目录

[1. 数据库连接 2](#_Toc12759)

[2. 使用try-with-resource的方式自动关闭连接 1](#_Toc13459)

[3. 增、删、改、查 1](#_Toc14081)

[4. 获取总数 2](#_Toc10758)

[5. 预编译Statement (PreparedStatement) 3](#_Toc21374)

[6. execute与executeUpdate 3](#_Toc15150)

[7. 特殊操作 4](#_Toc21696)

[8. 使用事务 6](#_Toc24410)

[9. ORM 8](#_Toc7696)

[10. DAO 10](#_Toc22798)

[11. 数据库连接池 14](#_Toc19105)

1. 数据库连接

**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
**public class** TestJDBC {  
 **public static void** main(String args[]){  
 Connection conn = **null**;  
 Statement s = **null**;  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 *//1. System.out.println("数据库驱动加载成功");* conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 *//2. System.out.println("连接成功，获取连接对象：" + c);* s = conn.createStatement();  
 String sql = **"insert into hero values(null,"** + **"'提莫'"** + **","** + 313.0f + **","** + 50 + **")"**;  
 s.execute(sql);  
 *//3. System.out.println("执行插入语句成功");* }**catch** (Exception e){  
 e.printStackTrace();  
 }**finally** {  
 *//先关闭Statement* **if**(s != **null**)  
 **try**{  
 s.close();  
 }**catch** (SQLException e){

e.printStackTrace();  
 }  
 *//后关闭Connection* **if**(conn != **null**)  
 **try**{  
 conn.close();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
 }  
}

1. 使用try-with-resource的方式自动关闭连接

如果觉得以上关闭连接的方式很麻烦，可以参考关闭流的方式，使用****try-with-resource****的方式自动关闭连接，因为Connection和Statement都实现了AutoCloseable接口

代码：

**public class** TestJDBC {  
 **public static void** main(String args[]){  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 **try**(  
 Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 Statement s = conn.createStatement();  
 ){  
 String sql = **"insert into hero values(null,"** + **"'鲁班'"** + **","** + 105.0f + **","** + 200 + **")"**;  
 s.execute(sql);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
}

1. 增、删、改、查

增加:

String sql =

**"insert into hero values(null,"** + **"'鲁班'"** + **","** + 105.0f + **","** + 200 + **")"**;  
s.execute(sql);

删除:

String sql = **"delete from hero where id = 109"**;  
s.execute(sql);

修改：

String sql = **"update hero set name = '英雄update' where id = 105"**;  
s.execute(sql);

查询：

String sql = **"select** *\** **from hero"**;  
ResultSet rs = s.executeQuery(sql); *//执行查询语句,并把结果返回给ResultSet***while** (rs.next()){  
 **int** id = rs.getInt(**"id"**);  
 String name = rs.getString(2);  
 **float** hp = rs.getFloat(**"hp"**);  
 **int** damage = rs.getInt(4);  
 System.***out***.printf(**"%d\t%s\t%f\t%d%n"**,id, name, hp, damage);  
}

显示前5条数据： select \* from hero limit 0,5

1. 获取总数

**public static void** main(String args[]){  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 **try**(Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 Statement s = conn.createStatement();  
 ){  
 String sql = **"select** *count***(***\****) from hero"**;  
 ResultSet rs = s.executeQuery(sql);  
 **int** total = 0;  
 **while**(rs.next()){  
 total = rs.getInt(1);  
 }  
 System.***out***.println(**"表Hero中总共有："** + total + **"条数据"**);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
}

1. 预编译Statement (PreparedStatement)

与创建Statement不同的是，需要根据sql语句创建PreparedStatement

除此之外，还能够通过设置参数，指定相应的值，而不是Statement那样使用字符串拼接

**public class** TestJDBC {  
 **public static void** main(String args[]) {  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 String sql = **"insert into hero values(null,?,?,?)"**;  
 **try** (Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**, **"root"**, **""**);  
 PreparedStatement ps = conn.prepareStatement(sql);  
 ) {  
 ps.setString(1, **"安妮"**);  
 ps.setFloat(2, 313.0f);  
 ps.setInt(3, 50);  
 ps.execute();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. execute与executeUpdate

相同点：都可以执行增加，删除，修改

不同点：

不同1：  
execute****可以执行查询语句****  
然后通过getResultSet，把结果集取出来  
executeUpdate****不能执行查询语句****

s.execute(sqlSelect);  
ResultSet rs = s.getResultSet();  
**while** (rs.next()) {  
 System.***out***.println(rs.getInt(**"id"**));  
}

不同2:  
execute****返回boolean类型****，true表示执行的是查询语句，false表示执行的是insert,delete,update等等  
executeUpdate****返回的是int****，表示有多少条数据受到了影响

**boolean** isSelect = s.execute(sqlSelect);  
System.***out***.println(isSelect);  
*// executeUpdate返回的是int，表示有多少条数据受到了影响*String sqlUpdate = **"update Hero set hp = 300 where id < 100"**;  
**int** number = s.executeUpdate(sqlUpdate);  
System.***out***.println(number);

1. 特殊操作

获取自增长id：SQL代码

CREATE TABLE hero (

id int(11) AUTO\_INCREMENT,

...

}

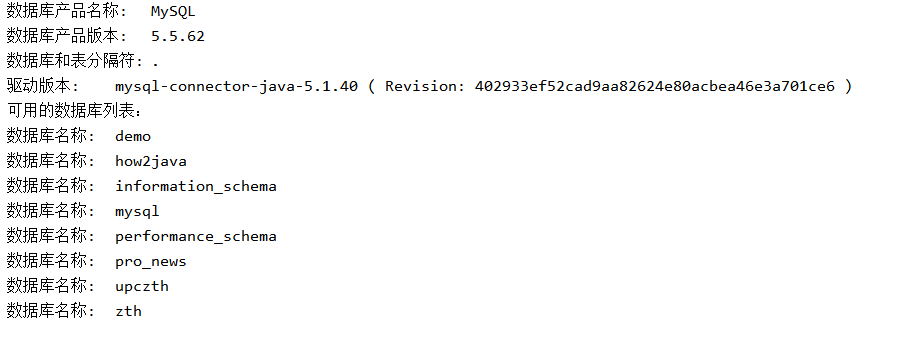
java代码：

**public class** TestJDBC {  
 **public static void** main(String[] args) {  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 String sql = **"insert into hero values(null, ?, ?, ?)"**;  
 **try** (Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 PreparedStatement ps = conn.prepareStatement(sql, Statement.***RETURN\_GENERATED\_KEYS***);  
 ){  
 ps.setString(1, **"元哥"**);  
 ps.setFloat(2, 616);  
 ps.setInt(3, 100);  
 ps.execute();  
  
 ResultSet rs = ps.getGeneratedKeys();  
 **if**(rs.next())  
 System.***out***.println(rs.getInt(1));  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
}

元数据：和数据库服务器相关的数据，比如数据库版本，有哪些表，表有哪些字段，字段类型是什么等等。

**public class** TestJDBC {  
 **public static void** main(String[] args) {  
 **try**{  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 String sql = **"insert into hero values(null, ?, ?, ?)"**;  
 **try** (Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);){  
 DatabaseMetaData dbmd = conn.getMetaData();  
 *// 获取数据库服务器产品名称* System.***out***.println(**"数据库产品名称:\t"**+dbmd.getDatabaseProductName());  
 *// 获取数据库服务器产品版本号* System.***out***.println(**"数据库产品版本:\t"**+dbmd.getDatabaseProductVersion());  
 *// 获取数据库服务器用作类别和表名之间的分隔符 如test.user* System.***out***.println(**"数据库和表分隔符:\t"**+dbmd.getCatalogSeparator());  
 *// 获取驱动版本* System.***out***.println(**"驱动版本:\t"**+dbmd.getDriverVersion());  
  
 System.***out***.println(**"可用的数据库列表："**);  
 *// 获取数据库名称* ResultSet rs = dbmd.getCatalogs();  
  
 **while** (rs.next()) {  
 System.***out***.println(**"数据库名称:\t"**+rs.getString(1));  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
}

输出：



1. 使用事务

在事务中的多个操作，要么都成功，要么都失败。

通过conn.setAutoCommit(false); 关闭自动提交

使用c.commit(); 进行手动提交

当c.setAutoCommit(false);时，事务是不会提交的  
只有执行使用 c.commit(); 才会提交进行

*// 有事务的前提下  
// 在事务中的多个操作，要么都成功，要么都失败*c.setAutoCommit(**false**);  
*// 加血的SQL*String sql1 = **"update hero set hp = hp +1 where id = 22"**;  
s.execute(sql1);  
*// 减血的SQL  
// 不小心写错写成了 updata(而非update)*String sql2 = **"updata hero set hp = hp -1 where id = 22"**;  
s.execute(sql2);  
*// 手动提交*c.commit();

事务练习：设计一个代码，删除表中前10条数据，但是删除前会在控制台弹出一个提示：  
是否要删除数据(Y/N)  
如果用户输入Y，则删除  
如果输入N则不删除。  
如果输入的既不是Y也不是N，则重复提示

**代码：**

**import** java.sql.\*;  
**import** java.util.Scanner;  
  
**public class** TestJDBC {  
 **public static void** main(String[] args) {  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 **try** (Connection c = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**, **""**);  
 Statement st4Query = c.createStatement();  
 Statement st4Delete = c.createStatement();  
 Scanner s = **new** Scanner(System.***in***);)  
 {  
 c.setAutoCommit(**false**);  
 ResultSet rs = st4Query.executeQuery(**"select id from hero order by id asc limit 10,10"**);  
 **while**(rs.next()){  
 **int** id = rs.getInt(1);  
 System.***out***.println(**"试图删除id="** + id + **" 的数据"**);  
 st4Delete.execute(**"delete from hero where id = "** + id);  
 }  
  
 **while**(**true**){  
 System.***out***.println(**"是否要删除数据（Y/N）"**);  
 String str = s.next();  
 **if**(str.equals(**"Y"**)){  
 c.commit();  
 System.***out***.println(**"提交删除"**);  
 **break**;  
 }**else if**(str.equals(**"N"**)){  
 System.***out***.println(**"放弃删除"**);  
 **break**;  
 }  
 }  
 } **catch** (SQLException e) {  
 *//* ***TODO Auto-generated catch block*** e.printStackTrace();  
 }  
 }  
}

1. ORM

ORM=Object Relationship Database Mapping 对象和关系数据库的映射

1. 根据id返回一个Hero对象

**public class** TestJDBC {  
 **public static** Hero get(**int** id){  
 Hero hero = **null**;  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 **try**(Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 Statement s = conn.createStatement();  
 ){  
 String sql = **"select** *\** **from hero where id = "** + id;  
 ResultSet rs = s.executeQuery(sql);  
 **if**(rs.next()){  
 hero = **new** Hero();  
 hero.**id** = id;  
 hero.**name** = rs.getString(2);  
 hero.**hp** = rs.getFloat(3);  
 hero.**damage** = rs.getInt(4);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** hero;  
 }  
 **public static void** main(String[] args) {  
 Scanner s = **new** Scanner(System.***in***);  
 **int** id = s.nextInt();  
 Hero h = *get*(id);  
 System.***out***.println(h.**name**);  
 }  
}

1. 练习ORM



**代码：**

**public** **static** List<Hero> list() {

        List<Hero> heros = **new** ArrayList<>();

**try** {

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

        } **catch** (ClassNotFoundException e) {

            e.printStackTrace();

        }

**try** (Connection c = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8",

                "root", "admin"); Statement s = c.createStatement();) {

            String sql = "select \* from hero";

            ResultSet rs = s.executeQuery(sql);

**while** (rs.next()) {

                Hero hero = **new** Hero();

**int** id = rs.getInt(1);

                String name = rs.getString(2);

**float** hp = rs.getFloat("hp");

**int** damage = rs.getInt(4);

                hero.name = name;

                hero.hp = hp;

                hero.damage = damage;

                hero.id = id;

                heros.add(hero);

            }

        } **catch** (SQLException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        }

**return** heros;

    }

1. DAO

DAO = DataAccessObject 数据访问对象 -> 把数据库相关的操作装在这个类里面，其它地方看不到JDBC的代码。

**DAO接口：**

**public interface** DAO {  
 **public void** add(Hero hero);  
 **public void** update(Hero hero);  
 **public void** delete(**int** id);  
 **public** Hero get(**int** id);  
 **public** List<Hero> list();  
 **public** List<Hero> list(**int** start, **int** count);  
}

**HeroDAO:**

改进：

1.把驱动的初始化放在了构造方法HeroDAO里：

因为驱动初始化只需要执行一次，使代码更简洁。

**public** HeroDAO() {  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
}

2.提供了一个getConnection方法返回连接

所有的数据库操作都需要事先拿到一个数据库连接Connection，以前的做法每个方法里都会写一个，如果要改动密码，那么每个地方都需要修改。 通过这种方式，只需要修改这一个地方就可以了。 代码变得更容易维护，而且也更加简洁。

**public** Connection getConnection() **throws** SQLException {  
**return**DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**, **"root"**,**""**);  
}

代码：

**package** jdbc;  
  
**import** charactor.Hero;  
  
**import** java.sql.\*;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** HeroDAO **implements** DAO{  
 **public** HeroDAO(){  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }  
 }  
 **public** Connection getConnection() **throws** SQLException {  
 **return** DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 }  
 **public int** getTotal(){  
 **int** total = 0;  
 **try**(Connection conn = getConnection(); Statement s = conn.createStatement();){  
 String sql = **"select** *count***(***\****) from hero"**;  
 ResultSet rs = s.executeQuery(sql);  
 **while**(rs.next()){  
 total = rs.getInt(1);  
 }  
 System.***out***.println(**"total:"** + total);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** total;  
 }  
  
 @Override  
 **public void** add(Hero hero) {  
 String sql = **"insert into hero values(null, ?, ?, ?)"**;  
 **try**(Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement(sql);){  
 ps.setString(1, hero.**name**);  
 ps.setFloat(2, hero.**hp**);  
 ps.setInt(3, hero.**damage**);  
 ps.execute();  
 ResultSet rs = ps.getGeneratedKeys();  
 **if**(rs.next())  
 hero.**id** = rs.getInt(1);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** update(Hero hero) {  
 String sql = **"update hero set name = ?, hp = ?, damage = ? where id = ?"**;  
 **try** (Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement(sql);){  
 ps.setString(1, hero.**name**);  
 ps.setFloat(2, hero.**hp**);  
 ps.setInt(3, hero.**damage**);  
 ps.setInt(1, hero.**id**);  
 ps.execute();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** delete(**int** id) {  
 **try**(Connection conn = getConnection(); Statement s = conn.createStatement();){  
 String sql = **"delete from hero where id = "** + id;  
 s.execute(sql);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public** Hero get(**int** id) {  
 Hero hero = **null**;  
 **try**(Connection conn = getConnection(); Statement s = conn.createStatement();){  
 String sql = **"select** *\** **from hero where id = "** + id;  
 ResultSet rs = s.executeQuery(sql);  
 **while**(rs.next()){  
 hero.**name** = rs.getString(2);  
 hero.**hp** = rs.getFloat(**"hp"**);  
 hero.**damage** = rs.getInt(4);  
 hero.**id** = id;  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** hero;  
 }  
  
 @Override  
 **public** List<Hero> list() {  
 **return** list(0, Short.***MAX\_VALUE***);  
 }  
  
 @Override  
 **public** List<Hero> list(**int** start, **int** count) {  
 List<Hero> heros = **new** ArrayList<>();  
 String sql = **"select** *\** **from hero order by id desc limit ?, ?"**;  
 **try** (Connection conn = getConnection(); PreparedStatement ps = conn.prepareStatement(sql);){  
 ps.setInt(1, start);  
 ps.setInt(2, count);  
 ResultSet rs = ps.executeQuery();  
  
 **while**(rs.next()){  
 Hero h = **new** Hero();  
 h.**id** = rs.getInt(1);  
 h.**name** = rs.getString(2);  
 h.**hp** = rs.getFloat(3);  
 h.**damage** = rs.getInt(4);  
 heros.add(h);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** heros;  
 }  
}

TestJDBC:

**public class** TestJDBC {  
 **public static void** main(String[] args) {  
 DAO dao = **new** HeroDAO();  
 List<Hero> l = dao.list(0, 5);  
 **for**(Hero h : l){  
 System.***out***.println(h.**id** + **" "** + h.**name** + **" "** + h.**hp** + **" "** + h.**damage**);  
 }  
 }  
}

1. 数据库连接池

原理：创建关闭连接需要时间，（多线程）且连接数有限，消耗光后续线程发起的数据库连接就会失效，因此就会使用数据库连接池。

连接池在使用之前，就会创建好一定数量的连接。如果有任何线程需要使用连接，那么就从连接池中借用，而不是自己重新创建。使用完，则会把这个连接归还给线程池供下一次或其它线程使用。倘若发生多线程并发情况，连接池中的连接被借用光了，那么其它线程就会临时等待，直到有连接被归还回来，再继续使用。整个过程，这些连接都不会被关闭，而是被不断循环利用，从而节约了启动和关闭的时间。

**package** jdbc;  
*//ConnectionPool***import** java.util.\*;  
**import** java.sql.\*;  
  
**public class** ConnectionPool {  
 List<Connection> **conns** = **new** ArrayList<>();  
 **int size**;  
 **public** ConnectionPool(){  
 **size** = 5;  
 init();  
 }  
 **public** ConnectionPool(**int** size){  
 **this**.**size** = size;  
 init();  
 }  
 **public void** init(){  
 **try**{ *//这里不能使用try-with-resource的方式,因为这些连接都需要是“活”的,不要被自动关闭了.* Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 **for**(**int** i = 0; i < **size**; ++ i){  
 Connection conn = DriverManager.*getConnection*(**"jdbc:mysql://127.0.0.1:3306/how2java?characterEncoding=UTF-8"**,**"root"**,**""**);  
 **conns**.add(conn);  
 }  
 }**catch** (ClassNotFoundException e){  
 e.printStackTrace();  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 }  
 **public synchronized** Connection getConnection(){  
 **while**(**conns**.isEmpty()){  
 **try** {  
 **this**.wait();  
 }**catch** (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 Connection conn = **conns**.remove(0);  
 **return** conn;  
 }  
 **public synchronized void** returnConnection(Connection c){  
 **conns**.add(c);  
 **this**.notifyAll();  
 }  
}

**package** jdbc;  
*//测试类***import** java.sql.Connection;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
**public class** TestConnectionPool {  
 **public static void** main(String[] args){  
 ConnectionPool cp = **new** ConnectionPool(3);  
 **for**(**int** i = 0; i < 100; ++ i){  
 **new** WorkingThread(**"working thread"** + i, cp).start();  
 }  
 }  
}  
  
**class** WorkingThread **extends** Thread{  
 **private** ConnectionPool **cp**;  
 **public** WorkingThread(String name, ConnectionPool cp){  
 **super**(name);  
 **this**.**cp** = cp;  
 }  
 **public void** run(){  
 Connection conn = **cp**.getConnection();  
 System.***out***.println(**this**.getName() + **":/t 获取了一根连接，并开始工作"**);  
 **try**(Statement s = conn.createStatement()){  
 Thread.*sleep*(1000);  
 s.execute(**"select** *\** **from hero"**);  
 }**catch** (SQLException | InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
}