

数据库操作

1、MySQL数据库

MySQL是一个小型关系型数据库管理系统，开发者为瑞典MySQL AB公司。在2008年1月16号被Sun公司收购。目前MySQL被广泛地应用在Internet上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库。

目前，MySQL的官方网站的网址是：www.mysql.com。

一、MySQL的特性

与其他的大型数据库（如Oracle、DB2、SQL Server等）相比，MySQL自有它的不足之处，如规模小、功能有限（MySQL Cluster的功能和效率都相对较差）等，但是这丝毫没有减少它受欢迎的程度。对于一般的个人使用者和中小型企业来说，MySQL提供的功能已经绰绰有余，而且由于MySQL是开放源码软件，因此可以大大降低总体拥有成本。

总结起来，MySQL大致有以下几个特性：

1. 使用C和C++编写，并使用了多种编译器进行测试，保证源代码的可移植性；
2. 支持AIX、FreeBSD、HP-UX、Linux、Mac OS、Novell Netware、OpenBSD、OS/2 Wrap、Solaris、Windows等多种操作系统；
3. 为多种编程语言提供了API。这些编程语言包括C、C++、Eiffel、Java、Perl、PHP、Python、Ruby和Tcl等；
4. 支持多线程，充分利用CPU资源；
5. 优化的SQL查询算法，有效地提高查询速度；
6. 既能够作为一个单独的应用程序应用在客户端服务器网络环境中，也能够作为一个库而嵌入到其他的软件中提供多语言支持，常见的编码如中文的GB2312、BIG5，日文的Shift_JIS等都可以用作数据表名和数据列名；
7. 提供TCP/IP、ODBC和JDBC等多种数据库连接途径；
8. 提供用于管理、检查、优化数据库操作的管理工具；
9. 可以处理拥有上千万条记录的大型数据库。

二、MySQL的管理

可以使用命令行工具管理MySQL数据库，也可以从MySQL的网站下载图形管理工具MySQL Administrator和MySQL Query Browser。

phpMyAdmin是由php写成的MySQL数据库管理程序，让管理者可用Web界面管理MySQL数据库。

另外，还有其他的GUI管理工具，例如早先的mysql-front 以及 ems mysql manager、navicat、SQLyog等等。

三、MySQL中建数据库、数据表

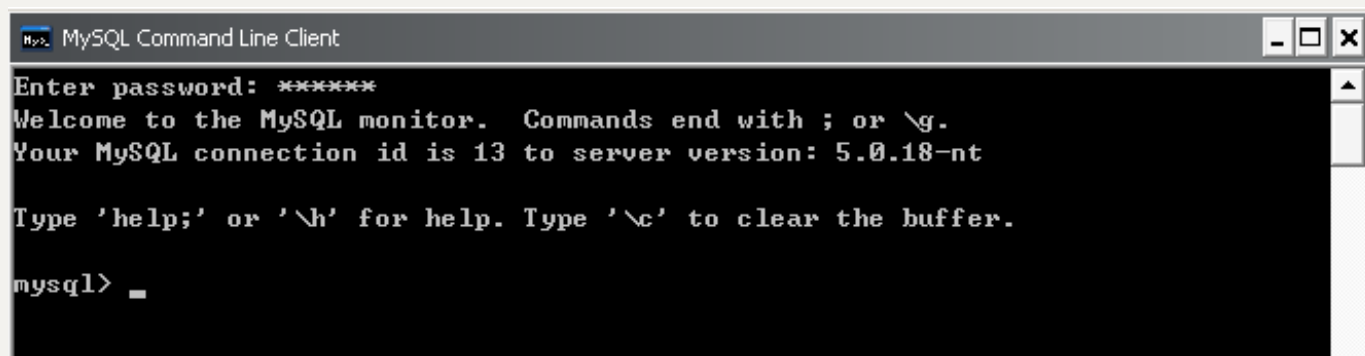
这里假设MySQL数据库安装好后的登录密码为123456，用户名为root。

首先我们介绍一下命令行方式完成建数据库、数据表的操作。

依次点击"开始程序MySQLMySQL Server 5.0MySQL Command Line Client"进入命令行客户端，如图所示。

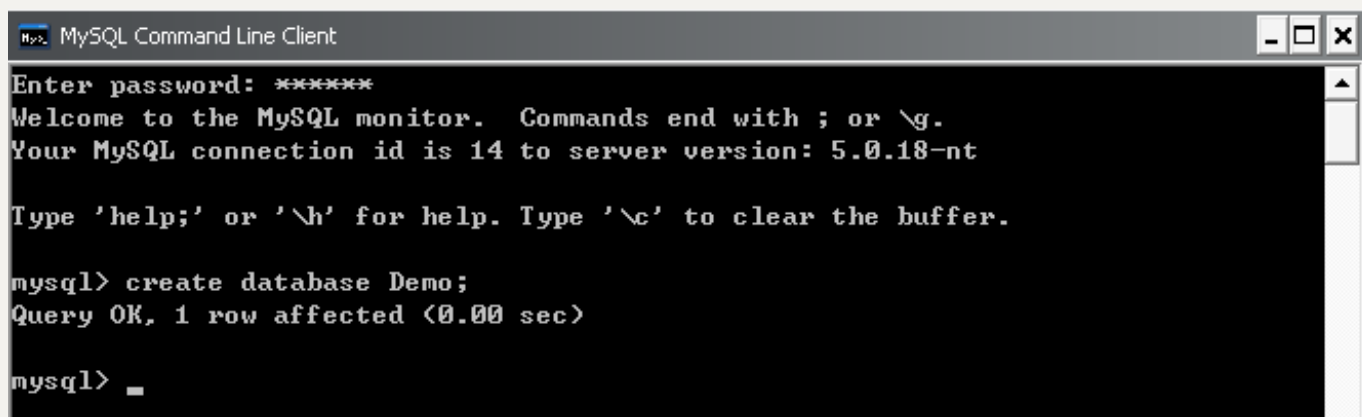


我们输入上面假设的管理员密码123456，进入如图所示。



我们就可以使用命令行方式进行数据库和数据库表的创建了。

首先我们创建一个名为Demo的数据库，我们在图6-4所示的光标处输入"create database Demo;"并回车，如图所示，看到"Query OK,1 row affected"等信息，就表示创建成功。



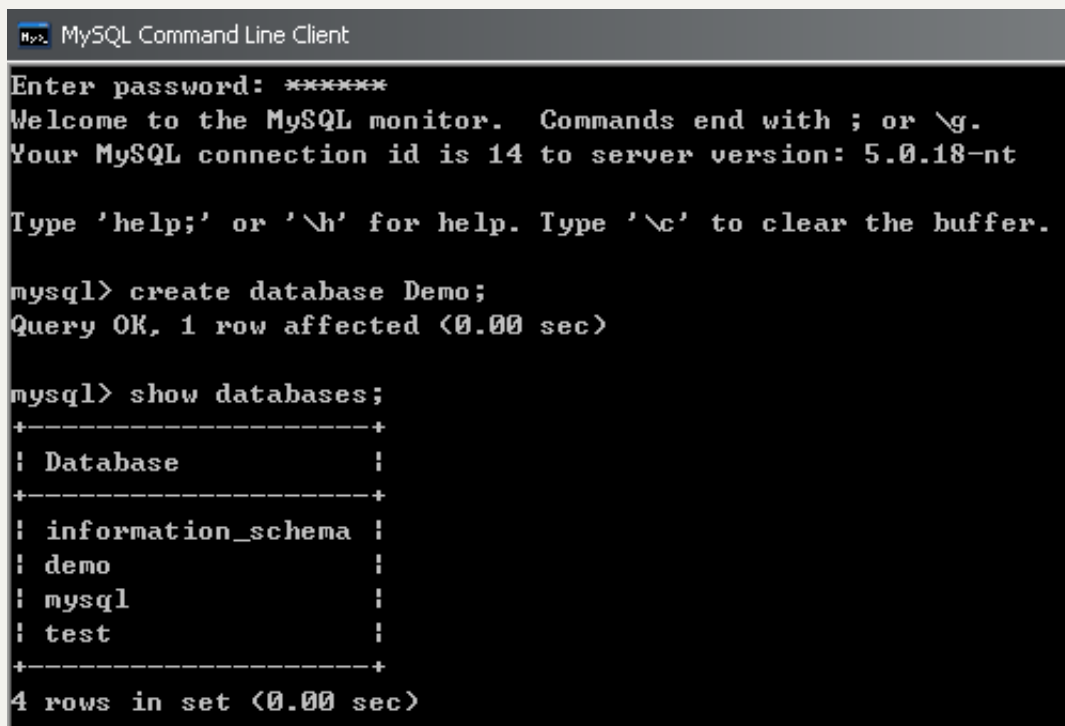
```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 5.0.18-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database Demo;
Query OK, 1 row affected (0.00 sec)

mysql> _
```

下面我们查看刚新建的数据库，输入命令"show databases;"，如图所示，可以看到demo的数据库在其中。



```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 5.0.18-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database Demo;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| demo       |
| mysql      |
| test       |
+-----+
4 rows in set (0.00 sec)
```

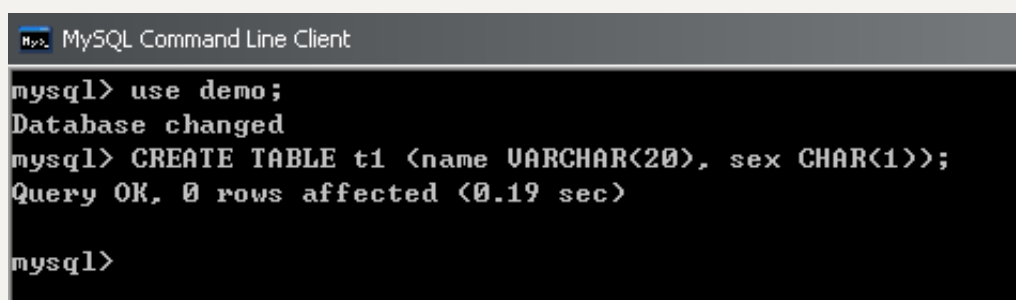
下面在刚刚新建的demo数据库中新建一个名为t1的数据表，表字段有两个，分别为name和sex。

依次输入下面两个命令：

步骤1：输入"use demo;"后会显示"Database changed"信息，说明数据库切换成功。

步骤2：输入"CREATE TABLE t1(name VARCHAR(20), sex CHAR(1));"。

运行上面两个命令后，显示结果如图所示。



```
mysql> use demo;
Database changed
mysql> CREATE TABLE t1 (name VARCHAR(20), sex CHAR(1));
Query OK, 0 rows affected (0.19 sec)

mysql>
```

显示刚新建的表，输入命令"show tables;"，结果如图所示。



```
mysql> show tables;
+-----+
| Tables_in_demo |
+-----+
| t1              |
+-----+
1 row in set (0.00 sec)

mysql>
```

进一步查看表t1的结构，输入命令"DESCRIBE t1;"，结果如图所示。

```
MySQL Command Line Client

mysql> show tables;
+-----+
| Tables_in_demo |
+-----+
| t1              |
+-----+
1 row in set (0.00 sec)

mysql> DESCRIBE t1;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20)   | YES  |     | NULL    |       |
| sex   | char(1)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.03 sec)
```

2、JDBC简介

2.1 什么是JDBC

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语言编写的类和接口组成。JDBC为数据库开发人员提供了一个标准的API, 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用纯Java API 编写数据库应用程序。

有了JDBC, 向各种关系数据发送SQL语句就是一件很容易的事。程序员只需用JDBC API写一个程序, 而不用为专门数据库写专门程序, 另外, 还可以很好地利用JAVA的"一次编译、到处运行"的优势, 编写的数据库应用程序不需要考虑平台相关性。

2.2 JDBC的用途

一般来说, JDBC可以完成以下工作:

- 和一个数据库建立连接;
- 向数据库发送SQL语句;
- 处理数据库返回的结果。

2.3 JDBC的入门

本节内容主要讲解JDBC如何与数据库建立一个连接、如何向数据库发送SQL语句、如何处理数据库返回的结果。

1. 建立连接

建立一个连接又包含2个步骤：加载驱动程序、建立连接。

1. 加载驱动程序

大多数的数据库产品都有相应的JDBC驱动程序jar包，我们只需上网下载相应的驱动程序即可。下载完成后导入到工程内，加载驱动程序很简单，只需一行代码。

比如MySQL驱动程序类名为org.gjt.mm.mysql.Driver，那么将用以下的代码装载驱动程序：

```
Class.forName("org.gjt.mm.mysql.Driver");
```

实际上，上面org.gjt.mm.mysql.Driver是MySQL数据库的JDBC驱动程序类的其中一种写法。

这里，我们不需要创建驱动程序类的实例，是由驱动管理器类（即DriverManager类）去登记它。调用了Class.forName()后会自动加载驱动程序类。

加载了驱动后，接下来我们就可以建立与数据库的连接。

2. 建立连接

下面就是用适当的驱动程序类与数据库建立连接。

下面是一般的格式：

```
Connection conn = DriverManager.getConnection(url, "username",  
"password");
```

这里有3个参数，第二、三个参数比较简单，就是提供登录数据库管理系统的用户名和密码，关键是第一个url比较难。

如果直接采用第三方JDBC驱动程序，那么url的形式要由根据特定的JDBC驱动程序决定。

比如，连接MySQL数据库，那么url的形式：

```
"jdbc:mysql://127.0.0.1/jspdev?useUnicode=true&characterEncoding=gb2312"
```

此时，如果登录DBMS的用户名和密码分别为admin和123，那么建立连接的代码就如下形式：

```
String url="jdbc:mysql://127.0.0.1/db ";
```

```
Connection conn = DriverManager.getConnection(url, "admin",  
"123");
```

2. 向数据库发送SQL语句

主要是利用上面返回的连接对象创建Statement对象，接着是利用Statement对象的各个方法执行不同的SQL语句，从而实现对数据库的不同操作。

实际上有三种Statement对象：Statement、PreparedStatement（它从Statement继承而来）和CallableStatement（它从PreparedStatement继承而来）。

它们都专用于发送特定类型的SQL语句：

- Statement 对象用于执行不带参数的简单SQL 语句；
- PreparedStatement 对象用于执行带或不带 IN 参数的预编译SQL语句；
- CallableStatement 对象用于执行对数据库存储过程的调用。

1. 创建 Statement 对象

Statement对象用数据库连接对象调用createStatement方法创建。

如下面代码所示：

```
Connection conn=DriverManager.getConnection(url,user,password);
```

```
Statement stmt=conn.createStatement();
```

为了执行Statement对象，SQL语句将作为参数提供给它的方法调用：

```
String sql="select * from admin";
```

```
ResultSet rs=stmt.executeQuery(sql);
```

上面代码返回一个记录集，包含了检索出的所有数据。

2. 使用 Statement 对象执行语句

Statement 接口提供了三种执行SQL语句的方法：executeQuery、executeUpdate 和 execute。使用哪一个方法由SQL语句所产生的内容决定。

- executeQuery

用于产生单个结果集的语句，例如 SELECT 语句。

- executeUpdate

用于执行INSERT、UPDATE或DELETE语句以及 SQL DDL（数据定义语言）语句。

- execute

用于执行返回多个结果集、多个更新计数或二者组合的语句。

1. 语句完成

当连接处于自动提交模式时，其中所执行的语句在完成时将自动提交或还原。语句在已执行且所有结果返回时，即认为已完成。对于返回一个结果集的 executeQuery 方法，在检索完 ResultSet 对象的所有行时该语句完成。对于方法 executeUpdate，当它执行时语句即完成。但在少数调用方法 execute 的情况中，在检索所有结果集或它生成的更新计数之后语句才完成。

2. 关闭 Statement 对象

Statement对象将由 Java

垃圾收集程序自动关闭。而作为一种好的编程风格，应在不需要Statement对象时显式地关闭它们，将立即释放资源，有助于避免潜在问题。

3. 处理数据库返回的结果

如果是查询数据，那么返回的是一个记录集，我们就需要对记录集进行处理。

SQL语句对数据库的查询操作将返回一个ResultSet对象，ResultSet对象是以统一形式的列组织的数据行组成。ResultSet对象一次只能看到一个数据行，使用next()方法走到下一数据行，获得一行数据后，ResultSet对象可以使用getxxxx方法获得字段值，将位置索引（第一列使用1，第二列使用2等等）或字段名传递给getxxxx方法的参数即可。

如下所示，是ResultSet对象的若干方法。

返回类型	方法名称
boolean	next()
byte	getBytes(int columnIndex)
Date	getDate(int columnIndex)
double	getDouble(int columnIndex)
float	getFloat(int columnIndex)
int	getInt(int columnIndex)
long	getLong(int columnIndex)
String	getString(int columnIndex)
byte	getBytes(String columnName)
Date	getDate(String columnName)

double	getDouble(String columnName)
float	getFloat(String columnName)
int	getInt(String columnName)
long	getLong(String columnName)
String	getString(String columnName)

例如，返回一个名为rs的记录集，那么循环输出每条记录的第一个字段和第二个字段的内容，代码如下：

```
while(rs.next()){%>
```

```
    第一个字段内容为: <%=rs.getString(1)%><br>
```

```
    第二个字段内容为: <%=rs.getString(2)%><br>
```

```
<%}%>
```

如果是执行更新操作，那么返回的是受影响的行数。比如往数据表插入一条记录，那么最后会返回一个整数1（如果将executeUpdate执行后结果赋值给某个变量的话），代码如下：

```
String sql="INSERT INTO table1 VALUES (对应的字段列表值)"
```

```
stmt.executeUpdate(sql);
```

3、数据库的连接过程

1). 安装JDBC驱动

由于MySQL数据库版本是mysql-5.7.17，所以我们也要找到相应的JDBC驱动版本，这里使用的是mysql-connector-java-5.0.3-bin.jar（可以在百度等搜索引擎搜索）。

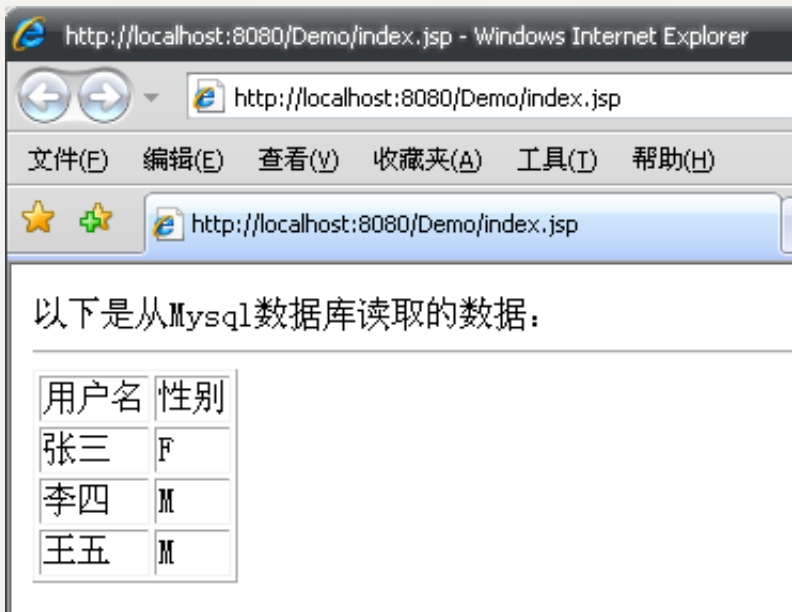
2). 操作实践

新建web工程，名为Demo，将mysql-connector-java-5.0.3-bin.jar拷贝到web工程内，再新建一个JSP页面，用于测试。连接的数据库是上面用命令行创建的demo数据库，里面含有一个表t1。

index.jsp

```
<%@ page language="java" import="java.sql.*" pageEncoding="gb2312"%>
<html>
<body>
    以下是从Mysql数据库读取的数据:
    <hr>
    <table border=1>
        <tr>
            <td>用户名</td>
            <td>性别</td>
        </tr>
    <%
Class.forName("org.gjt.mm.mysql.Driver");
Connection con= DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/demo ","root","123");
Statement stmt=con.createStatement();
ResultSet rst=stmt.executeQuery("select * from t1;");
while(rst.next())
{
    out.println("<tr>");
    out.println("<td>"+rst.getString("name")+"</td>");
    out.println("<td>"+rst.getString("sex")+"</td>");
    out.println("</tr>");
}
//关闭连接、释放资源
rst.close();
stmt.close();
con.close();
%>
    </table>
</body>
</html>
```

运行结果，如图所示。



以上就是连接MySQL数据库的测试页面。

4、数据操作

数据的4大基本操作：查询、更新、添加和删除。

这里新建一个名为demo的MySQL数据库进行演示。

demo数据库中新建一个表student，表中有4个字段，如图所示。

对象

student@demo (localhost)

📄

📄 +

📄 ←

📄 -

字段

索引


外键

触发器

选项

注释

SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		关键字ID
username	char	20	0	<input type="checkbox"/>	<input type="checkbox"/>		姓名
math	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		数学
english	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		英语

连接代码如下：

```
Class.forName("org.gjt.mm.mysql.Driver ");
String url= jdbc:mysql://localhost:3306/demo";
// 用户名root, 密码123
Connection conn = DriverManager.getConnection(url,"root","123");
```

本节的所有操作都要用到这个连接对象conn，所以提到开头部分来写。本节里面凡是提到连接对象conn就是指这里创建的数据库连接对象。

4.1 添加记录

要添加记录到数据库，必须首先建立与数据库的连接，这里直接使用已创建好的连接对象conn，使用上面创建的conn对象调用方法createStatement()创建Statement对象。

```
Statement stmt=conn.createStatement();
```

接着再利用stmt对象调用executeUpdate方法来执行insert语句。

实例：

在web应用Demo中新建一个文件夹score，里面全部存放以MySQL数据库为存储数据的成绩管理模块，新建一个insert.jsp作为数据添加页面，所有代码都在insert.jsp内完成。

insert.jsp源代码如下：

```
<%@ page language="java" import="java.sql.*" pageEncoding="gb2312"%>
<html>
<body>
    插入数据到MySQL数据库:
    <hr>
    <% //加载驱动
        Class.forName("org.gjt.mm.mysql.Driver ");
        String url= jdbc:mysql://localhost:3306/demo";
        // 用户名root, 密码123
        Connection conn = DriverManager.getConnection(url,"root","123");
```

```
Statement stmt = conn.createStatement();//创建Statement对象
String sql = "insert into student(username,math,english) values('张三',80,90)";
stmt.executeUpdate(sql);
//关闭连接、释放资源
stmt.close();
conn.close();
%>
</body>
</html>
```

运行结果

部署web应用，启动TOMCAT后执行insert.jsp，接着打开数据库中的student表，可以看到数据已经正常地插入到了表中，如图所示。

	id	username	math	english
	1	张三	80	90

添加数据

同样，我们修改插入数据内容，并多次执行insert.jsp后，可以发现数据表中插入了多条数据，如图所示。

	id	username	math	english
	1	张三	80	90
	2	李四	70	90
	3	王五	77	80
	4	赵六	87	70
	5	赵五	77	70
	6	赵四	79	60
	7	老李	99	70
	8	张六	59	70

添加多条数据后结果

4.2 查询记录

要查询数据库中的记录，必须首先建立与数据库的连接。

使用上面创建的conn对象调用方法createStatement()创建Statement对象。

```
Statement stmt=conn.createStatement();
```

最后利用stmt对象调用executeQuery方法进行查询。

```
ResultSet rs=sql.executeQuery("SELECT * FROM student");
```

返回一个记录集对象rs，该对象一次只能看到一行数据，使用next()方法到下一行数据，获得一行数据后，rs可以使用getxxxx方法获得字段值，将位置索引（第一列使用1，第二列使用2等等）或字段名传递给getxxxx方法的参数即可。

1). 参数查询

根据用户输入的条件进行查询。

在Demo工程内的文件夹score中添加query3.jsp和query3_result.jsp两个页面，在query3.jsp输入用户名，在query3_result.jsp显示相应的数据。

这里需要构造一个带where条件的SQL语句，即

```
String sql = "select * from student where username='"+param+"'";
```

query3.jsp

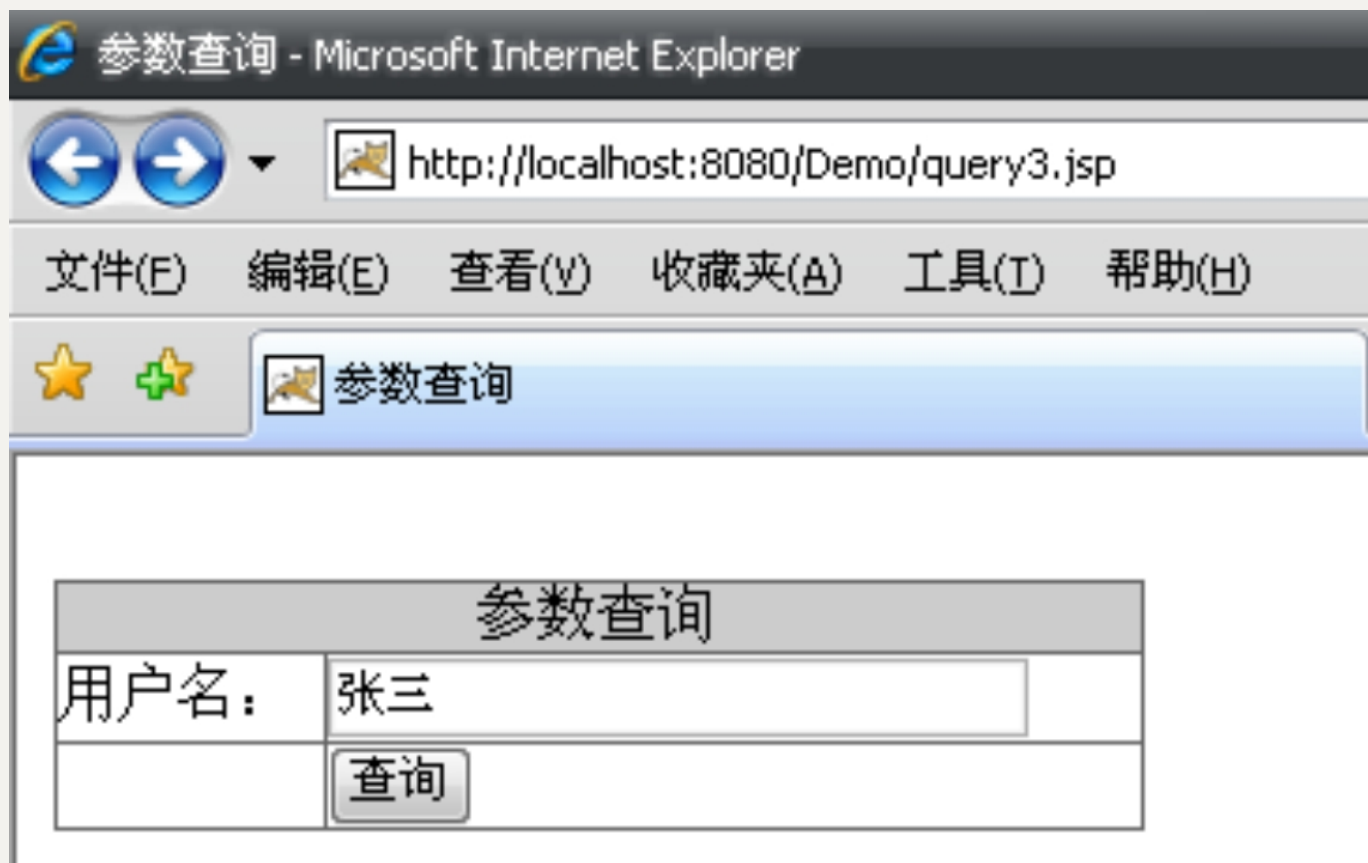
```
<%@ page language="java" pageEncoding="gb2312"%>
<html>
  <head>
    <title>参数查询</title>
  </head>
  <body>
    <br>
    <form name="form1" method="post" action="query3_result.jsp">
      <table width="291" border="0" cellpadding="0" cellspacing="1"
bgcolor="#666666">
        <tr>
          <td colspan="2" align="center" bgcolor="#CCCCCC">参数查询</td>
        </tr>
        <tr>
          <td width="68" bgcolor="#FFFFFF">用户名: </td>
          <td width="207" bgcolor="#FFFFFF"><input name="username"
type="text" id="username" size="25"></td>
        </tr>
        <tr>
          <td bgcolor="#FFFFFF">&nbsp;</td>
          <td bgcolor="#FFFFFF"><input type="submit" name="Submit" value="查
询"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```


query3_result.jsp

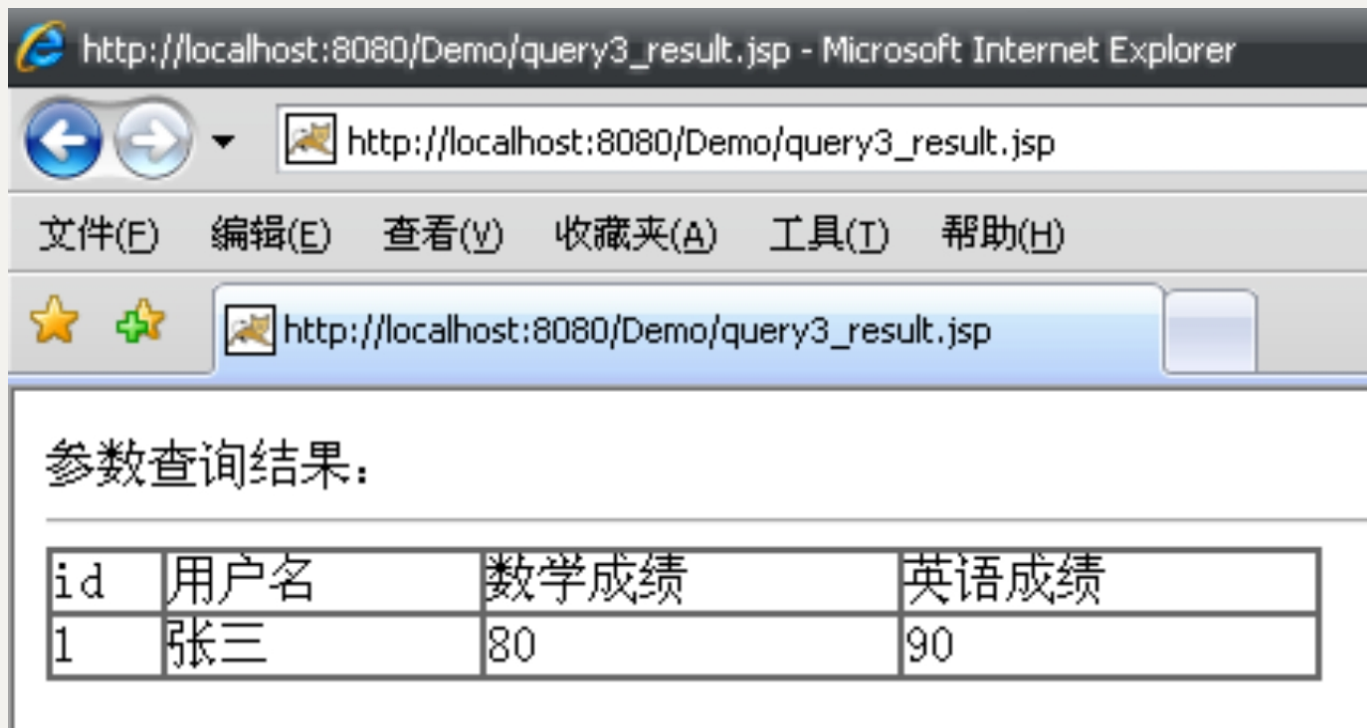
```
<%@ page language="java" import="java.sql.*" pageEncoding="gb2312"%>
<html>
  <body>
    参数查询结果:
    <hr>
    <table width="400" border="1" cellpadding="0" cellspacing="0"
bordercolor="#666666">
      <tr>
        <td bgcolor="#FFFFFF">id</td>
        <td bgcolor="#FFFFFF">用户名</td>
        <td bgcolor="#FFFFFF">数学成绩</td>
        <td bgcolor="#FFFFFF">英语成绩</td>
      </tr>
      <% //加载驱动
        Class.forName("org.gjt.mm.mysql.Driver ");
        String url= jdbc:mysql://localhost:3306/demo";
        // 用户名root, 密码123
        Connection conn = DriverManager.getConnection(url,"root","123");
        //创建Statement对象
        Statement stmt = conn.createStatement();
        //发送SQL语句
        request.setCharacterEncoding("gb2312");
        String param = request.getParameter("username");
        String sql = "select * from student where username='"+param+"'";
        ResultSet rs = stmt.executeQuery(sql);
        //显示记录
        while(rs.next()){
          out.print("<tr>");
          out.print("<td>"+rs.getInt("id")+"</td>");
          out.print("<td>"+rs.getString("username")+"</td>");
          out.print("<td>"+rs.getInt("math")+"</td>");
          out.print("<td>"+rs.getInt("english")+"</td>");
          out.print("</tr>");
        }
        //关闭连接、释放资源
        rs.close();
        stmt.close();
      <%>
    </table>
  </body>
</html>
```

```
conn.close();  
%>  
</table>  
</body>  
</html>
```

“ 执行query3.jsp并输入“张三”后，显示结果如图所示。



参数查询 【1】



参数查询 【2】

说明数据表中有“张三”这条记录。

2. 通配符查询

根据上面的参数查询，我们可以发现一个问题，就是只能精确查询，如果需要模糊查询，怎么办呢？那就需要用到这里要讲的通配符查询。

可以用SQL语句操作符LIKE进行模式匹配，使用“%”代替一个或多个字符，用一个下划线“_”代替一个字符。

比如要查找所有姓“张”的人的成绩，那么sql语句可以如下：

```
String sql = " select * from student where username like '张%' ";
```

4.3 更新记录

要更新数据库中的记录，必须首先建立与数据库的连接。

使用上面创建的conn对象调用方法createStatement()创建Statement对象。

```
Statement stmt=conn.createStatement();
```

最后利用stmt对象调用executeUpdate方法进行更新。

```
String sql = "UPDATE student SET math =100 WHERE username like '张%' ;  
stmt.executeUpdate(sql);
```

注意：如果上面不加WHERE条件，那么表中所有人的数学成绩都被更新为100分。

4.4 删除记录

要删除数据库中的记录，必须首先建立与数据库的连接。

使用上面创建的conn对象调用方法createStatement()创建Statement对象。

```
Statement stmt=conn.createStatement();
```

最后利用stmt对象调用executeUpdate方法进行删除。

```
String sql = "delete from student where id=1" ;  
stmt.executeUpdate(sql);
```

注意：如果上面不加WHERE条件，那么表中所有数据都被删除。