JDBC 课堂笔记

今日内容介绍

* 1.jdbc介绍
* 2.快速入门
* 3.jdbc api详解(重点)
* 4.使用dao模式登录操作

1. sql注入
2. PreparedStatement(重点)

* 5.jdbc处理大数据(text, blob)
* 6.jdbc批处理

复习：

1. 什么是外键 ？

为什么需要外键 ？

1. 什么是外键约束 ？
2. 一对一， 一对多， 多对多 的概念 ？
3. 内连接， 有 两种方式使用：

什么 是 内连接 ？

多表查询 ， 2 表查询， 3 个，4 个表 。

标准方式： select \* from A inner join B on A.dept\_id = B.id

隐含方式（使用比较多）： select \* from A, B where A.dept\_id = B.id

Select \* from A,B,C where A.d\_id = B.id and B.id = C.cno

查询结果特点： 结果在 A 表里面 有记录，在 B表里面 有记录， 结果是 2 个表的交集（

1. 外连接( 概念） 使用不多

左连接： select \* from A left outer join B on A.dept\_id = B.id

右连接： select \* from A right outer join B on A.dept\_id = B.id

1. 子查询 ( 面试题比较多）

什么是子查询 ？

Select \* from A where id > any[all] ( select \* from B where id = 100)

1. JDBC 是 什么 ？
2. JDBC 是 sun 公司定义的一系列规范 ？
3. 其他

复习：

1. ResultSet 内容比较多

resultSet 是 什么 ？

查询结果

resultSet 如何遍历 ？

While(rs.next()) {

}

resultSet， 如何取值 ， 2 种方式 ？

getString, getInt:

getString(“columnname”)

getString(index\_id) (1 --- n)

1. 关闭资源

在jdbc， connection 是 一个 非常耗资源的对象 ， 数据库连接池。

1. JdbcUtils 类 （基本java 代码）

我们 大部分使用 静态方法， utils 类里面 大部分都是静态方法

static { } , 是什么 ？ 有 什么作用 ？

Properties 类： map， Hashtable

ResourceBundle : 能干什么 ？

方法的重载 ，

1. CRUD

Statement.

executeQuery(), executeUpdate(), boolean execute()

1. 滚动结果集（了解)

ResultSet,

1. DAO 模式（ 重点， 面试题）

持久层。

1. Sql注入 ？

同 java 没什么 关系 ？只要是 sql语句进行拼串 的方式就存在 风险 。

1. preparedStatement： 准备， 预编译的 语句。

Select \* from user where id = ? , 数据库 执行计划（数据库的机器语言）

Conn.prepraredStatement(sql);

占位符号如何赋值 ？

Pstmt.setString(1, “tiger”);

Pstmt.executeQuery()

1. 其他

----------------------------------------------

# jdbc介绍

## 问题:jdbc是什么?

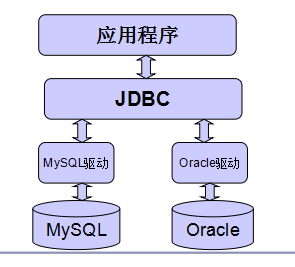
JDBC（Java Data Base Connectivity,java数据库连接)

简单说:就是可以直接通过java语言，去操作数据库。

API: Application Programming Interface , 应用编程接口。

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用[Java语言](https://baike.baidu.com/item/Java%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/jdbc/_blank)编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，

jdbc是一套标准,它是由一些接口（sun）与类（数据库厂家实现）组成的。



学习中涉及到的类与接口

它们主要在两个包下

java.sql

类:DriverManger

接口 Connection Statement ResultSet PreparedStatement

CallableStatement（它是用于调用存储过程）

javax.sql

接口 DataSource（数据源、 就是 数据库连接池）

## 什么是驱动? （Driver）

驱动程序即添加到[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/192" \t "https://baike.baidu.com/item/%E9%A9%B1%E5%8A%A8/_blank)中的一小块[代码](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81/86048" \t "https://baike.baidu.com/item/%E9%A9%B1%E5%8A%A8/_blank)，其中包含有关硬件设备的信息。有了此信息，计算机就可以与设备进行通信。驱动程序是硬件厂商根据操作系统编写的配置文件，可以说没有驱动程序，计算机中的硬件就无法工作。

-------------------------------------------------------------------

# jdbc入门

编写一个jdbc入门代码，完成对数据库操作.

create table user(

id int primary key auto\_increment,

username varchar(20) unique not null,

password varchar(20) not null,

email varchar(40) not null

);

INSERT INTO user VALUES(NULL,'tom','123','tom@163.com');

INSERT INTO user VALUES(NULL,'fox','123','fox@163.com');

1.下载驱动

将驱动jar包复制到lib下.

1. 创建一个JdbcDemo1类



// 1.注册驱动

DriverManager.registerDriver(new Driver());

// 2.获取连接对象

Connection con = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/day17", "root", "abc");

// 3.通过连接对象获取操作sql语句Statement

Statement st = con.createStatement();

// 4.操作sql语句

String sql = "select \* from user";

// 操作sql语句(select语句),会得到一个ResultSet结果集

ResultSet rs = st.executeQuery(sql);

// 5.遍历结果集

// boolean flag = rs.next(); // 向下移动，返回值为true，代表有下一条记录.

// int id = rs.getInt("id");

// String username=rs.getString("username");

// System.out.println(id);

// System.out.println(username);

while(rs.next()){

int id=rs.getInt("id");

String username=rs.getString("username");

String password=rs.getString("password");

String email=rs.getString("email");

System.out.println(id+" "+username+" "+password+" "+email);

}

//6.释放资源

rs.close();

st.close();

con.close();

----------------------------------------------------------------------

# jdbc操作详解

## 注册驱动

DriverManager.registDriver(new Driver());

1.DriverManager类

它是java.sql包下的一个驱动管理的工具类,可以理解成是一个容器(Vector),可以装入很多数据库驱动

它的registDriver方法分析

public static synchronized void registerDriver(java.sql.Driver driver)

参数:java.sql.Driver

我们传递的是 com.mysql.jdbc.Driver;

在com.mysql.jdbc.Driver类中有一段静态代码块:

static {

try {

java.sql.DriverManager.registerDriver(new Driver());

} catch (SQLException E) {

throw new RuntimeException("Can't register driver!");

}

}

上述代码的问题:

1.在驱动管理器中会装入两个mysql驱动.

解决方案:使用反射

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

分析:使用反射的方式来加载驱动有什么好处?

1.只加载一次，装入一个驱动对象.

2.降低耦合，不依赖于驱动.

2.可以通过DriverManager来获取连接对象

Connection con=DriverManager.getConection(String url,String user,String password);

url作用:就是用于确定使用哪一个驱动.

mysql url: jdbc:mysql://localhsot:3306/数据库名.

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

总结:DriverManager作用:

1.注册驱动

2.获取连接Connection

-----------------------------------------------------------

关于url

url格式

主协议 子协议 主机 端口 数据库

jdbc : mysql ://localhost:3306/day17

mysql的url可以简写:

前提：主机是localhost 端口是3306

jdbc:mysql:///day17

了解:在url后面可以带参数

useUnicode=true&characterEncoding=UTF-8

------------------------------------------------------------------

## Connection详解

java.sql.Connection，它代表的是一个连接对象。简单说，就是我们程序与数据库连接。

Connection作用:

1.可以通过Connection获取操作sql的Statement对象。

Statement createStatement() throws SQLException

示例:

Statement st=con.createStatement();

了解:

1.可以获取执行预处理的PreparedStatement对象.

PreparedStatement prepareStatement(String sql) throws SQLException

2.可以获取执行存储过程的 CallableStatement

CallableStatement prepareCall(String sql) throws SQLException

2.操作事务

setAutoCommit(boolean flag);开启事务

rollback();事务回滚

commit();事务提交

------------------------------------------------------------------

## Statement详解

java.sql.Statement用于执行sql语句.

Statement作用:

1.执行sql

DML:insert update delete

int executeUpdate(String sql)

利用返回值判断非0来确定sql语句是否执行成功。

DQL:select

ResultSet executeQuery(String sql)

可以通过execute方法来执行任何sql语句.

execute(String sql)：用于向数据库发送任意sql语句

2.批处理操作

addBatch(String sql); 将sql语句添加到批处理

executeBatch();批量执行

clearBatch();清空批处理.

---------------------------------------------------------------------

## ResultSet详解

java.sql.ResultSet它是用于封装select语句执行后查询的结果。

常用API

1.next()方法

public boolean next();

用于判断是否有下一条记录。如果有返回true,并且让游标向下移动一行。

如果没有返回false.

2.可以通过ResultSet提供的getXxx()方法来获取当前游标指向的这条记录中的列数据。

常用:

getInt()

getString()

getDate()

getDouble()

参数有两种

1.getInt(int columnIndex);

2.getInt(String columnName);

如果列的类型不知道，可以通过下面的方法来操作

getObject(int columnIndex);

getObject(String columnName);

----------------------------------------------------------------

## 关闭资源

Jdbc程序运行完后，切记要释放程序在运行过程中，创建的那些与数据库进行交互的对象，这些对象通常是ResultSet, Statement和Connection对象。

特别是Connection对象，它是非常稀有的资源，用完后必须马上释放，如果Connection不能及时、正确的关闭，极易导致系统宕机。Connection的使用原则是尽量晚创建，尽量早的释放。

为确保资源释放代码能运行，资源释放代码也一定要放在finally语句中。

=================================================================================

# jdbc示例----CURD（CRUD）

1.查询

1.查询全部

2.条件查询---根据id

2.修改

3.删除

4.添加

关于JdbcUtils抽取:

只抽取到Connection。

public static Connection getConnection() throws ClassNotFoundException,

SQLException {

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

// 2.获取连接

Connection con = DriverManager.getConnection("jdbc:mysql:///day17",

"root", "abc");

return con;

}

关于上述的抽取JdbcUtils的缺点:

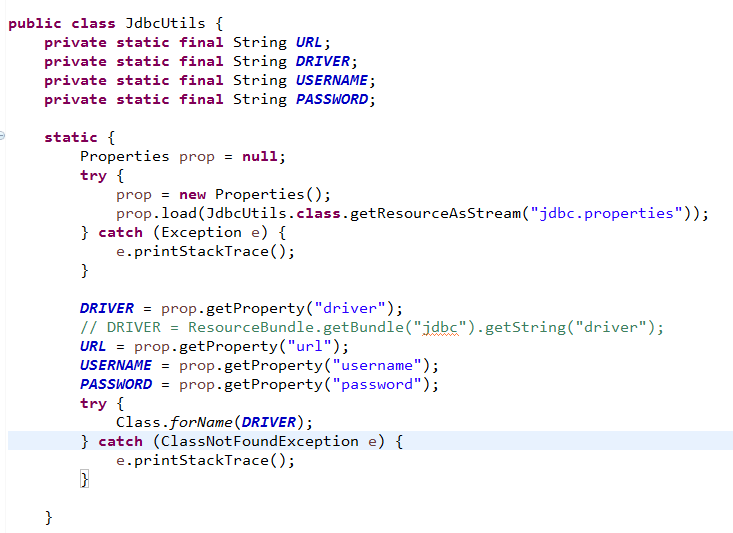
1.它只能针对于mysql数据库。

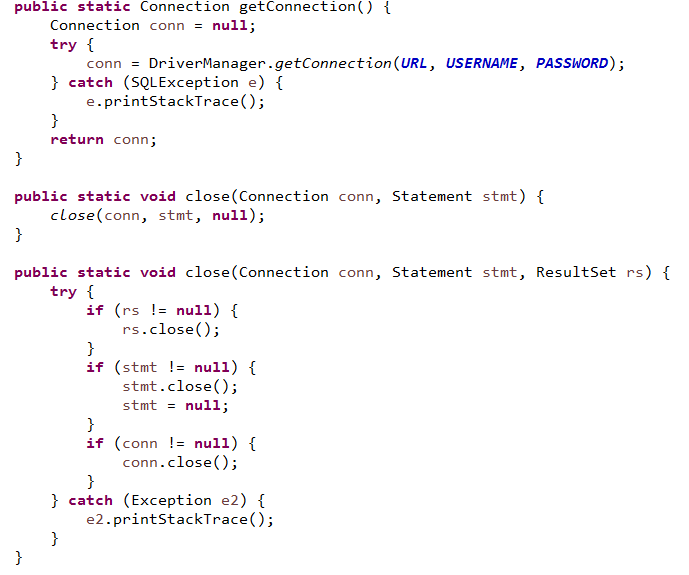
2.每一次调用，都会注册一次驱动.

对于上述问题，对JdbcUtils进行修改:

1.将关于连接数据库的信息定义到配置文件中。

读取配置文件进行加载.





public class JdbcUtils {

private static final String DRIVERCLASS;

private static final String URL;

private static final String USERNAME;

private static final String PASSWORD;

static {

DRIVERCLASS = ResourceBundle.getBundle("jdbc").getString("driverClass");

URL = ResourceBundle.getBundle("jdbc").getString("url");

USERNAME = ResourceBundle.getBundle("jdbc").getString("username");

PASSWORD = ResourceBundle.getBundle("jdbc").getString("password");

}

static {

try {

// 将加载驱动操作，放置在静态代码块中.这样就保证了只加载一次.

Class.forName(DRIVERCLASS);

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

public static Connection getConnection() throws SQLException {

// 2.获取连接

Connection con = DriverManager.getConnection(URL, USERNAME, PASSWORD);

return con;

}

}

=============================================================================

# 滚动结果集:(了解)

默认得到的ResultSet它只能向下遍历(next()),对于ResultSet它可以设置成是滚动的，可以向上遍历，

或者直接定位到一个指定的物理行号.

问题:怎样得到一个滚动结果集?

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(sql);

这是一个默认结果集:只能向下执行，并且只能迭代一次。

Statement stmt = con.createStatement(

ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery(sql);

这个就可以创建滚动结果集.

简单说，就是在创建Statement对象时，不使用createStatement();

而使用带参数的createStatement(int,int)

Statement createStatement(int resultSetType,

int resultSetConcurrency)

throws SQLException

resultSetType - 结果集类型，它是 ResultSet.TYPE\_FORWARD\_ONLY、ResultSet.TYPE\_SCROLL\_INSENSITIVE 或 ResultSet.TYPE\_SCROLL\_SENSITIVE 之一

resultSetConcurrency - 并发类型；它是 ResultSet.CONCUR\_READ\_ONLY 或 ResultSet.CONCUR\_UPDATABLE 之一

第一个参数值

ResultSet.TYPE\_FORWARD\_ONLY 该常量指示光标只能向前移动的 ResultSet 对象的类型。

ResultSet.TYPE\_SCROLL\_INSENSITIVE 该常量指示可滚动但通常不受 ResultSet 底层数据更改影响的 ResultSet 对象的类型。

ResultSet.TYPE\_SCROLL\_SENSITIVE 该常量指示可滚动并且通常受 ResultSet 底层数据更改影响的 ResultSet 对象的类型。

第二个参数值

ResultSet.CONCUR\_READ\_ONLY 该常量指示不可以更新的 ResultSet 对象的并发模式。

ResultSet.CONCUR\_UPDATABLE 该常量指示可以更新的 ResultSet 对象的并发模式。

以上五个值，可以有三种搭配方式

ResultSet.TYPE\_FORWARD\_ONLY ResultSet.CONCUR\_READ\_ONLY 默认

ResultSet.TYPE\_SCROLL\_INSENSITIVE ResultSet.CONCUR\_READ\_ONLY

ResultSet.TYPE\_SCROLL\_SENSITIVE ResultSet.CONCUR\_UPDATABLE

常用API

next()：移动到下一行

previous()：移动到前一行

absolute(int row)：移动到指定行

beforeFirst()：移动resultSet的最前面

afterLast() ：移动到resultSet的最后面

updateRow() ：更新行数据



==================================================================================

# DAO模式

DAO模式（Data Access Object 数据访问对象）：在持久层通过DAO将数据源操作完全封装起来，业务层通过操作Java对象，完成对数据源操作

\* 业务层无需知道数据源底层实现 ，通过java对象操作数据源

DAO模式结构 ：

1、数据源（MySQL数据库）

2、Business Object 业务层代码，调用DAO完成 对数据源操作 UserService

3、DataAccessObject 数据访问对象，持久层DAO程序，封装对数据源增删改查，提供方法参数都是Java对象

4、TransferObject 传输对象（值对象） 业务层通过向数据层传递 TO对象，完成对数据源的增删改查



使用dao模式完成登录操作

1.web层

login.jsp LoginServlet User

2.service层

UserService

3.dao层

UserDao

-------------------------

# sql注入

由于没有对用户输入进行充分检查，而SQL又是拼接而成，在用户输入参数时，在参数中添加一些SQL 关键字，

达到改变SQL运行结果的目的，也可以完成恶意攻击。

示例:

在输入用户名时 tom' or '1'='1

这时就不会验证密码了。

解决方案:



# PreparedStatement(重点)

它是一个预处理的Statement，它是java.sql.Statement接口的一个子接口。

## PreparedStatement使用:

1.在sql语句中，使用"?"占位

String sql="select \* from user where username=? and password=?";

2.得到PreparedStatement对象

PreparedStatement pst=con.prepareStatement(String sql);

3.对占位符赋值

pst.setXxx(int index,Xxx obj);

例如:

setInt()

setString();

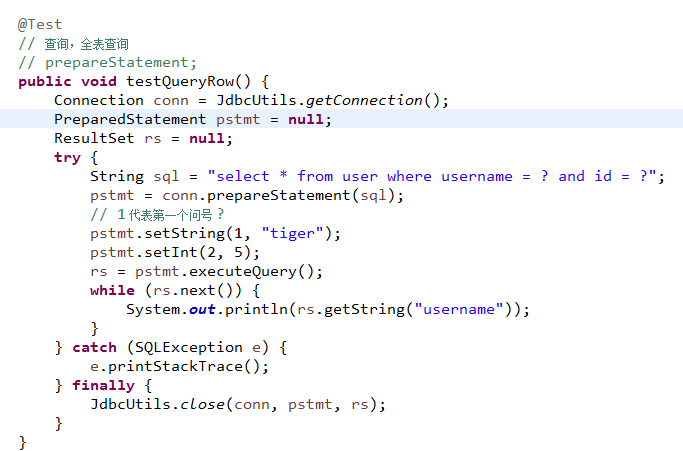
参数index,代表的是"?"的序号.注意：从1开始。

4.执行sql

DML： pst.executeUpdate();

DQL: pst.executeQuery();

注意：这两方法无参数



## 关于PreparedStatement优点:

1.解决sql注入(具有预处理功能)

2.不需要在拼sql语句。

3、 执行效率高， 因为sql 语句 是预编译 。



---------------------------------------------------------------------

# jdbc处理大数据 (了解 )

mysql中有大数据

blob 大二进制

TINYBLOB(255)、BLOB(64kb)、MEDIUMBLOB(16m)和LONGBLOB(4g)

text(clob) 大文本

TINYTEXT(255)、TEXT(64kb)、MEDIUMTEXT(16m)和LONGTEXT(4g)

对于大数据操作，我们一般只有两种 insert select

演示1:

大二进制操作

create table myblob(

id int primary key auto\_increment,

content longblob

)

向表中插入数据

问题1:java.lang.AbstractMethodError: com.mysql.jdbc.PreparedStatement.setBinaryStream(ILjava/io/InputStream;)V

原因:mysql驱动不支持setBinaryStream(int,InputStream);

setBinaryStream(int, InputStream) 可以了 。

修改成

pst.setBinaryStream(1, fis,file.length());

原因:因为mysql驱动不支持setBinaryStream(int,InputStream,long);

解决:

mysql驱动支持setBinaryStream(int,InputStream,int);

注意：如果文件比较大，那么需要在my.ini文件中配置

max\_allowed\_packet=64M

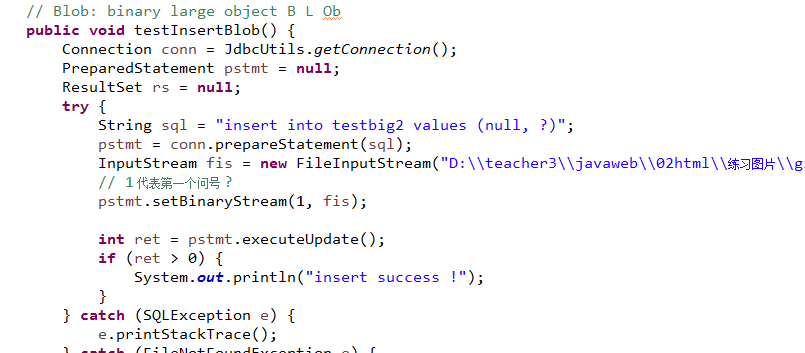
总结:

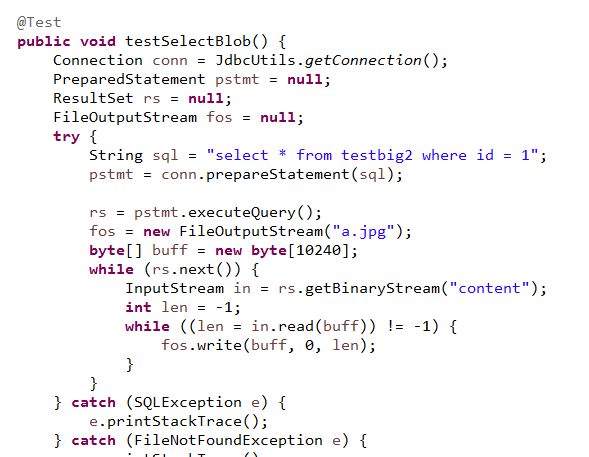
存

pst.setBinaryStream(1, fis, (int) (file.length()));

取

InputStream is = rs.getBinaryStream("content");





--------------------------------------------------------------

演示:存储大文本

create table mytext(

id int primary key auto\_increment,

content longtext

)

存储

File file = new File("D:\\java1110\\workspace\\day17\_3\\a.txt");

FileReader fr = new FileReader(file);

pst.setCharacterStream(1, fr, (int) (file.length()));

获取:

Reader r = rs.getCharacterStream("content");

----------------------------------------------------------------------------------

# jdbc批处理

一次可以执行多条sql语句.

在jdbc中可以执行sql语句的对象有Statement,PreparedStatement，它们都提供批处理.

## 1.Statement执行批处理

addBatch(String sql); 将sql语句添加到批处理

executeBatch(); 执行批处理

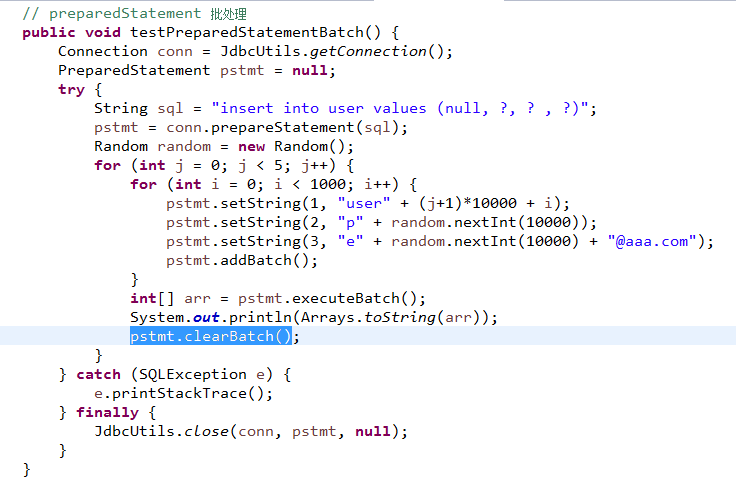
clearBatch();

## 2.PreparedStatement执行批处理

addBatch();

executeBatch();

clearBatch();



以上两个对象执行批处理区别?

1.Statement它更适合执行不同sql的批处理。它没有提供预处理功能，性能比较低。

2.PreparedStatement它适合执行相同sql的批处理，它提供了预处理功能，性能比较高。

注意;mysql默认情况下，批处理中的预处理功能没有开启，需要开启

1.在 url下添加参数

url=jdbc:mysql:///day17?useServerPrepStmts=true&cachePrepStmts=true&rewriteBatchedStatements=true

2.注意驱动版本

Mysql驱动要使用mysql-connector-java-5.1.13以上

## 练习：

插入 20 万条不同的数据 ， 同大家展示一下 索引的功能和作用 。 30W， sql

---------------------------------------------------------------------

# 作业:

1.注册+登录案例

采用dao模式 使用PreparedStatement操作

1. 使用PreparedStatement完成CRUD

# 作业2

