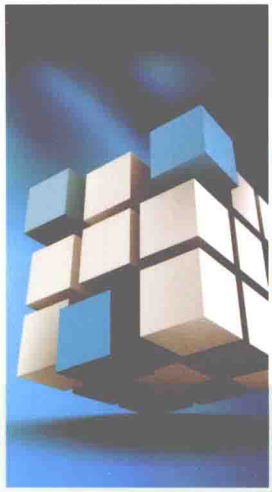
普通高等职业教育 计算机系列规划教材

Java EE SSH框架应用开发项目教程



◆ 彭之军 主编
◆ 刘 波 陈志凌 副主编

SOFTWARE

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

配备
源码、电子课件

普通高等职业教育计算机系列规划教材

Java EE SSH 框架应用开发项目教程

彭之军　主　编

刘　波　陈志凌　副主编

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书以 Java EE Web 开发的知识点为主线，以 Oracle 数据库附带的表为基础，第 1～4 章讲解了 JDBC、JSP、Servlet、AJAX 在 Java EE 中的使用方法，第 5～12 章重点而详细地介绍了 Struts2、Spring3 及 Hibernate4 框架的主要内容和最新内容。在本书的最后一章，以一个综合性的案例——书籍管理系统，完整地介绍了使用 SSH 开源框架开发的全过程，内容包括目前主流的表示层技术——jQuery 技术的详细讲解。

本书可作为应用型本科 Java EE 企业级开发课程、高职高专相关专业课程教材和教学参考书，也可作为培训机构的教材，以及供从事 Java EE 应用系统开发的用户学习和参考。

图书在版编目（CIP）数据

Java EE SSH 框架应用开发项目教程 / 彭之军主编. —北京：电子工业出版社，2015.6
（普通高等职业教育计算机系列规划教材）
ISBN 978-7-121-26343-9

Ⅰ. ①J… Ⅱ. ①彭… Ⅲ. ①JAVA 语言－程序设计－高等职业教育－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 130315 号

策划编辑：徐建军（xujj@phei.com.cn）
责任编辑：徐建军　　特约编辑：俞凌娣　张祖凤
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：17.5　字数：448 千字
版　　次：2015 年 6 月第 1 版
印　　次：2015 年 6 月第 1 次印刷
印　　数：3 000 册　　定价：38.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言
Preface

Java 语言已经成为企业开发的主要语言之一。对于企业级 Java 开发，基于 Struts2、Hibernate3 和 Spring4 的开源框架已经是主流的基础框架。通过本书，希望读者能快速地进入 Java EE 开发的殿堂。

本书的三位作者都有丰富的企业开发经验和多年的职业教育经验。

刘波（广州科苑北大青鸟培训中心校长、高级讲师）：我从事 Java EE 的开发与教学，算下来已经 10 个年头了，而对 Java 的接触就更早一些。应该说在工作和教学过程中积累了不少经验，一直很想有机会或渠道与大家分享这些心得。于是就有了这本书。在写这本书的时候，我尽量用年轻人比较容易接受的语言把一些复杂的技术要点解释得通俗一点，而且加入了大量自己的经验和技巧。想必接触这本书的大部分人应该是年轻人和初入 Java EE 殿堂的人吧，这本书适合于入门者或正在学习 Java EE 的读者，也可以是一本 Java EE 开发人员的参考书籍，希望通过学习这本书，能给你的工作和学习带来帮助。

陈志凌（广州科苑北大青鸟培训中心高级讲师）：本书源于广东东软学院彭之军老师的邀请，有幸同彭老师和广州科苑刘波老师一起通力合作完成。这本书凝聚了本人多年来在北大青鸟的教学经验，涵盖了目前流行的 SSH 开源框架中的核心内容。本书针对初学 Java 开源框架的读者，由浅入深、简明易懂地引领他们掌握 Java EE 流行的三大框架。在编写这本书的过程中，我体会了辛苦和快乐。每写一页都要反复推敲，反复验证每行代码。完成这本书也是本人在职业生涯中的一次提升，热切期待它能成为 Java EE 学习者的益友。

彭之军（广东东软学院专职教师）：我曾经从事 Java 开发和职业技术培训，现在广东东软学院计算机系从事软件开发方面的教学工作。本书邀请到其他两位作者执笔，特别是 Hibernate 的性能分析以及第 13 章 jQuery 和 SSH 的整合使用，为本书增色不少。立足于理论与实践相结合，相信读者只要认真研读，完成上机实验，必定能收获良多。

本书由彭之军、刘波、陈志凌共同编写完成，其中彭之军编写了第 1～3 章和第 12 章，刘波编写了第 4、10、11、13 章，陈志凌编写了第 5～9 章，全书由彭之军统稿，并由植挺生主审，徐婉珍参审。

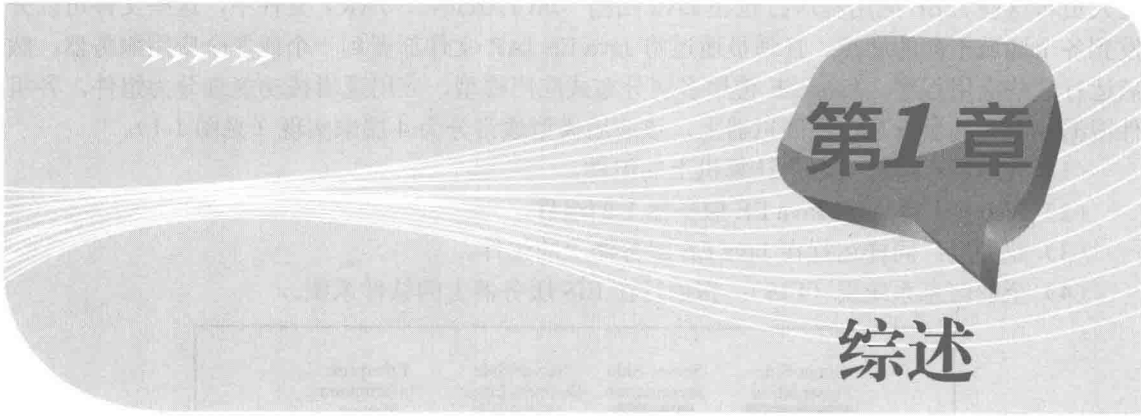
感谢刘波和陈志凌先生的高效和通力的合作，感谢广东东软学院计算机系的植挺生和徐婉珍老师，以及其他提出宝贵意见的老师，还要感谢计算机系领导的大力支持。

为了方便教师教学，本书配有电子教学课件及相关资源，请有此需要的老师登录华信教育资源网（www.hxedu.com.cn）注册后免费进行下载，本书的案例和教学 PPT 也可以在 51cto 博客（cnjava.blog.51cto.com）上获取。如有问题可在网站留言板留言或与电子工业出版社联系（E-mail：hxedu@phei.com.cn）。

教材建设是一项系统工程，需要在实践中不断加以完善及改进，书中难免存在疏漏和不足，恳请同行专家和读者给予批评和指正。

编者

目录
Contents



从 Sun 公司 1995 年正式发布 Java 到现在已经有近二十年了。Java 也随着 Java EE（Java platform Enterprise Edition，Java 平台企业版）的发布成为大中型信息系统的首选开发语言。

如果你有 Java SE 的学习或开发经验，就会发现目前使用 Java 开发 C/S 模式（Client/Server，客户端/服务器端）的程序日渐减少，而使用 Java EE 来开发 B/S（Browser/Server，浏览器/服务器）模式的程序已成为企业信息系统的主流。

本书主要介绍使用 Java EE Web 的主流企业级开发框架来开发信息系统。虽然一般使用 Java EE 来开发大中型系统，但本书是以小型系统来讲解知识点，这样的好处是降低读者学习的难度。小型系统中简单的业务可以让读者将重点放到 Java EE 知识体系的学习而不必花太多的时间在令人费解的业务上。

1.1 Java EE 技术和相关框架

长期以来，Java EE（Java 企业版，以前也简称为 J2EE）已成为各行业（金融、电信、零售、商业，等等）开发和部署企业级应用程序的首选平台。这是因为 Java EE 提供了一个基于标准的平台，可以用来构建强壮和高扩展性的分布式应用程序，以支持类似从银行核心业务到在线购物平台的所有业务。无须讳言，开发 Java EE 应用程序也是一项非常艰巨的任务。

首先，开源的 Java 平台提供了丰富的选项，数目繁多的框架、实用的工具库、集成开发环境（IDE）以及各种工具，使得开发工作更具有挑战性。“工欲善其事，必先利其器”。因此选择合适的技术是非常重要的。选择使用良好的架构和技术，才可能构建易于维护、复用和扩展的程序。

1.1.1 Java EE 应用程序架构

J2EE 应用程序由一些组件构成，包含了例如 JavaServer Pages（JSP）、Servlet 和 Enterprise

JavaBeans（EJB）等模块。开发人员通过以上介绍的这些组件来构建大型分布式应用程序。开发人员将这些 J2EE 应用程序打包在 Java 归档（Java Archive，JAR）文件中，这些文件可以分发到各个地域不同的站点。管理员通过将 Java EE JAR 文件部署到一个或多个应用服务器，然后运行这些应用程序。Java EE 使用多层分布式应用模型，应用逻辑按功能划分为组件，各组件根据他所在的层分布在不同机器上。该应用模型通常分为 4 层来实现（见图 1-1）。

（1）客户层：运行在客户计算机上的组件。

（2）Web 层：运行在 Java EE 服务器上的组件。

（3）业务层：同样运行在 Java EE 服务器上的组件。

（4）企业信息系统层（EIS）：指运行在 EIS 服务器上的软件系统。

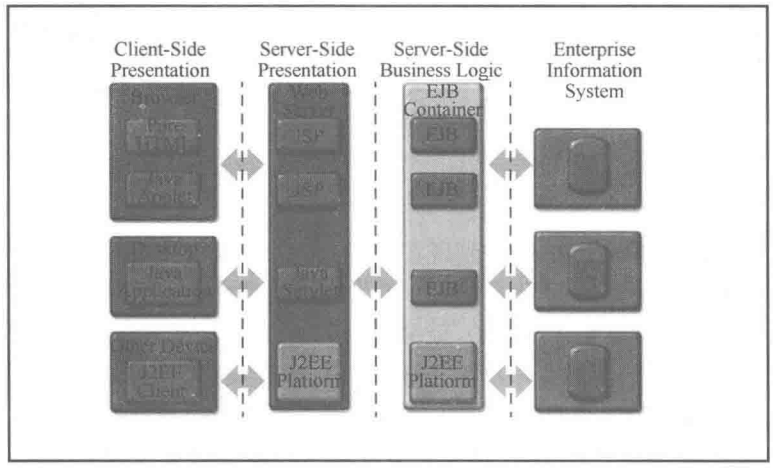


图 1-1　Java EE 平台 4 层结构图

Java EE 平台使得分布式多层应用程序的开发变得更为容易。应用程序的各个组件可以基于功能来进行划分。不同层上的组件可以使用一种名为 MVC 的架构模式来建立协作关系。

1979 年，Trygve Reenshaug 在《Applications Programming in Smalltalk-80:How to Use Model-view-Controller》一文中首次提出了 MVC 的概念。简单地说，MVC 是将一个应用程序划分为三个不同但又相互协作的组件。这三个核心部件分别是模型（Model）、视图（View）、控制器（Controller）。

在 MVC 结构中，模型（Model）代表应用程序的数据（Data）和用于控制访问和修改这些数据的业务规则（Business rule）。通常模型被用来作为对现实世界中一个处理过程的软件，当定义一个模型时，可以采用一般的简单建模技术。

当模型发生改变时，它会通知视图（View），并且为视图提供查询模型相关状态的能力。同时，它也为控制器（Controller）提供访问封装在模型内部的应用程序功能的能力。

一个视图（View）用来组织模型的内容。它从模型那里获得数据并指定这些数据如何表现。当模型变化时，视图负责维持数据表现的一致性，同时将用户要求告知控制器（Controller）。

控制器（Controller）定义了应用程序的行为。它负责对来自视图的用户要求进行解释，并把这些要求映射成相应的行为，这些行为由模型负责实现。在独立运行的 GUI 客户端，用户要求可能是一些鼠标单击或是菜单选择操作。在一个 Web 应用程序中，它们的表现形式可能是一些来自客户端的 GET 或 POST 的 HTTP 请求。模型所实现的行为包括处理业务和修改模

型的状态。根据用户要求和模型行为的结果，控制器选择一个视图作为对用户请求的应答。通常一组相关功能集对应一个控制器。如图 1-2 所示为 MVC 三个组件之间的协作关系图。

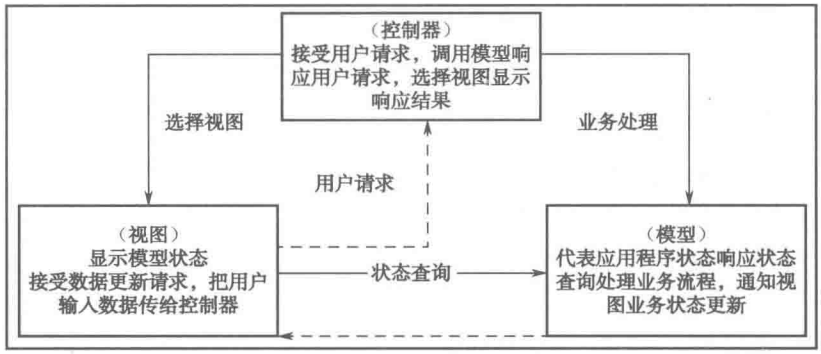


图 1-2 MVC 三个组件之间的协作关系图

Struts 是 Java 语言领域中最早实现 MVC 模块的框架，早在 2000 年，Craig McClanahan 采用了 MVC 的设计模式开发 Struts。随着时间的推移，软件开发领域新技术、新方法和新思想的出现，Struts 中的很多地方已不能适应最新的需求，所以 Struts 和另外一个著名的 Web 框架 WebWork 合并，将新的框架称为 Struts2。Struts2 借助它的历史积淀和优秀的设计成为了企业 Java EE 开发中采用率最高的 Web 框架。

1.1.2 对象关系映射框架

面向对象开发方法是当今的主流，但是同时我们不得不使用关系型数据库，所以在企业级应用开发的环境中，对象关系映射（ORM）是一种耗时的工作。围绕对象关系映射和持久数据的访问，在 Java 领域中发展起来了一些 API 和框架。Hibernate 就是其中的佼佼者。它不仅管理 Java 类到数据库表的映射（包括 Java 数据类型到 SQL 数据类型的映射），还提供数据查询和获取数据的方法，可以大幅度减少开发时手动使用 SQL 和 JDBC 处理数据的时间。

1.1.3 Spring 框架

Spring 是一个 Java EE 开源框架。它是于 2003 年兴起的一个轻量级的 Java 开发框架，由 Rod Johnson 在其著作 Expert One-On-One J2EE Development and Design 中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的，Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。

1.2 本书的结构

本书将会在第 1～3 章介绍如何使用 JSP+Servlet+EL+JSTL+JDBC 的方式来开发小型 Web 系统。

在第 4～12 章介绍如何使用 Ajax 技术及 Struts2、Hibernate 和 Spring 这三个框架来开发大中型 Web 系统。

Oracle 公司收购 Sun 公司后，Java 语言和 Oracle 公司的 Oracle 数据库结合得更加紧密了。本书就以 Oracle 10g 数据库的示例数据库为参照，开发一个简单的员工部门管理系统（HR-System）。

本书案例基于如下版本使用。

- IDE：MyEclipse 8 或以上。
- 数据库服务器：Oracle 10g。
- Web 服务器：Tomcat6.0。

Oracle 10g 数据库的示例数据库。第一张表名为 Dept。脚本如下：

```
CREATE TABLE DEPT(
    DEPTNO NUMBER(2) CONSTRAINT PK_DEPT PRIMARY KEY,
        DNAME VARCHAR2(14) ,
        LOC VARCHAR2(13)
);
```

第二张表名为 EMP。脚本如下：

```
CREATE TABLE EMP(
    EMPNO NUMBER(4) CONSTRAINT PK_EMP PRIMARY KEY,
    ENAME VARCHAR2(10),
    JOB VARCHAR2(9),
    MGR NUMBER(4),
    HIREDATE DATE,
    SAL NUMBER(7,2),
    COMM NUMBER(7,2),
    DEPTNO NUMBER(2) CONSTRAINT FK_DEPTNO REFERENCES DEPT
);
```

本系统有员工模块和部门模块，员工模块具有的功能如下：

（1）可以对员工进行基本信息维护；

（2）可以对部门进行系统维护；

（3）可以改变员工所属部门（从一个部门调往另一个部门）。

代码说明：为了节约篇幅，本书的所有程序清单省略了 import 语句。规范代码请参照本书的相关电子资料。

1.3 JDBC 的使用

1.3.1 JDBC 系统的数据访问层

Java 中访问数据库离不开 JDBC。JDBC 提供对独立于数据库统一的 API，用以执行 SQL 命令。在 SUN Java JDBC 的模块中，JDBC 1.0 的操作都放在 java.sql.*包中。后期发布的 JDBC 2.0，增加了 javax.sql.*这个包。

API 常用的类、接口如下。

（1）DriverManager 类。

管理 JDBC 驱动的服务类，主要通过它获取 Connection 数据库链接，常用方法如下：

```
Connection getConnection(String url, String user, String password)
```

该方法获得 url 对应的数据库的连接。

（2）Connection 接口。

常用数据库操作方法：

```
Statement createStatement :
```

该方法返回一个 Statement 对象。

```
PreparedStatement prepareStatement(String sql)
```

该方法返回预编译的 Statement 对象，即将 SQL 语句提交到数据库进行预编译。

```
CallableStatement prepareCall(String sql)
```

该方法返回 CallableStatement 对象，该对象用于存储过程的调用。

上面的三个方法都是返回执行 SQL 语句的 Statement 对象，PreparedStatement、CallableStatement 的对象是 Statement 的子类，只有获得 Statement 之后才可以执行 SQL 语句。

Statement 用于执行 SQL 语句的 API 接口，该对象可以执行 DDL 语句、DCL 语句，也可以执行 DML 语句，还可以执行 SQL 查询语句，当执行查询语句是返回结果集时，主要方法如下：

```
ResultSet executeQuery(String sql)
```

该方法用于执行查询语句，并返回查询结果对应的 ResultSet 对象，该方法只用于查询语句。

int executeUpdate（String sql）：该方法用于执行 DML 语句，并返回受影响的行数；该方法也可以执行 DDL，执行 DDL 返回 0。

使用 JDBC 对部门表进行操作，详细的步骤如下。

步骤 1

首先，加载对应的数据库的驱动，虽然 JDBC 刚发布时使用 ODBC-JDBC 桥连接的方式，但是在生产环境，首推使用数据库厂商提供的 JDBC 驱动程序来和 Java 进行交互。所以我们要下载 Oracle 10g 对应的 JDBC 驱动类。它被厂商压缩成了 Java 的标准压缩格式，名为 OJDBC14.jar 文件。

将此 jar 文件导入到 Java 项目中，为了简化导入 jar 文件的过程，读者可以使用 MyEclipse 建立一个 Java Web 工程，直接将所需要的 jar 包复制到此 Web 工程的 WebRoot→WEB-INF→lib 文件夹下，MyEclipse 会自动完成 jar 包的配置。

步骤 2

加载驱动类的一般方法如下：

```
Class.forName("驱动类名");
```

对于 Oracle 数据库的驱动类加载的方式如下：

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

步骤 3

得到 Connection 对象。方法如下：

```
Connection con=DriverManager.getConnection(String url, Stirng user, String pass)
```

当使用 DriverManager 来获取链接，需要传入 3 个参数：数据库的 URL、用户名、密码。在连接 Oracle 数据库的情形下，URL 为：

```
jdbc:oracle:thin:@localhost:1521:orcl
```

其中 orcl 为数据库的名字，localhost:1521 分别为数据库的 IP 和端口号。

步骤 4

通过 Connection 对象得到语句对象：

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

步骤 5

通过 Statement 语句对象来执行 SQL 语句。

例如，执行一个添加记录的操作会如下所示：

```
String sql = "insert into DEPT(DEPTNO,DNAME,LOC)values(100,'开发部','广州')";
int row = stmt.executeUpdate(sql);
```

而执行一个查询操作会稍微复杂一些，必须要通过 ResultSet 对象来处理数据库返回的结果。示例如下：

```
String sql = "select DEPTNO,DNAME,LOC from DEPT";
        ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                int dno = rs.getInt("deptno");
                String dname = rs.getString("dname");
                String loc = rs.getString("loc");
                System.out.println("部门编号：" + dno + "，部门名称：" + dname+ "，部门所在地区：" +
loc);
            }
```

步骤 6

关闭 Statement 语句对象和 Connection 对象。

注意关闭顺序，先打开后关闭。

```
rs.close();//如果是 sql 查询语句有返回结果
stmt.close();
conn.close();
```

以上就是一个完整的 JDBC 执行过程。

程序清单 1-1 是对 DEPT 表添加一条记录查询所有部门信息所需的代码。

程序清单 1-1

```
package com.newboy.ch1;
public class JdbcDemo {
    public static void main(String[] args) {
        executeInsert();
        executeQuery();
    }
     // 添加一条部门信息
    public static void executeInsert() {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 加载驱动类，必须要捕获异常
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:orcl", "scott", "tiger");
            stmt = conn.createStatement();
            String sql = "insert into DEPT(DEPTNO,DNAME,LOC)values(100,'开发部','广州')";
            int row = stmt.executeUpdate(sql);
            if (row > 0) {
                System.out.println("执行成功");
            } else {
                System.out.println("执行失败");
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                stmt.close();// 关闭数据库 Statement 对象
                conn.close(); // 关闭数据库连接对象
            } catch (SQLException e) {
                e.printStackTrace();
            }  }//end of finally
    } //end of method executeInsert
     // 查询所有部门的部门信息并在控制台打印
    public static void executeQuery() {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 加载驱动类，必须要捕获异常
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:orcl", "scott", "tiger");
```

```
                stmt = conn.createStatement();
                String sql = "select DEPTNO,DNAME,LOC from DEPT";
                rs = stmt.executeQuery(sql);
                while (rs.next()) {
                        int dno = rs.getInt("deptno");        // deptno 为数据库的列名，以下相同
                        String dname = rs.getString("dname");
                        String loc = rs.getString("loc");
                        System.out.println("部门编号：" + dno + ", 部门名称：" + dname
                                + ", 部门所在地区：" + loc);
                }
        } catch (ClassNotFoundException e) {
                e.printStackTrace();
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        rs.close();                           // 关闭结果集对象
                        stmt.close();                         // 关闭数据库 Statement 对象
                        conn.close();                         // 关闭数据库连接对象
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }//end of finally
    } //end of method executeQuery
} //end of class
```

以上的程序演示了使用 Statement 来完成查询和插入功能。

1.3.2 PreparedStatement 接口

由于 Statement 接口有 SQL 语句注入的风险，所以一般情况下都用 PreparedStatement 接口来取代它。

该 PreparedStatement 接口继承 Statement，但与之在 2 个方面有所不同：

（1）PreparedStatement 实例包含已编译的 SQL 语句。这就是语句“准备好”。包含于 PreparedStatement 对象中的 SQL 语句可具有一个或多个 IN 参数。IN 参数的值在 SQL 语句创建时未被指定。相反的，该语句为每个 IN 参数保留一个问号（？）作为占位符。每个问号的值必须在该语句执行之前，通过适当的 setXXX 方法来提供。

（2）由于 PreparedStatement 对象已预编译过，所以其执行速度要快于 Statement 对象。因此，多次执行的 SQL 语句经常创建为 PreparedStatement 对象，以提高效率。

作为 Statement 的子类，PreparedStatement 继承了 Statement 的所有功能。另外，它还添加了一整套方法，用于设置发送给数据库以取代 IN 参数占位符的值。同时，三种方法 execute、executeQuery 和 executeUpdate 已被更改以使之不再需要参数。这些方法的 Statement 形式（接受 SQL 语句参数的形式）不应该用于 PreparedStatement 对象。

在企业开发中，为了减少代码重复，会将一些多次使用的代码片段重构成父类的方法。在 JDBC 中，也有很多重复性的代码，例如，Connection 对象的创建，Statement 对象和 ResultSet

对象的创建，都会多次出现。所以我们将这些功能代码封装在一个父类 BaseDao 中，代码见程序清单 BaseDao.java。

程序清单 BaseDao.java

```
package org.newboy.ch1;
public class BaseDao {
    private static final String DRIVE = "oracle.jdbc.driver.OracleDriver";
    private static final String URL = "jdbc:oracle:thin:@localhost:1521:orcl";
    private static final String DBUSER = "system";
    private static final String DBPWD = "123456";
    protected Connection con;
    protected PreparedStatement ps;
    protected ResultSet rs;

//1. 取得数据连接 @return Connection
    public  Connection getCon() {
        try {
            Class.forName(DRIVE);
            con = DriverManager.getConnection(URL, DBUSER, DBPWD);
        } catch (ClassNotFoundException e) {
            e.print$tackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return con;
    }
//2. 关闭数据连接相关对象
    public static void closeAll(ResultSet rs, Statement st, Connection con) {
        try {
            if (rs != null)
                rs.close();
            if (st != null)
                st.close();
            if (con != null)
                con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    /** 3. 执行增、删、改操作 */
    public  int executeSQL(String sql, Object[] param) {
        int rows = 0;
        try {
            con = getCon();
            ps = con.prepareStatement(sql);
            if (param != null) {
                for (int i = 0; i < param.length; i++) {
                    //数据库兼容其他类型转为 string 型
```