Web开发技术

第六章： JDBC数据访问接口

第六章： JDBC数据访问接口

第六章： JDBC数据访问接口

序 数据库介绍

- 数据库常用的术语和基本概念：

数据、数据库、数据库管理系统、数据库系统

数据、数据库、数据库管理系统、数据库系统是与数据库技术密切相关的四个基本概念。

第六章： JDBC数据访问接口

- 数据是数据库中存储的基本对象。数据的种类很多：文字、图形、图像、声音、学生的档案记录、货物的运输情况等等都是数据。

- 数据的定义:描述事物的符号记录称为数据。

- 描述事物的符号:可以是数字、也可以是文字、图形、图像、声音、语言等，数据有多种表现形式，它们都是经过数字化后存入计算机。

第六章： JDBC数据访问接口

- 在计算机中，为了存储和处理事物，就要抽出对这些事物感兴趣的特征组成一个记录来描述。
- 例如：在学生档案中，人们最感兴趣的是：学生姓名、性别、年龄、出生年月、籍贯、所在系别、入学时间，那么可以这样描述：

  (李明，男，22，1984，哈尔滨，计算机，2006)
- 这里的学生记录就是数据。了解其含义的可以明白它代表的意思，不了解其语义的则无法理解其含义。

第六章： JDBC数据访问接口

- 数据的形式并不能完全表达其内容，需要经过解释。所以数据和关于数据的解释是不可分的。
- 数据的解释是指对数据的说明，数据的含义称为数据的语义，数据与其语义是不可分的。

第六章： JDBC数据访问接口

- （举例）

学生登记表

学　号	姓　名	年　令	性　别	系　名	年　级
95004	李　萍	19	女	社会学	95
95006	黄大鹏	20	男	商品学	95
95008	张文斌	18	女	法律学	95
…	…	…	…	…	…

第六章： JDBC数据访问接口

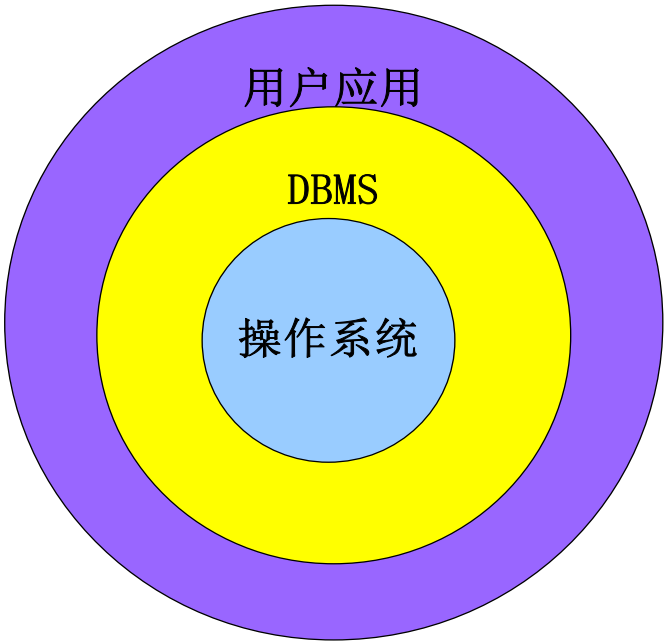
- 所谓数据库就是长期储存在计算机内、有组织的、可共享的数据集合。

- 特点：数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度，较高的数据独立性和可为各种用户共享。

- 例如：图书馆的图书数据库、机场的航班数据库、银行数据库

第六章： JDBC数据访问接口

- DBMS（Database Management System），是计算机程序的集合，用于创建和维护数据库
  - 位于操作系统和用户应用之间
  - 总是基于某种数据模型
  - 数据库厂商的产品通常指DBMS，如Oracle、SQL Server、DB2、Informix等。



第六章： JDBC数据访问接口

1 数据库应用开发简介

- 作为有效的数据存储和组织管理工具，数据库的应用日益广泛
- 目前主流的数据库产品有Oracle、SQL Server、DB2和SyBase MySQL等多种。
- 在数据库开发领域中，有三方面需要掌握：SQL语言、ODBC数据访问接口和JDBC数据库访问接口。

第六章： JDBC数据访问接口

1.1 SQL语言概述

- SQL（Structured Query Language）是使用关系模型的数据库语言，用于和各类数据库连接，提供通用的数据管理和查询功能。
- SQL语言最初由IBM公司开发，实现了关系数据库中的信息检索。后几经修改和完善，被国际标准化组织确定为国际标准，目前执行的是1992年制定的SQL-92标准。
-

第六章： JDBC数据访问接口

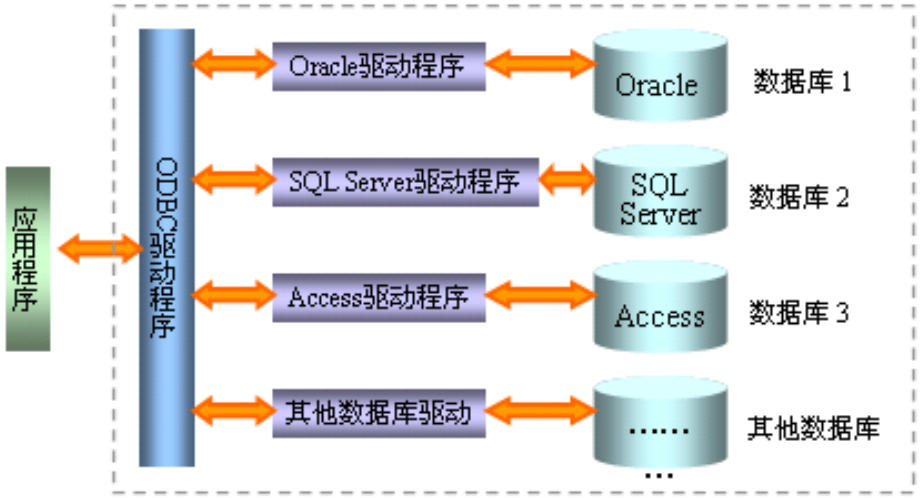
1.1 SQL语言概述

- SQL可以为各种支持SQL-92标准的数据库管理系统（DBMS）所接受和处理，通常各种DBMS都提供图形用户界面，以使用户直接对数据库进行操作。但SQL语言本身并不是完整的编程语言，还需要与其他高级编程语言配合，才能实现应用程序对数据库的访问操作。

第六章： JDBC数据访问接口

1.2 ODBC数据访问接口

- 开放式数据库互连ODBC（Open DataBase Connectivity）



- 是微软公司开发的一套开发数据库系统应用程序接口规范，它支持应用程序以标准的ODBC函数和SQL语句操作各种不同的数据库。

2 SQL语言基础应用

- SQL（Structured Query Language）是关系型数据库的标准语言，是由国际标准组织提出的，各种关系型数据库都支持SQL指令，Oracle在基本的SQL基础上进行了扩充。

- SQL语句有如下的两大特点
- （1）SQL是一种类似于英语的语言，很容易理解和书写。
- （2）SQL语言是非过程化的语言（第四代语言）

第六章： JDBC数据访问接口

2 SQL语言基础应用

SQL分类	描述
数据定义语言（DDL）	数据定义语言（DDL）用于定义、修改或者删除数据库对象，如Create Table等
数据查询语言（DQL）	数据查询语句（Data Query Language，DQL）用于对数据进行检索。如最常用的Select语句
数据操纵语言（DML）	数据操纵语言（DML）用于访问、建立或者操纵在数据库中已经存在数据，如Select、Insert、Update和Delete等等。
事务控制语言（TCL）	事务控制语言（Transact Control Language）管理DML语句所做的修改，是否保存修改或者放弃修改。
数据控制语言（DCL）	数据控制语言（DCL）管理对数据库内对象的访问权限和授予和回收，如Grant、Revoke等等。

2.1 基本SQL语句

- 基本的SQL语句包括数据查询语言DQL和数据操纵语言DML。也就是对数据库最常用的四大基本操作：
- 查询（Select）；
- 插入（Insert）；
- 更新（Update）；
- 删除（Delete） 。

第六章： JDBC数据访问接口

2.1 基本SQL语句

1.DQL的3种基本格式

1. 基本句型一：（最简单的SELECT语句）

SELECT 字段名 FROM 数据表

例1. SELECT * FROM grade

功能说明：将grade表中的所有字段取出来。

例2. SELECT 学号,姓名 FROM grade

功能说明：将grade表中学号和姓名字段取出来。

例3. SELECT学号,姓名,语文+数学+英语 as 总成绩 FROM grade

功能说明：将grade表中的学号和姓名取出来，并将语文、数学和英语成绩相加产生虚拟列总成绩。

2.1 基本SQL语句

1.DQL的3种基本格式

2.基本句型二：使用条件查询

SELECT字段名 FROM数据表 WHERE 筛选条件

测试句型如下。

例1. SELECT * FROM grade WHERE数学>60

功能说明：把所有数学成绩大于60分的记录选出来。

例2. SELECT * FROM grade WHERE数学=300 or 语文=300

功能说明：把数学成绩等于300分或者语文成绩等于300分的人选出来。

第六章： JDBC数据访问接口

2.1 基本SQL语句

1.DQL的3种基本格式

例3. Like子句基本格式一：“_”匹配。

功能说明：每个下划线匹配一个任意字符，注意只匹配一个字符。比如：姓名 like ‘_敏’，匹配姓名以“敏”字结尾且字数等于二的所有数据记录，如：“张敏”。

例4. Like子句基本格式二：“%”匹配。

比如：姓名 Like ’%敏%’，匹配姓名中出现“敏”的所有数据记录，如：“周惠敏”，“于敏”、“敏大”、“敏二”等。

比如要在数据库中查询姓江的人，只要利用一条SQL语句就可以了，SELECT * FROM 数据库表 WHERE 姓名 Like ’江%’。

2.1 基本SQL语句

1.DQL的3种基本格式

3.基本句型三：（进行排序）

SELECT字段名 FROM 数据表 ORDER BY 字段名

测试句型如下。

（1）SELECT * FROM grade ORDER BY数学　　注：从低到高排序

功能说明：从grade表中取出所有字段，并按数学成绩排序。

（2）SELECT * FROM grade ORDER BY数学,语文

功能说明：从grade表中取出所有字段，并按数学成绩排序，如果数学成绩相同则按照语文成绩排序。

2.1 基本SQL语句

1.DQL的3种基本格式

3.基本句型三：（进行排序）

SELECT字段名 FROM 数据表 ORDER BY 字段名

测试句型如下。

（3）SELECT * FROM grade ORDER BY数学 desc 注：从高到低排序

功能说明：从**grade**表中取出所有字段，并按数学成绩倒序。

（4）SELECT top 5 * FROM grade

功能说明：从**grade**表中取出前五条记录的所有字段。

2.1 基本SQL语句

2. DML的基本格式

（1）DELETE指令：删除数据记录。

基本语法：DELETE FROM 数据表 WHERE 条件

例：DELETE from grade WHERE数学=0

功能说明：删除所有数学成绩为零的记录，如果没有WHERE子句，则删除所有记录。

2.1 基本SQL语句

2. DML的基本格式

（2）UPDATE指令：更新数据记录。

基本语法：UPDATE 数据表 SET 字段值=新值 WHERE条件

例1：UPDATE grade SET 数学=数学+10　说明：将grade表中所有人的成绩加10分

例2：UPDATE grade SET 数学=100 WHERE 姓名 like '%敏%'

功能说明：将姓名中含有敏的人的数学成绩更新为100分

2.1 基本SQL语句

2. DML的基本格式

（3）INSERT INTO指令：添加数据记录。

基本格式1：INSERT INTO 数据表VALUES （字段新值）

基本格式2：INSERT INTO 数据表（字段一，字段二，……）

VALUES （字段新值）

其中关键字两种格式的区别是：当values含有数据库表所有字段的值，并且顺序和数据库字段一致时，就可以省略数据库表后面的字段名称。

2. 1 基本SQL语句

2. DML的基本格式

例1：INSERT INTO grade(学号, 姓名, 数学) VALUES (1234, '周润发',70)

例2：INSERT INTO grade VALUES (5678, '周润发',70,80,90)

功能说明：该语句等价于：

INSERT INTO grade(学号, 姓名, 语文, 数学, 英语) VALUES (5678, '周润发',70,80,90)

2.2聚合函数

聚合函数在信息管理系统经常使用，功能是做一些基本的统计和计算。

聚合函数有5个，分别是

SUM函数；

AVG函数；

COUNT函数

MAX函数和MIN函数。

2.2聚合函数

- （1）SUM函数，功能是算出某个字段的总值。
  - 例. SELECT SUM(数学) As Total FROM grade
  - 功能说明：求出所有学生数学成绩总和，这个数值的列名为Total。

（2）AVG函数，功能是算出某个字段的平均值。

  - 例. SELECT AVG(数学) As Average FROM grade
  - 功能说明：求出所有学生数学成绩平均分，这个数值的列名为Average。

2.2聚合函数

- （3）COUNT函数，功能是算出返回记录的行数。
  - 例. SELECT COUNT(*) As Counts FROM grade
  - 功能说明：求出满足条件的记录总数。
- （4）MAX函数，功能是算出某个字段的最大值。
  - 例. SELECT MAX(数学) As First FROM grade
  - 功能说明：求出所有学生数学成绩的最高分，这个数值的列名为First。
- （5）MIN函数，功能是算出某个字段的总值。
  - 例. SELECT MIN(数学) As Last FROM grade
  - 功能说明：求出所有学生数学成绩的最低分，这个数值的列名为Last。

2.3 分组查询

- 分组查询包括GROUP BY和HAVING关键字。比如计算某班男生女生的数学平均分，利用可以利用分组查询完成。grade表中添加一列“性别”，如person1.mdb库中的grade表，
- 可以使用“SELECT 性别, AVG(数学) as 平均分 FROM grade GROUP BY 性别”得到男生和女生的平均分
- person1.data

2.3 分组查询

- 计算所有男生的数学平均成绩，有两种方法：
- （1）SELECT 性别, AVG(数学) as 平均分 FROM grade GROUP BY 性别 HAVING 性别='男'
- （2）SELECT 性别, AVG(数学) as 平均分 FROM grade WHERE 性别='男' GROUP BY 性别
- person1.data

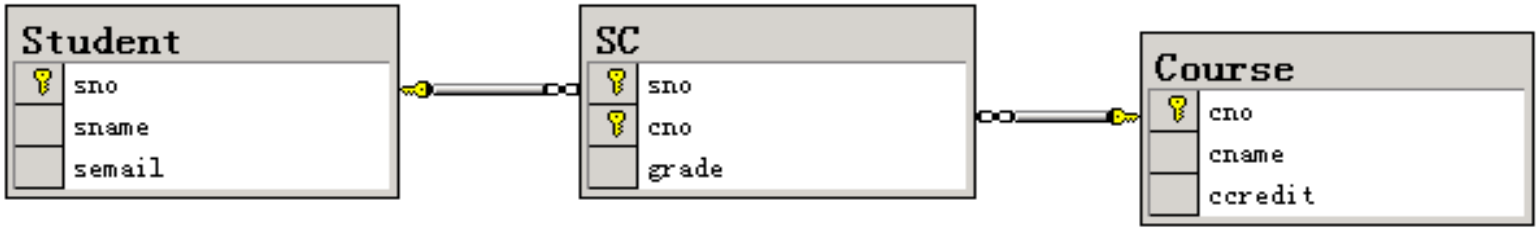
2.3 分组查询

- 在使用分组查询的时候，有4点需要注意。
  - （1）WHERE子句必须放在GROUP BY子句之前。
  - （2）HAVING子句中只能包含分组字段或者聚合函数。
  - （3）SELECT语句选择的列只能是分组字段或者聚合函数。
  - （4）HAVING必须放在GROUP BY子句之后。

第六章： JDBC数据访问接口

2.4 交叉查询

- 考虑3个表：学生表（Student）、课程表（Course）和选课表（SC）表的关系如图



2.4 交叉查询

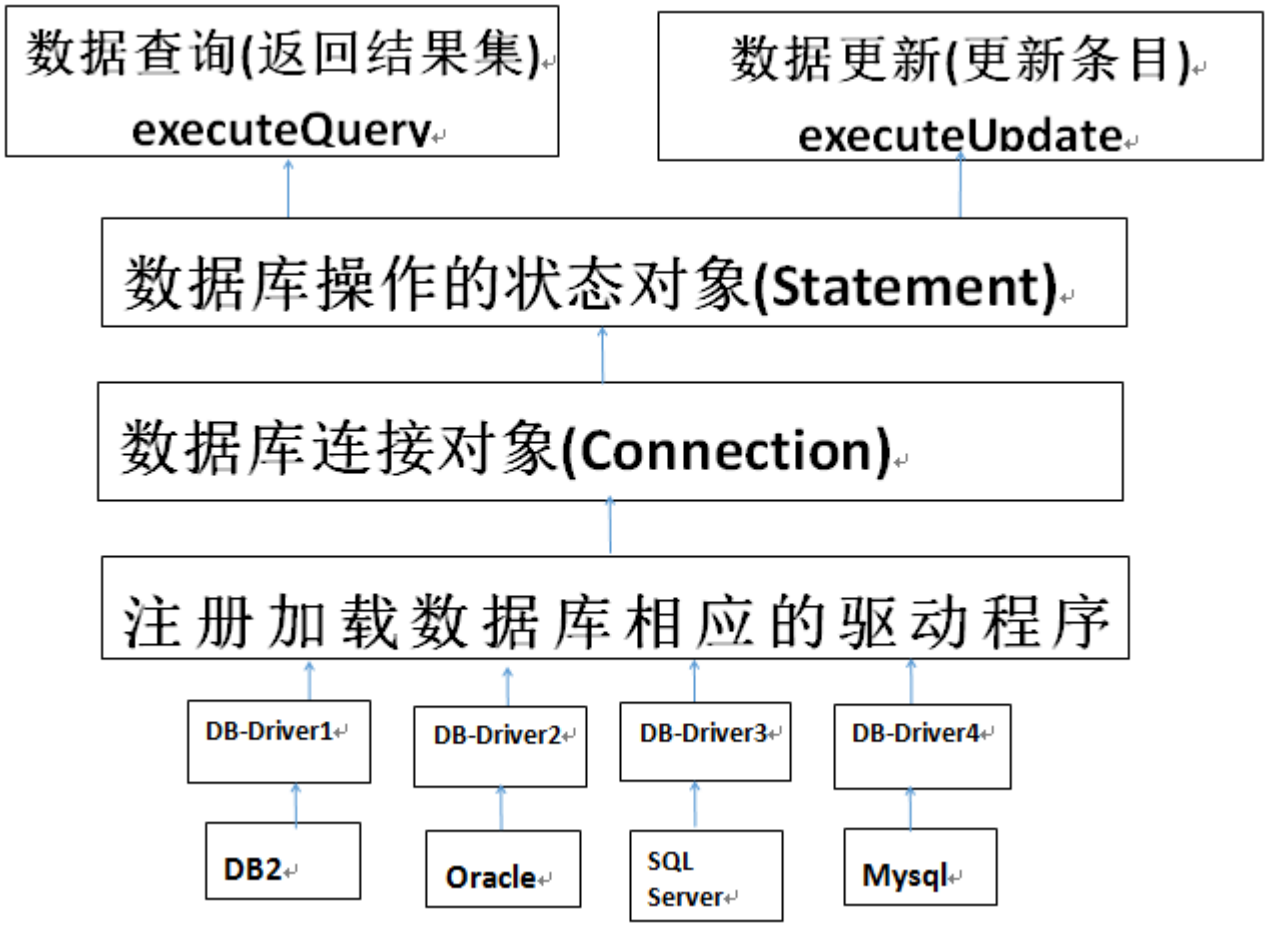
- （1）学生表中的sno表示学生学号，sname表示学生姓名。
- （2）课程表中cno表示课程编号，cname表示课程的名称。
- （3）选课表中的sno表示学生学号，cno表示课程编号。
- 查找选择课程为“软件工程”的所有同学姓名？可以利用如下的SQL语句。

```
SELECT b.sname FROM sc a, student b, course c
WHERE a.cno=c.cno
AND a.sno=b.sno and c.cname='软件工程'
```

- 其中“sc a”表示给表sc起个别名为a，同样“student b”是给student表起个别名b。该查询实现了三个表之间的交叉查询。

第六章： JDBC数据访问接口

3 JDBC的结构



3 JDBC的结构

3.1 JDBC驱动数据库的四种类型

- （1）JDBC-ODBC桥加ODBC驱动程序：JDBC-ODBC桥产品利用ODBC驱动程序提供JDBC访问。在服务器上必须可以安装ODBC驱动程序。
- （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。

第六章： JDBC数据访问接口

3.1 JDBC驱动数据库的四种类型

- （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。
- （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。
- 第3类和第4类驱动程序将成为JDBC访问数据库的首选方法。第1类和第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。

3 JDBC的结构

连接数据库的方法

（1）找到数据库的物理实体

（2）打开数据库（建立数据库的连接）

（3）发送SQL语句

（4）返回发送结果

3 JDBC的结构

3.2 数据驱动程序

- 使用JDBC的第一步是安装驱动程序。大多数数据库都有JDBC驱动程序，常用的JDBC驱动程序如图所示。

第六章： JDBC数据访问接口

- 开发使用数据库的应用： JAR文件的绝对路径

CLASSPATH环境变量中

- Web应用：

JAR文件

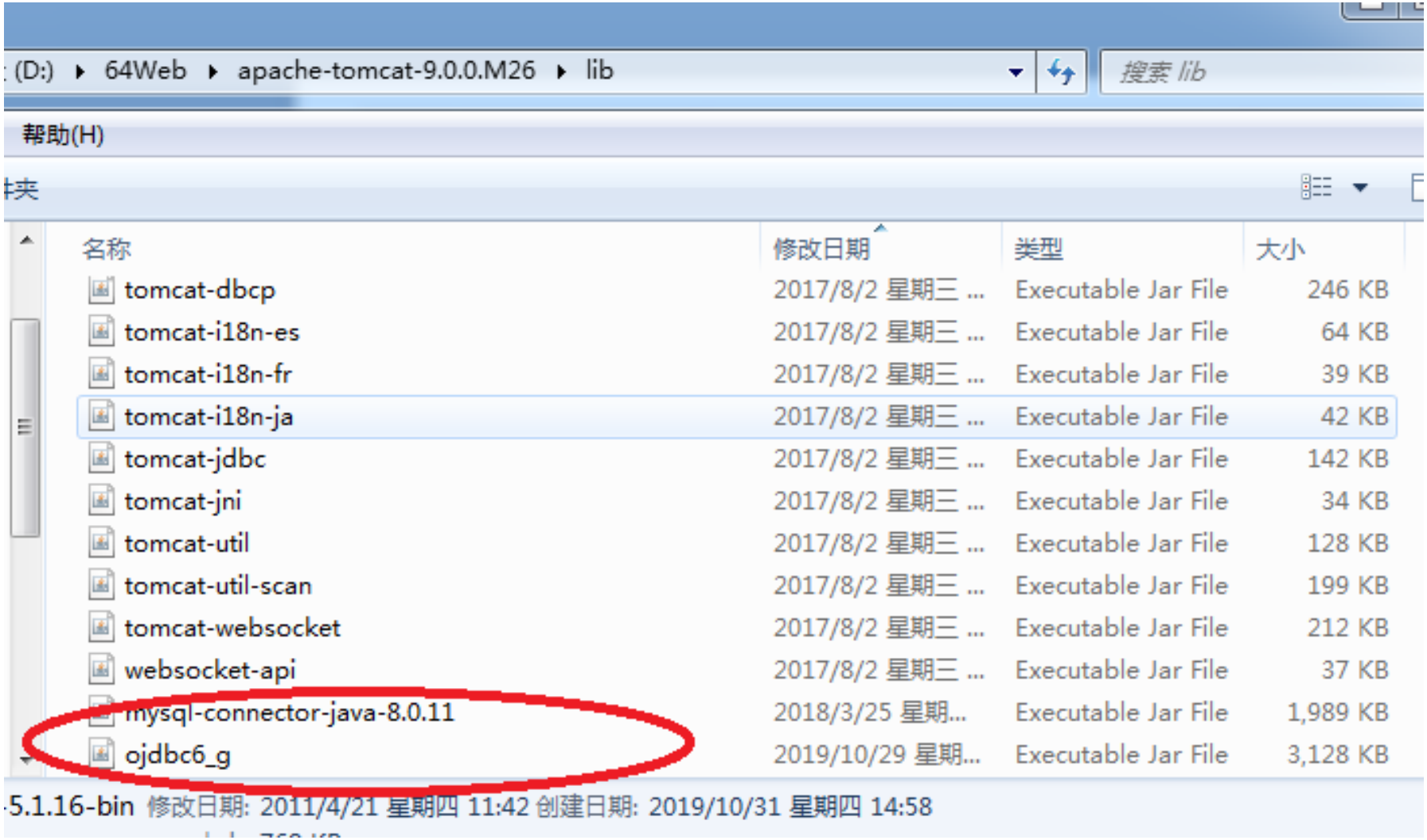
WEB-INF/lib

- 多个应用使用同一个数据库

JAR文件

{tomcat的web安装目录}/lib

第六章： JDBC数据访问接口



第六章： JDBC数据访问接口

- JDBC的几个参数 MySQL Connector/J
  - driver → com.mysql.jdbc.Driver
  - URL → jdbc:mysql://host:3306/dbName（主机名或IP地址；数据库名字）
  - User
  - password

3 JDBC的结构

3.2 数据驱动程序

- 加载oralce的JDBC驱动程序：
- Class.forName(“oralec.jdbc.driver.OracleDriver”)
- 加载Mysql的JDBC驱动程序
- Class.forName(“ com.mysql.jdbc.Driver”)

```
<%Connection conn=null;
Statement stmt=null;
ResultSet rs=null;
try{
Class.forName(" com.mysql.jdbc.Driver ");  }
catch(ClassNotFoundException ce){
System.out.println(ce.getMessage());}
try{
conn=DriverManager.getConnection("jdbc:mysql://host:3306/dbName
,"username","password");
stmt=conn.createStatement();
rs=stmt.executeQuery("select * from grade");
while(rs.next()){
out.print(rs.getString("学号"));
out.print(rs.getString("姓名"));
out.print(rs.getString("语文"));
out.print(rs.getString("数学"));
out.print(rs.getString("英语"));
out.print("<BR>");   } }
catch(SQLException e){
System.out.println(e.getMessage()); }
finally{ stmt.close();conn.close();  }  %>
```

3 JDBC的结构

3.3 Connection对象

- 建立是建立与数据库之间的连接，也就是创建一个Connection的实例。DriverManager类的getConnection()方法将建立数据库的连接：
  - public static Connection getConnection(String URL, String user, String password) throws SQLException
- 在程序的最后，应该关闭Connection对象：
- public void close() throws SQLException

3 JDBC的结构

3.4 Statement对象

- Statement对象用于将SQL语句发送到数据库中。
- 存在3种Statement对象：
- （1）Statement：发送基本的sql语句。
- （2）PreparedStatement（从Statement继承而来）：发送带有参数的sql语句或基本sql语句
- （3）CallableStatement（从PreparedStatement继承而来）：调用数据库中的存储过程。

<%Connection conn=null;
Statement stmt=null;
ResultSet rs=null;
try{
Class.forName(" com.mysql.jdbc.Driver "); }
catch(ClassNotFoundException ce){
System.out.println(ce.getMessage());}
try{
conn=DriverManager.getConnection("jdbc:mysql://host:3306/dbName
,"username","password");
stmt=conn.createStatement();
rs=stmt.executeQuery("select * from grade");
while(rs.next()){
out.print(rs.getString("学号"));
out.print(rs.getString("姓名"));
out.print(rs.getString("语文"));
out.print(rs.getString("数学"));
out.print(rs.getString("英语"));
out.print("
");   } }
catch(SQLException e){
System.out.println(e.getMessage()); }
finally{ stmt.close();**conn.close(); }** %>

3 JDBC的结构

连接数据库的方法

（1）找到数据库的物理实体

（2）打开数据库（建立数据库的连接）

（3）发送SQL语句

（4）返回发送结果

3 JDBC的结构

3.4 Statement对象

- Statement接口提供了两种执行SQL语句的常用方法：
- public ResultSet executeQuery(String sql) throws SQLException
- 用于产生单个ResultSet的语句，例如SELECT语句。
- public int executeUpdate(String sql) throws SQLException
- 用于执行INSERT、UPDATE或DELETE语句以及SQL DDL语句，例如CREATE TABLE和DROP TABLE。该方法返回一个整数，指示受影响的行数。

3 JDBC的结构

3.5 ResultSet对象

- ResultSet包含符合SQL语句执行结果所有行，并且它通过一套get方法提供了对这些行中数据的访问，常用的get方法有：
- int getInt(int columnIndex)，取得当前行中第columnIndex列的整数的值。
- int getInt(String columnName)，取得当前行中列名为columnName的整数的值。
- Date getDate(int columnIndex)，取得当前行中第columnIndex列的日期的值。
- Date getDate(String columnName)，取得当前行中列名为columnName的日期的值。
- public String getString(int columnIndex)，取得当前行中第columnIndex列的字符串的值。
- public String getString(String columnName)，取得当前行中列名为columnName的字符串的值。

3 JDB

案例

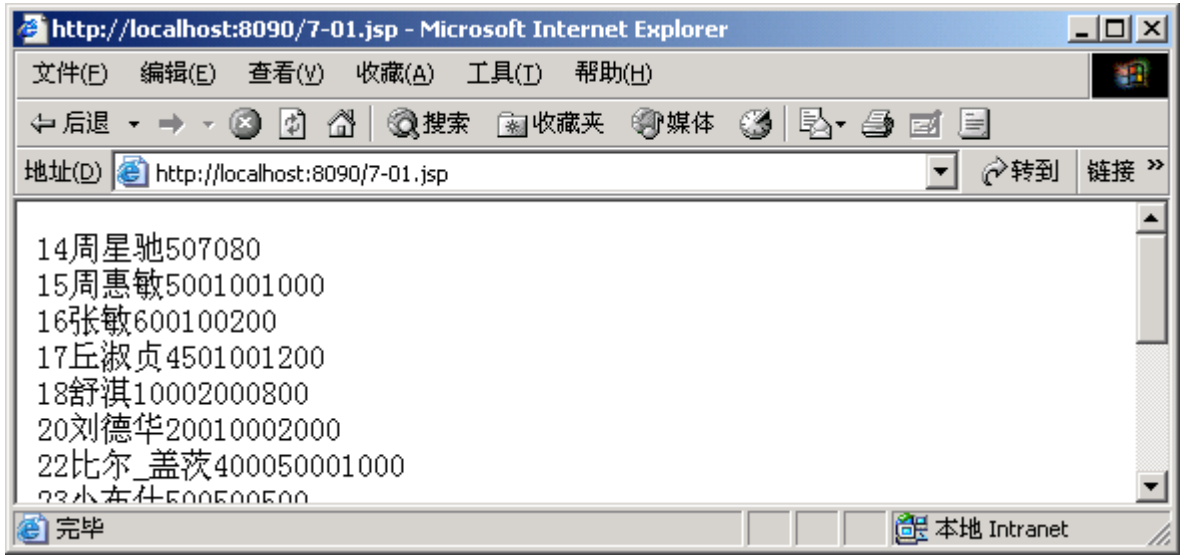
```
<%Connection conn=null;
Statement stmt=null;
ResultSet rs=null;
try{
Class.forName(" com.mysql.jdbc.Driver ");  }
catch(ClassNotFoundException ce){
System.out.println(ce.getMessage());}
try{
conn=DriverManager.getConnection("jdbc:mysql://host:3306/dbName
,"username","password");
stmt=conn.createStatement();
rs=stmt.executeQuery("select * from grade");
while(rs.next()){
out.print(rs.getString("学号"));
out.print(rs.getString("姓名"));
out.print(rs.getString("语文"));
out.print(rs.getString("数学"));
out.print(rs.getString("英语"));
out.print("<BR>");   } }
catch(SQLException e){
System.out.println(e.getMessage()); }
finally{ stmt.close();conn.close();  }  %>
```

第六章： JDBC数据访问接口

3 JDBC的结构

案例6-1 MySql的JDBC本地原生读取数据库数据

案例名称：使用JDBC-本地API读取Mysql数据库



3 JDBC的结构

案例6-2 JDBC本地API读取Mysql数据库

利用ExecuteUpdate方法来执行数据操作语句（Insert,Update和Delete）；

案例名称：使用JDBC-ODBC 桥操作Access数据库

程序名称：6-2.jsp

```
String sql="update grade set 数学=数学+10";
int iback=stmt.executeUpdate(sql);
out.print("有"+iback+"条数据被修改!");
Delete from grade where 学号=1001
Insert into grade values('1','张敏
','1','1','1')
```

3 JDBC的结构

3.5 PreparedStatement对象

- PreparedStatement对象表示预编译SQL语句，SQL语句经过预编译后存储在PrepareStatement对象当中，可以用来进行高效的多次执行，PreparedStatement对象是Conncention的preparedStatement()方法创建的；

- Public PreparedStatement preparedStatement(String sql) throws SQLException

3 JDBC的结构

3.5 PreparedStatement对象

- 在生成PreparedStatement对象的字符串中可以通过”?”代表一个产生变化的参数，随后通过循环语句生成这一系列语句，方便SQL的生成。
- Public void setInt(int parameterindex,int x) throws SQLExecption
- 可以将第parameterindex个参数赋值为x，通过addBatch()方法将其加到一个批次作业，通过executeBatch()执行所有加入批次的作业，最后应该关闭PreparedStatement对象。

3 JDBC的结构

3.5 PreparedStatement对象

案例名称：使用PreparedStatement
程序名称：6-3.jsp

```
try{
conn=DriverManager.getConnection(“jdbc:mysql://localhost:3306/dbname”,”username”,“password”);
pst=conn.prepareStatement("Insert into grade(学号)values(?)");
for(int i=101;i<111;i++){
pst.setInt(1,i);
pst.addBatch();
}
pst.executeBatch();
```

3 JDBC的结构

3.5 PreparedStatement对象

案例名称：使用PreparedStatement
程序名称：6-4.jsp

```
try{
conn=DriverManager.getConnection("
(“jdbc:mysql://localhost:3306/dbname”,”username”,“password”);
pst=conn.prepareStatement("update grade set 数学=数学+(?)
where 学号=(?)");
for(int i=101;i<110;i++){
pst.setInt(1,1000);
pst.setInt(2,i);
pst.addBatch(); }
int i[]=pst.executeBatch();
if(i[0]>0){
out.print("批量处理成功"); }
```

4 JDBC四种数据访问格式

4.1 格式一：执行基本SQL语句

案例名称：数据访问基本格式一

```
Class.forName("JDBC驱动程序");
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);
Statement stmt=conn.createStatement();
ResultSet rs=stmt.executeQuery("DQL语句");//如果是数据查询
stmt.executeUpdate("DML语句");//如果是数据操作
```

程序6-1.jsp和程序6-2.jsp

4 JDBC四种数据访问格式

4.2 格式二：实现分页

- 在JDBC1.0中，结果只能利用next方法每次向前移动一行。在JDBC2.0中，增加了一类新的结果集，称为可滚动结果集

案例名称：数据访问基本格式二

```
Class.forName("JDBC驱动程序");
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);
Statement stmt = connect.createStatement(结果集类型,结果集并发性);
ResultSet rs=stmt.executeQuery("DQL语句");//如果是数据查询
stmt.executeUpdate("DML语句");//如果是数据操作
```

4 JDBC四种数据访问格式

4.2 格式二：实现分页

Statement createStatement(int resultSetType, int resultSetConcurrency)

- 结果集类型有3种**resultSetType**：
  - **ResultSet.TYPE_FORWORD_ONLY**：指定**ResultSet**对象是不可滚动，这是默认值。
  - **ResultSet.TYPE_SCOLL_INSENSITIVE**：指定**ResultSet**对象是可滚动的，但是对数据库中修改不敏感。
  - **ResultSet.TYPE_SCOLL_SENSITIVE**：指定**ResultSet**对象是可滚动的，而且对数据库的修改敏感。

4 JDBC四种数据访问格式

4.2 格式二：实现分页

Statement createStatement(int resultSetType, int resultSetConcurrency)

- 结果集并发性有2种**resultSetConcurrency**：
  - 结果集的并发性（**Concurrency**）决定**ResultSet**对象是否可以修改数据库中的行。可以使用**ResultSet**类中定义的**int**常量来指定结果集的并发性。
  - **ResultSet.CONCUR_READ_ONLY**，指定**ResultSet**对象不能修改数据库，默认值。
  - **ResultSet.CONCUR_UPDATABLE**，指定**ResultSet**对象可以修改数据库。

4 JDBC四种数据访问格式

4.2 格式二：实现分页

- 在可滚动结构集中，可以使用多个方法灵活的在行之间移动：
- **next()**方法：移动到结果集下一行，没有下一行返回**false;**否则返回**true**；
- **privious()**方法：移动到前一行，没有前一行返回**false;**否则返回**true**；
- **first()**方法：移动到第一行，结果集中没有行返回**false;**否则返回**true**；
- **last()**方法：移动到最后一行，结果集中没有行返回**false;**否则返回**true**；
- **beforeFirst()**方法：移动到第一行前面位置；
- **afterlast()**方法：移动到最后一行后面的位置；
- **absolute(int rowNumber)**方法：移动到**rowNumber**指定的行；
- **relative(int relativeRowNumber)**方法：移动到相对于当前行的某一行；

4 JDBC四种数据访问格式

4.2 格式二：实现分页

- 在可滚动结构集中，如果不知道当前行的位置，以下方法检查：
- **getRow()**方法：返回当前行的行号（**int**值），没有当前行返回**0**；
- **isFirst()**方法：如果当前是第一行，返回**true;**否则返回**false**；
- **isLast()**方法：如果当前是最后一行，返回**true**；否则返回**false**；
- **isBrforeFirst()** 方法：如果当前位置在第一行之前，返回**true**；否则返回**false**；
- **isAfterLast()**方法：如果当前位置在最后一行之后，返回**true**；否则返回**false**；

4 JDBC四种数据访问格式

4.2 格式二：实现分页

- 在使用可更新结果集的时候，可以对**ResultSet**对象本身进行添加、删除和修改：
- **updateString()**方法更新字符串列；
- **updateDate()**方法更新日期型列；
- **updateInt()**方法更新数字列。

4 JDBC四种数据访问格式

4.3 格式三：执行带参数的SQL语句

案例名称：数据访问基本格式三

```
Class.forName("JDBC驱动程序");
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);
PreparedStatement stmt = connect.PreparedStatement (带参数的SQL语句);
pst.setInt(设置参数的值);
ResultSet rs = pst.executeQuery();     //如果是数据查询
stmt.executeUpdate();                  //如果是数据操作
```

4 JDBC四种数据访问格式

4.4 格式四：执行存储过程

案例名称：数据访问基本格式四

```
Class.forName("JDBC驱动程序");
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);
CallableStatement cs = con.prepareCall("{call 存储过程名(?,?)}")
cs.setInt(1, 设置参数的值);     //设置输入参数的值
cs.execute ();                  //执行存储过程
String strBack = cs.getString(2)   //得到输出参数的值
```

5 使用JDBC访问数据库

5.1 格式1：读取Excel数据

- Connection对象不仅可以连接数据库，也可以连接excel。

案例名称：访问**excel**

程序**6-5.jsp**

```
<%@ page import="java.sql.*"%>
 <% Connection conn=null; Statement stmt=null ;ResultSet rs=null;
  try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
 catch (ClassNotFoundException ce){out.print(ce.getMessage());}
 try{ conn=DriverManager.getConnection("jdbc:odbc:person");
 stmt=conn.createStatement();
 String sql="select * from [Sheet1$]"; rs=stmt.executeQuery(sql);
 while(rs.next()){ out.print("编号："+rs.getString(1));  out.print("姓名："+rs.getString(2));
   out.print("座位："+rs.getString(3));    out.print("<br>");}
 rs.close(); stmt.close(); conn.close();}
 catch (SQLException e){System.out.print(e.getMessage()); } %>
```

5 使用JDBC访问数据库

5.1 格式1：实现事务管理

- 事物是一些时间的集合，执行一条SQL语句可以理解位一个事件。事物中包含多个事件，每当一个事件都能执行成功的时候，事物才执行，如果任何一个事件不能执行成功，事物不被执行。

```
boolean defaultCommit=conn.getAutoCommit();
conn.setAutoCommit(false);
try{ stmt.excuteUpdate(SQL1); stmt.excuteUpdate(SQL2); conn.commit();}
catch(Exception e){ conn.rollback(); e.printStackTrace();}
finally{ if(stmt!=null){ stmt.close();}
If (conn!=null){ conn.close();} }
conn.setAutoCommit(defaultCommit);
```

第六章： JDBC数据访问接口

案例名称： 事物管理

程序**6-6.jsp**

```
<%@ page import="java.sql.*"%>
 <%
  Connection conn=null;  Statement stmt=null;  boolean defaultcommit=false;
  String SQL1="insert into grade (学号) values(9001)";
  String SQL2="update grade set 姓名='张三' where 学号=9001";
  try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  }
  catch(ClassNotFoundException ce){ out.print(ce.getMessage()); }
  try{ conn=DriverManager.getConnection("jdbc:odbc:grade");
  defaultcommit=conn.getAutoCommit();
  conn.setAutoCommit(false);
  stmt=conn.createStatement();
  stmt.executeUpdate(SQL1);  stmt.executeUpdate(SQL2);  conn.commit(); }
  catch(Exception e){
  conn.rollback();  e.printStackTrace(); out.print(e.getMessage());  }
  finally{  conn.setAutoCommit(defaultcommit);   if(stmt!=null){  stmt.close();  }
  if(conn!=null){conn.close();  } }out.print("提交成功！ ！ ");
  %>
```

5 使用JDBC访问数据库

5.1 格式1：动态合成SQL语句

- 一般情况下，需要根据用户的输入来合成SQL语句，

程序：6-7.jsp 动态执行sql语句

```
<%Connection conn=null; Statement stmt=null;ResultSet rs=null;
  request.setCharacterEncoding("GBK");
  String str=request.getParameter("keywords");
  String SQL="select * from grade where 姓名 like'%"+str+"%'";
  System.out.print(SQL);
  try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
  catch(ClassNotFoundException ce){   out.print(ce.getMessage());}
  try{ conn=DriverManager.getConnection("jdbc:odbc:grade");
  stmt=conn.createStatement(); rs=stmt.executeQuery(SQL);
 while(rs.next()){
out.print(rs.getString("学号")); out.print("|"); out.print(rs.getString("姓名"));
out.print("|");out.print(rs.getString("语文")); out.print("|");out.print(rs.getString("数
学")); out.print("|"); out.print(rs.getString("英语"));  out.print("<br>");  }  }
  catch(SQLException e){out.print(e.getMessage());  }
  finally{  stmt.close(); conn.close(); }%>
<FORM action="" method=post>
   请输入关键字：<INPUT type="text" name="keywords" value= "<%=str%>">
   <INPUT type="submit" value="执行"> </FORM>
```

5.2 格式2：实现分页显示

- 使用ResultSet的absolute()方法直接定位当某一条记录上。

程序：6-8.jsp 输出某一条记录

```
<%@ page import="java.sql.*"%>
<% Connection conn=null; Statement stmt=null; ResultSet rs=null;
 String SQL="select * from grade";
 try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
 catch(ClassNotFoundException ce){ out.print(ce.getMessage()); }
 try{conn=DriverManager.getConnection("jdbc:odbc:grade");
 stmt=conn.createStatement(
 ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
 rs=stmt.executeQuery(SQL);rs.absolute(5);out.print(rs.getString("学号"));
  out.print("|"); out.print(rs.getString("姓名")); out.print(rs.getString("语文"));
 out.print("|");out.print(rs.getString("数学")); }
 catch (SQLException e){out.print(e.getMessage());}
 finally{ stmt.close(); conn.close(); }  %>
```

程序：6-9.jsp 实现分页

```
<% Connection conn=null; Statement stmt=null;ResultSet rs=null;
 String strSQL=""; String SQL="";  int pagesize=5;  int Page=1;
 int totalpage=1; int totalrecord=0;
 try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  }
 catch(ClassNotFoundException ce){ out.print(ce.getMessage()); }
try{ conn=DriverManager.getConnection("jdbc:odbc:grade");
stmt=conn.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
strSQL="select count(*) as recordcount from grade";
rs=stmt.executeQuery(strSQL);
if(rs.next())
totalrecord=rs.getInt("recordcount");//计算总行数
SQL="select * from grade";
rs=stmt.executeQuery(SQL);
if(totalrecord%pagesize==0){
totalpage=totalrecord/pagesize; }
 else{totalpage=(int)Math.floor(totalrecord/pagesize)+1;}
```

第六章： JDBC数据访问接口

```
if(totalpage==0)totalpage=1;
if(request.getParameter("Page")==null||request.getParameter("Page").equals(""))
Page=1; try{ Page=Integer.parseInt(request.getParameter("Page")); }
catch(java.lang.NumberFormatException e){
Page=1; }
if(Page<1) Page=1; if(Page>totalpage) Page=totalpage;
rs.absolute((Page-1)*pagesize+1); out.print("<table border='1'>");
for(int ipage=1; ipage<=pagesize; ipage++){
out.print("<tr><td>"+rs.getString("学号")+"<td>");
 out.print("<td>"+rs.getString("姓名")+"<td>");
  out.print("<td>"+rs.getString("语文")+"<td>");
    out.print("<td>"+rs.getString("数学")+"<td>");
      out.print("<td>"+rs.getString("英语")+"<td>");
      if(!rs.next()) break; }
out.print("</table>"); }
 catch(SQLException e){ out.print(e.getMessage()); }
 finally{ stmt.close(); conn.close(); }
 %>
```

第六章： JDBC数据访问接口

程序：6-10.jsp 实现分页

```
<form action="7-10.jsp" method=post>
  <%  if(Page==1){
  out.print("<a href=7-10.jsp?Page="+(Page+1)+">下一页</a>");
  out.print("<a href=7-10.jsp?Page="+totalpage+">最后一页</a>");}
  else if(Page==totalpage){
  out.print("<a href=7-10.jsp?Page=1>第一页</a>");
  out.print("<a href=7-10.jsp?Page="+(Page-1)+">上一页</a>");
  }  else{
  out.print("<a href=7-10.jsp?Page=1>第一页</a>");
  out.print("<a href=7-10.jsp?Page="+(Page-1)+">上一页</a>");
   out.print("<a href=7-10.jsp?Page="+(Page+1)+">下一页</a>");
  out.print("<a href=7-10.jsp?Page="+totalpage+">最后一页</a>");
  }  %>
页数：<%=Page%>/<%=totalpage%>  输入页数：<INPUT type="text" name="Page"
size=2><INPUT type="submit" value="go">
    </form>
```

第六章： JDBC数据访问接口

程序：6-11.jsp 实现超链接显示

```
out.print("<td><a href=view.jsp?id="+rs.getString("学号")+">"+rs.getString("姓名
")+"</a><td>");
```

程序：view.jsp 显示超链接

```
<%  Connection conn=null;Statement stmt=null;ResultSet rs=null;
try{  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
catch(ClassNotFoundException ce){ out.print(ce.getMessage()); }
 String str=request.getParameter("id");
  try{  conn=DriverManager.getConnection("jdbc:odbc:grade");
stmt=conn.createStatement();
String SQL="select * from grade where 学号="+str; rs=stmt.executeQuery(SQL);
if(rs.next()){  out.print(rs.getString("姓名")+"的成绩为：");
out.print("<br>语文成绩："+rs.getString("语文"));
out.print("<br>数学成绩："+rs.getString("数学"));
out.print("<br>英语成绩："+rs.getString("英语"));}  }
catch(SQLException e){  out.print(e.getMessage());}
finally{  stmt.close();  conn.close();  }  %>
```

5 使用JavaBean访问数据库

- Select 使用executeQuery()方法，返回rs对象；
- DML语句使用executeUpdate()方法，返回整数；
- 所以可以在JavaBean中队select和DML提供处理的方法。

程序：javabean操作数据库 dbconn.java

```
 package db;
import java.sql.*;
public class dbconn {
String DBDriver="sun.jdbc.odbc.JdbcOdbcDriver";String
connstr="jdbc:odbc:grade"; Connection conn=null;Statement stmt=null;ResultSet
rs=null;
public dbconn(){
try{Class.forName(DBDriver);}
catch(ClassNotFoundException ce){System.out.println(ce.getMessage());} }
        public ResultSet executeQuery(String sql){
                try{conn=DriverManager.getConnection(connstr);
                        stmt=conn.createStatement();
                        rs=stmt.executeQuery(sql);}
        catch(SQLException e){System.out.println(e.getMessage());}return rs;}
        public int executeUpdate(String sql){
                int result=0;try{conn=DriverManager.getConnection(connstr);
                        stmt=conn.createStatement();
                        result=stmt.executeUpdate(sql);}
                catch(SQLException e){System.out.println(e.getMessage());}
                return result;}}
```

5 使用JavaBean访问数据库

程序：javabean操作数据库 7-12.jsp

```
<jsp:useBean id="conn" class="db.dbconn" scope="page"></jsp:useBean>
  <title>My JSP '7-12.jsp' starting page</title>
    </head>
  <body>
<%
String sql="select * from grade";
ResultSet rs=conn.executeQuery(sql);
while(rs.next()){
out.print(rs.getString("学号"));
out.print(rs.getString("语文"));
out.print(rs.getString("数学"));
out.print(rs.getString("英语"));
out.print("<br>");
}
%>
```

Web开发技术

下　课！！