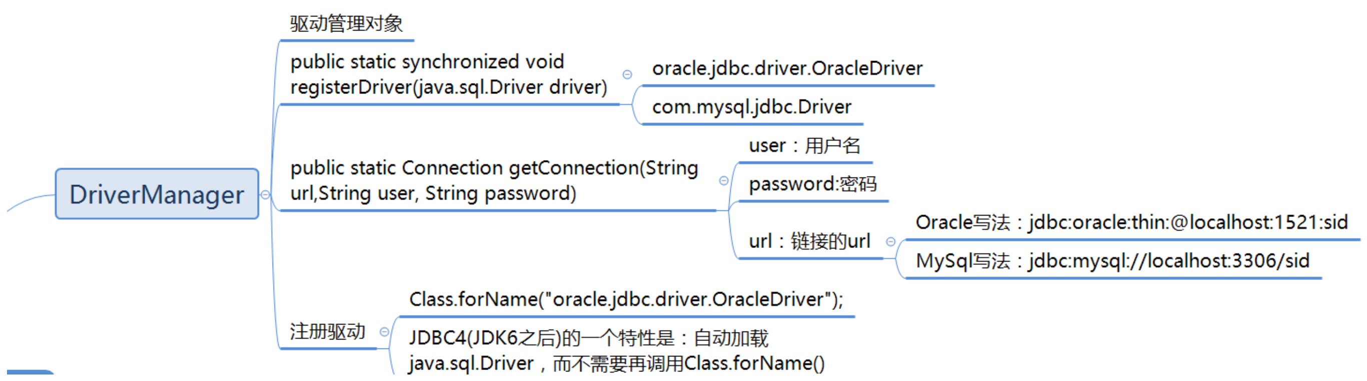
----

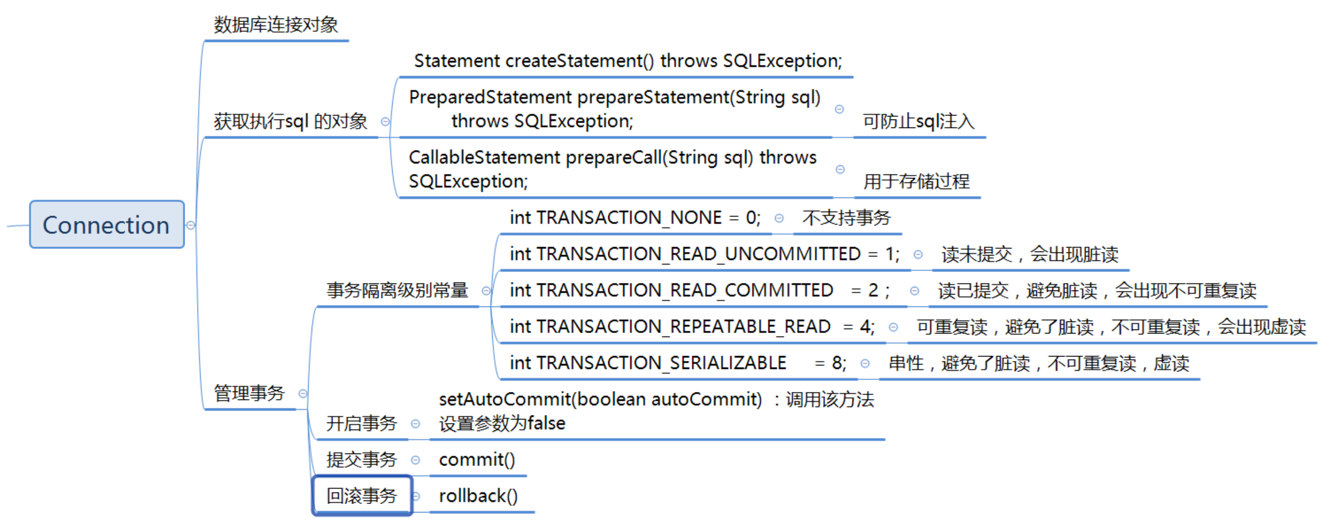
 * 回顾
   * DDL：create table,alter table modify,add,drop,rename... to...
     drop table,truncate table, delete
   * DML: insert,update,delete
   * DQL: select * from table_name where condiction group by,having,order by(asc desc),limit
   * JDBC：JAVA DataBase Connectivity
   * 常见的API：DriverManager，Connection，Statement，ResultSet
   * HelloWorld JDBC

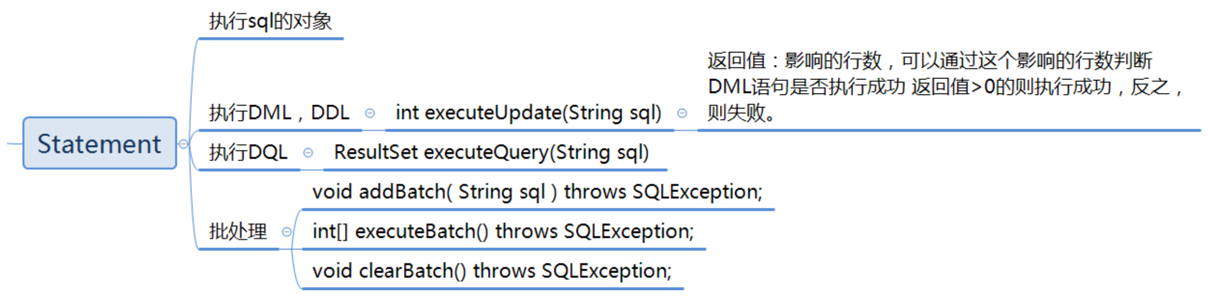
* 能够理解JDBC的概述
   * JDBC：Java DataBase Connectivity (Java数据库连接)
   * 由sun公司提供一套Java操作数据库的接口标准，而且由各大厂商提供驱动(jar 包)。
      * Oracle提供Oracle驱动的实现（jar包）
      * MySQL提供MySQL驱动的实现 （jar包）
 * 能够掌握JDBC常见的API

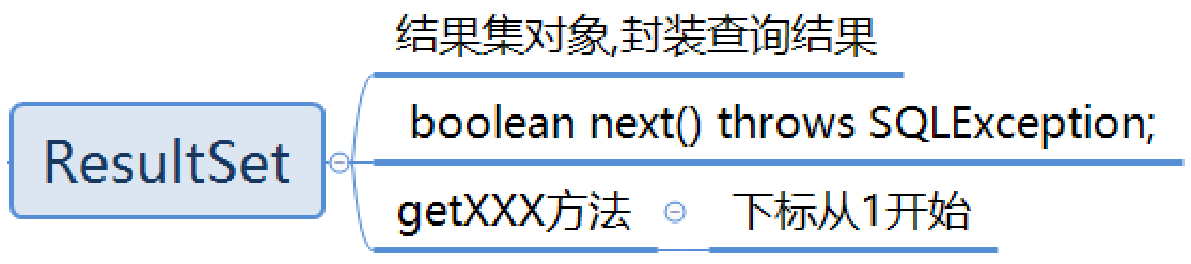

sqlplus
用户名，密码，url(IP,协议（@））:端口号，数据库名字
发送sql
获得结果
1 加载驱动（Oracle 驱动）--DriverManager
2 获得链接：通过DriverMananger提供的方法
用户密，密码，url（Connection）
3 发送SQL：可以通过Connection获得一辆发送
SQL车：Statement，并且可以获得结果集
4 处理结果
5 释放链接
数据库：Oracle
服务器
Java程序


JDBC常见的API
DriverManager
Connection
Statement
ResultSet


DriverManager
驱动管理对象
public static synchronized void registerDriver(java.sql.Driver driver)
oracle.jdbc.driver.OracleDriver
com.mysql.jdbc.Driver
public static Connection getConnection(String url,String user, String password)
user：用户名
password:密码
url：链接的url
Oracle写法：jdbc:oracle:thin:@localhost:1521:sid
MySql写法：jdbc:mysql://localhost:3306/sid
注册驱动
Class.forName("oracle.jdbc.driver.OracleDriver");
JDBC4(JDK6之后)的一个特性是：自动加载 java.sql.Driver，而不需要再调用Class.forName()


Connection
数据库连接对象
获取执行sql 的对象
Statement createStatement() throws SQLException;
PreparedStatement prepareStatement(String sql) throws SQLException;
可防止sql注入
CallableStatement prepareCall(String sql) throws SQLException;
用于存储过程
管理事务
事务隔离级别常量
int TRANSACTION_NONE = 0;
不支持事务
int TRANSACTION_READ_UNCOMMITTED = 1;
读未提交，会出现脏读
int TRANSACTION_READ_COMMITTED = 2；
读已提交，避免脏读，会出现不可重复读
int TRANSACTION_REPEATABLE_READ = 4;
可重复读，避免了脏读，不可重复读，会出现虚读
int TRANSACTION_SERIALIZABLE = 8;
串性，避免了脏读，不可重复读，虚读
开启事务
setAutoCommit(boolean autoCommit) ：调用该方法设置参数为false
提交事务
commit()
回滚事务
rollback()


Statement
执行sql的对象
执行DML，DDL
int executeUpdate(String sql)
返回值：影响的行数，可以通过这个影响的行数判断DML语句是否执行成功 返回值>0的则执行成功，反之，则失败。
执行DQL
ResultSet executeQuery(String sql)
批处理
void addBatch(String sql) throws SQLException;
int[] executeBatch() throws SQLException;
void clearBatch() throws SQLException;



* 能够掌握JDBC的HelloWorld的开发

```java
public static void main(String[] args) throws SQLException {
        // 1 加载驱动（Oracle 驱动）--DriverManager
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
//      2 获得链接：通过DriverMananger提供的方法
//              用户密，密码，url（Connection）
        String user="scott";
        String password="tiger";
        String url="jdbc:oracle:thin:@192.168.1.121:1521:orcl";
        Connection conn = DriverManager.getConnection(url, user, password);
```

```
//      3 发送SQL：可以通过Connection获得一辆发送SQL车：Statement，并且可以获得结果
        Statement statement = conn.createStatement();
        String sql="select * from emp";
        ResultSet rs=statement.executeQuery(sql);
//      4 处理结果
        while(rs.next()) {
            String empNo=rs.getString(1);// 数组下标是从开始，JDBC不从零开始,从1开
            String eName=rs.getString(2);
            String job=rs.getString(3);
            String mgr=rs.getString(4);
            String hireDate=rs.getString(5);
            String sal=rs.getString(6);
            String comm=rs.getString(7);
            String deptNo=rs.getString(8);
            System.out.println(empNo+":"+eName+":"+job+":"+mgr+":"+hireDate+":'
        }
//      5 释放链接
        rs.close();
        conn.close();
    }
结果：

7369:SMITH:CLERK:7902:1980-12-17 00:00:00:800:null:20
7499:ALLEN:SALESMAN:7698:1981-02-20 00:00:00:1600:300:30
7521:WARD:SALESMAN:7698:1981-02-22 00:00:00:1250:500:30
7566:JONES:MANAGER:7839:1981-04-02 00:00:00:2975:null:20
7654:MARTIN:SALESMAN:7698:1981-09-28 00:00:00:1250:1400:30
7698:BLAKE:MANAGER:7839:1981-05-01 00:00:00:2850:null:30
7782:CLARK:MANAGER:7839:1981-06-09 00:00:00:2450:null:10
7788:SCOTT:ANALYST:7566:1987-04-19 00:00:00:3000:null:20
7839:KING:PRESIDENT:null:1981-11-17 00:00:00:5000:null:10
7844:TURNER:SALESMAN:7698:1981-09-08 00:00:00:1500:0:30
7876:ADAMS:CLERK:7788:1987-05-23 00:00:00:1100:null:20
7900:JAMES:CLERK:7698:1981-12-03 00:00:00:950:null:30
7902:FORD:ANALYST:7566:1981-12-03 00:00:00:3000:null:20
7934:MILLER:CLERK:7782:1982-01-23 00:00:00:1300:null:10

* 注册的方式
 * Class.forName("oracle.jdbc.driver.OracleDriver");
 * jdk1.6之后：注册驱动，可以省略不写

* 正确关闭释放资源的方法
public static void main(String[] args){
        String user="scott";
        String password="tiger";
        String url="jdbc:oracle:thin:@192.168.1.121:1521:orcl";
        Connection conn = null;
        Statement statement =null;
        ResultSet rs=null;
        try {
            conn = DriverManager.getConnection(url, user, password);
//          3 发送SQL：可以通过Connection获得一辆发送SQL车：Statement，并且可以获得
            statement = conn.createStatement();
            String sql="select * from emp";
            rs=statement.executeQuery(sql);
//          4 处理结果
            while(rs.next()) {
                String empNo=rs.getString(1);// 数组下标是从开始，JDBC不从零开始，
                String eName=rs.getString(2);
                String job=rs.getString(3);
                String mgr=rs.getString(4);
                String hireDate=rs.getString(5);
                String sal=rs.getString(6);
                String comm=rs.getString(7);
                String deptNo=rs.getString(8);
                System.out.println(empNo+":"+eName+":"+job+":"+mgr+":"+hireDate
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
//          5 释放链接
            if(rs!=null) {
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(statement!=null) {
                try {
```

```java
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conn!=null) {
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }

        }
    }

```

* 能够掌握JDBC工具类的编写

```java
public class ConnectionUtils {
    private static String userName;
    private static String password;
    private static String url;

    static {
        Properties prop=new Properties();
        try {
            prop.load(ConnectionUtils.class.getClassLoader().getResourceAsStrea
            userName=prop.getProperty("username");
            password=prop.getProperty("password");
            url=prop.getProperty("url");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(url, userName, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }


    /**
     * 是否资源的方法
     * @param conn
     * @param st
     * @param rs
     */
    public static void close(Connection conn,Statement st,ResultSet rs) {
        if(rs!=null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(st!=null) {
            try {
                st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(conn!=null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

    }

    public static void closeRs(ResultSet rs) {
        close(null, null, rs);
    }

    public static void closeSt(Statement st) {
        close(null, st, null);
    }

    public static void closeConn(Connection con) {
        close(con,null,null);
    }
}

* 测试
public static void main(String[] args){
        Connection conn = null;
        Statement st =null;
        ResultSet rs=null;
        try {
            conn=ConnectionUtils.getConnection();
//          3 发送SQL：可以通过Connection获得一辆发送SQL车：Statement，并且可以获得
            st = conn.createStatement();
            String sql="select * from emp";
            rs=st.executeQuery(sql);
//          4 处理结果
            while(rs.next()) {
                String empNo=rs.getString(1);// 数组下标是从开始，JDBC不从零开始,
                String eName=rs.getString(2);
                String job=rs.getString(3);
                String mgr=rs.getString(4);
                String hireDate=rs.getString(5);
                String sal=rs.getString(6);
                String comm=rs.getString(7);
                String deptNo=rs.getString(8);
                System.out.println(empNo+":"+eName+":"+job+":"+mgr+":"+hireDate
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            ConnectionUtils.close(conn, st, rs);
        }
    }

```

* 能够掌握JDBC的CRUD的编写

```
package com.lg.test1;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class Test2 {

    private Connection conn;
    private Statement st;
    private ResultSet rs;
    @Before
    public void setUp() throws SQLException {
        conn = ConnectionUtils.getConnection();
        st = conn.createStatement();
    }

    @Test
    public void testAdd() throws SQLException {
        String sql = "INSERT INTO stu(sid,sname,age,gender) VALUES('S_1013','xi
        // 3 insert
        int result = st.executeUpdate(sql);
        if (result > 0) {
            System.out.println("insert ok");
        }
    }
```

```java

    @Test
    public void testUpdate() throws SQLException {
        String sql = "UPDATE stu SET sname='xiaohong' WHERE sid='S_1013'";
        // 3 insert
        int result = st.executeUpdate(sql);
        if (result > 0) {
            System.out.println("UPDATE ok");
        }
    }

    @Test
    public void testDelete() throws SQLException {
        String sql = "DELETE FROM stu WHERE sid='S_1013'";
        // 3 insert
        int result = st.executeUpdate(sql);
        if (result > 0) {
            System.out.println("DELETE ok");
        }
    }

    @Test
    public void testQueryAll() throws SQLException {
        String sql = "SELECT * FROM stu;
        // 3 insert
        ResultSet rs = st.executeQuery(sql);
        // st.executeUpdate(sql)
        // 5 处理结果
        while (rs.next()) {
            // JDBC 下标是从1开始的
            String sid = rs.getString(1);
            String sname = rs.getString(2);
            int age = rs.getInt(3);
            String sex = rs.getString(4);
            System.out.println(sid + ":" + sname + ":" + age + ":" + sex);
        }
    }

    @After
    public void finish() {
        ConnectionUtils.close(conn, st, rs);
    }
}

* StudentDao的编写
  * DAO(Data Access Object) 数据访问对象是一个面向对象的数据库接口
public class Student {
    private String sid;
    private String name;
    private int age;
    private String gender;
    public Student() {
        super();
    }

    public Student(String sid, String name, int age, String gender) {
        super();
        this.sid = sid;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String getSid() {
        return sid;
    }
    public void setSid(String sid) {
        this.sid = sid;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + age;
        result = prime * result + ((gender == null) ? 0 : gender.hashCode());
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + ((sid == null) ? 0 : sid.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Student other = (Student) obj;
        if (age != other.age)
            return false;
        if (gender == null) {
            if (other.gender != null)
                return false;
        } else if (!gender.equals(other.gender))
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        if (sid == null) {
            if (other.sid != null)
                return false;
        } else if (!sid.equals(other.sid))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "User [sid=" + sid + ", name=" + name + ", age=" + age + ", genc
    }

}

public interface StudentDao {
    // CRUD
    // 添加学生
    boolean add(Student student);

    // 根据学号，删除学生
    boolean delStudent(String sid);

    // 通过学号，修改姓名
    boolean updateStudent(String sid,String name);

    // 通过学号获得的学生
    public Student getStudent(String sid);

    // 查询所有的学生
    public List<Student> getStudents();

    // 查询部分学生(分页)
    public List<Student> getStudents(int offset,int row);
}

public class StudentDaoImpl implements StudentDao {

    @Override
    public boolean add(Student student) {
        String sql = "INSERT INTO stu(sid,sname,age,gender) VALUES('" + student
                + "','" + student.getAge() + "','" + student.getGender() + "')'
        Connection conn = ConnectionUtils.getConnection();
        Statement st = null;
        int result = 0;
        try {
            st = conn.createStatement();
            result = st.executeUpdate(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, null);
        }
        return result > 0;
    }

    @Override
    public boolean delStudent(String sid) {
        Connection conn = ConnectionUtils.getConnection();
        Statement st = null;
        String sql = "DELETE FROM stu WHERE sid='" + sid + "'";
        // 3 insert
        int result = 0;
        try {
            st = conn.createStatement();
            result = st.executeUpdate(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, null);
        }
        return result > 0;
    }

    @Override
    public boolean updateStudent(String sid, String name) {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "UPDATE stu SET sname='" + name + "' WHERE sid='" + sid +
        Statement st = null;
        int result = 0;
```

```java
        try {
            st = conn.createStatement();
            result = st.executeUpdate(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, null);
        }
        return result > 0;
    }

    @Override
    public Student getStudent(String sid) {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "select * from stu where sid='" + sid + "'";
        Statement st = null;
        ResultSet rs = null;
        Student student = null;
        try {
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            if (rs.next()) {
                student = new Student(rs.getString(1), rs.getString(2), Integer
                        rs.getString(4));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, rs);
        }
        return student;
    }

    @Override
    public List<Student> getStudents() {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "select * from stu";
        Statement st = null;
        ResultSet rs = null;
        List<Student> students = new ArrayList<Student>();
        try {
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            while (rs.next()) {
                try {
                    Student student = new Student(rs.getString(1), rs.getString
                            rs.getString(4));
                    students.add(student);
                } catch (NumberFormatException e) {
//                  e.printStackTrace();
                    Student student=new Student();
                    student.setSid(rs.getString(1));
                    student.setName(rs.getString(2));
                    students.add(student);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, rs);
        }
        return students;
    }

    @Override
    public List<Student> getStudents(int offset, int row) {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "select sid,sname,age,gender from (select rownum rn,sid,sn
        Statement st = null;
        ResultSet rs = null;
        List<Student> students = new ArrayList<Student>();
        try {
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            while (rs.next()) {
                try {
                    Student student = new Student(rs.getString(1), rs.getString
                            rs.getString(4));
                    students.add(student);
                } catch (NumberFormatException e) {
```

```java
//                      e.printStackTrace();
                        Student student=new Student();
                        student.setSid(rs.getString(1));
                        student.setName(rs.getString(2));
                        students.add(student);
                    }
                }
            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                ConnectionUtils.close(conn, st, rs);
            }
            return students;
        }

}

public class StudentDaoTest {
    StudentDao studentDao;
    @Before
    public void setUp() {
        studentDao=new StudentDaoImpl();
    }
    @Test
    public void testAdd() {
        studentDao.add(new Student("S_1012","xiaoming",20,"male"));
    }
    @Test
    public void testUpdate() {
        studentDao.updateStudent("S_1012", "xiaobai");
    }
    @Test
    public void testDel() {
        studentDao.delStudent("S_1012");
    }

    @Test
    public void testGetStudent() {
        Student student = studentDao.getStudent("S_1009");
        System.out.println(student);
    }
    @Test
    public void testGetStudents() {
        List<Student> students = studentDao.getStudents();
        for(Student student:students) {
            System.out.println(student);
        }
    }
    @Test
    public void testGetStudents2() {
        List<Student> students = studentDao.getStudents(2,5);
        for(Student student:students) {
            System.out.println(student);
        }
    }
}

```

* 能够使用PreparedStatement防止SQL注入

```
* 创建用户表
 create table t_user(
     username varchar2(50) primary key not null,
     password varchar2(50)
 );
 insert into t_user values('xiaohei','202cb962ac59075b964b07152d234b70');
 insert into t_user values('xiaobai','202cb962ac59075b964b07152d234b70');
 commit;

public class User {
    private String name;
    private String password;
    public User() {
        super();
    }

    public User(String name, String password) {
        super();
        this.name = name;
```

```java
        this.password = password;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + ((password == null) ? 0 : password.hashCode(
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        User other = (User) obj;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        if (password == null) {
            if (other.password != null)
```

```java
                return false;
        } else if (!password.equals(other.password))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "User [name=" + name + ", password=" + password + "]";
    }
}

public interface UserDao {
    // 可以SQL注入的
    User getUser(String username,String password);
    // 防止SQL注入的
    User getUser2(String username,String password);
}

public class UserDaoImpl implements UserDao{

    @Override
    public User getUser(String username, String password) {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "select * from t_user where username='"+username+"' and pa
        Statement st = null;
        ResultSet rs = null;
        User user = null;
        System.out.println(sql);
        try {
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            if (rs.next()) {
                user = new User(rs.getString(1), rs.getString(2));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, rs);
        }
        return user;
    }


    @Override
    public User getUser2(String username, String password) {
        Connection conn = ConnectionUtils.getConnection();
        String sql = "select * from t_user where username=? and password=?";
        PreparedStatement st = null;
        ResultSet rs = null;
        User user = null;
        System.out.println(sql);
        try {
            // 先预编译好
            st =conn.prepareStatement(sql);
            st.setString(1, username);
            st.setString(2, password);
            rs = st.executeQuery();
            if (rs.next()) {
                user = new User(rs.getString(1), rs.getString(2));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            ConnectionUtils.close(conn, st, rs);
        }
        return user;
    }

}

* 测试
public class MD5Utils {
    /**
     * 使用md5的算法进行加密
     */
    public static String md5(String plainText) {
        byte[] secretBytes = null;
        try {
            secretBytes = MessageDigest.getInstance("md5").digest(
                    plainText.getBytes());
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("没有md5这个算法！");
        }
        String md5code = new BigInteger(1, secretBytes).toString(16);// 16进制数
        // 如果生成数字未满32位，需要前面补0
        for (int i = 0; i < 32 - md5code.length(); i++) {
            md5code = "0" + md5code;
        }
        return md5code;
    }

}

    @Test
    public void test1() {
        UserDao userDao=new UserDaoImpl();
 //     User user = userDao.getUser("xiaohei", MD5Utils.md5("123"));
//      User user = userDao.getUser("xiaohei123", MD5Utils.md5("56")+"' or '1'=
//      select * from t_user where username='xiaohei123' and password='9f61408e
//      User user =userDao.getUser2("xiaohei", MD5Utils.md5("123"));
        User user = userDao.getUser2("xiaohei123", MD5Utils.md5("56")+"' or '1'
        if(user!=null) {
            System.out.println("登录成功");
        }else {
            System.out.println("登录失败");
        }
    }

* sql注入
select * from t_user where username='xiaohei123' and password='202cb962ac59075b
```

* 能够掌握JDBC的批处理

 * 概述：程序向数据库发送一批SQL语句执行，这时应避免向数据库一条条的发送执行，提升执行效率。

 * JDBC实现批处理有两种方式：Statement和Preparedstatement

* Statement.addBatch(sql)方式实现批处理：

    * 优点：可以向数据库发送多条不同的SQL语句。

    * 缺点：

      * SQL语句没有预编译。

      * 当向数据库发送多条语句相同，但仅参数不同的SQL语句时，需重复写上很多条SQL语句。

* 采用PreparedStatement.addBatch()实现批处理

    * 优点：发送的是预编译后的SQL语句，执行效率高。

    * 缺点：只能应用在SQL语句相同，但参数不同的批处理中。因此此种形式的批处理经常用于在同一个表中批量插入数据，或批量更新表的数据。

```
* 创建表testbatch表
 create table testbatch(
    name varchar2(50)
);
* Statement 批处理
@Test
    public void test1() throws SQLException {
         // 1 获得连接
        Connection conn = ConnectionUtils.getConnection();
        String sql1 = "insert into testbatch(name) values('xiaohei0')";
        String sql2 = "insert into testbatch(name) values('xiaohei1')";
        String sql3 = "insert into testbatch(name) values('xiaohei2')";
        String sql4 = "insert into testbatch(name) values('xiaohei3')";
        String sql5 = "insert into testbatch(name) values('xiaohei4')";
        String sql6 = "insert into testbatch(name) values('xiaohei5')";
        Statement st = conn.createStatement();
        st.addBatch(sql1);
        st.addBatch(sql2);
        st.addBatch(sql3);
        st.addBatch(sql4);
        st.addBatch(sql5);
        st.addBatch(sql6);
        // 打包一起发过去
        st.executeBatch();
        st.clearBatch();
        ConnectionUtils.close(conn, st, null);
    }

* 使用PreparedStatement的批处理
 * 不使用批处理花的时间:
    @Test
    public void test2() throws SQLException {
        long start = System.currentTimeMillis();
        // 1 获得连接
        Connection conn = ConnectionUtils.getConnection();
        String sql = "insert into testbatch(name) values(?)";
        PreparedStatement psmt = conn.prepareStatement(sql);
        for (int i = 1; i < 10008; i++) {
            psmt.setString(1, "xiaobai" + i);
            psmt.executeUpdate();
        }
        long end = System.currentTimeMillis();
        System.out.println("不使用批处理花的时间:" + (end - start));
        ConnectionUtils.close(conn, psmt, null);
    }
 * 结果:
    不使用批处理花的时间:11935
 * 使用批处理
 @Test
    public void test2() throws SQLException {
        long start = System.currentTimeMillis();
        // 1 获得连接
        Connection conn = ConnectionUtils.getConnection();
        String sql = "insert into testbatch(name) values(?)";
        PreparedStatement psmt = conn.prepareStatement(sql);
        for (int i = 1; i < 10008; i++) {
            psmt.setString(1, "xiaobai" + i);
            psmt.addBatch();
            // 每次发一千条
            if (i % 1000 == 0) {
                psmt.executeBatch();
                psmt.clearBatch();
            }
        }
        psmt.executeBatch();
        psmt.clearBatch();
        long end = System.currentTimeMillis();
        System.out.println("使用批处理花的时间:" + (end - start));
        ConnectionUtils.close(conn, psmt, null);
    }
* 结果:
    使用批处理花的时间:848
```