JDBC第一天　　2007年6月4日

一、JDBC 原理概述

1，JDBC 是一套协议，是JAVA开发人员和数据库厂商达成的协议，也就是由 Sun 定义一组接口，由数据库厂商来实现，

并规定了 JAVA开发人员访问数据库所使用的方法的掉用规范。

2，JDBC 的实现是由数据库厂商提供，以驱动程序形式提供。

3，JDBC 在使用前要先加载驱动。

JDBC 对于使用者要有一致性，对不同的数据库其使用方法都是相同的。

4、driver 开发必须要实现 Driver 接口。

JDBC 驱动程序的类型

目前比较常见的 JDBC 驱动程序可分为以下四个种类：

（1）JDBC-ODBC 桥加 ODBC 驱动程序

JavaSoft 桥产品利用 ODBC 驱动程序提供 JDBC 访问。

注意，必须将 ODBC 二进制代码（许多情况下还包括数据库客户机代码）加载到使用该驱动程序的每个客户机上。

因此，这种类型的驱动程序最适合于企业网（这种网络上客户机的安装不是主要问题），或者是用 Java 编写的三层结构的应用程序服务器代码。

（2）本地 API

这种类型的驱动程序把客户机 API 上的 JDBC 调用转换为 Oracle、Sybase、Informix、DB2 或其它 DBMS 的调用。

注意，象桥驱动程序一样，这种类型的驱动程序要求将某些二进制代码加载到每台客户机上。

（3）JDBC 网络纯 Java 驱动程序

这种驱动程序将 JDBC 转换为与 DBMS 无关的网络协议，之后这种协议又被某个服务器转换为一种 DBMS 协议。

这种网络服务器中间件能够将它的纯 Java 客户机连接到多种不同的数据库上。所用的具体协议取决于提供者。

通常，这是最为灵活的 JDBC 驱动程序。有可能所有这种解决方案的提供者都提供适合于 Intranet 用的产品。

为了使这些产品也支持 Internet 访问，它们必须处理 Web 所提出的安全性、通过防火墙的访问等方面的额外要求。

几家提供者正将 JDBC 驱动程序加到他们现有的数据库中间件产品中。

（4）本地协议纯 Java 驱动程序

这种类型的驱动程序将 JDBC 调用直接转换为 DBMS 所使用的网络协议。

这将允许从客户机机器上直接调用 DBMS 服务器，是Intranet 访问的一个很实用的解决方法。

由于许多这样的协议都是专用的，因此数据库提供者自己将是主要来源，有几家提供者已在着手做这件事了。

据专家预计第（3）、（4）类驱动程序将成为从 JDBC 访问数据库的首方法。

第（1）、（2）类驱动程序在直接的纯 Java 驱动程序还没有上市前会作为过渡方案来使用。

对第（1）、（2）类驱动程序可能会有一些变种，这些变种要求有连接器，但通常这些是更加不可取的解决方案。

第（3）、（4）类驱动程序提供了 Java 的所有优点，包括自动安装（例如，通过使用 JDBC 驱动程序的 appletapplet 来下载该驱动程序）。

5、JDBC 的 API

java.sql 包和 javax.sql 包
DriverManager 类（驱动管理器），它可以创建连接，它本身就是一个创建 Connection 的工厂(Factory)。
Driver 接口　入口
Connection 接口，会根据不同的驱动产生不同的连接
Statement 接口，发送 sql 语句
ResultSet 接口（结果集），是用来接收 select 语句返回的查寻结果的。其实质类似于集合。

以上的资源都需要释放，释放的是数据库的资源

JDBC 应用步骤
1，注册加载一个 driver 驱动
2，创建数据库连接（Connection）
3，创建一个 Statement（发送 sql）
4，执行 sql 语句
5，处理 sql 结果（select 语句）
6，关闭 Statement
7，关闭连接 Connection。

注意：6，7 两个步骤势必须要做的，因为这些资源是不会自动释放的，必须要自己关闭

访问 Oracle 的数据库的驱动名字叫 o 加到环境变量 PATH 中。
jdbc14.jar，这个 jar 文件中出访的驱动程序的.class 文件
要使用这个驱动程序，要先将他

一，注册加载驱动 driver，也就是强制类加载

1、Class.forName(driver);

driver = "oracle.jdbc.driver.OracleDriver";

2、Driver d=new Driver 类();

Driver d = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);

3、编译时利用虚拟机的系统属性

java -Djdbc.drivers=oracle.jdbc.driver.OracleDriver 类名（文件）

Oracle 的 Driver 的全名 oracle.jdbc.driver.OracleDriver

mysql 的 Driver 的全名 com.mysql.jdbc.Driver

SQLServer 的 Driver 的全名 com.microsoft.jdbc.sqlserver.SQLServerDriver

二，创建连接

DriverManager.getConnection(String url,String username,String password);

Connection 连接是通过 DriverManager 的静态方法 getConnection(.....)来得到的，这个方法的实质是把参数传到实际的 Driver 中的 connect()方法中来获得数据库连接的。

Oracle 的 URL 值是由连接数据库的协议和数据库的 IP 地址及端口号还有要连接的库名（DatebaseName）

Oracle URL 的格式

jdbc:oracle:thin:（协议）@XXX.XXX.X.XXX:XXXX（IP 地址及端口号）:XXXXXXX（所使用的库名）

例：jdbc:oracle:thin:@192.168.0.39:1521:TARENADB

MySql URL 的写法

例： jdbc:mysql://192.168.8.21:3306/test

SQLServer URL 的写法

例：jdbc:microsoft:sqlserver://192.168.8.21:1433

java -Djdbc.drivers=驱动全名 类名

使用系统属性名，加载驱动 -D 表示为系统属性赋值

使用 Connection 对象获得一个 Statement，Statement 中的 executeQuery(String sql) 方法可以使用 select 语句查询，并且返回一个结果集 ResultSet 通过遍历这个结果集，

可以获得 select 语句的查寻结果，ResultSet 的 next()方法会操作一个游标从第一条记录的前边开始读取，直到最后一条记录。

executeUpdate(String sql) 方法用于执行 DDL 和 DML 语句，可以 update，delete 操作。注意：要按先 ResultSet 结果集，后 Statement，最后 Connection 的顺序关闭资源，因为 Statement 和 ResultSet 是需要连接是才可以使用的，

所以在使用结束之后有可能起他的 Statement 还需要连接，所以不能现关闭 Connection。

作业：修改 StudentDao 的设计以及实现和测试程序，来完成从命令行传递学生的信息。

JDBC 第二天　　2007 年 6 月 5 日

一、提问
如何进行代码复用
　继承复用、组合复用
　私有复用：一个方法在一个类的内部使用
　工具方法：使用静态方法，使用类名直接调用

二、Statement
execute(sql); 当不知道执行的 SQL 语句是什么类型的时候执行 ，返回值是 boolean
executeQuery(sql); 执行查询语句
executeUpdate(sql); 执行更新语句

三、PreparedStatement
可以使用参数替代 sql 语句中的某些参数使用 "?"代替，他先将带参数的 sql 语句发送到数据库，进行编译，然后 PreparedStatement 会将参数发送给数据库。
在使用 PreparedStatement 时，在设置相应参数时，要指明参数的位置和类型，以及给出参数值
根据不同的参数类型使用不同的 setXXX(参数的位置，参数值)来设置参数

例：

```
public void insert(Student s){
        Connection con=ConnectionFactory.getConnection();//建立连接
        String sql="insert into student(id,name) values(?,?)";
        PreparedStatement ps=null;
        try {
            ps=con.prepareStatement(sql);//创建一个 PreparedStatement
      int index=1;
            ps.setInt(index++,s.getStuId());   //为参数赋值
            ps.setString(index++,s.getName());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally{
            if(ps!=null)
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
```

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

CallableStatement 是可以用非 sql 语句来访问数据库，他是通过调用存储过程（PL/SQL）来访问数据库的。可以直接使用连接来调用 prepareCall(...)方法，来执行这个存储过程，"..." 是存储过程的名字。

对于系统时间要去数据库时间
TimeStamp 和 Date 都可以保存时间
TimeStamp 可以保存时、分、秒的数据，Date 只保存日期年月的信息。

SQLException 是检查异常必须处理要么 throws ，要么 try{}catch(){}
getErrorCode()可以获得错误码，可以对错误进行查询。

四、源数据
JDBC 中有两种源数据,一种是数据库源数据,另一种是 ResultSet 源数据。

源数据就是描述存储用户数据的容器的数据结构。

```
ResultSet rs=ps.executeQuery(sql);
ResultSetMetaData m=rs.getMetaData();
```

getColumnCount(),获得实际列数
getColumnName(int colnum)，获得指定列的列名
getColumnType(int colnum)，获得指定列的数据类型
getColumnTypeName(int colnum)，获得指定列的数据类型名

```
//打印结果集
public static void printRS(ResultSet rs)throws SQLException{
    ResultSetMetaData rsmd = rs.getMetaData();
    while(rs.next()){
        for(int i = 1 ; i < = rsmd.getColumnCount() ; i++){
            String colName = rsmd.getColumnName(i);
            String colValue = rs.getString(i);
            if(i>1){
                System.out.print(",");
            }
```

```
                System.out.print(name+"="+value);
            }
            System.out.println();
        }
}
```

五、数据库源数据

DatabaseMetaData
getURL(),获得连接数据库的 URL
getDatabaseProductName() 获得数据库产品的名称
getDriverVersion() 获得 JDBC 驱动程序的 String 形式的版本号
getTables()获得数据库中该用户的所有表
getUserName() 获得数据库用户名。

六、事务（Transaction）

事务是针对原子操作的，要求原子操作不可再分，要求原子操作必须同时成功同时失败。
事务是捆绑的原子操作的边界。
JDBC 中使用事务，先要使用连接调用 setAutoCommite(false)方法，把自动提交（commit）置为 false。打开事务就要关闭自动提交。不用事务是要把 setAutoCommite(true)

在处理事务时，在发送 sql 语句后执行成功并确认时，就在 try 块中使用连接调用 commit()方法来发送提交信息，
在发送 sql 语句后执行失败时，会在 catch 语句块中使用连接调用 rollback()方法来发送回滚信息，也可以在需要时做回滚操作（主观原因）。

七、JDBC 事务并发产生的问题和事务隔离级别

1，脏读（dirty read），读取到了没有提交的数据。
2，不可重复读（UnPrpeatable Read），两次读取到了不同的数据，就是要保持在同一时间点上两次读取到的数据相同，
　不能够使查询数据时进行改变。
3，幻读（phantom），在两次查询同一时间点数据时，数据数量发生改变，要保持在同一时间点上两次读取到的数据相同。

事务隔离级别

TRANSACTION_NONE 不使用事务。
TRANSACTION_READ_UNCOMMITTED 可以读取为提交数据。
TRANSACTION_READ_COMMITTED 可以避免脏读，不能够读取没提交的数据，最常用的隔离级别　大部分数据库的默认隔离级别
TRANSACTION_REPEATABLE_READ 可以避免脏读，重复读取，

TRANSACTION_SERIALIZABLE 可以避免脏读，重复读取和幻读，（事务串行化）会降低数据库效率

以上的五个事务隔离级别都是在 Connection 类中定义的静态常量，使用 setTransactionIsolation(int level) 方法可以设置事务隔离级别。

八，异常的处理
try{}
catch(SQLException){}
try{}
catch(Exception){}

作业：自己写一个 Sqlplus 应用

JDBC 第三天　　2007 年 6 月 6 日

一、JDBC2.0 的新特性
　可滚动结果集（可双向滚动）
　批处理更新　***
　可更新结果集

1、批处理更新
Statement
addBatch(String sql)，　方法会在批处理缓存中加入一条 sql 语句
executeBatch() ，执行批处理缓存中的所有 sql 语句。

PreparedStatement

addBatch() 将一组参数添加到此 PreparedStatement 对象的批处理命令中。
executeBatch() 将一批命令提交给数据库来执行，如果全部命令执行成功，则返回更新计数组成的数组。
PreparedStatement 中使用批量更新时，要先设置好参数后使用 addBatch()方法加入缓存。
注意：批量更新中只能使用更新或插入语句

execute(String sql),这个方法的返回值是 boolean 类型，如果返回 true 就表示 sql 是一个 select 语句，可以通过 getResultSet()获得结果集，如果是 false，sql 就是 DML 语句或者是 DDL 语句。

2、可滚动结果集（可双向滚动），这种结果集不但可以双向滚动，相对定位，绝对定位，并且可以修改数据信息。

滚动特性
next()，此方法是使游标向下一条记录移动。
previous() ，此方法可以使游标上一条记录移动，前提前面还有记录。
absolute(int row)，可以使用此方法跳到指定的记录位置。定位成功返回 true，不成功返回 false，返回值为 false，则游标不会移动。
afterLast() ，游标跳到最后一条记录之后。
beforeFirst() ，游标跳到第一条记录之前。（跳到游标初始位）
first()，游标指向第一条记录。
last()，有彪指向最后一条记录。
relative(int rows) ，相对定位方法，参数值可正可负，参数为正，游标从当前位置向下移动指定值，参数为负，游标从当前位置向上移动指定值。

TYPE_FORWARD_ONLY ，该常量指示指针只能向前移动的 ResultSet 对象的类型。

TYPE_SCROLL_INSENSITIVE ，该常量指示可滚动但通常不受其他的更改影响的 ResultSet 对象的类型。
TYPE_SCROLL_SENSITIVE ，该常量指示可滚动并且通常受其他的更改影响的 ResultSet 对象的类型。

要使用可滚动结果集时，要在 Statement 创建时指定参数，才可以使用
Statement st=null;
st=con.createStatement(ReusltSet.TYPE_SCROLL_INSENSITIVE,ResuleSet.CONCUR_UPDATABLE)

ResultSet 结果集中，先使用 moveToInsertRow()，将游标移到和结果集结构类似的缓冲区中
然后可以使用 updateXxx(int column,columnType value)方法来更新指定列数据，
再使用 insertRow() 方法插入记录，
最后将游标指回原位，moveToCurrentRow() 。

能否使用可更新结果集，要看使用的数据库驱动是否支持，
还有只能用于单表且表中有主键字段（可能会是联合主键），不能够有表连接，会取所有非空字段且没有默认值。

能否使用 JDBC2.0 ResultSet 的新特性要看数据库驱动程序是否支持。

3、SQL3.0 中的行类型

Array，数组
Sturct，结构
Blob，大的二进制数据文件。
Clob，大文本文件对象。
在使用上述大对象的时候，在使用 JDBC 插入记录时要先插入一个空的占位对象，然后使用 select blobdata from t_blob where id = " + id + " for update 这样的语法来对获得的大对象，进

行实际的写入操作 Blod 通过 getBinaryOutputStream() 方法获取流进行写入。getBinaryStream()方法获得流来获取 blob 中存储的数据。
clob 的操作也和 blob 相同。 getAsciiStream() 方法用于读取存储的文本对象，getAsciiOutputStream()方法之获得流用来向文件对象写入的。

JDBC 第四天　　2007 年 6 月 7 日

一、ID 的 High/Low 算法
　　高位数字分别与低位数字相匹配，得到的数字是唯一的
　　减少与数据库的交互

二、ORM
１、类映射成表
　　类名与表名对应
２、属性定义映射成列，类型之间必须是兼容的
３、类关系映射成表关系

一对一双向关系
内存中都保存对方的一个引用
数据库中，表 b 的 id 是主键，也是外键，引用 a 表的 id 主键　－－　share pk
　　表 b 中有一个字段 aid 是外键，引用 a 表的主键，并且有唯一约束　　－－　pk+fk
共享主键：

```
create table car_pk (
    id number(10,0) not null,
    name varchar2(15),
    serial varchar2(30),
    manufacturer varchar2(50),
    producedate date,
    primary key (id)
);

create table engine_pk (
    id number(10,0) not null,
    model varchar2(20),
    manufacturer varchar2(50),
    producedate date,
    primary key (id)
);

alter table engine_pk
    add constraint fk_engine_car_pk
```

```
        foreign key (id)
        references car_pk(id);
```

外键＋唯一约束

```
create table car_fk (
        id number(10,0) not null,
        name varchar2(15) not null,
        serial varchar2(30) not null,
        manufacturer varchar2(50) not null,
        producedate date,
        primary key (id)
);

create table engine_fk (
        id number(10,0) not null,
        model varchar2(20) not null,
        manufacturer varchar2(50) not null,
        producedate date,
        carid number(10,0) unique,
        primary key (id)
);

alter table engine_fk
        add constraint fk_engine_car_fk
        foreign key (carid)
        references car_fk(id);
```

实体对象：在内存中有 id 属性的
值对象：没有 id 的，依赖其他对象存在

一对多关系
一的一方保存多一方的一个集合，最好使用 set，保证无重复元素
多的一方保存一一方的一个对象的引用

```
public class Order implements Serializable{
        private int id;
        private String owner;
        private String phone;
        private String address;
        private Set<Item> items = new HashSet<Item>();
}
public class Item implements Serializable{
        private int id;
        private String product;
```

```
    private int amount;
    private Order order;
}

create table ec_item (
    id number(10,0) not null,
    product varchar2(15) not null,
    amount number(10,0) not null,
    orderid number(10,0) not null,
    primary key (id)
);

create table ec_order (
    id number(10,0) not null,
    owner varchar2(15) not null,
    phone varchar2(15) not null,
    address varchar2(50),
    primary key (id)
);

alter table ec_item
    add constraint fk_item_order
    foreign key (orderid)
    references ec_order(id);
```

多对多
双方都保存对方的多个引用
例子：学生选课

```
public class TarenaCourse implements Serializable{
    private int id;
    private String name;
    private int period;
    private Set<TarenaStudent> students = new HashSet<TarenaStudent>();
}
public class TarenaStudent implements Serializable{
    private int id;
    private String name;
    private Date birthday;
    private Set<TarenaCourse> courses = new HashSet<TarenaCourse>();
}

create table student (
    id number(10,0) not null,
```

```
    name varchar2(15) not null,
    birthday date,
    primary key (id)
);

create table student_course (
    sid number(10,0) not null,
    cid number(10,0) not null,
    primary key (sid, cid)
);

create table course (
    id number(10,0) not null,
    name varchar2(15) not null,
    perion number(10,0),
    primary key (id)
);

alter table student_course
    add constraint fk_student
    foreign key (sid)
    references student(id);

alter table student_course
    add constraint fk_course
    foreign key (cid)
    references course(id);
```

通过学生姓名找课程

```
select c.name from cource c,student s,student_course sc
where c.id=sc.cid and    s.id=sc.sid
and s.name = 's1'
```

三、继承关系

```
public abstract class Computer implements Serializable{
    private int id;
    private int price;
    private String manufacturer;
}

public class Desktop extends Computer{
    private boolean isLCD;
}
```

```
public class Notepad extends Computer{
	private float weight;
	private float thickness;
}
```

1、建 3 张表　table per class
子类中保存父类的主键作为外键

```
create table computer_tpc (
	id number(10,0) not null,
	price number(10,0) not null,
	manufacturer varchar2(30) not null,
	primary key (id)
);

create table desktop_tpc (
	computerid number(10,0) not null,
	islcd char(1),
	primary key (computerid)
);

create table notepad_tpc (
	computerid number(10,0) not null,
	weight float,
	thickness float,
	primary key (computerid)
);

alter table desktop_tpc
	add constraint fk_desk_computer_tpc
	foreign key (computerid)
	references computer_tpc(id);

alter table notepad_tpc
	add constraint fk_note_computer_tpc
	foreign key (computerid)
	references computer_tpc(id);
```

查找所有电脑的配制（只要是电脑就能被查出来）

```
select c.id,c.price,d.islcd,n.weight,n.thickness
from computer c, desktop d,notepad n
where	c.id = d.computerid(+)
and	c.id = n.computer(+)
```

2、建 2 张表

```
create table desktop (
        id number(10,0) not null,
        price number(10,0) not null,
        manufacturer varchar2(30) not null,
        islcd char(1),
        primary key (id)
);

create table notepad (
        id number(10,0) not null,
        price number(10,0) not null,
        manufacturer varchar2(30) not null,
        weight float,
        thickness float,
        primary key (id)
);
```

3、建 1 张表

```
create table computer_tph (
        id number(10,0) not null,
        category char(1) not null,
        price number(10,0) not null,
        manufacturer varchar2(30) not null,
        islcd char(1),
        weight float,
        thickness float,
        primary key (id)
);
```

JDBC 第五天　　2007 年 6 月 8 日

一、JDBC2.0 扩展

1、JDBC DataSource

DataSourse（数据源），包含了连接数据库所需的信息，可以通过数据源或的数据库连接，有时由于某些连接数据库的信息会变更，

所以经常使用包含数据库连接信息的数据源。

JDBC 取连接有 2 种方式：Driver Manager 和 数据源

2、JNDI 和 DataSourse

主要功能：定位服务
JNDI，（命名路径服务）也用于存储数据，但是他所存储的是一写零散的信息。
JNDI 的方法是在 javax.naming 包下

InitialContext 连接，初始化上下文，这个类的提供者一般也是服务器的提供者
查找和绑定
查找由我们做，绑定我们并不关心，只配制数据源就好了

代替 DriverManager 定位数据源
遍布式企业的数据源的属性可以存储在同一个目录（JNDI）中
以这种方式集中管理用户名、密码、数据库名和 JDBC URL
创建连接：

```
Context jndiContext = new InitialContext();
DataSource source = (DataSource)jndiContext.lookup(" ");
COnnection    con = source.getConnection();
```

3、连接池
要提供连接池数据源，带缓存的连接
带缓存的连接，即可池化的连接，其 close()方法，在物理上并没有被关闭，而是保留在一个队列中并被反复使用。

4、分布式事务
事务分为 JDBC 事务和 JTA
JDBC 事务，由容器管理
JTA，分布式事务，由容器管理