

Web应用开发 之 JDBC数据库访问



本章内容

- 数据库简介
- 数据库连接步骤
- JDBC API
- 数据源
- DAO设计模式

一、数据库简介

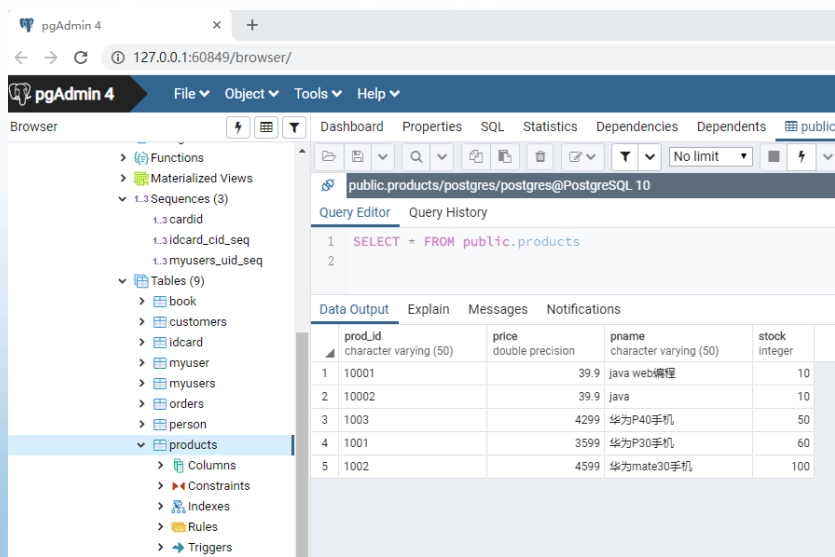
- PostgreSQL
- MySQL
- SQL Server
- ORACLE
- 国产：OceanBase、南大通用、武汉达梦、人大金仓、神舟通用

1、PostgreSQL数据库简介

- PostgreSQL是以加州大学计算机系开发的POSTGRES 4.2版本为基础的对象关系型数据库管理系统 (ORDBMS)
- 目前是最重要的开源数据库产品开发项目之一，有着非常广泛的用户
- PostgreSQL支持事务、子查询、多版本并发控制、数据完整性检查等特性，并且支持多语言的应用开发
- 能在包括Linux、FreeBSD和Windows等多种平台下运行

PostgreSQL的下载和安装

- 计算机系统必须满足下面要求：
 - CPU: Intel或AMD的32位或64位CPU。
 - 操作系统: Windows、Linux、FreeBSD (UNIX)、MAC。
 - 磁盘格式: 文件系统为NTFS格式。
 - 用户: 以系统管理员身份安装PostgreSQL。
- 目前最新版本是12. 2。
- 下载地址: <http://www.postgresql.org/download>



2、MySQL数据库简介

- MySQL是目前最为流行的开放源代码的数据库，是完全网络化的跨平台的关系型数据库系统，它是由瑞典的MySQL AB 公司（开发人Michael Widenius and David Axmark）于1995年开始发布
- 具有功能强大、支持跨平台、运行速度快、支持面向对象、安全性高、成本低、支持各种开发语言、数据存储量大、支持强大的内置函数等特点
- 2008年2月26日，Sun公司并购MySQL AB 公司，2010年1月27日，Oracle公司并购了Sun公司，现在MySQL属于Oracle。
- 版本：开源版MySQL8.0
- 下载地址<http://dev.mysql.com/downloads>

如何管理MySQL数据库

- 使用MySQL命令行工具
- MySQL Workbench（官方免费工具）
- Navicat或SQLyog（收费软件）

MySQL Workbench

Local instance MySQL Router x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- emailsys
- pmanagement
- rjpg
- shop
 - Tables
 - ausertable
 - busertable
 - carttable
 - department
 - focustable
 - goodstable
 - goodstype
 - noticetable
 - orderbasetable
 - orderdetail
 - post
 - quit
 - staff
 - transfer
 - Views
 - Stored Procedures
 - Functions
- sprinatest

ausertable department SQL File 10* ausertable busertable SQL File 11* product products products goodstable

1 • SELECT * FROM shop.goodstable;

Result Grid

	id	gname	goprice	grprice	gstore	gpicture	goodstype_id
	12	iphone11	6999	5999	50	20191130110309040.jpg	14
▶	13	HUAWEI Mate 30	5999	5499	99	20191130123824810.jpg	14
	14	Adidas加厚羽绒棉衣	588	278	12	20191130123838334.jpg	15
	15	阿迪达斯加绒运动套装	678	358	8	20191130123846910.jpg	15
	16	Java EE框架整合开发	119	98.2	29	20191130124123653.jpg	16
	17	Spring Boot项目实战	69	47.6	15	20191130123904571.jpg	16
	18	Java编程思想（第4版）	108	70.2	60	20191130123912378.jpg	14
	19	Java程序设计教程（第3版）	69	53.2	118	20191130123815348.jpg	16
	20	小米9	3299	2699	20	20191130123929418.jpg	14
	21	vivo iQOO Pro高通骁龙855Plus	3599	3299	17	20191130123936916.jpg	14
	22	地图上的中国史	398	353.8	20	20191130123955437.jpg	16

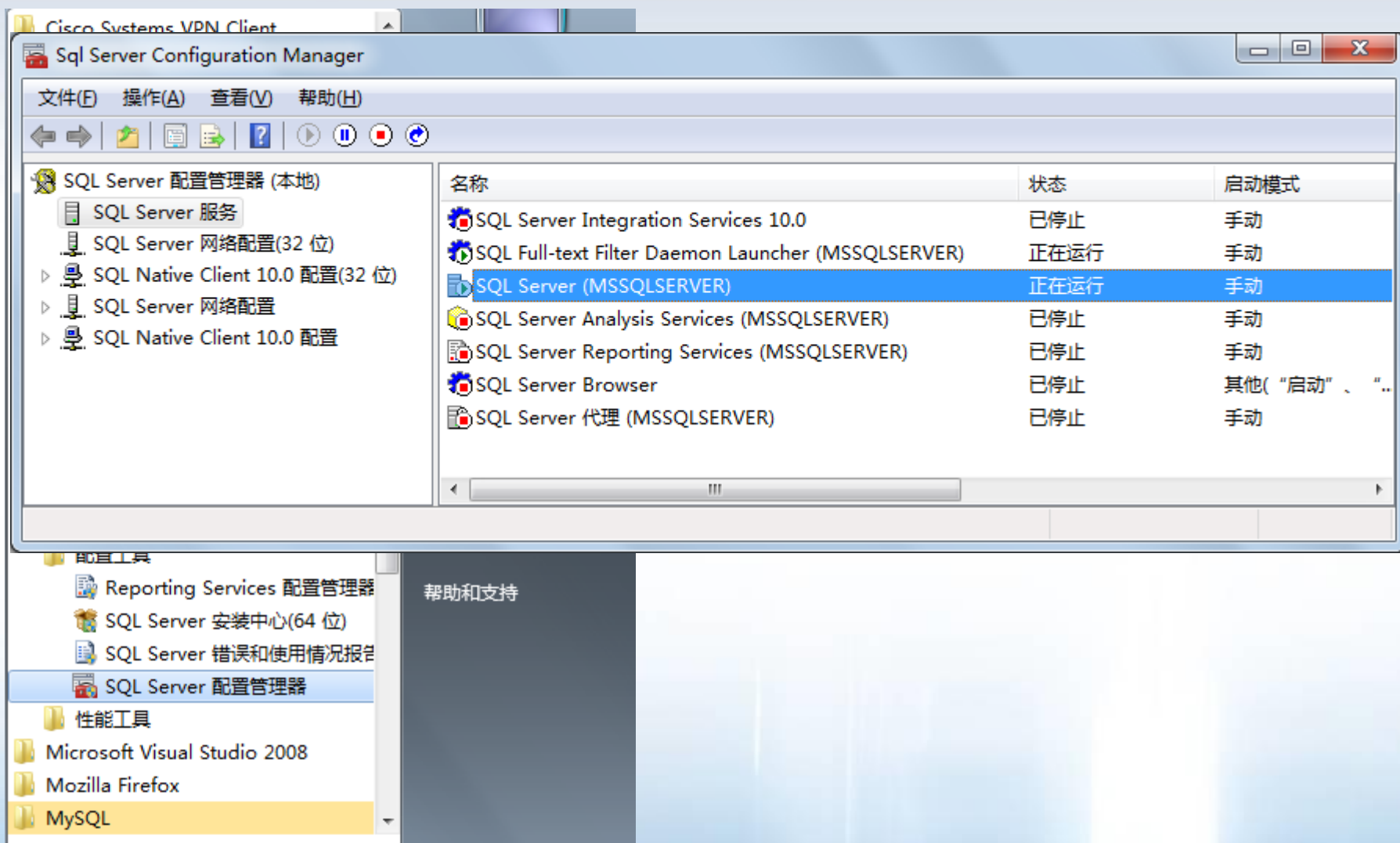
goodstable 1 x

Output

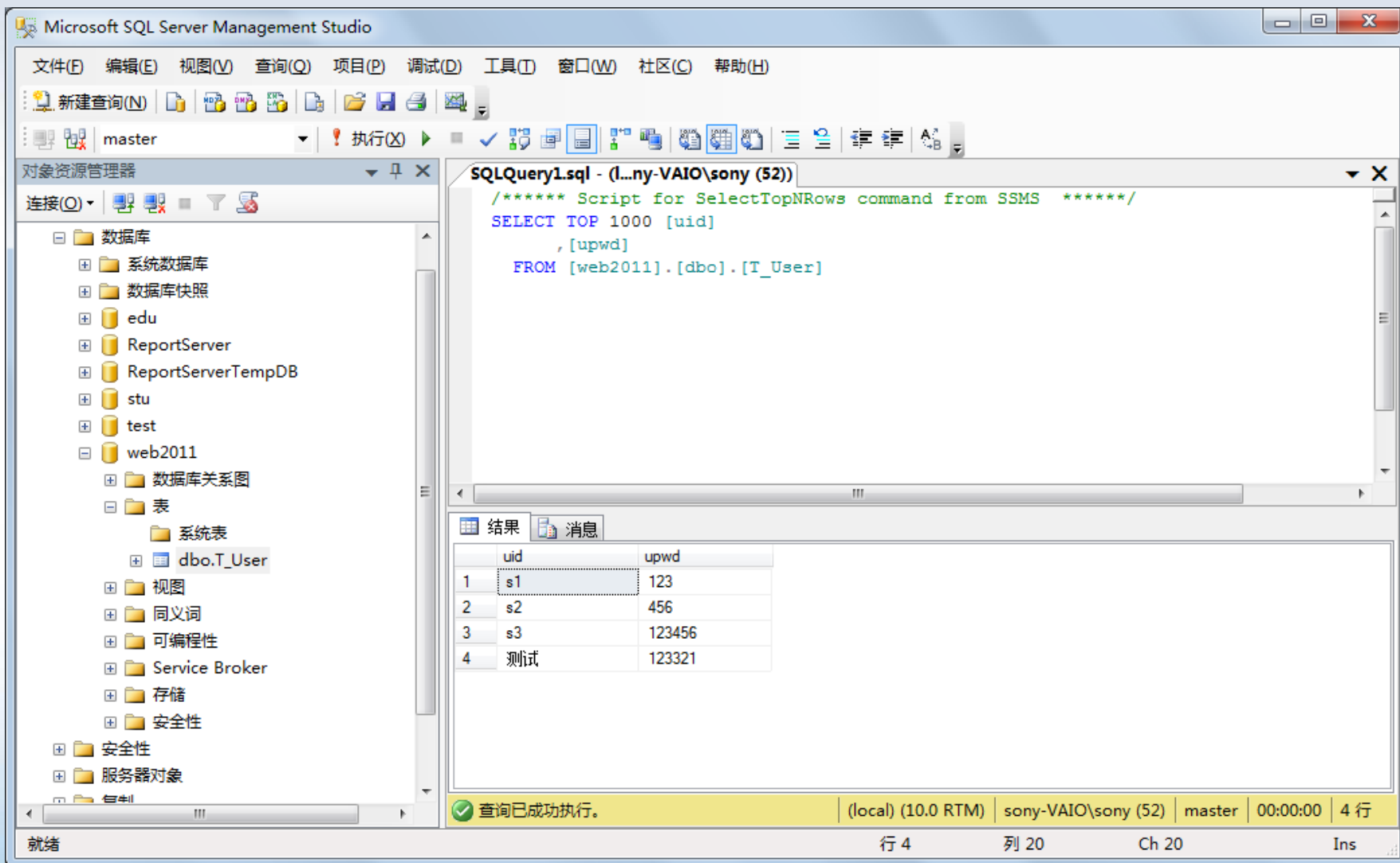
3、SQL Server数据库简介

- SQL Server是由Microsoft开发和推广的关系数据库管理系统（DBMS），提供了一个可靠的、高效的、智能化的数据平台
- 经历7.0，2000，2005，2008、2012、2014、2016等版本，其中SQL Server 2008分为SQL Server 2008企业版、标准版、工作组版、Web版、开发者版、Express版（免费）等

在配置工具—>Sql Server配置管理器中启动Sql Server2008服务



用Sql Server management studio打开 Sql Server2008控制台



Microsoft SQL Server Management Studio

文件(F) 编辑(E) 视图(V) 查询(Q) 项目(P) 调试(D) 工具(T) 窗口(W) 社区(C) 帮助(H)

新建查询(N)

master

执行(X)

对象资源管理器

连接(O)

数据库

- 系统数据库
- 数据库快照
- edu
- ReportServer
- ReportServerTempDB
- stu
- test
- web2011
 - 数据库关系图
 - 表
 - 系统表
 - dbo.T_User
 - 视图
 - 同义词
 - 可编程性
 - Service Broker
 - 存储
 - 安全性
- 安全性
- 服务器对象

SQLQuery1.sql - (I...ny-VAIO\sony (52))

```
/****** Script for SelectTopNRows command from SSMS *****/  
SELECT TOP 1000 [uid]  
        , [upwd]  
FROM [web2011].[dbo].[T_User]
```

结果 消息

	uid	upwd
1	s1	123
2	s2	456
3	s3	123456
4	测试	123321

就绪

查询已成功执行。 (local) (10.0 RTM) sony-VAIO\sony (52) master 00:00:00 4 行

行 4 列 20 Ch 20 Ins

4、ORACLE数据库

- Oracle是目前关系型数据库管理系统市场中应用最广泛的产品之一，最新的Oracle版本是Oracle 19c，普遍使用的版本是11g和12c
- 劳伦斯·埃里森、Bob Miner和Ed Oates在1977年建立了软件开发实验室咨询公司（SDL，Software Development Laboratories）。SDL开发了Oracle软件的最初版本
- ORACLE数据库在物理上是存储于硬盘的各种文件。它是活动的，可扩充的，随着数据的添加和应用程序的增大而变化
- ORACLE数据库在逻辑上是由许多表空间构成
- ORACLE的详细信息参见百度百科
<http://baike.baidu.com/view/15020.htm>

用PL/SQL Developer管理Oracle数据库

PL/SQL Developer - gxt66ZJ01SMS_192.168.0.13

File Project Edit Session Debug Tools Macro Documents Reports Window Help

SQL Window - Query data of table YHSJ_ZHUANYE_ZJUTSMS_192.168.0.13

SQL Output Statistics

```
select * from yhsj_zhuanye t
```

My objects

- Recent objects
- Recycle bin
- Functions
- Procedures
- Packages
- Package bodies
- Types
- Type bodies
- Triggers
- Java sources
- Jobs
- Queues
- Queue tables
- Libraries
- Directories
- Tables
 - ADC_AUTHENLOGIN
 - ADC_ECREQUEST
 - ADC_MEMBERSHIP
 - ADC_PERSONALBUSISYN
 - ADC_SERVICESTATEREQUE
 - AD_COLLEGE

	ZYID	ZYNAME	COLLEGEID	SCHOOLC	ZYCODE
1	B9601B2D1E7F40CFB8BD3561FF212C3D	动漫设计与制作	97D2962200484D72B26507277A55A7BC	10340	67030500
2	46E3A7786DD444F691D487A5A6C75BFE	电子商务	AB4683961D8B494DAC16ED8BD9FEE861	13743	62040500
3	16A0F4BF311F449E9DFFE045D0888201	英语	B135737A1BC3437989960AB4253D359D	10337	05020100
4	786ECFE89A9149AD89BA181E21B205D0	竞赛	B135737A1BC3437989960AB4253D359D	10337	
5	17C8E9BEC9B143E59D7288C9C09A95E5	视觉传达设计	C711BEC6B5CF48DAA0DFE49F4E947244	10337	67010300
6	692288B5BC80409D8039167F1A5D5076	工业设计	C711BEC6B5CF48DAA0DFE49F4E947244	10337	08030300
7	2FAC9055DAF6481F8326FCFDEA6DFB9F	动画	C711BEC6B5CF48DAA0DFE49F4E947244	10337	05041800
8	976491D7BDDA46C7B95ED2E49ADA52D9	环境艺术	C711BEC6B5CF48DAA0DFE49F4E947244	10337	05040800
9	30F10BE11F0B4F21BBFE4D8807CEEE6	多媒体与网页设计	C711BEC6B5CF48DAA0DFE49F4E947244	10337	
10	6F4F94C456194BCBA0BB15600563A035	艺术设计(环境艺术设计)	C711BEC6B5CF48DAA0DFE49F4E947244	10337	05040800
11	ABED261037F340E89575F9DE72A5A6BC	法律	395B4BB3F38B45B0AC0A0E7AC4A46D90	10337	03010100
12	FD4DABE73250457584C32524775D154B	建筑经济	395B4BB3F38B45B0AC0A0E7AC4A46D90	10337	56050300
13	DC21CF0B891442D3A9BDBAB9DC534148	计算机科学与技术	C3423B39E8C04214927364FC5AB42E22	10337	08060500

5、OceanBase

- **OceanBase** 是阿里巴巴和蚂蚁金服完全自主研发的金融级分布式关系型数据库，实现了数千亿条记录、数百**TB**数据上的跨行跨表事务。
- 创造了 **4200** 万次/秒处理峰值的纪录。
- **2019年TPC-C**基准测试（被誉为“数据库领域世界杯”）中，性能**2**倍于第二名**Oracle**
- 下载地址<https://oceanbase.alipay.com/>

二、数据库连接步骤

- 加装驱动程序
- 建立连接对象
- 创建语句对象
- 执行SQL语句并处理结果
- 关闭建立的连接

2.1 加装驱动程序

- 要使应用程序能够访问数据库，首先必须加载驱动程序。驱动程序是实现了Driver接口的类，它一般由数据库厂商提供。
 - PostgreSQL驱动包： `postgresql-9.3-1104.jdbc4.jar`
 - MySQL驱动包： `mysql-connector-java-8.0.11`
 - Sql Server驱动包： `sqljdbc42.jar`
 - Oracle驱动包： `ojdbc8.jar`
- 在Web应用程序中，将驱动程序打包文件复制到：
 - Tomcat安装目录的lib目录中
 - Web应用程序的WEB-INF\classes\lib目录中
 - Eclipse工程的build path中

2.1 加载驱动程序

- 加载JDBC驱动程序最常用的方法是使用Class类的 `forName()` 静态方法，该方法的声明格式为：

```
public static Class<?> forName(String  
className)throws ClassNotFoundException
```

- 参数className为一字符串表示的完整的驱动程序类的名称。
- 找不到驱动程序将抛出ClassNotFoundException 异常

2.1 加载驱动程序

- 加载PostgreSQL数据库驱动程序语句：
`Class.forName("org.postgresql.Driver");`
- 加载mysql数据库驱动程序语句：
`Class.forName("com.mysql.jdbc.Driver");`
`Class.forName("com.mysql.cj.jdbc.Driver");`
- 加载sqlServer数据库驱动程序语句：
`Class.forName("com.microsoft.sqlserver.
jdbc.SQLServerDriver")`
- 加载Oracle数据库驱动程序语句：
`Class.forName("oracle.jdbc.driver.OracleDrive");`

2.2 建立连接对象

- 驱动程序加载成功后应使用DriverManager类的getConnection()建立数据库连接对象。
- DriverManager类维护一个注册的Driver类的列表。

1. DriverManager 类

- 建立数据库连接的方法是调用DriverManager类的静态方法getConnection(), 该方法的声明格式为:

```
public static Connection getConnection(String dburl)
```

```
public static Connection getConnection(String dburl,  
String user,String password)
```

2. JDBC URL

- 连接PostgreSQL数据库JDBC URL为:

`jdbc:postgresql://dbServerIP:5432/dbname`

- 连接mysql数据库JDBC URL为:

`jdbc:mysql://dbServerIP:3306/dbname?user=username
&password=password`

- 连接sqlserver数据库JDBC URL为:

`jdbc:Microsoft:sqlserver://dbServerIP:1433;database=dbname`

- 连接Oracle数据库JDBC URL为:

`jdbc:oracle:thin:@dbServerIP:1521/orclpdbservice`

2. JDBC URL

- 连接PostgreSQL数据库，数据库名为paipaistore、用户名为paipaistore、口令为paipaistore:

```
String dburl =
```

```
    "jdbc:postgresql://localhost:5432/paipaistore";
```

```
Connection conn = DriverManager.getConnection(  
    dburl, "paipaistore", "paipaistore");
```

2.3 创建语句对象

- 通过**Connection**对象，可创建语句对象。对于不同的语句对象，可以使用**Connection**接口的不同方法创建。
 - 创建一个简单的**Statement**对象可以使用**createStatement()**
 - 创建**PreparedStatement**对象应该使用**prepareStatement()**
 - 创建**CallableStatement**对象应该使用**prepareCall()**

2.3 创建语句对象

- 创建一个简单的Statement对象语句:

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

- 创建一个预编译的PreparedStatement对象语句:

```
String sql= "select prod_id,pname from products"
```

```
PreparedStatement pstmt = conn. prepareStatement(sql);
```

2.4 执行SQL语句并处理结果

- 执行SQL语句使用Statement对象的方法。对于查询语句，调用`executeQuery(String sql)`返回ResultSet。
- ResultSet对象保存查询的结果集，再调用ResultSet的方法可以对查询结果的每行进行处理。

```
String sql = "SELECT prod_id,pname FROM products" ;  
ResultSet rst = stmt.executeQuery(sql) ;  
while(rst.next()){  
    out.print(rst.getString(1)+"\t") ;  
}
```


3.4 执行SQL语句并处理结果

- 对于CREATE、ALTER、DROP、INSERT、UPDATE、DELETE等须使用语句对象的
`executeUpdate(String sql)`。
- 该方法返回值为整数，用来指示被影响的行数。

2.5 关闭建立的对象

- 在 **Connection** 接口、**Statement** 接口和 **ResultSet** 接口中都定义了 **close()**。当这些对象使用完毕后应使用 **close()** 关闭。
- 如果使用 **try-with-resources** 语句，则可以自动关闭这些对象。

2.6示例：访问postgresql数据库

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.sql.*"%>
<%
Class.forName("org.postgresql.Driver");
Connection conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/postgres", "postgres", "postgres");
Statement st = conn.createStatement();
st.setMaxRows(20);
String sqlStr = "select * from products";
ResultSet rs = st.executeQuery(sqlStr);
%>
<table border=1>
<tr><td>商品编号</td><td>商品名称</td><td>价格</td><td>库存量</td></tr>
<%
while (rs.next()) {
%>
<tr>
<td><%=rs.getString("prod_id")%></td>
<td><%=rs.getString("pname")%></td>
<td><%=rs.getDouble("price")%></td>
<td><%=rs.getString("stock")%></td>
</tr>
<%
}
rs.close();
st.close();
conn.close();
%>
</table>
```

products

General Columns Constraints Advanced Parameters Security SQL

Inherited from table(s)

	Name	Data type	Length	Precision
<input checked="" type="checkbox"/>	prod_id	character varying	50	
<input checked="" type="checkbox"/>	price	double precision		
<input checked="" type="checkbox"/>	pname	character varying	50	
<input checked="" type="checkbox"/>	stock	integer		

http://localhost:8080/chapter07/testpostgreSQL.jsp

商品编号	商品名称	价格	库存量
10001	java web编程	39.9	10
10002	java	39.9	10
1003	华为P40手机	4299.0	50
1001	华为P30手机	3599.0	60
1002	华为mate30手机	4599.0	100

2.6示例：访问MySQL数据库

```
<%@ page import="java.sql.*"%>
<%
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "root", "111111");
Statement st = conn.createStatement();
st.setMaxRows(20);
String sqlStr = "select * from products";
ResultSet rs = st.executeQuery(sqlStr);
%>
<table border=1>
<tr>
<td>商品编号</td>
<td>商品名称</td>
<td>价格</td>
<td>库存量</td>
</tr>
<%
while (rs.next()) {
%>
<tr>
<td><%=rs.getString("prod_id")%></td>
<td><%=rs.getString("pname")%></td>
<td><%=rs.getDouble("price")%></td>
<td><%=rs.getString("stock")%></td>
</tr>
<%
}
rs.close();
st.close();
conn.close();
%>
</table>
```

products										
products - Table										
Table Name: products Schema: testdb										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
prod_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
pname	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
price	DOUBLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
stock	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

prod_id	pname	price	stock
10001	华为P40手机	4188	100
10002	华为P40 Pro手机	6188	50
10003	联想ThinkPad X280	7199	20

http://localhost:8080/chapter07/testmysql.jsp			
商品编号	商品名称	价格	库存量
10001	华为P40手机	4188.0	100
10002	华为P40 Pro手机	6188.0	50
10003	联想ThinkPad X280	7199.0	20

2.6示例：访问SQLServer数据库testSqlServer.jsp

```
<%@ page import="java.sql.*" %>
<%
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");//
.microsoft.jdbc.sqlServer.SQLServerDriver");
Connection conn = DriverManager.getConnection("jdbc:sqlserver://localhost:1433;DatabaseName=web"
,web","web");
Statement st=conn.createStatement();
st.setMaxRows(20);
String sqlStr="select * from Users";
ResultSet rs=st.executeQuery(sqlStr);
<%
while(rs.next())
{%>
<%=rs.getString("uid")%>
<%=rs.getString("upwd")%>
<%
}
rs.close();
st.close();
conn.close();
%>
```



2. 7访问数据库的应用示例

- 根据用户输入的商品号从数据库中查询该商品信息，或者查询所有商品信息。
- 采用MVC设计模式：
 - ①视图：`queryProduct.jsp`、`displayProduct.jsp`、`displayAllProduct.jsp`和`error.jsp`，
 - ②模型：`Product`类
 - ③控制器：`QueryProductServlet`类

2.7访问数据库的应用示例:查询商品信息

- 1、创建数据库表products。

```
CREATE TABLE products (  
    prod_id character(3) NOT NULL,  
    pname character varying(30) NOT NULL,  
    price numeric(8,2),  
    stock integer, CONSTRAINT product_pkey  
        PRIMARY KEY (prod_id)  
)
```

◆为测试需求，建表后录入几条数据。

2. 7访问数据库的应用示例: 查询商品信息

- 2、设计JavaBeans类Product存放商品信息，表的字段对应Product类的成员变量。

```
package com.model;
import java.io.Serializable;

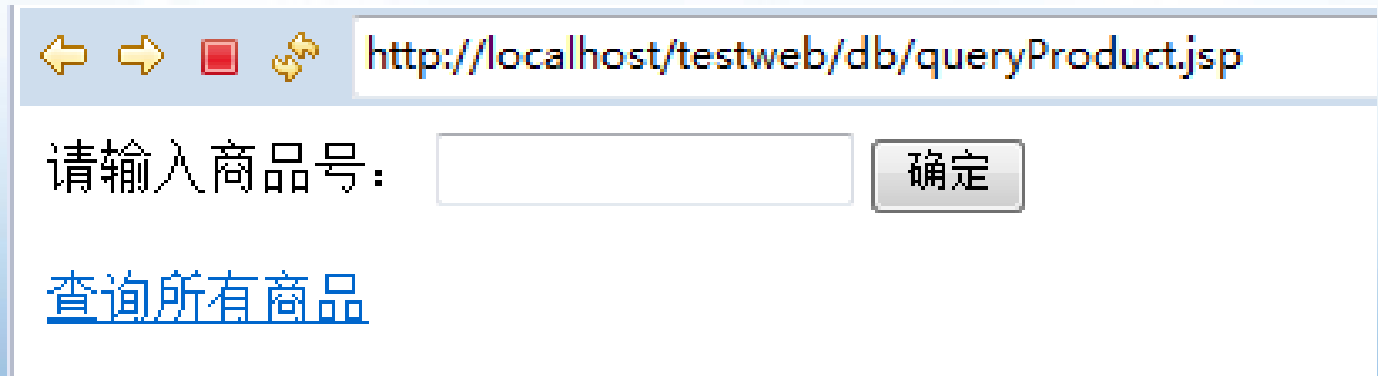
public class Product implements Serializable{
    private String prod_id;
    private String pname;
    private double price;
    private int stock;
    public Product() { }
    //setter and getter
}
```


2.7 访问数据库的应用示例：查询商品信息

3、编写查询商品页面queryProduct.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<html>
<head><title>商品查询</title></head>
<body>
<form action = "queryproduct.do" method="post">
    请输入商品号：
    <input type = "text" name="productid" size="15">
    <input type = "submit" value = "确定">
</form>
<p><a href="queryproduct.do" >查询所有商品</a></p>
</body>
</html>
```



← → □ ↻ http://localhost/testweb/db/queryProduct.jsp

请输入商品号:

[查询所有商品](#)

2.7访问数据库的应用示例:查询商品信息

- 4、编写Servlet处理查询商品页面
`QueryProductServlet.java`。

(1) 当用户在文本框中输入商品号, 单击“确定”按钮, 将执行doPost()

(2) 当用户单击“查询所有商品”链接时, 将执行doGet()。

QueryProductServlet.java

```
package com.demo;

import java.io.*;
import java.sql.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.model.Product;
import javax.servlet.annotation.WebServlet;

@WebServlet("/queryproduct.do")
public class QueryProductServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    Connection dbconn = null;
    public void init() {
        String driver = "org.postgresql.Driver";
        String dburl = "jdbc:postgresql://127.0.0.1:5432/postgres";
        String username = "postgres";
        String password = "postgres";
        try {
            Class.forName(driver);
            dbconn = DriverManager.getConnection(dburl, username, password);
        } catch (ClassNotFoundException e1) {
            System.out.println(e1);
        } catch (SQLException e2) {
        }
    }
}
```

QueryProductServlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String productid = request.getParameter("productid");
    try {
        String sql = "SELECT * FROM products WHERE prod_id = ?";
        PreparedStatement pstmt = dbconn.prepareStatement(sql);
        pstmt.setString(1, productid);
        ResultSet rst = pstmt.executeQuery();
        if (rst.next()) {
            Product product = new Product();
            product.setProd_id(rst.getString("prod_id"));
            product.setPname(rst.getString("pname"));
            product.setPrice(rst.getDouble("price"));
            product.setStock(rst.getInt("stock"));
            request.getSession().setAttribute("product", product);
            response.sendRedirect("displayProduct.jsp");
        } else {
            response.sendRedirect("error.jsp");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

QueryProductServlet.java

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    ArrayList<Product> productList = null;
    productList = new ArrayList<Product>();
    try {
        String sql = "SELECT * FROM products";
        PreparedStatement pstmt = dbconn.prepareStatement(sql);
        ResultSet result = pstmt.executeQuery();
        while (result.next()) {
            Product product = new Product();
            product.setProd_id(result.getString("prod_id"));
            product.setName(result.getString("pname"));
            product.setPrice(result.getDouble("price"));
            product.setStock(result.getInt("stock"));
            productList.add(product);
        }
        if (!productList.isEmpty()) {
            request.getSession().setAttribute("productList", productList);
            response.sendRedirect("displayAllProduct.jsp");
        } else {
            response.sendRedirect("error.jsp");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public void destroy() {
    try {
        dbconn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

2.7 访问数据库的应用示例：查询商品信息

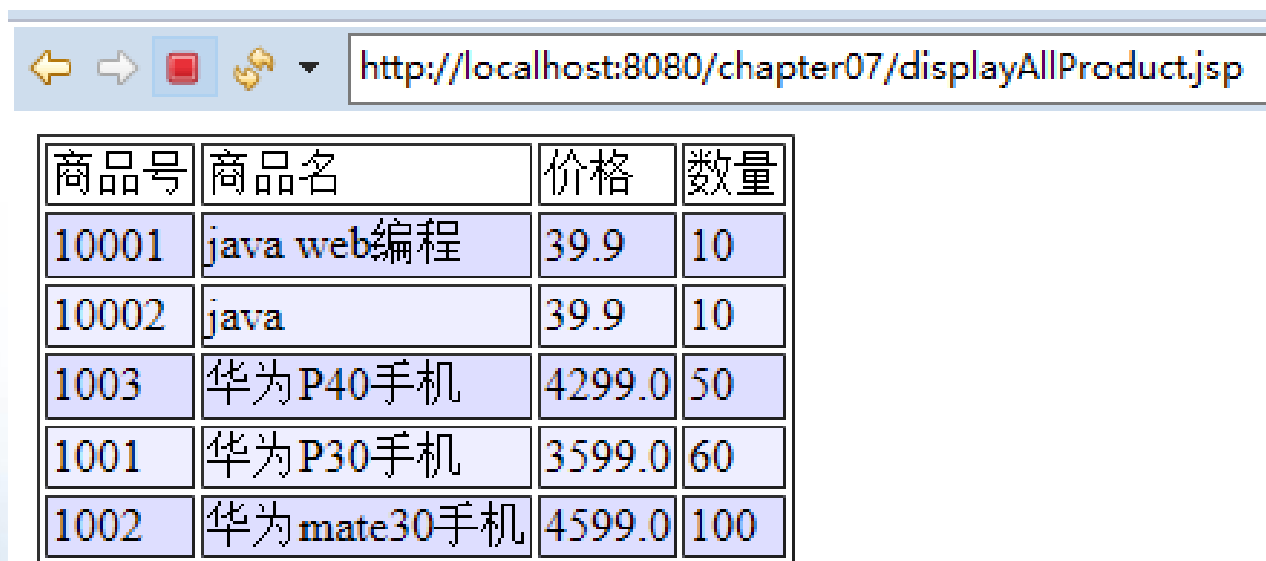
- 5、编写查询结果页面
- **displayProduct.jsp**显示查询一件商品。

```
<%@ page contentType="text/html; charset=utf-8" %>
<html>
<head><title>商品信息</title></head>
<body>
<table>
<tr><td>商品号: </td><td>${product.prod_id}</td>
</tr>
<tr><td>商品名: </td><td>${product.pname}</td>
</tr>
<tr><td>价格: </td><td>${product.price}</td>
</tr>
<tr><td>库存量: </td><td>${product.stock}</td>
</tr>
</table>
</body></html>
```



2. 7访问数据库的应用示例: 查询商品信息

- 5、编写查询结果页面
- `displayAllProduct.jsp`显示所有商品信息。



The screenshot shows a web browser window with the address bar displaying the URL `http://localhost:8080/chapter07/displayAllProduct.jsp`. Below the browser window, a table displays the query results for all products.

商品号	商品名	价格	数量
10001	java web编程	39.9	10
10002	java	39.9	10
1003	华为P40手机	4299.0	50
1001	华为P30手机	3599.0	60
1002	华为mate30手机	4599.0	100

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.*,com.model.Product"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<head>
    <title>显示所有商品</title>
</head>
<body>
    <table border="1">
        <tr>
            <td>商品号</td>
            <td>商品名</td>
            <td>价格</td>
            <td>数量</td>
        </tr>
        <c:forEach var="product" items="${sessionScope.productList}" varStatus="status">
            <!--为奇数行和偶数行设置不同的背景颜色-->
            <c:if test="${status.count%2==0}">
                <tr style="background: #eeeeff">
            </c:if>
            <c:if test="${status.count%2!=0}">
                <tr style="background: #dedeff">
            </c:if>
            <!--用EL访问作用域变量的成员-->
            <td>${product.prod_id}</td>
            <td>${product.pname}</td>
            <td>${product.price}</td>
            <td>${product.stock}</td>
        </tr>
        </c:forEach>
    </table>
</body>
</html>
```


2.7 访问数据库的应用示例：查询商品信息

6、编写错误页面error.jsp

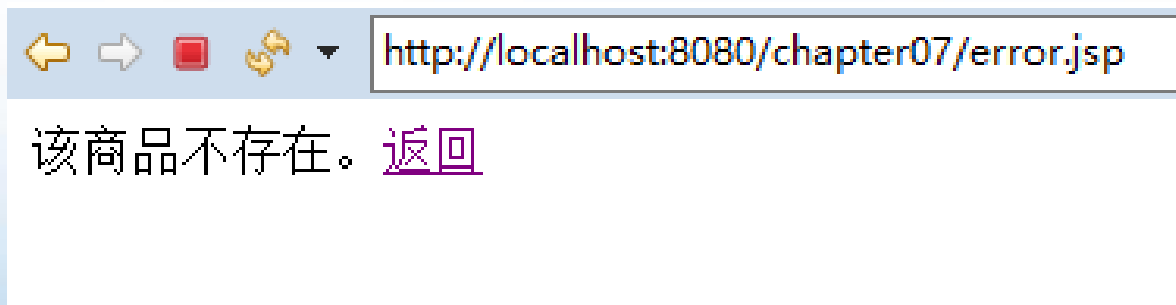
- 当查询的商品不存在时，显示错误页面。

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

```
<html><body>
```

```
    该商品不存在。<a href="queryProduct.jsp">返回</a>
```

```
</body></html>
```



小 结

- Java程序是通过JDBC API访问数据库。
- JDBC API定义了Java程序访问数据库的接口。
- 访问数据库首先应该建立到数据库的连接。

传统的方法是通过DriverManager类的
getConnection()建立连接对象。使用这种方法很容易产生性能问题。

小 结

➤ 数据库连接步骤：

- (1) 加载驱动程序。
- (2) 建立连接对象。
- (3) 创建语句对象。
- (4) 执行语句获得结果。
- (5) 关闭建立的对象。

➤ 测试访问数据库的示例：创建数据表，编写javabeans类，编写查询jsp页面，编写处理查询的servlet，编写结果显示jsp页面

作业

1、测试查询商品信息的例子,使之能正常运行, 并增加如下功能。

- **（1）** 分别增加添加、修改和删除功能；
- **（2）** 增加一个商品管理页面，包含添加商品、查询商品、查看所有商品等功能，查询结果列表的后面增加修改和删除功能。

作业

2、实现某师生健康码管理系统的用户注册和登录功能，具体要求如下：

（1）用户注册**register.jsp**。如果是教师，则可输入姓名、身份证号、工号、密码、学院、角色（系统管理员、校级管理员、院系级管理员、普通教师）等信息，注册的信息写入教师表**dbt_teacher**中；如果是学生，则可输入姓名、身份证号、学号、密码、学院、专业、班级等信息，注册的信息存入学生表**dbt_student**。要求密码采用**SHA256**加密（可用**JDK**自带的**java.security.MessageDigest**实现）后存入数据库。要求教师的身份证号、工号和学生的身份证号、学生不能重复注册，否则跳转至**failed.jsp**提示“您输入的身份证号、工号或学号已被注册！”。

（2）用户登录**login.jsp**。选择用户类型(教师或学生)，输入登录名、密码，教师的登录名为工号，学生的登录名为学号。如果教师用户登录成功后，跳转至**teacher.jsp**显示该教师的姓名、身份证号、工号、密码、学院、角色等信息；如果是学生用户登录成功后，跳转至**student.jsp**显示该学生的姓名、身份证号、学号、密码、学院、专业、班级等信息；如果登录失败，则跳转至**error.jsp**，显示“您的用户名或密码错误”，3秒后自动跳转至登录页面**login.jsp**。