第5章　数据库技术的使用

主要内容

1. JDBC简介
2. JDBC常用接口
3. 连接数据库
4. 在JSP中使用JDBC访问数据库

5.1 JDBC简介

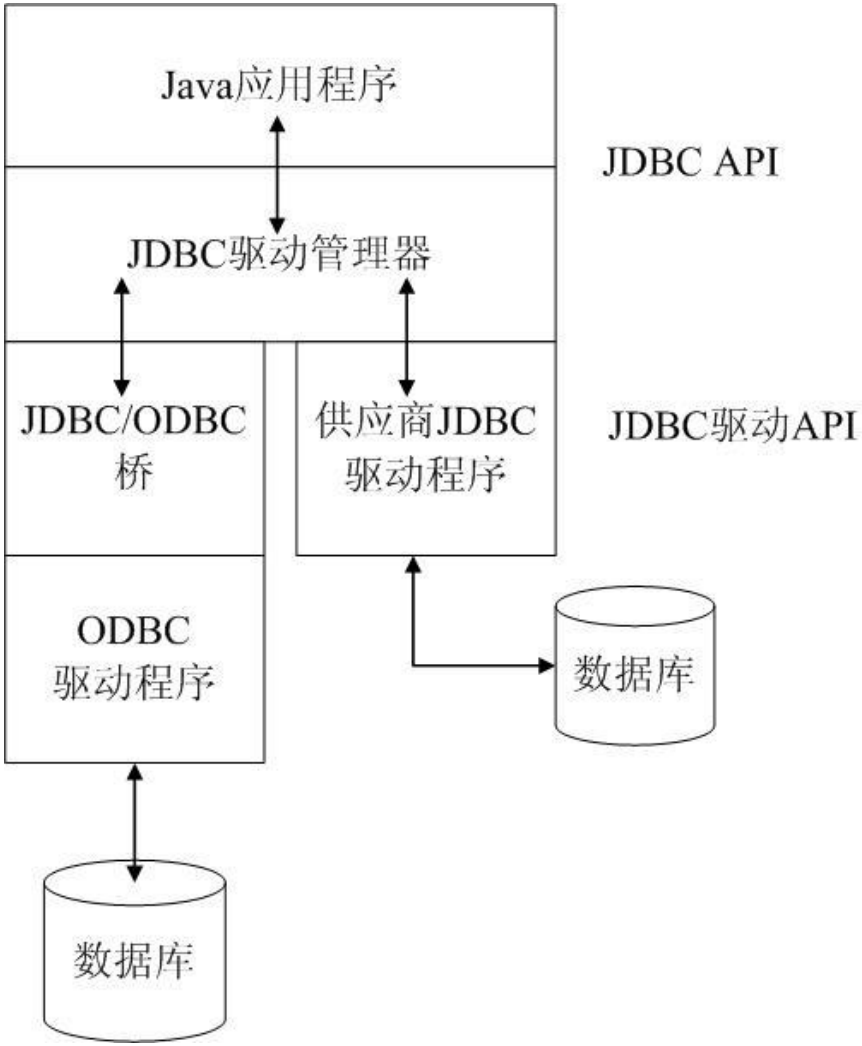
❖JDBC简介

JDBC是**J**ava **D**ata**b**ase **C**onnectivity（Java数据库连接）的缩写，编程人员可以通过这个API接口连接到数据库，并使用结构化查询语言（SQL）完成对数据库的查找和更新。

JDBC的目标是屏蔽不同的数据库驱动程序之间的差别，为开发者提供一个标准的、纯Java的数据库程序设计接口，为在Java中访问不同类型的数据库提供技术支持。

5.1 JDBC简介

JDBC体系结构



5.1 JDBC简介

JDBC接口包含两层：

1) JDBC API：负责与JDBC驱动程序管理器API进行通信，供应用程序开发人员使用。

2) JDBC驱动API：JDBC驱动程序管理器与实际连接到数据库的第三方驱动程序进行通信，供各数据库供应商和工具开发商使用。

5.1 JDBC简介

JDBC API的作用主要有以下三个方面：

- 建立与数据库的连接。
- 向数据库发起查询、更新请求。
- 处理数据返回的结果。

5.1 JDBC简介

❖数据库驱动程序

JDBC驱动程序可以归结为以下四类：

- Type 1：JDBC-ODBC桥
- Type 2：本地API
- Type 3：JDBC-Net
- Type 4：本地协议

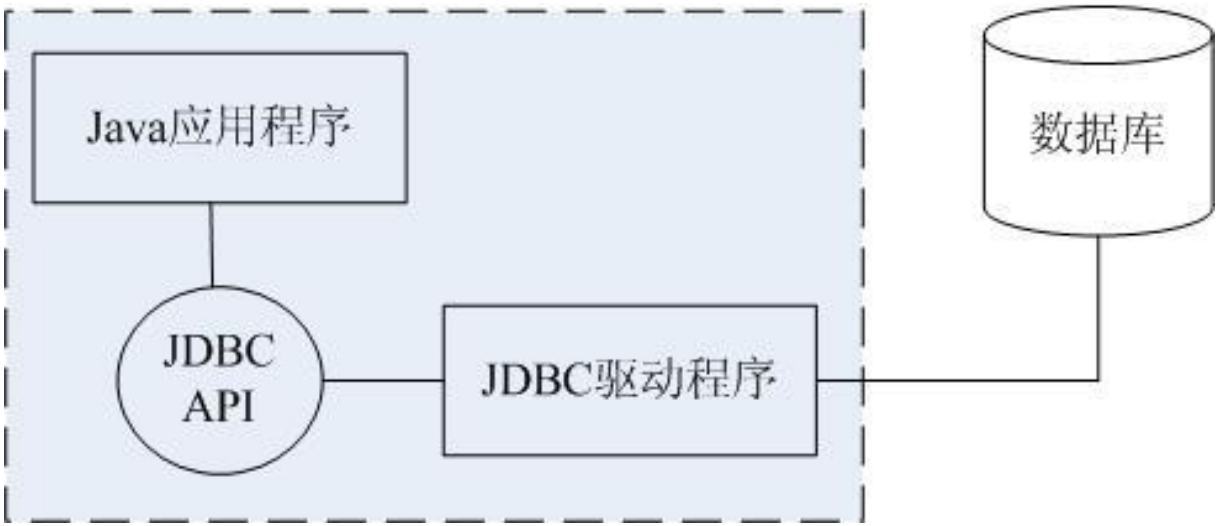
5.1 JDBC简介

Type 4：本地协议

这种类型的JDBC驱动程序完全用Java编写，采用数据库支持的网络协议把JDBC API调用转换为直接的网络调用。

这类驱动程序是最高效的数据访问方式，目前常见的Java应用程序都使用这一类的JDBC驱动程序访问数据库。但是，访问不同厂商的数据库，需要不同的JDBC驱动程序。Oracle、Microsoft和Sybase等几个主要的数据库厂商都为各自的数据库产品提供了这种类型的驱动，可以去该公司官网进行匹配下载。

5.1 JDBC简介



通过本地协议的纯Java驱动程序访问数据库

主要内容

1. JDBC简介
2. JDBC常用接口
3. 连接数据库
4. 在JSP中使用JDBC访问数据库

5.2 JDBC常用接口

JDBC 包括java.sql和javax.sql两个核心包。

其中包括了一些常用的接口，如：Driver、Connection、Statement和ResultSet等，每一个厂家的JDBC驱动程序中都提供了对这些接口的实现类。

5.2 JDBC常用接口

❖Driver接口

所有JDBC驱动程序都必须实现Driver接口，在该接口中提供了一个重要的方法connect()，它用来建立到数据库的连接，其定义如下：

Connection connect (String url, Properties info) throws SQLException

5.2 JDBC常用接口

不同厂商实现Driver接口的类名是不相同的，下面列出了几个常用数据库的JDBC驱动类名：

- SQL Server

  com.microsoft.sqlserver.jdbc.SQLServerDriver

- Oracle

  oracle.jdbc.driver.OracleDriver

- MySQL

  com.mysql.jdbc.Driver

5.2 JDBC常用接口

❖DriverManager类

DriverManager是驱动程序管理类，负责建立数据库的连接以及管理JDBC驱动程序。

5.2 JDBC常用接口

DriverManager的常用方法

方法	描述
registerDriver(Driver driver)	注册JDBC驱动程序。
getConnection(String url, String user, String password)	建立和数据库的连接，返回Connection对象。
setLoginTimeout(int seconds)	设定等待数据库连接的最长时间。

5.2 JDBC常用接口

❖Connection接口

Connection代表与数据库的连接，主要方法如下：

方法	描述
getMetaData()	返回数据库的元数据，它包含了数据库的相关信息，如当前数据库连接的用户名、使用的JDBC驱动程序，以及数据库的版本等。
createStatement()	创建并返回Statement对象。
prepareStatement()	创建并返回PreparedStatement对象。
close()	关闭当前连接以及释放由它所创建的JDBC资源。

5.2 JDBC常用接口

❖Statement接口

在建立与数据库的连接后，需要对数据库进行访问，执行SQL语句。Statement执行不带参数的简单SQL语句。

5.2 JDBC常用接口

方法	描述
executeQuery(String sql)	执行参数sql指定的SQL语句，并返回一个ResultSet对象用于查看执行的结果。
executeUpdate(String sql)	执行参数sql指定的Insert、Update或者Delete语句，也可以用于执行DDL语句，如create table。
execute(String sql)	执行返回多个结果集的SQL语句。通常情况下不使用该方法，除非要执行一个返回多个结果集的存储过程或者动态执行一个未知的SQL语句。
executeBatch()	该方法允许向数据库提交一批命令，然后一起执行。
getConnection()	获取对数据库的连接。
getFetchSize()	获取返回的数据库结果集行数。
setFetchSize(int rows)	设置返回的数据库结果集行数。
close()	关闭Statement对象及它所对应的结果集。

5.2 JDBC常用接口

PreparedStatement接口

PreparedStatement接口继承了Statement接口，但它包含了经过预编译的SQL语句，因此获得更高的执行效率。

在PreparedStatement语句中可以包含多个用“?”代表的字段，在程序中可以利用setXXX方法设置该字段的内容，从而增强了程序设计的动态性。

```
PreparedStatement pstmt=conn.prepareStatement(“UPDATE SCORE SET
C_SCORE=? WHERE ID=?”);
    pstmt.setInt(1, 100);          //设置第一个字段C_SCORE，1表示第一个?
    pstmt.setInt(2, 1);            //设置第二个字段ID，2表示第二个?
```

5.2 JDBC常用接口

❖ResultSet接口

ResultSet用来存放数据库查询操作获得的结果，它包含了满足SQL语句条件的所有行。

5.2 JDBC常用接口

ResultSet的常用方法

方法	描述
next()	将结果集中的当前行向后移动一行。如果已经到达最后一行的后面，则返回false。注意：初始情况下必须调用该方法才能转到第一行。
getXXX(int columnNumber)	用给定的列序号返回该列的值，并将值转换成指定的类型（XXX指数据类型，如int、double、String等）。
getXXX(String cloumnName)	用给定的列标签返回该列的值，并将值转换成指定的类型（XXX指数据类型，如int、double、String等）。
close()	关闭当前的结果集。

5.2 JDBC常用接口

❖通过JDBC访问数据库的步骤

在Java程序中，通过JDBC访问数据库的步骤如下：

①装载并注册数据库的JDBC驱动程序；

②建立与数据库的连接；

③创建Statement对象；

④调用SQL语句访问数据库；

⑤处理ResultSet中的记录集；

⑥关闭ResultSet、Statement和Connection对象。

主要内容

1. JDBC简介
2. JDBC常用接口
3. 连接数据库
4. 在JSP中使用JDBC访问数据库

5.3 连接数据库

❖数据库URL

要建立与数据库的连接，首先要创建指定数据库的URL。

连接通常是通过数据库的URL对象，利用DriverManager的getConnection( )方法建立的。

数据库URL对象与网络资源的统一资源定位类似，其构成的一般格式如下：

jdbc:subProtocol:other stuff

5.3 连接数据库

实例

- MS SQL server

jdbc:Microsoft:sqlserver://localhost:1433;DatabaseName=STUDENT

5.3 连接数据库

❖注册驱动程序

DriveManager类用于选择数据库驱动程序和创建新的数据库连接。注意，只有在驱动管理器中注册过的驱动程序才可以被激活。在JDBC中，有两种注册驱动程序的方式：

- 将驱动程序添加到java.lang.System的属性jdbc.drivers中。
- 在程序中利用Class.forName( )方法注册指定的驱动程序。
- Class.forName( )方法注册SQL SERVER示例

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver"
);
```

5.3 连接数据库

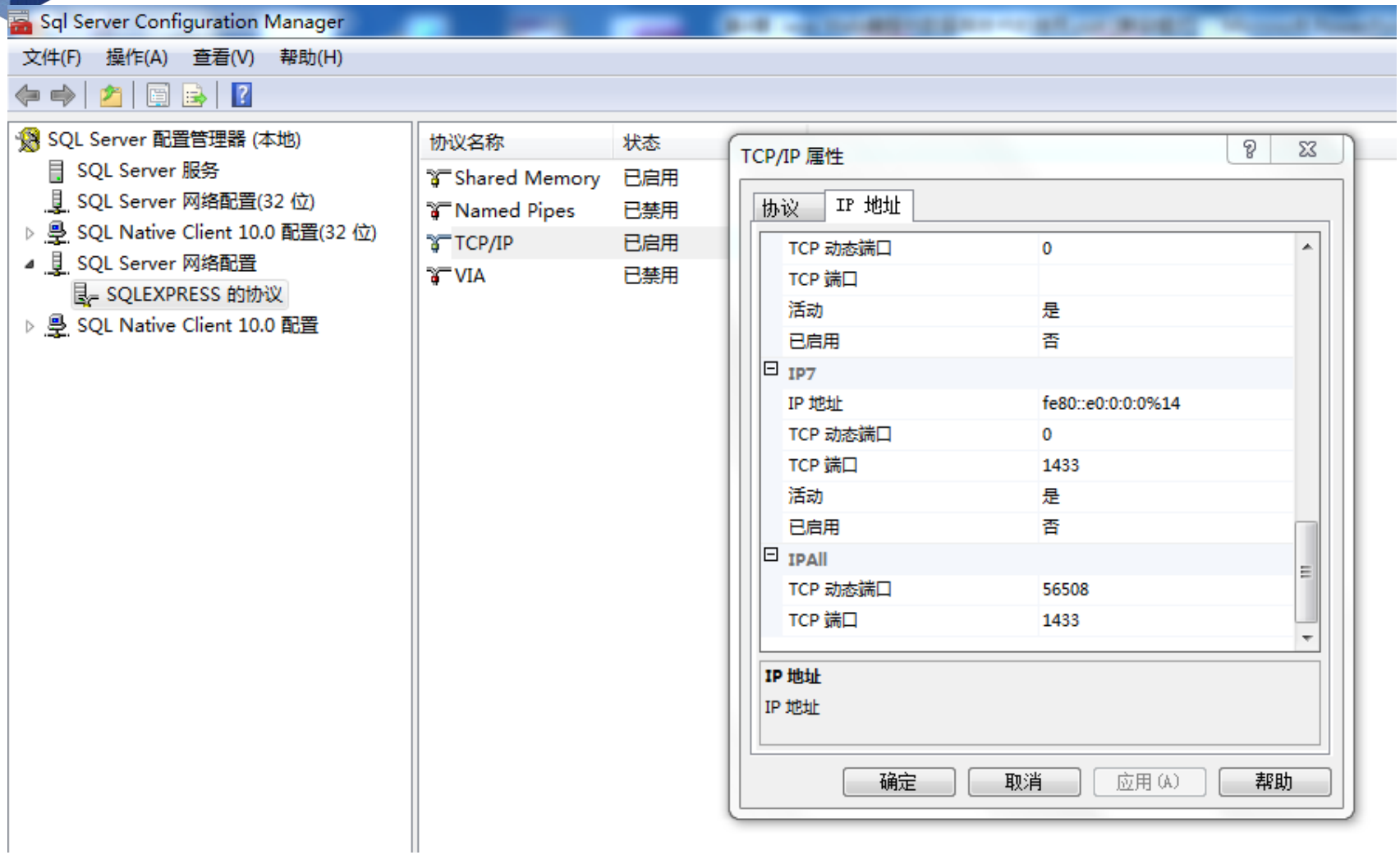
❖ 使用专用JDBC驱动程序连接数据库时需要做一些配置，以SQL Server 为例

1) 修改SQL Server 网络配置

启动SQL Server Configuration Manager，选择“SQL Server 网络配置→MSSQLSERVER的协议”，启用TCP/IP协议。

2) 修改IPALL属性TCP端口为1433.

5.3 连接数据库


Sql Server Configuration Manager
文件(F) 操作(A) 查看(V) 帮助(H)
SQL Server 配置管理器 (本地)
SQL Server 服务
SQL Server 网络配置(32 位)
SQL Native Client 10.0 配置(32 位)
SQL Server 网络配置
SQLEXPRESS 的协议
SQL Native Client 10.0 配置
协议名称
状态
Shared Memory 已启用
Named Pipes 已禁用
TCP/IP 已启用
VIA 已禁用
TCP/IP 属性
协议
IP 地址
TCP 动态端口 0
TCP 端口
活动 是
已启用 否
IP7
IP 地址 fe80::e0:0:0:0%14
TCP 动态端口 0
TCP 端口 1433
活动 是
已启用 否
IPAll
TCP 动态端口 56508
TCP 端口 1433
IP 地址
IP 地址
确定
取消
应用(A)
帮助

2) 更改SQL Server 的验证模式

启动SQL Server Management Studio，并以“Windows身份验证方式”连接数据库。

连接后，在左边的窗口选中实例，然后单击右键，在弹出的菜单中选择“属性”。在服务器属性对话框中，选择“安全性”，然后将服务器身份验证模式修改为“SQL Server和Windows身份验证模式”。

修改后，重新启动SQL Server 的服务。

5.3 连接数据库

修改身份验证模式

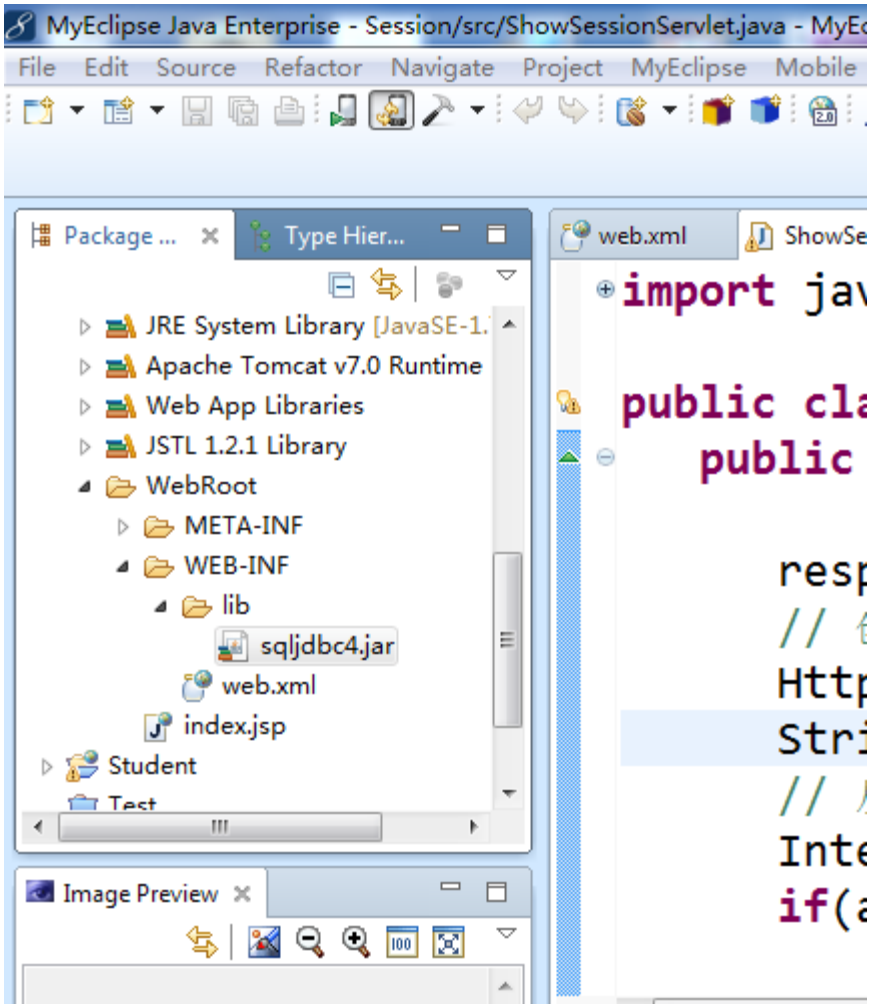
3) 使用JDBC Driver

Microsoft SQL Server JDBC Driver 提供了两个类库文件：sqljdbc.jar和sqljdbc4.jar，具体使用哪个文件取决于使用的Java运行时环境（JRE）设置：

1)sqljdbc.jar类库提供了对JDBC 3.0的支持，它要求使用5.0版的Java运行时环境，在JRE 6.0上使用会引发异常；

2)sqljdbc4.jar类库提供了对JDBC 4.0的支持，它不仅包括了sqljdbc.jar的所有功能，还包括新增的JDBC 4.0的方法，该类库要求使用6.0或者更高版本的Java运行时环境。

导入数据驱动sqljdbc4.jar



主要内容

1. JDBC简介
2. JDBC常用接口
3. 连接数据库
4. 在JSP中使用JDBC访问数据库

5.4 在JSP中使用JDBC访问数据库

在JSP页面中使用JDBC访问数据库可以采用下面5个步骤：

①在页面的开头使用page指令导入java.sql.*；

②打开数据库连接；

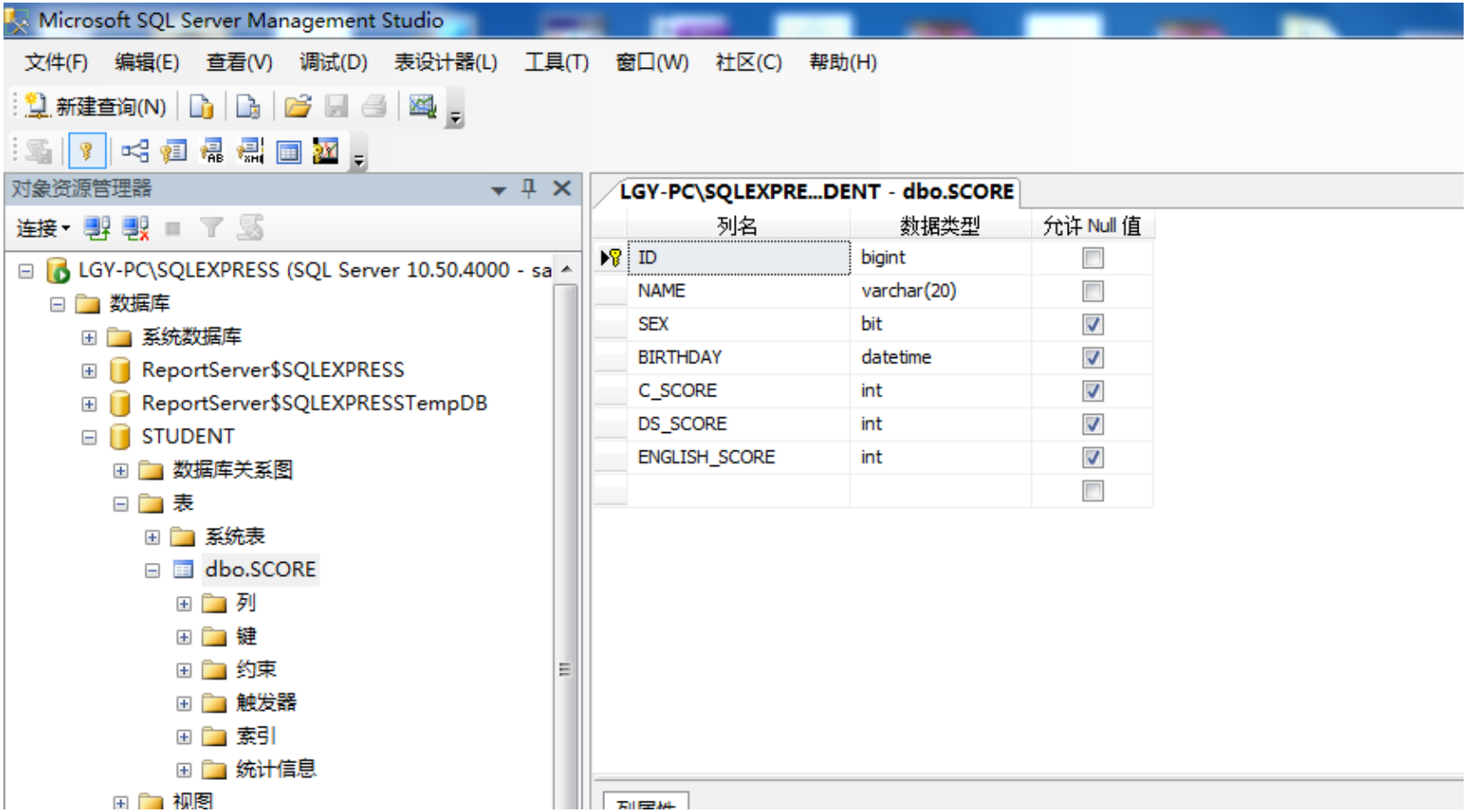
③执行SQL语句；

④处理返回结果；

⑤关闭数据库连接。

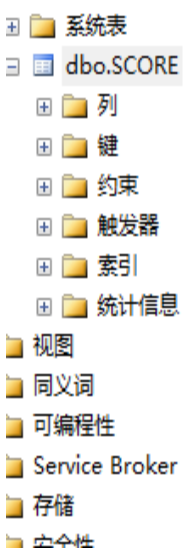
```
<%@ page language="java" import="java.sql.*" pageEncoding="UTF-8"%>
<html>
<head><title>创建学生成绩表</title></head>
<body>
<%
   Connection connection = null;
   Statement stmt = null;
   final String url ="jdbc:sqlserver://localhost:1433;databaseName=STUDENT";
   final String username = "sa"; //访问数据库的用户名
   final String password = "sasa"; //访问数据库的密码
   try {
      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      connection = DriverManager.getConnection(url, username, password);
      stmt = connection.createStatement();
      stmt.executeUpdate("CREATE TABLE SCORE ( "
            + "ID bigint NOT NULL ,"
            + "NAME varchar(20) NOT NULL ,“
            + "SEX bit,BIRTHDAY datetime," + "C_SCORE int,"
            + "DS_SCORE int," + "ENGLISH_SCORE int,"
            + "PRIMARY KEY (ID))");
            }
    catch (SQLException e)
        {e.printStackTrace();}
    finally {
        if(stmt!=null){ stmt.close(); }
         if(connection!=null){connection.close(); }}
         out.println("Success!");
%>
```

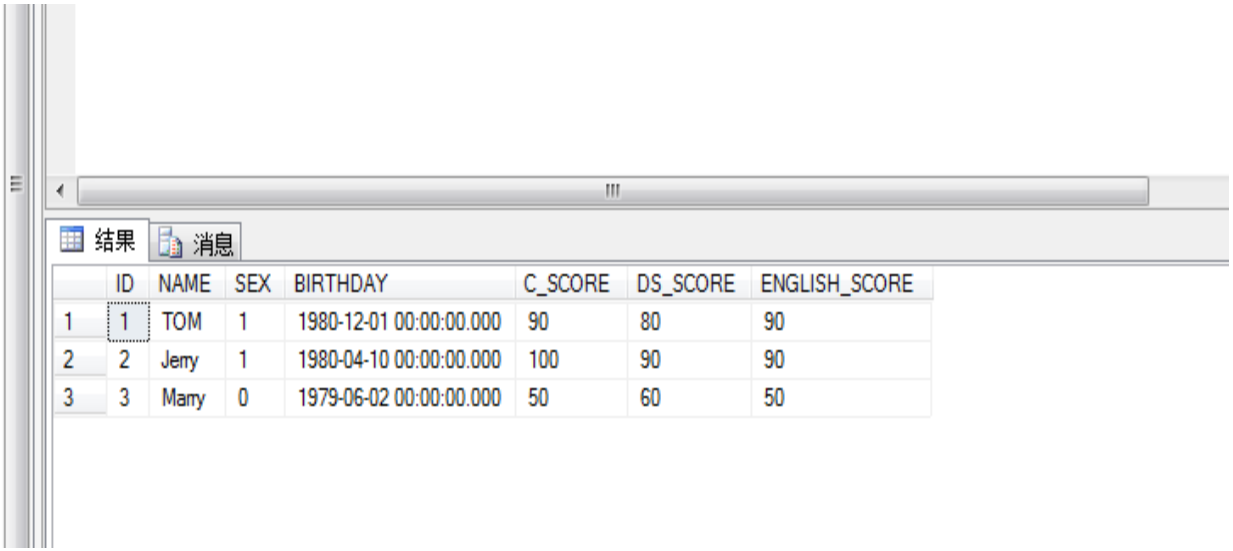
执行效果


Microsoft SQL Server Management Studio
文件(F) 编辑(E) 查看(V) 调试(D) 表设计器(L) 工具(T) 窗口(W) 社区(C) 帮助(H)
新建查询(N)
对象资源管理器
连接
LGY-PC\SQLEXPRESS (SQL Server 10.50.4000 - sa
数据库
系统数据库
ReportServer$SQLEXPRESS
ReportServer$SQLEXPRESSTempDB
STUDENT
数据库关系图
表
系统表
dbo.SCORE
列
键
约束
触发器
索引
统计信息
视图
LGY-PC\SQLEXPRE...DENT - dbo.SCORE
列名 数据类型 允许 Null 值
ID bigint
NAME varchar(20)
SEX bit
BIRTHDAY datetime
C_SCORE int
DS_SCORE int
ENGLISH_SCORE int

```
<%@ page language="java" import="java.sql.*" pageEncoding="UTF-8"%>
<html>
<head><title>插入学生成绩</title></head>
<body>
<%
    Connection connection = null;
    Statement stmt = null;
       final String url = "jdbc:sqlserver://localhost:1433;databaseName=STUDENT";
       final String username = "sa"; //访问数据库的用户名
       final String password = "sasa"; //访问数据库的密码
       try {
              Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
              connection = DriverManager.getConnection(url, username, password);
              stmt = connection.createStatement();
              stmt.addBatch("INSERT INTO SCORE (ID, NAME, SEX, BIRTHDAY, C_SCORE, DS_SCORE,
ENGLISH_SCORE) VALUES(1,'TOM',1,'12/01/1980',90,80,90)");
              stmt.addBatch("INSERT INTO SCORE (ID, NAME, SEX, BIRTHDAY, C_SCORE, DS_SCORE,
ENGLISH_SCORE) VALUES(2,'Jerry',1,'4/10/1980',100,90,90)");
              stmt.addBatch("INSERT INTO SCORE (ID, NAME, SEX, BIRTHDAY, C_SCORE, DS_SCORE,
ENGLISH_SCORE) VALUES(3,'Marry',0,'6/02/1979',50,60,50)");
              //此处省略其它插入语句
              stmt.executeBatch();
        }
       //省略部分语句
       out.println("Success!");
%>
</body>
</html>
```

执行效果





结果 消息

	ID	NAME	SEX	BIRTHDAY	C_SCORE	DS_SCORE	ENGLISH_SCORE
1	1	TOM	1	1980-12-01 00:00:00.000	90	80	90
2	2	Jerry	1	1980-04-10 00:00:00.000	100	90	90
3	3	Marry	0	1979-06-02 00:00:00.000	50	60	50

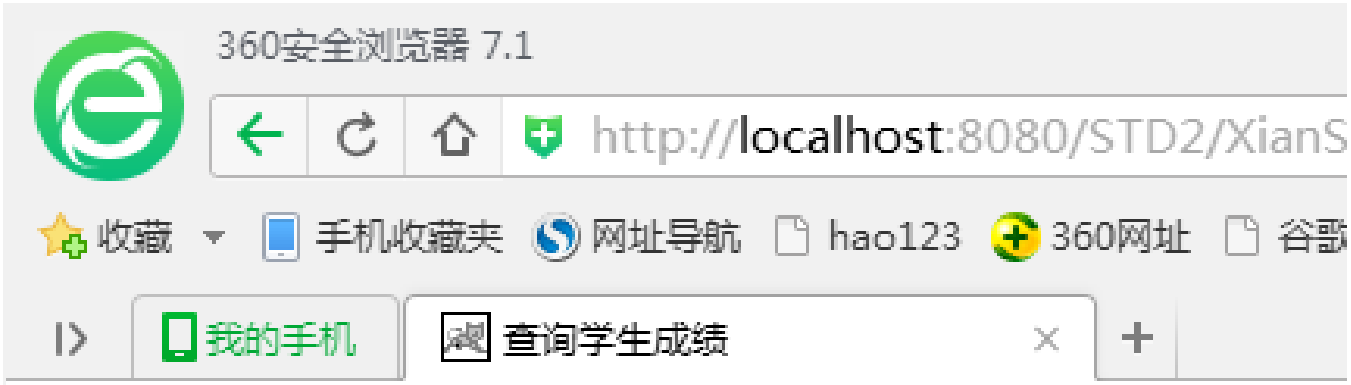
5.4 在JSP中使用JDBC访问数据库

```
<%@ page language="java" import="java.sql.*" pageEncoding="UTF-8"%>
<html>
<head><title>查询学生成绩</title></head>
<body>
<%
        Connection connection = null;
        Statement stmt = null;
        ResultSet rs = null;
        final String url = "jdbc:sqlserver://localhost:1433;databaseName=STUDENT";
        final String username = "sa"; //访问数据库的用户名
        final String password = "sasa"; //访问数据库的密码
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        connection = DriverManager.getConnection(url, username, password);
        stmt = connection.createStatement();
        rs = stmt.executeQuery("select ID, NAME, C_SCORE, DS_SCORE,
ENGLISH_SCORE from score");
%>
```

5.4 在JSP中使用JDBC访问数据库

```
<table>
        <tr>
            <td>ID</td><td>NAME</td>
            <td>C</td><td>Data Structure</td><td>English</td>
        </tr>
        <% while(rs.next()) { %>
        <tr>
            <td><%=rs.getInt(1)%></td>
            <td><%=rs.getString(2)%></td>
            <td><%=rs.getInt(3)%></td>
            <td><%=rs.getInt(4)%></td>
            <td><%=rs.getInt(5)%></td>
        </tr>
        <% }%>
</table>
        <% rs.close();
            stmt.close();
            connection.close(); %>
</body>
</html>
```

执行效果



ID	NAME	C	Data Structure	English
1	TOM	90	80	90
2	Jerry	100	90	90
3	Marry	50	60	50

❖删除数据

为了从数据库表中删除一行数据，可以借助于executeUpdate方法，例如，要将ID为1的学生记录从表中删除，可以使用下面的语句：

```
stmt.executeUpdate("DELETE FROM SCORE WHERE ID=1");
```

本章小结

JDBC的概念

JDBC的常用接口

在JSP中使用JDBC链接数据库