1.概念

JDBC(Java DataBase Connectivity)：可以为多种关系型数据库DBMS提供同一的访问方式，用Java来操作数据库

2.JDBC API主要功能：

i.连接数据库

ii.发送SQL语句

iii.返回结果集

具体是通过一下类/接口实现

DriverMananger：管理jsbc驱动

Connection：连接（通过DriverMananger产生）

Statement（PreparedStatement）：增删改查（通过Connection产生）

CallableStatement：调用数据库中的存储过程/存储函数（通过Connection产生）

ResultSet：返回的结果集（通过Statement或者CallableStatement产生）

1.

Connection产生操作数据库的对象

Connection产生Statement对象：createStatement()

Connection产生PreparedStatement对象：prepareStatement()

Connection产生CallableStatement对象：prepareCall();

Statement操作数据库

增删改：executeUpdate()

查询：executeQuery()

ResultSet：保存结果集 select * from xxx

next():光标下移，判断是否有下一条数据；true/false

previous()：光标上移；true/false

getXxx(字段名/位置)：获取具体的字段值

PreparedStatement操作数据库

public interface PreparedStatement extends Statement

因此

增删改：executeUpdate()

查询：executeQuery()

赋值操作：setXxx();

PreparedStatement和 Statement在使用时的区别：

a.Statement：

sql

executeUpdate(sql)

b.PreparedStatement

sql(可能存在占位符问号)

在创建PreparedStatement对象的同时，将sql预编译prepareStatement(sql)

executeUpdate()

setXxx()替换占位符?

推荐用PreparedStatement：原因如下：

a.编程更加简单(避免了字符串的拼接)

String name = "zs"

int age=23

stmt:

String sql = "insert into student(stuno,stuname) values(' "+name+"','"+age+"') ";//字符串拼接

stmt.executeUpdate(sql);

pstmt

String sql = "insert into student(stuno,stuname) values(?,?)";

pstmt = connection.prepareStatement(sql);//预编译SQL

pstmt.setString(1,name);

pstmt.setInt(2,age);

pstmt.executeUpdate();

b.提高性能(因为预编译操作)

需要重复增加100条数

stmt:

String sql = "insert into student(stuno,stuname) values(' "+name+"','"+age+"') ";//字符串拼接

for(100)

stmt.executeUpdate(sql);

pstmt

String sql = "insert into student(stuno,stuname) values(?,?)";

pstmt = connection.prepareStatement(sql);//预编译SQL

pstmt.setString(1,name);

pstmt.setInt(2,age);

for(100)

pstmt.executeUpdate();

3.安全（可以有效的防止sql的注入）

stmt：存在被sql注入的风险

sql注入：将客户输入的内容和开发人员的SQL语句混为一体

（如输入：用户名：任意值 ’ or 1=1 -- //--后面一定要有空格

账号：任意值）

分析：select count(*) from login where uname='" + name + "' and upwd='" + pwd + "'

将字符带入：

select count(*) from login where uname='任意值 ’ or 1=1 -- ’ and upwd='任意值'

//oracle通过--注释

pstmt：有效防止sql注入

3.jdbc访问数据库的具体步骤

a.导入驱动，加载具体的驱动类

b.与数据库建立连接

c.发送SQL，执行

d.处理结果集（查询）

4.数据库驱动

驱动jar

具体的驱动类

Oracle ojdbc-x.jar oracle.jdbc.OracleDriver

MySql mysql-connector-java-x.jar com.mysql.jdbc.Driver

SqlServrt sql jdcb-x.jar com.microsoft.sqlserver.jdbc.SQLServerDriver

//-x是指版本号

1连接字符串

Oracle jdbc:oracle:thin:@localhost:1521:ORCL

MySql jdbc:mysql://localhost:3306/数据库实例名

SqlServrt

jdbc:microsoft:sqlserver:localhost:1433;databasename=数据库实例名

1.jdbc总结（模板）

```
try{
a.导入驱动包，加载具体驱动类Class.forName("具体驱动类");
b.与数据库建立连接connection=DriverManager.getConnection(...);
c.通过connection，湖区操作数据库对象
（Statement\prepareStatement\callableStatement）
    stmt=connection.createStatement();
d.(查询)处理结果集rs=pstmt.executeQuery()
while(rs.next()){rs.getXxx(...);}
catch(ClassNotFoundException e){..}
catch(SQLException e){...}
catch(Exception e){..}
finally{
//打开顺序与关闭顺序相反;
if(rs!=null) rs.close();
if(stmt!=null) stmt.close();
if(connnection!=null) connnection.close();
}
}
```

2.CallableStatement:调用存储过程、存储函数

connection.prepareCall(存储过程/存储函数名)

参数格式：

存储过程：（没有返回值，用out参数代替返回值）

{call 存储过程名（参数列表）}

存储函数：（有返回值return）

{ ? = call 存储函数名（参数列表）}//赋值语句是:等于

```
create or replace procedure addTwoNum(num1 in number,num2 in number,result out
number)
```

```
as
begin
    result := num1+num2;
    end;
    /
```

以 / 结束

强调：

如果通过sqlplus访问数据库，只需要开启：OracleServiceSID

通过其他程序访问数据库（sql deveolper、navicate、JDBC），需要开启 OracleServiceSID、XxxListner

JDBC调用存储过程的步骤

a. cstmt=connection.prepareCall("....");

b.通过setXxx()处理输入参数 cstmt.setInt(1, 10);

c.通过registerOutParameter(...)处理输出对象的类型

d.cstmt.execute();执行存储函数

e.接收输出值（返回值）getXxx()

调用存储函数/

```
create or replace function addTwoNumfunction(num1 in number,num2 in number)
return number
as
result number; --在as和begin之间定义返回值的类型，注意标点符号
begin
    result := num1+num2;
    return result;
    end;
    /
```

JDBC调用存储函数

调用时注意参数 cstmt=connection.prepareCall("{? = call addTwoNumfunction(?,?)}");

3.处理CLOB/BLOB类型 / /在Oracle里

处理稍大型数据

ａ．存储路径：通过JDBC存储文件路径，然后根据IO操作处理，以字符串的形式存储到数据库中

获取时再用IO解析／／但受到路径的限制，路径必须存在

ｂ．

CLOB：大文本数据

varchar2():最多可以放4000个字节

BLOB：二进制（一切文件）／／任何电脑都可以，但笨拙

clob　:通过字符流，Reader　Writer

存：

1.先通过pstmt的占位符？来代替小说内容

2.再通过pstmt.setCharacterStream(2, reader,(int)file.length());将上一步的？替换为小说流，注意

第三个是int类型

取：

1.通过Reader reader = rs.getCharacterStream("NOVEL"); 将clob类型的数据保存到reader对象中

2.将Reader通过Writer输出即可。

blob:二进制：字节流 InputStream　OutStream//会有提示

与clob步骤基本一致　getBinaryStream(".....")　　setBinaryStream(".......");

4.JSP访问数据库

JSP就是在HTML中嵌套的Java代码，因此Java代码可以写在jsp中（<%　....%>）

导包操作：

Java项目：1.将jar包复制到工程中 2.右键该jar包：build path->add to build path

WEB项目：直接将jar包复制到WEB-INF/lib

核心：就是将Java中的JDBC代码复制到JSP中的<%.....%>

注意：如果JSP出现错误：The import Xxx cannot be resolved..

尝试解决步骤;

a.(可能jdk和tomcat的版本问题) 右键项目->build path,将其报错的libary或者lib删除后重新导入

b.清空各种缓存：右键项目——>Clean tomcat...　clean (Project -clean 或者进tomcat目录删除里面的文件)

c.重启电脑

d.如果类之前没有包，则将该类加入包中

5.JavaBean

刚才我们将jsp中的登陆操作代码转移到了LoginDao.java，其中LoginDao类就称之为JavaBean

作用：a.减轻了jsp的复杂度 b.提高代码复用率（以后任何地方的登陆操作，都可以通过调用LonginDao实现）

定义：就是一个java类，满足以下两点可以称之为Javabean

a.public修饰的类，public无参构造

b.所有属性都是private，并且提供 set/get （如果boonlean则get可以替换成is）

使用层面，JavaBean分为两大类

a.封装业务逻辑的JavaBean（LoginDao.java封装了登陆逻辑） 逻辑

可以将JSP中的JDBC代码，封装到Login.java 类中

b.封装数据的JavaBean（实体类，Student.java Person.java ） 数据

对应于数据库中的一张表

Login login = new Login(name,pwd);//即用Login对象封装了2个数据（用户名和密码）

封装数据的JavaBean对应于数据库中的一张表

封装业务逻辑的JavaBean用于操作一个封装数据的JavaBean

可以发现JavaBean可以简化代码（JSP->JSP+JAVA），提高代码复用(Login.java)

```
public void sleep(String name,String place,int time)
{.........}
public void sleep(Person per)
{       per.getName()
per.getPlace()          }
```