Sun™ ONE Studio 5
Web 应用程序教程

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

部件号码：817-3298-10
2003 年 10 月 , 修订 A

有关本文档的建议请发到：docfeedback@sun.com

目录

开始之前　11

1.　使用入门　19

获得并安装所需软件　19

启动软件　21

启动 IDE　21

启动应用程序服务器　21

确认 Sun ONE 应用程序服务器 7 作为缺省服务器　27

创建教程数据库表　27

2.　CDShopCart 简介　31

教程应用程序的功能　31

应用程序方案　32

应用程序功能说明　33

教程应用程序的用户视图　33

教程应用程序的结构　37

应用程序元素　38

服务组件详细说明　39

创建教程应用程序任务概述　40

创建 Web 模块　40

使用 JSTL 标记库　41
创建支持元素　41
测试应用程序　41
结尾注释　42

3. 创建 CDShopCart 应用程序　43
创建 Web 模块　43
什么是 Sun ONE Studio 5 Web 模块？　44
创建 CDShopCart Web 模块　44
设置 Web 应用程序的上下文根　46
使用 JSP 标记获取并显示数据库数据　47
什么是 JSP 标记？　47
使用 JSTL 标记　48
创建 [CD Catalog List] 页　51
创建 [Shopping Cart] 页和支持元素　58
创建 CartLineItem Bean　59
创建 Cart Bean　65
创建 [Shopping Cart] 页　68
测试 [Shopping Cart] 页　72
创建三个消息页　73
创建 [Empty Cart] 页　74
创建 [Place Order] 页　75
创建 [Cancel Order] 页　77
测试三个消息页　79

A. CDShopCart 源文件　81
ProductList.jsp 源　82
CartLineItem Bean 源　84
CartLineItemBeanInfo 源　87

Cart Bean 源　91

ShopCart.jsp 源　94

EmptyCart.jsp 源　96

PlaceOrder.jsp 源　96

CancelOrder.jsp 源　97

B. CDShopCart 数据库脚本　99

PointBase 数据库脚本　100

Oracle 数据库脚本　101

Microsoft SQLServer 数据库脚本　102

IBM DB2 数据库脚本　103

C. 利用 Oracle 数据库创建教程　105

将 IDE 连接到 Oracle 数据库　106

启用 Oracle Type 4 JDBC 驱动程序　106

将 IDE 连接到 Oracle Server　107

创建数据库表　108

在 IDE 中查看数据库表　109

利用 Oracle 数据库创建 CDShopCart 应用程序　110

索引　111

图

图 2-1 CDShopCart 应用程序的结构 38

图 3-1 [CD Catalog List] 页 51

图 3-2 [Shopping Cart] 页 58

图 3-3 [Empty Cart] 页 74

图 3-4 [Place Order] 页 76

图 3-5 [Cancel Order] 页 78

表

表 1-1 管理服务器属性值 23
表 1-2 CDCatalog 数据库表 30
表 1-3 CD 表记录 30

开始之前

欢迎使用“Sun™ ONE Studio 5 Web 应用程序”教程。在本教程中，您将学习如何使用 Sun ONE Studio 5, Standard Edition 中介绍的功能，即：

- 对于使用 Java™ Servlet 和 JavaServer Pages™ (JSP™) 技术的 web 应用程序的支持
- 使用来自于 Jakarta Project 的 “JSP 标准标记库” (JSTL) 引用实现的数据库访问
- Sun ONE 应用程序服务器的部署和执行 — 为测试教程应用程序

参见发行说明，以了解书中有关创建各种示例的环境列表。发行说明位于下列 web 页：

`http://forte.sun.com/ffj/documentation/index.html`

屏幕快照将随平台的不同而略有变化。虽然几乎所有过程都使用 Sun ONE Studio 5 软件界面，但有时您可能也需要在命令行中输入命令。这也会随平台的不同而略有差异。例如：Microsoft Windows 命令可能会如下所示：

```
C:>cd MyWorkDir\MyPackage
```

UNIX 命令可能如下所示：

```
% cd MyWorkDir/MyPackage
```

阅读本书须知

本教程将创建一个与数据库进行交互并动态显示生成内容的简单 web 应用程序。设计和体系结构符合 “Java 2 平台，企业版 (J2EE™) 蓝图”。如果您要了解使用 Sun ONE Studio 5, Standard Edition 的功能构建 web 应用程序组件的方法，则您将在完成本教程的过程中受益。

开始阅读本教程之前，应熟悉下列主题：

- Java 编程语言
- Java Servlet 语法
- 启用了 JDBC™ 的驱动程序语法
- JavaServer Pages 语法
- HTML 语法
- 来源于 Jakarta Project 的 JSP 标准标记库 (JSTL)
- 关系数据库概念 （如表和键）
- 使用选定数据库的方法

本书要求您对 Java Servlet 和 JavaServer Pages 概念有所了解，包括 web 应用程序。下列资源定义这些概念：

- *Java Servlet 规范，版本 2.3*
  `http://java.sun.com/products/servlet/download.html#specs`
- *JavaServer Pages 规范，版本 1.2*
  `http://java.sun.com/products/jsp/download.html#specs`

有关 JSTL 教程以及至其它有用信息的链接，参见：
`http://jakarta.apache.org/taglibs/tutorial.html`

注意 – Sun 不对本文档中提及的第三方 web 站点的可用性负责，也不为从这些站点和资源获得的任何内容、广告、产品以及其它材料提供担保或承担任何责任与义务。对由于直接和间接使用这些站点、资源或其拥有的内容、产品或服务，而导致或声称导致的任何损害和损失， Sun 均不负任何责任。

本书的结构

本手册用于自始至终进行阅读。教程中的每一章均基于前面各章中已开发的代码。

第 1 章介绍 CDShopCart 教程的软件要求，说明启动 Sun ONE Studio 5 集成开发环境 (IDE) 以及启动 Sun ONE 应用程序服务器 web 服务器的方法。本章同时介绍了创建教程数据库表的方法。

第 2 章介绍 CDShopCart 应用程序的体系结构。

第 3 章为您提供创建 CDShopCart 应用程序 （一个简单的联机购物车应用程序，用于购买音乐 CD）的步进式说明。

附录 A 提供教程应用程序的完整源文件。

附录 B 提供教程应用程序的数据库脚本文件。

附录 C 介绍如何适应于本教程以使用 Oracle 数据库创建并运行应用程序。

排版惯例

字体	含义	示例
AaBbCc123	命令、文件和目录的名称；计算机屏幕输出	编辑您的 .login 文件。 使用 ls -a 列出所有文件。 Search is complete.
AaBbCc123	键入的内容，以便与计算机屏幕输出相区别	% **su** Password:
AaBbCc123	书名、新词或术语以及要强调的词	请阅读 《*用户指南*》的第 6 章。 这些称作*类*选项。 您*必须*是超级用户才能执行此操作。
AaBbCc123	命令行变量；用实际的名称或值替换	要删除文件，请键入 rm *filename*。

相关的文档

Sun ONE Studio 5 文档包括以 Acrobat Reader (PDF) 格式提供的书籍、发行说明、联机帮助、示例应用程序的自述文件以及 Javadoc™ 文档。

联机文档

本部分介绍的文档可从 docs.sun.com℠ Web 站点和 Sun ONE Studio Developer Resources （Sun ONE Studio 开发人员资源）门户 (http://forte.sun.com/ffj/documentation) 的文档页面中找到。

docs.sun.com Web 站点 (http://docs.sun.com) 使您可以通过因特网阅读、打印和购买 Sun Microsystems 的手册。如果找不到手册，请参见与本地系统或网络上的产品一同安装的文档索引。

- 发行说明 （HTML 格式）

  每个 Sun ONE Studio 5 版本均提供了发行说明。介绍了最新的发行更改和技术说明。

  - 《Sun ONE Studio 5, Standard Edition 发行说明》- 部件号码 817-2337-10

- 入门指南 （PDF 格式）

  介绍如何在每个支持的平台上安装 Sun ONE Studio 5 集成开发环境 (IDE)，还包括其它相关信息，如系统需求、升级说明、应用程序服务器信息、命令行开关、已安装的子目录、数据库集成，以及有关如何使用更新中心的信息。

  - 《Sun ONE Studio 5, Standard Edition 入门指南》- 部件号码 817-3302-10
  - 《Sun ONE Studio 4, Moblie Edition 入门指南》- 部件号码 817-1145-10

- Sun ONE Studio 5 编程系列 （PDF 格式）

  本系列深入介绍了如何使用各种 Sun ONE Studio 5 功能以开发正确格式的 J2EE 应用程序。

  - 《构建 Web 组件》- 部件号码 817-3292-10

    描述如何使用 JSP 页、 servlet、标记库以及支持的类和文件生成一个作为 J2EE Web 模块的 Web 应用程序。

  - 《构建 J2EE 应用程序》- 部件号码 817-3290-10

    描述如何将 EJB 模块和 Web 模块组装到 J2EE 应用程序中，以及如何部署和运行 J2EE 应用程序。

  - 《构建 Enterprise JavaBeans 组件》- 部件号码 817-3288-10

    描述如何使用 Sun ONE Studio 5 EJB 生成器向导和 IDE 的其它组件生成 EJB 组件（会话 Bean、消息驱动 Bean 和包含容器管理持续性或 Bean 管理持续性的实体 Bean）。

  - 《构建 Web 服务》- 部件号码 817-3294-10

    描述如何使用 Sun ONE Studio 5 IDE 生成 Web 服务、如何通过 UDDI 注册表使 Web 服务可供其它服务使用，以及如何通过本地 Web 服务或 UDDI 注册表生成 Web 服务客户机。

  - 《使用 Java 数据库连接》- 部件号码 817-3296-10

    描述如何使用 Sun ONE Studio 5 IDE 的 JDBC 生产率增强工具，包括如何使用这些工具创建 JDBC 应用程序。

- Sun ONE Studio 5 教程 （PDF 格式）

  这些教程演示如何使用 Sun ONE Studio 5, Standard Edition 的主要功能：

  - 《Sun ONE Studio 5 Web 应用程序教程》- 部件号码 817-3298-10

    提供了生成简单的 J2EE Web 应用程序的分步说明。

- 《Sun ONE Studio 5 J2EE 应用程序教程》- 部件号码 817-3300-10

  提供了使用 EJB 组件和 Web Services 技术生成应用程序的分步说明。

- 《Sun ONE Studio 4, Mobile Edition 教程》- 部件号码 817-3861-10

  提供了为无线设备（如移动电话或个人数字助理 (PDA)）生成简单的应用程序的分步说明。此应用程序将与 Java 2 Platform, Micro Edition （J2ME™ 平台）兼容，并符合移动信息设备配置文件 (MIDP) 和联网的受限设备配置 (CLDC)。

您也可以在以下网址找到完整的教程应用程序：
`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

联机帮助

Sun ONE Studio 5 IDE 中提供了联机帮助。要打开帮助，请按帮助键 （在 Microsoft Windows 和 Linux 环境中按 [F1] 键，在 Solaris 环境中按 [Help] 键），或选择 “帮助” → “内容”。执行以上任意操作都将显示一个帮助主题列表和一个搜索工具。

示例

可以在以下 Sun ONE Studio Developer Resources （Sun ONE Studio 开发人员资源）门户下载演示特定 Sun ONE Studio 5 功能以及完整的教程应用程序的示例：

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

此站点包含本文档中使用的应用程序。

Javadoc 文档

许多 Sun ONE Studio 5 模块的 IDE 中均提供了 Javadoc 文档。请参考发行说明以了解如何安装此文档。

使用易读格式的文档

该文档以易读格式提供，以方便残障用户使用辅助技术进行阅读。您还可以按照下表所描述的信息找到文档的易读版本。

文档类型	易读版本的格式和位置
书籍和教程	HTML 格式，位于 `http://docs.sun.com`
迷你版教程	HTML 格式，位于 `http://forte.sun.com/ffj/tutorialsandexamples.html`
集成示例自述文件	HTML 格式，位于 *s1studio-install-directory*/examples 的示例子目录中
发行说明	HTML 格式，位于 `http://docs.sun.com`

与 Sun 技术支持联系

如果您有关于本产品的技术问题而本文档未予以解答，请访问：

`http://www.sun.com/service/contacting`

Sun 欢迎您提出意见和建议

Sun 致力于提高文档质量，并欢迎您提出宝贵的意见和建议。请通过电子邮件将您的意见发送至以下地址：

`docfeedback@sun.com`

请在电子邮件的主题行中包含文档的部件号码 (817-3298-10)。

第 1 章

使用入门

本章说明 Sun ONE Studio 5 “Web 应用程序”教程开始前必须要做的事情。本章包括以下主题：

- “获得并安装所需软件”，其后紧接着
- 第 21 页的 “启动软件”
- 第 27 页的 “创建教程数据库表”

注意 – 本书有几处引用了 *CDShopCart 应用程序文件*。这些文件包括教程应用程序的已完成版本、一个说明如何运行已完成应用程序的自述文件，以及用于创建所需数据库表的 SQL 脚本。这些文件被压缩成一个 zip 文件，可以从 Sun ONE Studio 5 开发人员资源门户下载，网址为：
`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

获得并安装所需软件

下列各项是创建和运行教程所必需的。

- Sun ONE Studio 5, Stardand Edition 软件，其中包括：
  - Sun ONE Studio 5, Stardand Edition 集成开发环境 (IDE)
  - Sun ONE 应用程序服务器 7 软件

    除非检测到已安装了 Sun ONE 应用程序服务器 7 的支持版本，否则 Sun ONE Studio 5, Stardand Edition 安装程序会安装全部两种产品。例如，“Solaris™ 9 操作环境第 2 次更新”包括 Sun ONE 应用程序服务器 7 的安装。

可从以下各处获得 Sun ONE Studio 5, Stardand Edition 软件：

- Sun ONE Studio 5, Stardand Edition 光盘
- Sun ONE 门户 (`http://wwws.sun.com/software/sundev/jde/`)
- Sun ONE 开发人员资源门户 (`http://forte.sun.com/ffj/`)

- Java™ 2 软件开发工具 (J2SE™ SDK)， 1.4.1_02 或更高版本

  如果系统中没有 J2SE SDK， Sun ONE Studio 5, Stardand Edition 安装程序将不会运行。如果系统中有 J2SE SDK，安装程序将启动，然后验证正在使用的 J2SE SDK 是不是 IDE 以及您平台上的 Sun ONE 应用程序服务器 7 所需的版本。如果没有所需版本，安装程序将会退出，同时显示一条消息，提示您在继续进行前必须安装正确的版本。可以从与 IDE 相同的位置获得 J2SE SDK。

- “PointBase 网络服务器”数据库软件

  本教程使用 PointBase 数据库。PointBase 随 Sun ONE Studio 5, Stardand Edition 软件一起安装，位于包含 Sun ONE 应用程序服务器 7 软件的子目录中。如果 IDE 和应用程序服务器是分别安装的，则应用程序服务器不一定包括 PointBase 软件。如果不包括，必须下载并手动安装 PointBase 软件。《Sun ONE 应用程序服务器 7 入门指南》中提供了说明。另外，附录 C 还描述了如何用 Oracle 数据库创建教程应用程序。

- 用于创建教程数据库表的 SQL 脚本

  教程 SQL 脚本在附录 B 中介绍。这些 SQL 脚本文件包括在 CDShopCart 教程的应用程序文件中，可从 Sun ONE Studio 5 开发人员资源门户获得。第 27 页的 “创建教程数据库表”介绍如何在 PointBase 数据库中安装教程数据库表。附录 C 介绍如何在 Oracle 数据库中安装教程表。

- web 服务器

  该教程是 web 应用程序，它需要一个 web 服务器。本教程使用 Sun ONE 应用程序服务器 7 的 web 服务器组件。

- web 浏览器

  您需要一个 web 浏览器来查看教程应用程序页面。使用 Netscape Communicator™ 或 Microsoft Internet Explorer 都可以。

可从发行说明或从 Sun ONE Studio 5 开发人员资源门户的文档页获知一般系统要求，该页位于 `http://forte.sun.com/ffj/documentation/`。

启动软件

本部分介绍在安装了软件之后，如何启动 Sun ONE Studio 5 IDE 和 Sun ONE 应用程序服务器 7。

启动 IDE

有几种方法可以启动 Sun ONE Studio 5 IDE。这里只介绍一种。要了解更多选择方法，参见 《Sun ONE Studio 5, Standard Edition 入门指南》。

启动 IDE：

- **运行程序可执行文件，启动 Sun ONE Studio 5 IDE**。
  - 在 Microsoft Windows 中，选择 [开始] → [程序] → [Sun Microsystems] → [Sun ONE Studio 5 SE] → [Sun ONE Studio 5 SE]
  - 在 Solaris、 UNIX 和 Linux 环境下，在终端窗口中运行 `runide.sh` 脚本，如下所示：

```
$ sh s1studio-install-directory/bin/runide.sh
```

变量 *s1studio-install-directory* 代表 IDE 的起始目录，其缺省值为 `$HOME/studio5_se`（UNIX 标准用户）或 `/opt/studio5_se`（UNIX 超级用户）。

启动应用程序服务器

在开始本部分之前，您必须具有对应用程序服务器域的写访问权限。缺省域在安装期间创建，并且需要具有超级用户特权 （Microsoft Windows 系统中的管理员特权或 Solaris 或 Linux 环境下的根特权）才能访问。如果您的用户 ID 具有超级用户特权，您可以使用所有缺省设置启动应用程序服务器，如下一部分所述。标准用户 （没有超级用户特权）必须使用第 23 页的 “启动管理服务器 （标准用户）”中所述的步骤。

启动管理服务器 （超级用户）

如果先前已启动了管理服务器，请确认它是否正在运行。有关信息，参见第 27 页的“确认 Sun ONE 应用程序服务器 7 作为缺省服务器”。如果是首次启动管理服务器，请从本部分开始。

启动管理服务器：

1. **在 IDE 中，选择 [资源管理器] 的 [运行环境] 标签。**

   [资源管理器] 的 [运行环境] 窗格显示其它节点当中的 [服务器注册表] 节点。此节点包含与所有已安装 web 及应用程序服务器相应的子节点，同时还包含一个节点，该节点显示哪些服务器是缺省服务器。

2. **选择 [服务器注册表] 节点。**

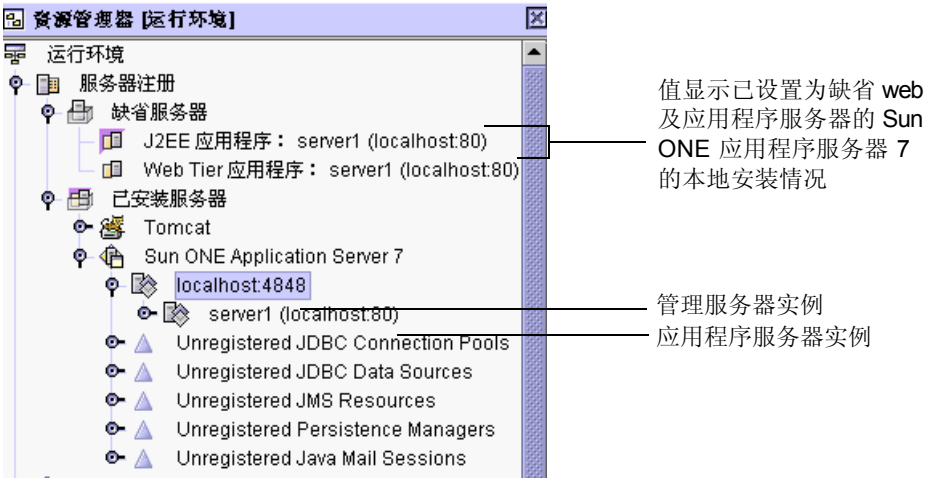
   弹出一个查询窗口，询问您是否要启动管理服务器。这是指缺省域的管理服务器，它只能由具有特权的用户运行。

3. **单击 [确定] 启动缺省管理服务器。**

   IDE 会启动缺省管理服务器并将 Sun ONE 应用程序服务器 7 配置为 IDE 的缺省应用程序服务器。

4. **展开 [服务器注册表] 节点、 [已安装的服务器] 节点和 [Sun ONE 应用程序服务器 7] 节点。**

   [资源管理器] 中的 [服务器注册表] 显示如下：



现在，请按第 26 页的 “启动应用程序服务器实例”中的说明启动服务器实例。

启动管理服务器 （标准用户）

如果您的用户 ID 没有超级用户特权，在开始这一部分之前，必须请超级用户为您创建一个域。步骤说明见 《Sun ONE Studio 5, Standard Edition 入门指南》。

如果先前已启动了 IDE、创建并启动了一个管理服务器，现在请使用第 27 页的“确认 Sun ONE 应用程序服务器 7 作为缺省服务器”中所述的步骤，确认它是否正在运行。如果是首次启动管理服务器，请从本部分开始。

在开始本部分所述步骤之前，需要几个域属性的值。管理员能为您提供这些值。使用下表记录这些属性值：

表 1-1 管理服务器属性值

管理服务器属性	值
管理服务器主机	
管理服务器端口	
用户名	
用户口令	
域	

启动管理服务器：

1. **在 IDE 中，选择 [资源管理器] 的 [运行环境] 标签。**

   [资源管理器] 的 [运行环境] 窗格显示其它节点当中的 [服务器注册表] 节点。此节点包含与所有已安装 web 及应用程序服务器相应的子节点，同时还包含一个节点，该节点显示哪些服务器是缺省服务器。

2. **选择 [服务器注册表] 节点。**

   弹出一个查询窗口，询问您是否要启动管理服务器。这是指缺省域的管理服务器，它只能由具有特权的用户运行。

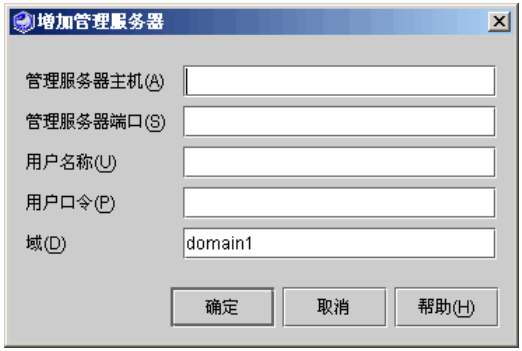
   如果您单击 [确定]，此操作会创建并启动一个您无法使用的管理服务器。单击 [取消]。

3. **将管理服务器添加到 IDE 中。**

   a. **展开 [服务器注册表] 节点和 [已安装的服务器] 节点。**

   b. **右键单击 [Sun ONE 应用程序服务器 7] 节点并选择 [增加管理服务器]。**

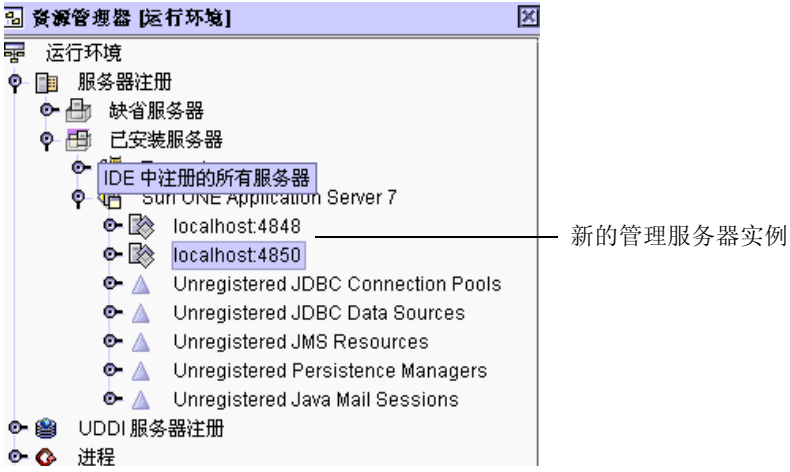
      会显示 [增加管理服务器] 对话框。



c. 键入表 1-1 **中的值，然后单击 [确定]**。

如果出现错误消息，声称 IDE 找不到管理服务器但如果它是本地的则会启动它，请单击 [确定] 关闭错误窗口。会出现一个进度窗口，显示管理服务器启动进展情况。

在 [资源管理器] 中会生成一个新的管理服务器节点。在下面的屏幕快照中，新管理服务器的主机为 localhost，端口号为 4850。



4. 创建应用程序服务器实例。

a. 右键单击新的管理服务器节点并选择 [创建服务器实例]。

会显示 [输入服务器实例值] 对话框。

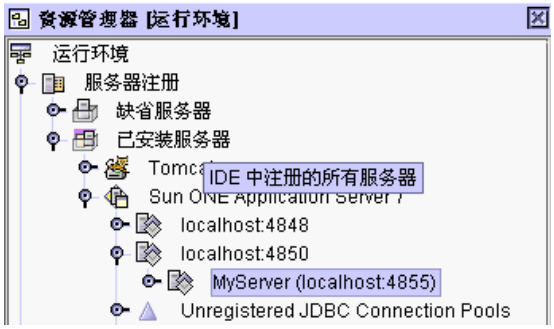
b. 键入名称和端口号。

例如，可以键入 `MyServer` 和 `4855`。

注意 – 在 Solaris 和 Linux 系统中， 1024 以下的端口号为保留值。请使用 1023 以上的端口号。在所有系统中都不要使用端口号 80，它是缺省应用程序服务器使用的端口号。

c. 单击 [确定]。

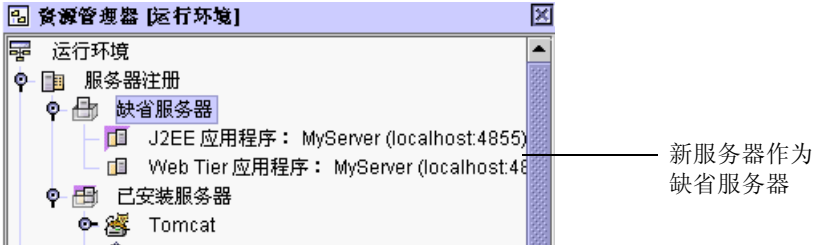
此操作将启动管理服务器，可以根据输出窗口和状态栏中的消息验证这一点。会在 IDE 中创建新的服务器实例。



5. 右键单击新的服务器实例并选择 [设置为缺省]，设置缺省应用程序及 web 服务器。

6. 展开 [缺省服务器] 节点，验证此操作。

J2EE 应用程序和 web 层应用程序的缺省服务器会将此新的服务器显示为缺省服务器。



现在按下一部分的说明启动服务器实例。

启动应用程序服务器实例

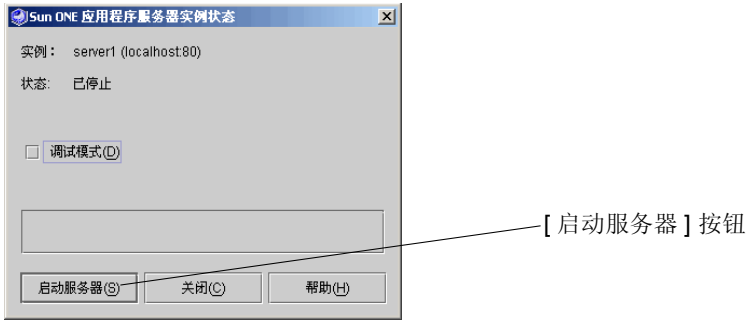
当您在开发期间进行测试部署和部署应用程序时，只要管理服务器正在运行， IDE 就会自动启动应用程序服务器实例。在这一部分，您将手动启动应用程序服务器实例，以便执行本章后面所述的一些操作。

所有用户均可按下列步骤启动服务器实例：

1. **右键单击应用程序服务器节点并选择 [状态]**。

注意 – 如果没有显示该节点，请选择管理服务器实例节点，然后选择 [刷新]。

将显示如图所示的 [Sun ONE 应用程序服务器实例状态] 对话框 （您的实例标签可能会有所不同）。



2. **单击 [启动服务器] 按钮**。

（如果对话框中出现的是 [停止服务器] 按钮，说明服务器已经在运行。）

在 Microsoft Windows 系统中，会出现一个命令窗口，显示进度消息。

当 [服务器实例状态] 窗口显示 [状态：正在运行] 时，服务器就被启动了。

3. **单击实例状态对话框中的 [关闭]**。

现在，请继续执行第 27 页的 “创建教程数据库表”。

确认 Sun ONE 应用程序服务器 7 作为缺省服务器

如果之前已启动了 Sun ONE 应用程序服务器 7，请按以下步骤确认它仍然是缺省服务器：

1. **在 IDE 中，选择 [资源管理器] 的 [运行环境] 标签。**
2. **展开 [服务器注册表] 节点及其 [缺省服务器] 子节点。**

   如果 [Web 层应用程序] 节点的标签是 *server-instance*（*server-hostname*：*server-port-number*），如图所示，则 Sun ONE 应用程序服务器 7 为缺省 web 服务器。转到下一部分。否则，继续进行下一步骤。

值显示已设置为缺省 web 服务器的 Sun ONE 应用程序服务器 7 的本地安装情况

3. **在 [已安装的服务器] 节点下找到 web 服务器实例，右键单击，然后选择 [设置为缺省]。**

   您的服务器会被设置为 J2EE 和 “Web 层”应用程序的缺省服务器。

创建教程数据库表

必须先在 “PointBase 网络服务器”数据库中创建并安装数据库表后，方能开始 CDShopCart 教程。请使用附录 B 中的 SQL 脚本创建这些表。脚本文件，`CDCatalog_pb.sql`，也可在 CDShopCart 教程的 `cdshop.zip` 文件中获得，该文件可从以下地址得到：

```
http://forte.sun.com/ffj/documentation/tutorialsandexamples.html
```

注意 – 如果 Sun ONE Studio 5 IDE 和 Sun ONE 应用程序服务器 7 不是一起安装的，则必须执行操作将 IDE 和应用程序服务器连接到 PointBase 数据库软件，这些操作在《Sun ONE Studio 5, Standard Edition 入门指南》中有述。在开始以下步骤之前，必须执行这些操作。

在 PointBase 数据库中安装教程表：

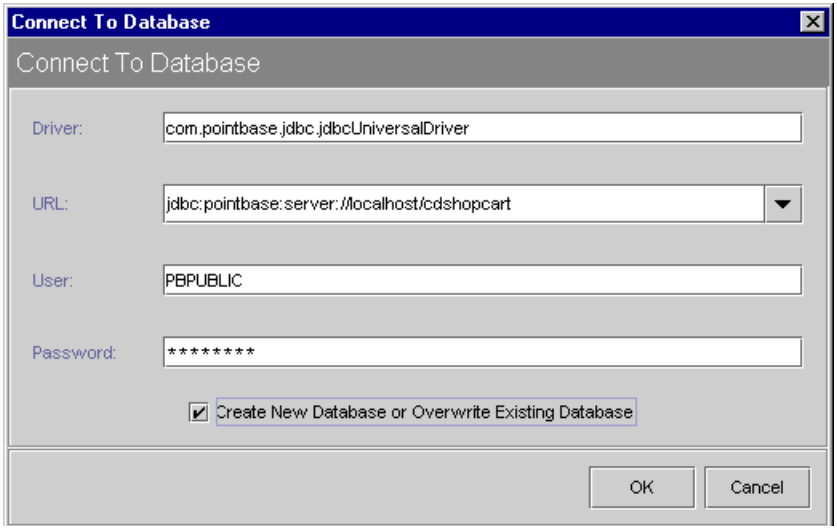
1. **选择 [工具] → [PointBase 网络服务器] → [启动服务器]，从 IDE 中启动“PointBase 服务器”。**

   会显示 [PointBase 网络服务器] 窗口。将此窗口最小化。

2. **选择 [工具] → [PointBase 网络服务器] → [启动控制台]，从 IDE 中启动“PointBase 控制台”。**

   出现 [连接到数据库] 对话框，显示缺省 `sample` 数据库的 PointBase 驱动程序值。

3. **将 [URL] 字段末尾的词 `sample` 更改为 `cdshopcart`，如图所示。**



4. **设置 [创建新数据库] 选项，然后单击 [确定]。**

   会显示 “PointBase 控制台”。等到显示以 “就绪”结尾的状态消息时，再继续往下进行。

5. **从第 100 页的“PointBase 数据库脚本”中复制 PointBase 脚本，并将其粘贴到“控制台”的 SQL 输入窗口中。**

   或者，如果您从教程源 zip 文件中得到了 `CDCatalog_pb.sql` 文件，也可执行以下操作：

   a. **选择 [文件] → [打开]，显示文件浏览器对话框。**

   b. **使用文件浏览器查找 `CDCatalog_pb.sql` 文件，然后单击 [打开]。**

6. 选择 **[SQL]** → **[执行全部]**。

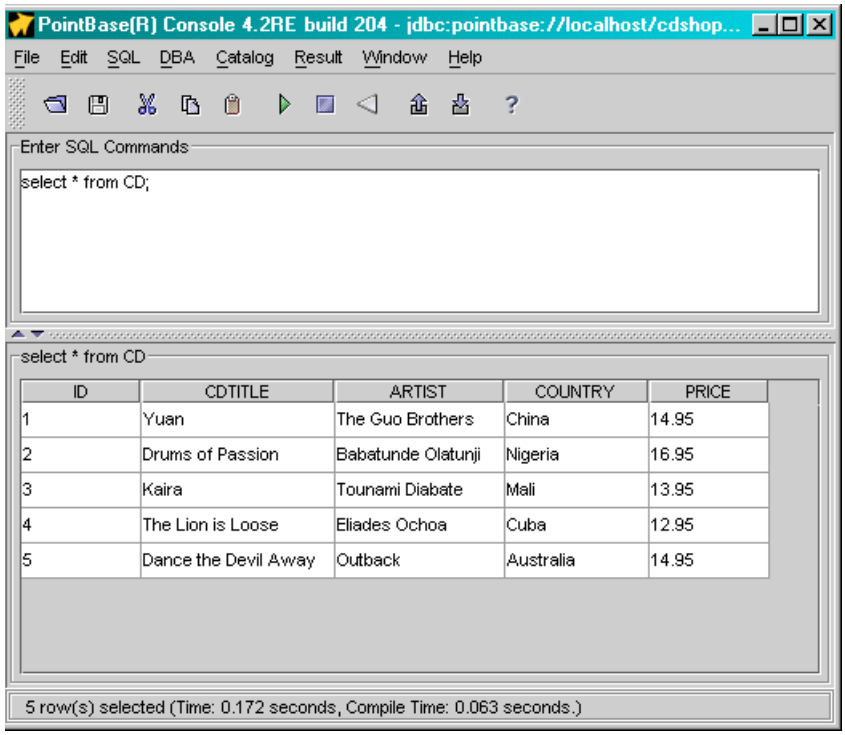
   会出现消息窗口，确认已执行了脚本。（请忽略开头为 “找不到表 ...” 的初始消息。该错误无害，它之所以出现是因为某个表尚有一条 DROP 语句未创建。如果您将来想再次运行此脚本来初始化该表，将会用到这条 DROP 语句。）

7. 清除 **SQL** 输入窗口 （**[窗口]** → **[清除输入]**），测试是否已创建了该表，然后键入：

```
select * from CD;
```

8. 选择 **[SQL]** → **[执行]**。

   控制台应显示 CD 表。



注意 – 如果您看到的内容与本表不同，请选择 [窗口] → [多窗口] 以更改显示类型。

---

9. 关闭 **[PointBase 控制台]** 窗口。

CDCatalog 脚本会创建如表 1-2 所示的数据库模式。

表 1-2 CDCatalog 数据库表

表名	列	主键	其它
CD	id	是	
	cdtitle		
	artist		
	country		
	price		

CD 表填充了表 1-3 中所示的记录。

表 1-3 CD 表记录

ID	CDtitle	Artist	Country	Price
1	Yuan	The Guo Brothers	China	14.95
2	Drums of Passion	Babatunde Olatunji	Nigeria	16.95
3	Kaira	Tounami Diabate	Mali	13.95
4	The Lion is Loose	Eliades Ochoa	Cuba	12.95
5	Dance the Devil Away	Outback	Australia	14.95

现在一切就绪，可以开始建立教程应用程序了。可以继续阅读第 2 章以了解将要建立的应用程序的概貌，也可直接转到第 3 章，开始建立。

第 2 章

CDShopCart 简介

在创建教程实例应用程序的过程中，您将学会如何生成具有 Sun ONE Studio 5, Standard Edition 功能的 web 应用程序的组件。

本章介绍将要生成的应用程序，首先罗列出应用程序的要求，然后介绍满足这些要求的结构。最后部分介绍如何使用 Sun ONE Studio 5 功能 — web 模块构造和 Apache 标记库 — 来创建应用程序。

本章被组织成下列部分：

- “教程应用程序的功能”，其后紧接着
- 第 33 页的 “教程应用程序的用户视图”
- 第 37 页的 “教程应用程序的结构”
- 第 40 页的 “创建教程应用程序任务概述”

教程应用程序的功能

教程实例应用程序，CDShopCart，是一个简单的用于购买音乐 CD 的联机购物车应用程序。客户使用 web 浏览器与应用程序界面进行交互，如下所示：

1. 客户从目录页选择要添加到购物车的项目。

2. 客户可以从目录页中添加更多项目，或者从购物车中删除现有项目。

3. 当客户准备进行购买时，应用程序会显示一条订购感谢消息并结束会话。

4. 其后，客户可以退出应用程序，或返回到订购页开始一个新的购物会话。

应用程序方案

CDShopCart 应用程序的交互以客户访问应用程序目录页开始，在客户完成订购或退出站点时结束。下列方案描述了客户与 CDShopCart 应用程序所进行的若干交互。对这些方案进行预排不仅会阐明应用程序的要求，而且会阐明在应用程序内部发生的交互。

1. 客户通过将浏览器指向应用程序主页的 URL 来启动应用程序。

   主页是 [CD Catalog List] 页，它显示可以获得的音乐 CD 列表及其相关信息：名称、CD id 号、演员名、演员国家以及价格。

2. 客户通过单击与 CD 相关联的 [Add] 按钮来选购 CD。

   此操作将令应用程序显示 [Shopping Cart] 页面，其中会列出所选 CD 名称及其 ID 号和价格。

3. 客户选购更多的 CD。

   客户单击 [Shopping Cart] 页上的 [Resume Shopping] 按钮，将会令应用程序重新显示 CD 目录页，以便做进一步的选择。客户可以随意多次重复此操作序列，甚至可以多次添加相同的 CD （这会在购物车中添加更多行相同的 CD — 每一 CD 都没有相应的 [Amount] 栏）。

4. 客户通过单击 [Shopping Cart] 页上与项目相关联的 [Delete] 按钮，从购物车中删除该项目。

   单击此按钮会重新显示除去了该项目的购物车，除非客户删掉了车中的最后一张 CD。删除最后一张 CD 后，会显示一个页面，报告购物车是空的。

5. 客户可以单击该页上的 [Resume Shopping] 按钮，返回到 [CD Catalog List] 页，或者单击 [Cancel Order] 按钮结束会话。（[Cancel Order] 页在方案 7 中讨论。）

6. 客户决定进行购买，并单击 [Shopping Cart] 页上的 [Place Order] 按钮。

   此操作显示一条感谢消息并结束会话。客户可以单击消息页上的 [Resume Shopping] 链接开始另一个会话，也可关闭浏览器或转到不同的 URL 来离开应用程序。

7. 客户通过单击 [Shopping Cart] 页上的 [Cancel Order] 按钮，随时取消订购。

   这会令应用程序显示 [Cancel Order] 消息页以结束会话。 [Cancel Order] 页中有一个 [Resume Shopping] 按钮，以防客户想要开始一个新的会话。

应用程序功能说明

给定将要使用 CDShopCart 应用程序的方案种类，下列各项将列出支持这些购物交互的应用程序用户界面的主要功能。

- 一组页间导航链接
- 通过显示列表提供的所有站点内容的主视图
- 已选购项目视图
- 目录页上用于添加每一项目的按钮
- 购物车页上用于删除项目的按钮
- 购物车页上开始进行结帐的按钮
- 购物车页上取消订购的按钮
- 结帐页上返回到主页开始新订购的按钮
- 空购物车页上返回到主页的按钮
- 空购物车页上取消订购的按钮

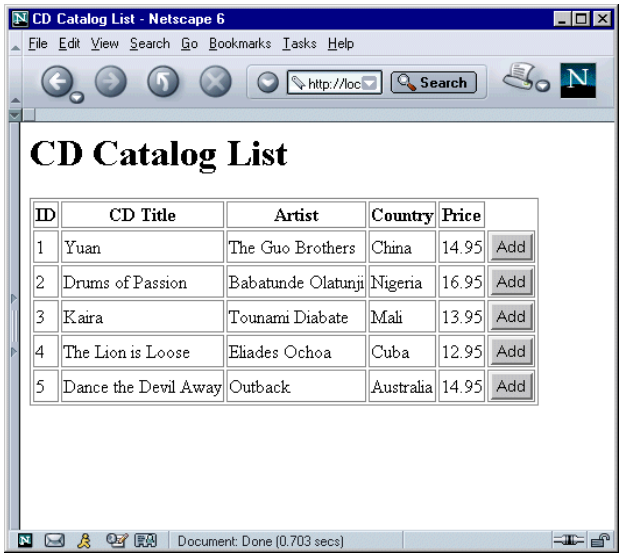
教程应用程序的用户视图

应用程序的用户视图阐明了第 31 页的 “教程应用程序的功能”中所述的方案和功能说明是如何实现的。

运行 CDShopCart 应用程序：

1. **应用程序以显示 CD 名称列表的 [CD Catalog List] 页开始。**

   该页是用 ProductList JSP 页创建的。



2. **要将某张 CD 添加到购物车中，单击该 CD 行中的 [Add] 按钮。**

   此操作将显示 [Shopping Cart] 页，其中含有所选的 CD。该页是由 ShopCart JSP 页创建的。

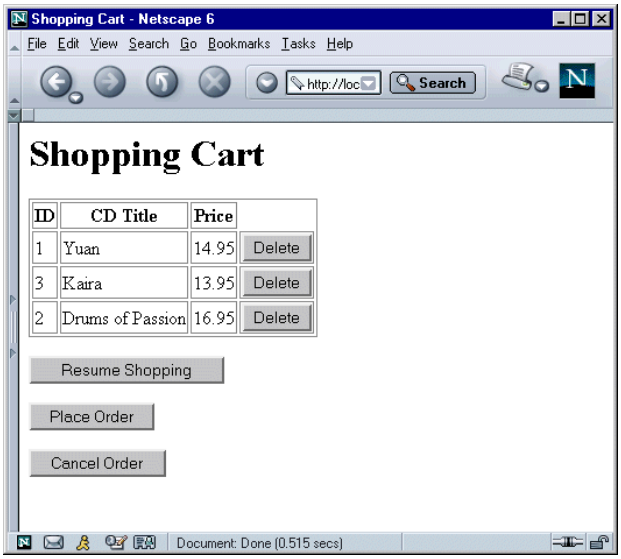
3. **要添加另一张 CD，单击 [Resume Shopping]，这会使您回到 [CD Catalog List] 页。**

4. **在相同或不同的 CD 上，单击 [Add]。**

   会重新显示 [Shopping Cart] 页，其中含有其它选择。

5. **重复步骤 2 和步骤 3 直到您选好了想要购买的所有 CD。**

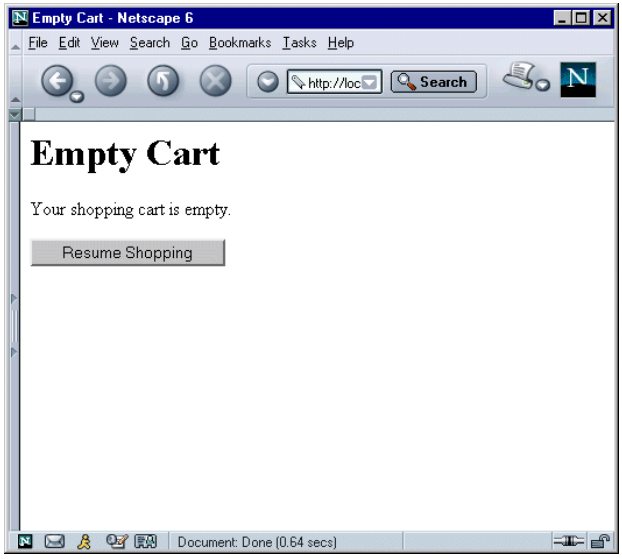
   [Shopping Cart] 会显示您所选的项目。如果选择了多个相同的项目，这些项目会分行显示。



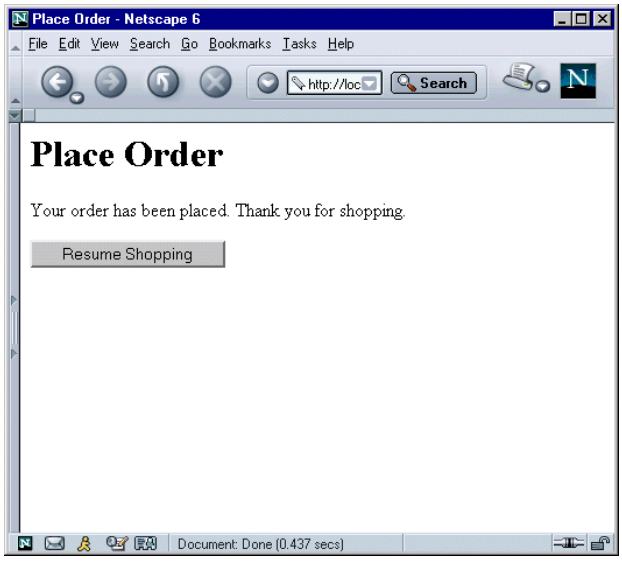
6. **要删除一个项目，单击该项目的 [Delete] 按钮。**

   表格会重新显示，其中去掉了已删除的项目。

如果删除表格中的最后一项，会显示 Empty Cart JSP 页：



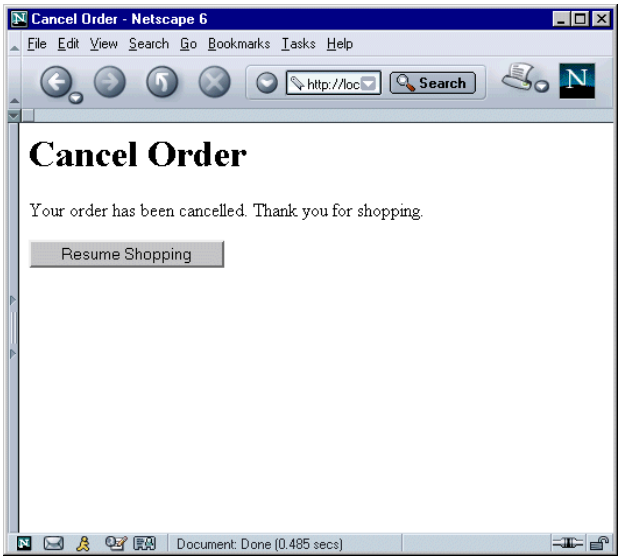
7. **单击 [Resume Shopping] 按钮返回到 [CD Catalog List] 页。**
8. **要执行一次订购，单击 [Shopping Cart] 页上的 [Place Order] 按钮。**
   将会显示 [Place Order] 页。该页是由 PlaceOrder JSP 页创建的。



既可以通过将浏览器指向不同的 URL 来退出应用程序，也可以通过单击 [Resume Shopping] 按钮来开始新的会话。

9. **或者，要取消订购，请在 [Shopping Cart] 页中，单击 [Cancel Order] 按钮。**

将会显示 [Cancel Order] 页。该页是由 CancelOrder JSP 页创建的。



可以单击 [Resume Shopping] 按钮来开始新的会话。

教程应用程序的结构

CDShopCart 应用程序是一个以 web 为中心的应用程序，它使用 web 客户机向 web 应用程序发送请求，并且从 web 应用程序那里接收结果。 *web 应用程序*由 web 组件及其支持类、bean 和 文件捆绑而成。*Web 组件*是服务器端的 J2EE 组件，例如，Servlet 和 JSP 页。

CDShopCart 应用程序由单个 web 模块组成。 *web 模块*是 J2EE 应用程序中可部署和使用的最小 web 资源单元。 Sun ONE Studio 5 IDE 实现了 *Java Servlet 规范 2.3 版*中介绍的一项功能，即 web 模块构造，它会自动创建所需的目录结构、所需数据对象的缺省版本以及 web 模块所需的其它特殊服务。

有关 web 模块及相关概念的详细信息，参见 《构建 Web 组件》。有关 web 模块构造的特定信息，参见联机帮助中 [JavaServlet 页和 Servlet] 下的 [开发 Web 模块] 部分。

图 2-1 展示了 CDShopCart 应用程序元素及其相互关系。

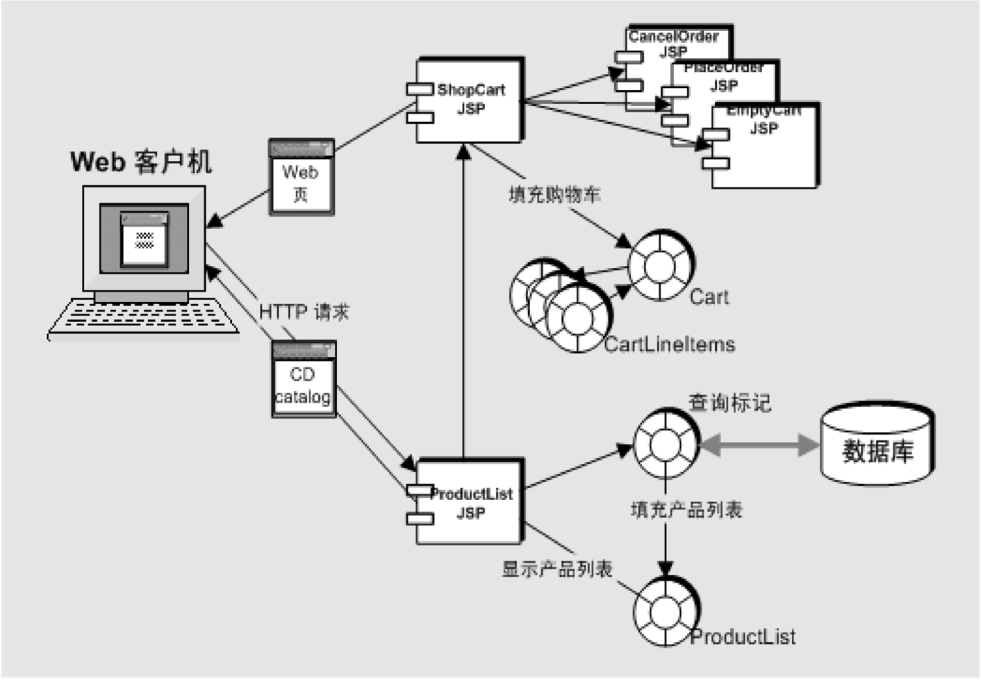


图 2-1 CDShopCart 应用程序的结构

应用程序元素

简单说来，图 2-1 中所示的元素有：

- *客户机*组件

  客户机组件是一种显示应用程序页的 web 浏览器。

- *服务*组件 （web 模块），其中包括：
  - ProductList JSP 页，它从数据库中检索产品数据并将其显示在 [CD Catalog List] web 页上的表格中。 ProductList 还提供了一个 [Add] 按钮，通过它，用户可以将 CD 添加到购物车中。
  - ShopCart JSP 页，它显示在 [Shopping Cart] web 页上的表格中选购的 CD，并且提供了 [Delete]、 [Place Order]、 [Cancel Order] 和 [Resume Shopping] 按钮。
  - EmptyCart JSP 页，它将在客户删除购物车中的最后一项时显示一条消息。其中包括一个 [Resume Shopping] 按钮。

- CancelOrder JSP 页，它显示订购已取消消息以及一个用于返回到 ProductList JSP 页的 [Resume Shopping] 按钮。
- PlaceOrder JSP 页，它显示订购已执行消息以及一个用于返回到 ProductList JSP 页的 [Resume Shopping] 按钮。
- Cart bean，它表示选购的项目。
- CartLineItem bean，它表示购物车行项目。
- CartLineItemBeanInfo beaninfo 组件，它指定 id 和 price 是 CartLineItem 的属性，不管是否已重载了 setter 方法

服务组件详细说明

CDShopCart 应用程序的服务组件是一个包括四个 JSP 页的 web 模块，假定有来自客户机的输入，这些 JSP 页将会协调应用程序的行为。此 web 模块的支持元素包括 JavaBeans 元素和一个 HTML 页文件。

- ProductList JSP 页

  此页为当前用户定位会话，或者如果会话不存在，则创建一个会话。`ProductList` 使用 Apache Foundation 的 “JSP 标准标记库” (JSTL) 中的 SQL 标记来从数据库中访问 CD 列表并使用核心标记将其显示在表格中。`ProductList` 还在其显示的每个 CD 行项目上提供了一个 [Add] 按钮。

- ShopCart JSP 页

  当用户单击 [ProductList] 页上的 [Add] 按钮时，会将行项目的数据传递给该 JSP 页，该页将会实例化一个由 CartLineItem 对象组成的 Cart 对象，然后使用 JSTL 标记库中的核心标记将它们显示在表格中。 ShopCart 在每个购物车项目上提供了一个 [Delete] 按钮。单击此按钮时，此页将使用一个 scriptlet 来删除该项目，更新表格数据，然后重新显示该表格。如果所删除的项目是购物车中的最后一项， ShopCart 会转至 [EmptyCart] 页。 ShopCart 还提供了 [Resume Shopping]、 [Cancel Order] 和 [Place Order] 按钮。使用这些按钮可分别转到 [ProductList] 页、 [CancelOrder] 页和 [PlaceOrder] 页。

- Cart JavaBeans 组件

  该 bean 具有一个 lineItems 属性，并且包括用于获取和删除 CartLineItem 对象的方法。该 bean 是由 [ShopCart] 页输入的。

- CartLineItem JavaBeans 组件

  该 bean 具有与 CD 相关的属性，并且包括用于获取和设置 Cart 行项目属性 （ID、 title、 artist、 country 和 price）的方法。

- CancelOrder JSP 页

  当用户单击 [ShopCart] 页上的 [Cancel Order] 按钮时，会调用该 JSP 页。该页将使会话失效，显示订购被取消消息，并提供一个用于返回到 [ProductList] 页的 [Resume Shopping] 按钮。

- EmptyCart JSP 页

  当用户从 [ShopCart] 页中删除最后一个项目时，会调用该 JSP 页。EmptyCart 显示购物车已空消息，并且包含一个 [Resume Shopping] 按钮。

- PlaceOrder JSP 页

  在购物车中有项目的情况下，当用户单击 [ShopCart] 页上的 [Place Order] 按钮时，会调用该 JSP 页。 PlaceOrder 将显示订购已执行消息，使会话失效，并且包含一个 [Resume Shopping] 按钮。

创建教程应用程序任务概述

教程仅有一章内容，在此章中，您将创建基本的应用程序。在创建教程应用程序之前，必须先安装和设置要运行的 Sun ONE Studio 5 软件，同时必须安装教程数据库表，如第 1 章中所述。

在第 3 章中，您会了解如何使用下列 Sun ONE Studio 5, Standard Edition 功能：

- Web 模块 （创建、开发和试运行）
- 用于连接数据库并与之交互的嵌入 JSTL 标记
- 用于在检索到的数据中进行迭代并呈现这些数据的嵌入 JSTL 标记

此外，您还可以创建支持元素：若干个 bean 和一个 HTML 页。

创建 Web 模块

Sun ONE Studio 5 IDE 提供了用于自动创建 web 应用程序分层目录结构的工具。该结构是 web 模块。您将在单个 web 模块构造中开发整个 CDShopCart 应用程序。

此教程并不试图提供关于如何开发 web 模块的完整信息。第 43 页的“创建 Web 模块”部分作为主题简介，概括了结构的基本元素以及用于创建结构的 （非常容易的）方法。要想更多地了解如何开发 web 模块，参见 《构建 Web 组件》和联机帮助。

使用 JSTL 标记库

在 CDShopCart web 模块中，将会创建用于取得 CD 目录数据以便在 [CD Product List] web 页上进行显示的 ProductList JSP 页。然后将创建显示用户选购 CD 的 ShopCart JSP 页。将会使用用于数据库访问和数据呈现功能的 JSTL 标记来完成此项任务。

第 47 页的 “使用 JSP 标记获取并显示数据库数据”部分介绍如何使用 SQL 标记来进行 JDBC 数据库连接并取得 CD 数据。然后介绍如何使用核心标记来在结果数据中进行迭代，以便 ProductList 能够在 web 页上的 HTML 窗体中显示这些数据。

ShopCart JSP 页显示由 ProductList JSP 页传递而来的 CD 数据。第 68 页的 “添加代码以从 Shopping Cart 表中添加或删除项目”部分示范如何使用核心 JSTL 标记在传递而来的值中进行迭代来查找各个字段值，以便能将这些值显示在购物车表格正确的栏中。

创建支持元素

ShopCart JSP 页的支持元素为两个 bean （Cart 和 CartLineItem）和三个 JSP 页（CancelOrder、 PlaceOrder 和 EmptyCart）。

第 59 页的 “创建 CartLineItem Bean”部分将为您展示，当用户单击项目上的 [Add] 按钮时，如何创建一个 bean，其对象包含自 ProductList 传递到 ShopCart 的行项目参数。然后，在第 65 页的 “创建 Cart Bean”中，您将了解如何创建一个 bean，其对象含有已选定的累积行项目。 Cart bean 具有用于向购物车中添加和从中删除行项目的方法。

在第 74 页的 “创建 [Empty Cart] 页”中，将创建一个显示购物车已空消息的 JSP 页。这是为了避免在 [Shopping Cart] 页上显示一个空窗体。

还将创建另两个 JSP 页，与 [ProductList] 和 [ShopCart] 相比，其逻辑程度较低。在第 75 页的 “创建 [Place Order] 页”中，将会创建一个感谢用户执行订购而后使会话失效的 JSP 页。在第 77 页的 “创建 [Cancel Order] 页”中，将创建一个类似的页，通知用户订购被取消并使会话失效。

测试应用程序

在整章中，在您创建了每个元素之后，便可紧接着对其进行测试。当您执行一个 web 模块的某一组件时， IDE 会自动将该 web 模块部署到它的内部容器中。

结尾注释

此教程应用程序设计得相当简洁，足以让您在相对短的时间内 （大约一天）完成创建过程。这样，其范围便受到了某些限制。例如：

- 没有错误处理。
- 没有调试过程。
- 没有如何创建用于部署的 WAR 文件的说明。

未来版本将具有这些过程。

尽管为使您能够快速完成，教程应用程序设计得比较简单，但您或许仍想要输入完整的应用程序，查看源文件，或者将方法代码复制并粘贴到您创建的方法中。 CDShopCart 应用程序可从 Sun ONE Studio 5 开发人员资源门户获得，网址为 `http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

第 3 章

创建 CDShopCart 应用程序

本章将逐步说明如何创建 CDShopCart 应用程序。在创建教程应用程序之前，必须先安装 Sun ONE Studio 5 软件并运行它，同时还必须安装教程数据库表，如第 1 章所述。

本章按下列主题进行组织：

- “创建 Web 模块”，其后紧接着
- 第 47 页的 “使用 JSP 标记获取并显示数据库数据”
- 第 58 页的 “创建 [Shopping Cart] 页和支持元素”
- 第 73 页的 “创建三个消息页”

在创建时对每一组件都要进行测试。在本章结尾，您将能够运行基本的应用程序，如第 2 章所述。

提示 – 本章描述的所有 JSP 页的完整源代码和 JavaBean 组件都可以在附录 A 中找到。

创建 Web 模块

CDShopCart 应用程序是一个 web 应用程序。 Web 应用程序由 web 模块组成。CDShopCart 应用程序是一个非常简单的应用程序，仅包含一个 web 模块。

本节将讲述如何使用 Sun ONE Studio 5, Standard Edition IDE 在一个模块内实现购物车功能。

什么是 Sun ONE Studio 5 Web 模块？

根据 *Java Servlet 规范，版本 2.3* 所述，“web 应用程序是一种结构化的目录分层结构。”此分层结构的根是容纳所有文件的文档根，这些文件是 web 应用程序的组成部分。对于与 web 应用程序相关联但不直接服务于客户机的项目，分层结构还包括一个特殊的非公共子目录 `WEB-INF`。`WEB-INF` 目录中的项目包括 web 部署描述符（`web.xml` 文件）、servlet 以及公用程序类，这些公用程序类被 web 应用程序加载器用来加载类。

应用程序的文件最终必须是 web 模块结构的一部分，这些文件被封装起来作为 WAR 文件（一种 Web 存档格式文件）并传送到 web 容器中。Sun ONE Studio 5 IDE 的 web 模块功能可以自动进行许多创建所需目录分层结构的过程，也可以自动用某些对象的缺省版本填写分层结构。

注意 – 本指南并不试图提供有关开发 web 模块的完整信息。要了解该任务，参见 《构建 Web 组件》一书。也可以咨询 Sun ONE Studio 5 在线帮助，了解有关 web 模块的详细信息。

创建 CDShopCart Web 模块

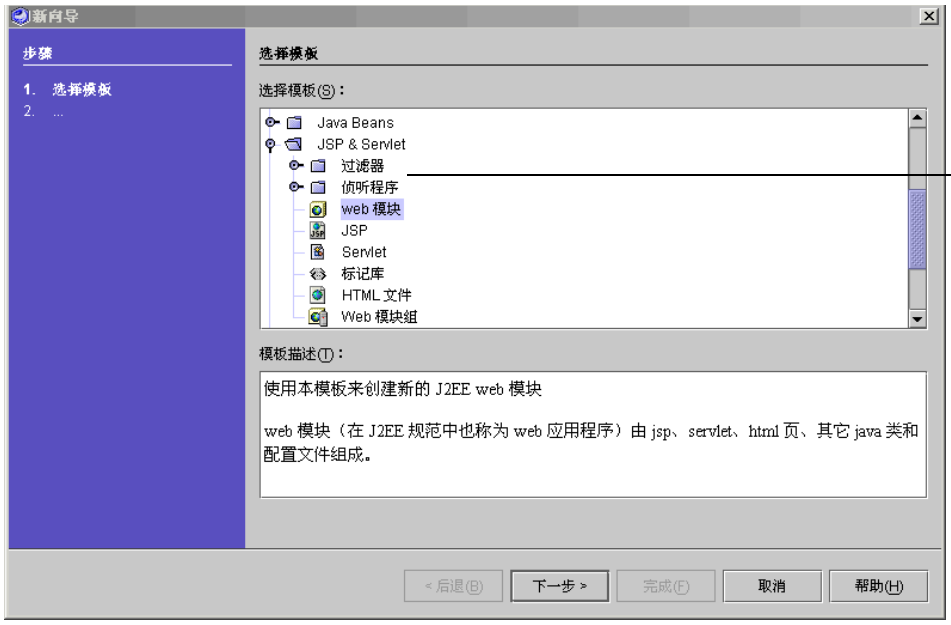
本节中，您将创建 CDShopCart 应用程序的 web 模块。您将使用 Sun ONE Studio 5 web 模块功能从头开始构建此 web 模块目录，尽管您也可以将现有目录转换为 web 模块。

要创建 CDShopCart web 模块：

1. **在 [资源管理器] 窗口的 [文件系统] 窗格中，选择缺省 （或任意）文件系统并选择 [文件] → [新建]，显示 [新建] 向导。**

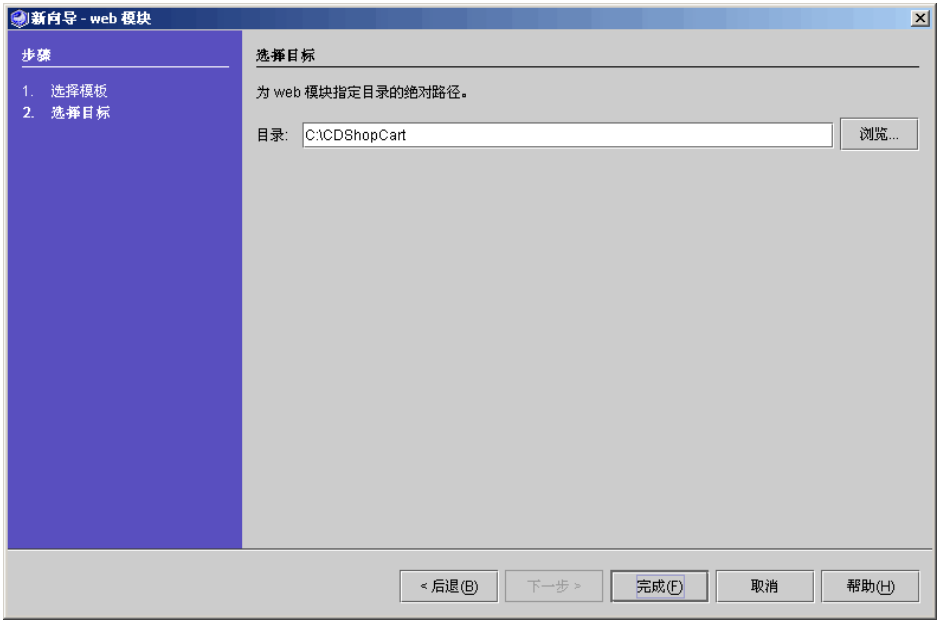
   您可以使用此向导从提供的模板创建多种不同的对象类型。

2. **打开 [JSPs & Servlets] 节点并选择 [Web 模块]。**



3. 单击 [下一步]，显示用来指定 **web** 模块目录名称的页。

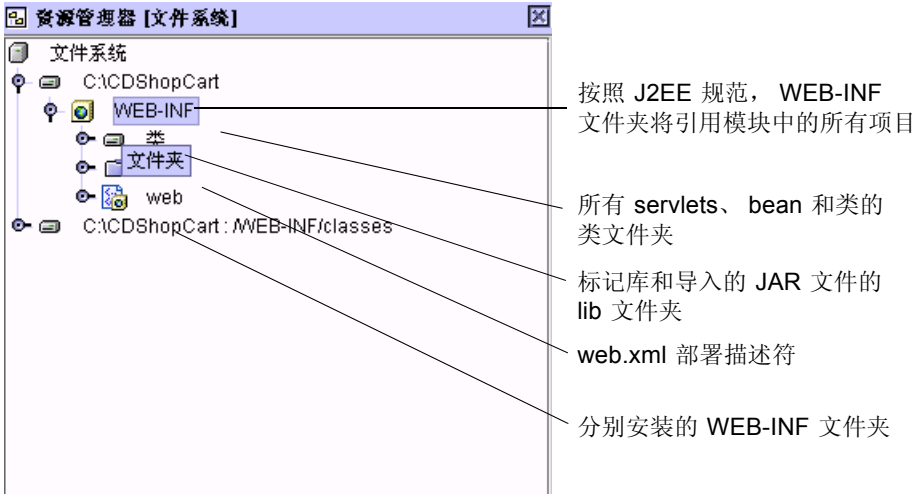
4. 在 [目录] 字段中，删除缺省目录文件规范和类型 *your-directory*\CDShopCart。



5. **单击 [完成]。**

   新的 CDShopCart web 模块将在 [资源管理器] 中创建。出现一个对话框，表明在 [缺省项目] 窗口中安装了一个 web 模块的备用视图。单击 [确定] 清除此消息。

6. **打开 web 模块中的节点，查看自动创建的内容。**



现在将准备创建应用程序的第一个组件 ProductList JSP 页。

设置 Web 应用程序的上下文根

为了部署 CDShopCart web 应用程序， web 服务器必须知道其上下文根。将其设置为 WEB-INF 文件夹的属性，如下所示：

1. **显示 WEB-INF 文件夹的属性。**

   属性窗口位于 [资源管理器] 下方。这是一个静态属性窗口，显示所有选定节点的属性。选择 WEB-INF 文件以显示其属性。或者，右键单击 WEB-INF 文件夹并选择 [属性]。一个动态属性窗口打开，其中仅显示 WEB-INF 文件夹的属性，而与所选内容无关。

2. **单击 [上下文根] 属性的值字段。**

3. **键入 `/CDShopCart` 并按 Return （或 Enter）键。**

使用 JSP 标记获取并显示数据库数据

在本节中，将创建一个名为 ProductList 的 JSP 页，以获取并显示 CD 产品数据。为了连接到数据库、检索表数据并格式化数据以便显示它们，您将使用 [JSP 标准标记库 (JSTL)] 标记。 JSTL 来自 Jakarta Project，属于 Apache Foundation 的一个项目。 JSTL 被嵌入在 Sun ONE Studio 5 中。

有关 JSTL 的完整信息 （包括描述、示例和指南），参见 Jakarta Project web 站点，网址为 `http://jakarta.apache.org/taglibs/index.html`。

什么是 JSP 标记？

JSP 文件的正文可包含两种类型的代码：固定模板数据和元素。

- 固定模板数据 - JSP 容器尚不清楚的代码类型，此数据经 HTTP 响应后不会改变。

  XML 和 HTML 代码都是固定模板数据的示例。您将使用 CDShopCart 应用程序中的 HTML 代码来创建标题、题目、表和按钮。

- 元素类型 - 包括三种不同的类型：
  - 指令 - 用来声明有关 JSP 页的全局信息，例如要导入哪些包、 JSP 页是否必须加入一个会话等。
  - 脚本元素 - 允许将 Java 代码嵌入到 JSP 文件中。
  - JSP 标记 - XML 样式的元素，提供了一种不必编写 Java 代码即可使用 Java 对象的途径。

    与每个标记相关联的 Java 类都实现标记的功能。

标准标记和定制标记

标准标记，在 JSP 规范文档中定义，在任何 JSP 容器中都可用。定制标记是在 XML 文档中定义的标记，被称为*标记库*。通过在指令元素中进行声明，可以使定制标记库在 JSP 页中可用。

JSTL 标记

“JSP 标准标记库”是由 Jakarta Project 生成的定制标记库。在 Sun ONE Studio 5 IDE 中嵌入的是 JAR 文件，它们包含 JSTL 标记并支持类和接口。这些文件包括：

- `standard.jar` - 包含多个 JSTL 标记库，包括核心和 SQL 标记
- `jstl.jar` - 包含 `standard.jar` 中 CDShopCart 所使用的 JSTL 库的支持接口和类

在这些 JAR 文件中将被使用的两个标记库描述符 (TLD) 文件是 `sql.tld`（用于数据库操作）和 `core.tld`（用于其它所有情况）。

使用 JSTL 标记

要使用 JSP 标记，您需要声明标记库，然后遵循特定标记的规定语法。

有关 JSTL 语法惯例的完整信息，参见 JSTL 文档，它可在 Jakarta Project web 站点得到，网址为 `http://jakarta.apache.org/taglibs/index.html`。

使用 `taglib` 指令

要在 JSP 页中使用标记，必须首先用 taglib 指令声明标记库。

tablib 指令必须声明该页使用给定 URI（统一资源标识符）的标记库并指定被用来调用库中操作的标记前缀。 URI 和 JSTL 标记前缀使用 Apache JSTL 惯例来定义。

指令的一般语法是：

```
<%@ taglib prefix="prefix" uri="http://java.sun.com/jstl/taglibname
```

例如，要声明 `core` taglib：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core
```

使用标记语法

JSP 标记以 XML 语法为基础并具有两种形式之一：

- 开始标记（元素名）加上可能的属性 / 属性值对、一个可选主体和一个匹配的结束标记
- 具有可能属性的空标记

将要在 CDShopCart 中使用的一个开始标记示例是查询标记，它用于从数据源获取数据。查询标记使用下列语法：

```
<sql:query var="stored_query" dataSource="${dataSource}" >
       body
</sql:query>
```

您将要使用的一个空标记示例是 sql 库 setDataSource 标记，它用于创建一个到数据库的 JDBC 连接。 setDataSource 标记使用下列语法：

```
<sql:setDataSource var="dataSource"
       url="driver_url"
       driver="driver_string"
       user="user_id" password="pwd" />
```

JSTL 标记的惯例是对于输出有关标记信息的任何标记属性都使用 “var”。注意，选择 “var” 是为了强调这是 JSP 变量而不是脚本变量这一事实，脚本变量使用惯例 “id” 来指明变量。

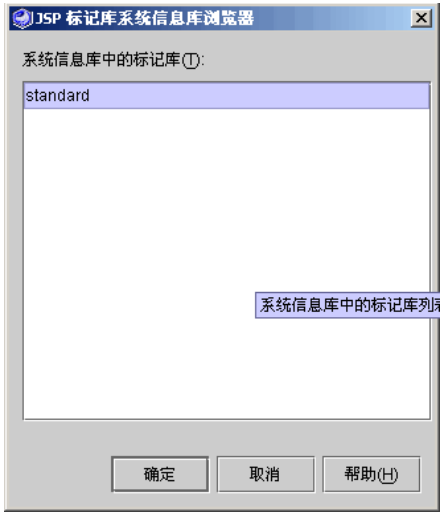
向 Web 模块增加 JSTL 标记库

把 JSTL 标记库 JAR 文件 `standard.jar` 和支持类及接口的 JAR 文件 `jstl.jar` 导入到 CDShopCart web 模块，因为您将使用由这些文件实现的操作。

要将 JSTL 标记库导入到 web 模块：

1. **在 [资源管理器] 中，右键单击 CDShopCart web 模块并选择 [增加 JSP 标记库] → [在标记库系统信息库中查找]。**

   [JSP 标记库系统信息库浏览器] 出现。

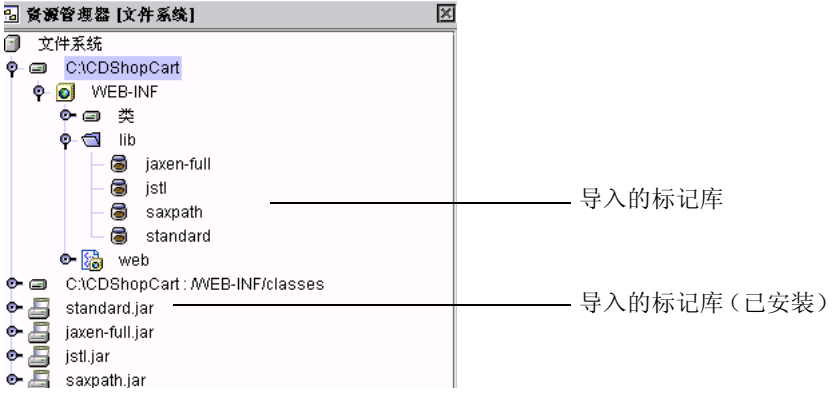


2. **确保选择了 `standard` 文件并单击 [确定]。**

   在 [资源管理器] 中，展开 `[WEB-INF]` 节点下的 `lib` 节点。

3. **检查以确保文件都位于 lib 节点下。**

   `jstl.jar` 和 `standard.jar` 文件，它们包含有本指南中使用的标记，显示在 `lib` 节点下。同时还显示了两个附加的 jar 文件。它们包含 XML 标记所需的 API，而这些标记不会再被用到。全部四个文件也都被分别安装。 [资源管理器] 如下所示：



提示 – 右键单击顶级 [文件系统] 节点并选择 [定制]。打开的窗口显示出 Sun ONE Studio 5 类路径中现在安装的所有文件。您应能够看到列表中包含两个 JAR 文件。

创建 [CD Catalog List] 页

创建从第 1 章所安装的数据库中检索数据的机制并在表中为用户显示这些数据。此机制包括：

- JSP 页 （产品列表）容纳全部代码
- 在代码中引用的标记库的标记库声明
- 连接到数据库的 setDataSource 标记
- 获取 CD 数据的查询标记
- 显示 CD 数据的迭代标记
- 每个 CD 行的 [Add] 按钮

创建的页看起来如图 3-1 所示。

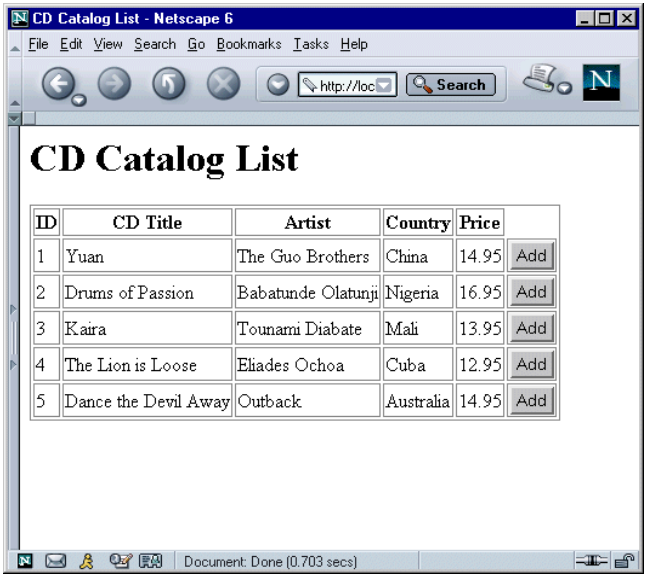


图 3-1 [CD Catalog List] 页

创建 ProductList JSP 页

创建使用标记从数据库检索 CD 数据并在表格中显示该数据的页面。此页的标题是 CD Catalog List，其产生机制是 ProductList JSP 页。

1. **在 [资源管理器] 中，右键单击 CDShopCart web 模块并从上下文菜单中选择 [新建] → [所有模板]。**

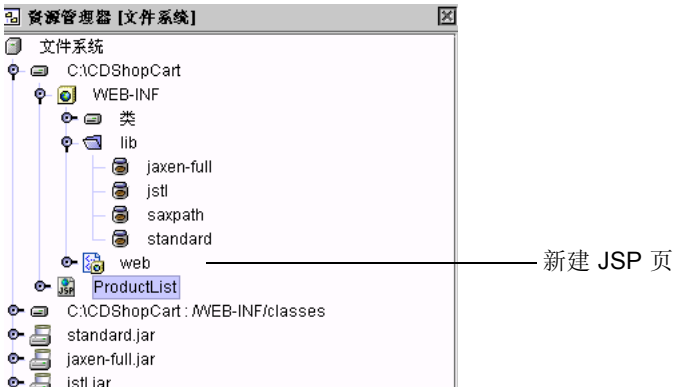
   [新建] 向导出现。

2. **展开 JSPs 和 Servlets 节点，选择它下面的 JSP 节点，单击 [下一步]。**

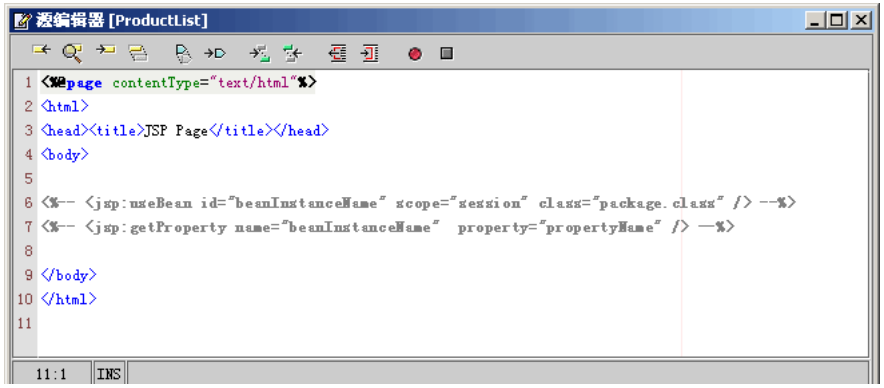
   [新建] 向导的 [新建对象页] 出现。

3. **在 [名称] 字段中键入 `ProductList` 并单击 [完成]。**

   ProductList JSP 页在 web 模块中显示出来。



JSP 页的骨架显示在 [源编辑器] 中。



声明标记库

必须首先声明标记库，如第 48 页的 “使用 `taglib` 指令”所述。将指令放到 JSP 页的主体上方，紧靠页标题的下面。下列过程显示了如何更改页标题并为两个标记库增加指令。

1. **在 [ProductList] 页的主体中，将文档标题更改为 CD Catalog List：**

```
<head><title>CD Catalog List</title></head>
```

2. **在此行下面，增加指令以导入 core 和 SQL 操作标记库，如下所示：**

```
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
```

使用 setDataSource 标记连接到数据库

您将要使用的第一个标记是 sql 库 setDataSource 标记，它将创建到数据库的 JDBC 连接。首先增加一个页标题 （用 <H1> 标记），然后在 <body> HTML 标记下增加 setDataSource 标记以连接到数据库。

1. **在 [ProductList] 页的 <body> 标记下创建一个页标题：**

```
<body>
<h1> CD Catalog List</h1>
```

2. **在标题下面，创建 JDBC 连接。**
   - 下列各项用于 PointBase 驱动程序：

```
<sql:setDataSource var="productDS"
url="jdbc:pointbase:server://localhost/cdshopcart"
driver="com.pointbase.jdbc.jdbcUniversalDriver"
user="PBPUBLIC" password="PBPUBLIC" />
```

   - 如果您正在使用 Oracle 数据库 （使用瘦驱动程序），则使用：

```
<sql:setDataSource var="productDS"
url="jdbc:oracle:thin:@hostname:port#:SID"
driver="oracle.jdbc.driver.OracleDriver"
user="userid" password="password" />
```

     缺省 Oracle 端口号是 1521。

   - 如果您正在使用有 Weblogic 驱动程序的 Microsoft SQLServer 数据库，则使用：

```
<sql:setDataSource var="productDS"
url="jdbc:weblogic:mssqlserver4:database@hostname:port#"
driver="weblogic.jdbc.mssqlserver4.Driver"
user="userid" password="password" />
```

     缺省 SQLServer 端口号是 1433。

使用查询标记获取 CD 数据

现在使用 sql 标记 query 查询数据库并获取包含数据行的单个结果集，然后将其存储在 productQuery 结果集中。然后将此 JSP 变量传递到 iterator 标记以显示结果。查询标记支持标准 SQL 语句 SELECT。

将查询标记紧靠驱动程序标记后放置。

要查询所有 CD 数据的数据库：

- **在 [ProductList] 页面中，紧靠驱动程序标记后面，创建一个查询，从数据库中选择所有 CD 数据 （使用上面创建的 productDS 数据源）：**

```
<sql:query var="productQuery" dataSource="${productDS}" >
SELECT * FROM CD
</sql:query>
```

使用迭代标记显示数据

您需要创建一个表并用数据填充表单元。首先，用 HTML 标记创建一个表，然后使用 forEach 标记去迭代获取的数据。使用 `out` 标记获取每行的字段数据。

forEach 的语法是：

```
<c:forEach var="$Current_collection_item" items="${Collection}" >
```

项目变量包含迭代目标的当前集合，而 var 包含该集合的当前项目。如果集合是一个结果集，则当前项目就是位于当前行的结果集对象。例如，要从名为 myResultSet 的结果中迭代一行：

```
<c:forEach var="row" items="${myResultSet.rows}" >
<TR>
<TD><c:out value="${row.col1}"/></TD>>
<TD><c:out value="${row.col2}" /></TD>
<TD><c:out value="${row.col3}" /></TD>
</TR>
</c:forEach>
```

要创建一个表并用数据填充其单元：

1. **启动一个 CD 数据表：**

```
<TABLE border=1>
<TR>
<TH>ID</TH>
<TH>CD Title</TH>
<TH>Artist</TH>
<TH>Country</TH>
<TH>Price</TH>
</TR>
```

2. **下一步，使用 forEach 和 out 标记填充表：**

```
<c:forEach var="row" items="${productQuery.rows}" >
<TR>
<TD><c:out value="${row.ID}"/></TD>
<TD><c:out value="${row.CDTITLE}"/></TD>
<TD><c:out value="${row.ARTIST}"/></TD>
<TD><c:out value="${row.COUNTRY}"/></TD>
<TD><c:out value="${row.PRICE}"/></TD>
```

在下一节中，将要向此代码添加更多内容。

为各个 CD 行创建 [Add] 按钮

CD 表的每行都包含一个 CD 的数据。要购买 CD，用户可单击 CD 行上的 [Add] 按钮。创建一个 HTML 窗体来定义用户输入区 （单击按钮），在此区域内嵌入被传递到购物车 JSP 页的信息。紧接着上述代码添加下列代码：

1. **在单元格内为表格创建一个窗体：**

```
<TD>
<form method=get action="ShopCart.jsp">
```

2. 指定产品 **ID**、标题和价格等的嵌入信息：

```
<input type=hidden name=cdId value="<c:out value="${row.ID}"/>">
<input type=hidden name=cdTitle value="<c:out value=
"${row.CDTITLE}"/>">
<input type=hidden name=cdPrice value="<c:out value=
"${row.PRICE}"/>">
```

3. 紧靠上述代码在其下方指定 **[Add]** 按钮：

```
<input type=submit name=operation value=Add>
```

4. 结束窗体、单元格和行：

```
</form>
</TD>
</TR>
```

5. 结束迭代和表：

```
</c:forEach>
</TABLE>
```

6. 选择 **[** 文件 **]** → **[** 保存 **]**，保存所做的工作。

提示 – 要查看此页已完成的源代码，参见第 82 页的 “ProductList.jsp 源”。

测试 ProductList JSP 页

通过编译 ProductList JSP 页、部署 web 应用程序，然后执行 [ProductList] 页来测试您的工作。 IDE 将自动启动缺省浏览器并显示该页。

要测试 [ProductList] 页：

1. **如果尚未启动，则通过选择 [工具] → [PointBase 网络服务器] → [启动服务器] 来启动 PointBase 网络服务器。**

   PointBase 服务器窗口显示出来。将其最小化。

2. **确保您的 Sun ONE 应用程序服务器 7 管理服务器正在运行，并且该应用程序服务器是缺省的 web 服务器。**

   有关信息，参见第 27 页的 “确认 Sun ONE 应用程序服务器 7 作为缺省服务器”。

3. **确保 web 模块的 [上下文根] 属性被设置为** `/CDShopCart`。

   参见第 46 页的 “设置 Web 应用程序的上下文根”。

4. **在 [资源管理器] 的 [文件系统] 窗格中，右键单击 CDShopCart web 模块，从上下文菜单中选择 [全部生成]。**

   观察输出窗口的消息区以了解状态消息。如果一切正常，您将会看到 [已完成]。如果输出窗口显示错误，则修复所有问题并重新执行此步骤，直到出现 [已完成] 消息为止。要了解源代码，参见第 82 页的 “ProductList.jsp 源”。

5. **右键单击 CDShopCart web 模块并选择 [部署]。**

   应用程序部署完毕后，进度表窗口被关闭。

6. **选择 ProductList JSP 页并单击工具栏上的 [执行] 按钮 () 来执行它。**

   或者，选择 [生成] → [执行]，或右键单击 [ProductList]，从上下文菜单中选择 [执行]。

   如果 Sun ONE 应用程序服务器 7 尚未运行， IDE 将启动它 （在 Microsoft Windows 系统中，您会看到一个包含登录消息的命令窗口，您可以将其最小化）。

   如果 Servlet 正在运行， IDE 将打开浏览器。几秒钟后， [CD Catalog List] 页出现，如图 3-1 所示。

   祝贺您！ 您已经成功创建了一个 JSP 页，它使用 JSTL 标记打开到数据库的连接并检索和显示其中的数据。现在创建购物车页。

创建 [Shopping Cart] 页和支持元素

现在创建用于显示项目（这些项目选自 [CD Catalog List] 页并要购买它们）的机制。此机制包括：

- 一个 bean (CartLineItem)，它包含所选 CD 行的属性，该 CD 行将作为参数从 [ProductList] 页传递。
- 一个 beaninfo 组件 (CartLineItemBeanInfo)，指定 id 和 price 是 CartLineItem 的属性，尽管有重载的 setter 方法
- 另一个 bean (Cart)，包含 CartLineItem 对象。
- ShopCart JSP 页，接收 Cart 对象并将其作为表中的一行显示出来
- 用于 [Shopping Cart] 页中所显示每一项目的一个 [Delete] 按钮
- [Cancel Order]、[Resume Shopping] 和 [Place Order] 按钮及其实现

在选择了一些项目后，您创建的 [Shopping Cart] 页看起来类似于图 3-2。

图 3-2 [Shopping Cart] 页

创建 CartLineItem Bean

创建一个行项目 bean，其对象可包含从 [CD Catalog List] (ProductList) 页传递到 [Shopping Cart] 页的参数。为此请在 bean 及其存取方法上创建三个属性。

注意 – J2SE 1.4 要求所有类都包含在包中，而本指南与 J2SE 1.4 相符合。WEB-INF 目录的类子目录不属于包。因此，您必须在类子目录下创建一个包来容纳类 (bean)。在本指南中，它被称为 ShopCart 包。

---

1. **展开 CDShopCart web 模块的 WEB-INF 节点，右键单击 [类] 节点，选择 [新建] → [Java 包]。**
2. **键入 `ShopCart` 作为包的名称并单击 [完成]。**

   新的 ShopCart 包将在 [资源管理器] 中显示出来。
3. **右键单击 ShopCart 包，选择 [新建] → [所有模板]。**

   [新建] 向导出现。
4. **展开 [Java Bean] 节点，选择 `Java Bean`，然后单击 [下一步]。**
5. **在 [名称] 字段中，键入 `CartLineItem`，单击 [完成]。**

   新的 CartLineItem bean 将显示在 [资源管理器] 中，其代码显示在 [源编辑器] 中。bean 附近的红色标记 ( ) 是 “需要编译”指示符。先不必理会，随后将对其进行编译。
6. **展开 bean 及其类，显示其内容。**
7. **右键单击 [Bean 模式] 节点，选择 [增加] → [属性]。**
8. **在 [新属性模式] 对话框中，定义 cdtitle 属性。**

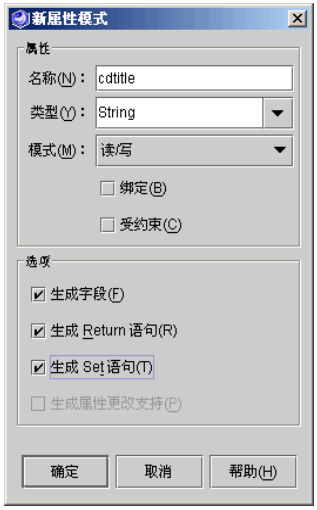
   a. **在 [名称] 字段中键入 `cdtitle`。**

   b. **选择 `String` 作为 [类型]。**

   c. **启用下列选项：**

   - 生成字段
   - 生成 return 语句
   - 生成 set 语句

对话框看起来应类似于：



9. 单击 **[** 确定 **]** 接受该信息并关闭对话框。

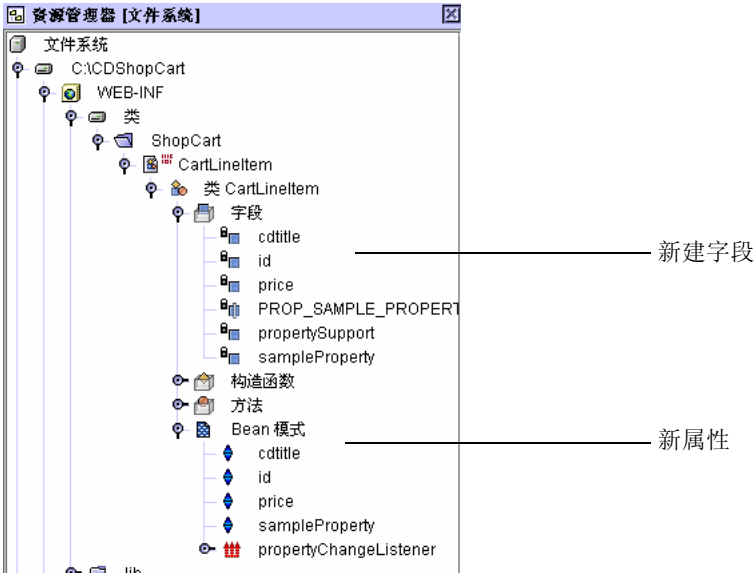
10. 类似地，如下所示创建 **id** 属性：

属性	值
名称	id
类型	int
启用的选项	生成字段 生成 return 语句 生成 set 语句

11. 利用同样的方法创建 `price`，如下所示：

属性	值
名称	price
类型	double
启用的选项	生成字段 生成 return 语句 生成 set 语句

12. 展开 **CartLineItem bean** 类的 **[** 字段 **]** 节点以查看创建的新字段。



13. 双击刚创建的新字段 （例如， **cdtitle**），在 **[** 源编辑器 **]** 中查看其代码。
14. 展开 **[** 方法 **]** 节点，查看每一字段新的 **get** 和 **set** 方法。

将 id 属性和 price 属性转换为字符串

从 ProductList JSP 页传递的参数都作为字符串传递。然而，由于 id 属性和 price 属性都不是字符串，因此必须对其进行转换。有效的转换方法是重载属性的 setter 方法并增加适当的代码。

要重载 setId 方法和 setPrice 方法：

1. 右键单击 **CartLineItem bean** 的 **[** 方法 **]** 节点，选择 **[** 增加方法 **]**。
2. 在出现的 **[** 编辑新方法 **]** 对话框中，定义 **setId** 方法。
   a. 在 **[** 名称 **]** 字段中键入 `setId`。
   b. 选择 `void` 作为 **[** 返回类型 **]**。
3. 在 **[** 方法参数 **]** 框中，单击 **[** 增加 **]** 按钮，以显示 **[** 输入方法参数 **]** 对话框。
4. 定义 **pId** 参数。
   a. 选择 `java.lang.String` 作为 **[** 类型 **]**。
   b. 键入 **`pId`** 作为 **[** 名称 **]**。

5. 单击 [确定]。

[新建方法] 对话框如下所示：

6. 单击 [确定]，创建方法并关闭对话框。

7. 双击 [资源管理器] 中的新方法，在 [源编辑器] 中显示其代码。

8. 将代码增加到新方法中，如下所示：

```
public void setId(java.lang.String pId) {
        int val = Integer.parseInt(pId);
        this.setId(val);
}
```

提示 – 如果向 [源编辑器] 输入代码时是通过复制和粘贴的方式，您可将光标放到 [源编辑器] 中并按下 Ctrl-Shift-F 自动重新格式化代码。

现在以类似方法创建 `setPrice` 方法。

9. **如下所示定义 setPrice 方法：**

属性	值
名称	setPrice
返回类型	void
参数类型	java.lang.String
参数名称	pPrice

10. **将下列代码添加到新方法中：**

```
public void setPrice(java.lang.String pPrice) {
        double val = Double.parseDouble(pPrice);
        this.setPrice(val);
}
```

11. **选择 CartLineItem bean ()，而不是类 ()，选择 [生成] → [编译]（或按 F9）**

如果 bean 的编译没有错误，则红色的“需要编译”标记将从 bean 节点删除，您即可创建 `Cart` bean。否则，检查键入的代码并重新编译。

提示 – 此 bean 的已完成源代码可在第 84 页的 “CartLineItem Bean 源”中找到。

创建 CartLineItemBeanInfo 组件

bean 属性是一个专用属性，它可以被一对公共存取 （get 和 set）方法访问，这是一个内置于 Introspector 类中的标准 Java 约定。由于您刚重载了 id 和 price 属性的 set 方法，因此 Introspector 将不会再把它们当做 bean 属性。

但是，您确实又希望将这些字段视为属性，因为仅当它们遵循 Java 约定时， JSTL 表达式语言才会使 bean 属性可用。例如，在下列代码中， “.” 操作符后面的项必须是属性：

```
<TD><c:out value="${row.id}"/></TD>
<TD><c:out value="${row.cdtitle}"/></TD>
```

您可利用 BeanInfo 组件避开这一难题，直接使用此组件指定 bean 的属性。为 CartLineItem bean 创建一个 BeanInfo 组件，然后对其进行编码，将 id 和 price 指定为属性。

要将 id 和 price 指定为 CartLineItem bean 的属性：

1. **右键单击 ShopCart 包，选择 [新建] → [所有模板]。**

   [新建] 向导出现。

2. **展开 [Java Bean] 节点，选择 [BeanInfo w/o 图标]，单击 [下一步]。**

3. **将 BeanInfo 命名为 `CartLineItemBeanInfo` 并单击 [完成]。**

   新的 CartLineItemBeanInfo 节点将在 [资源管理器] 中显示出来。

4. **展开 [BeanInfo] 节点及其 [方法] 节点。**

5. **双击 [getPdescriptor] 方法。**

   CartLineItemBeanInfo bean 的代码显示在 [源编辑器] 中 [getPdescriptor] 方法的定义处。

6. **将下列代码添加到 getPdescriptor 方法的正文中：**

```
if (properties == null) {
       try {
           PropertyDescriptor[] props = {
               new PropertyDescriptor("cdtitle",
CartLineItem.class),
               new PropertyDescriptor("id", CartLineItem.class),
               new PropertyDescriptor("price",
CartLineItem.class)};
               properties = props;
       } catch (IntrospectionException ex) {
           return null;
       }
}
```

提示 – 记住要按下 Control-Shift F 重新格式化粘贴或键入到 [源编辑器] 中的代码。

7. **选择 CartLineItemBeanInfo 节点并按 F9 编译 bean**。

   BeanInfo bean 编译时应当不会出现差错。

创建 Cart Bean

ShopCart JSP 页实例化或查找（如果它已存在）Cart 对象以容纳 CD 行项目对象，这些对象在用户单击 [添加] 按钮时被 ProductList JSP 页传递到 Cart 对象。cart 对象以 Cart bean 为基础。

要创建 Cart bean：

1. **右键单击 ShopCart 包，选择 [新建] → [Java Bean]**。

   注意此项目将出现在上下文菜单中。创建此快捷方式是为了便于您从 [新建] 菜单中选择项目。

2. **将 bean 命名为** `Cart` **并单击 [完成]**。

3. **右键单击 Cart bean 的 [Bean 模式] 节点，选择 [增加] → [属性]**。

4. **利用 [新属性模式] 对话框创建一个新的 lineItems bean 模式，如下所示：**

属性	值
名称	lineItems
类型	java.util.Vector
选定的选项	生成字段 生成 return 语句 生成 set 语句

   java.util.Vector 类型不在下拉列表中，因此需要键入它。

   现在必须把对 lineItems 字段的访问从专用改为公共。

5. **展开 Cart 类的 [字段] 节点并选择 lineItems 字段**。

6. **显示 lineItems 属性并单击 [修饰符] 值字段**。

注意 – 如果 [属性] 窗口显示在 [资源管理器] 窗口下，则只需选择一个项目即可在窗口中显示其属性。或者，您可以右键单击 lineItems 并从上下文菜单中选择 [属性]。

7. **单击省略号 (...) 按钮以显示 [修饰符] 对话框。**
8. **从 [访问] 列表中选择 [公共]。**

9. **单击 [确定] 接受更改。**

   现在添加实例化行项目对象的代码、返回选定项目的元素数量的方法以及从购物车中删除行项目的方法。

10. **打开 Cart bean 的 [构造函数] 节点并双击 Cart() 构造函数。**

    此操作将在 [源编辑器] 中显示 Cart() 构造函数。

11. **将下列代码 （粗体）添加到 Cart bean 的构造函数中以实例化新的 lineItems 对象，如下所示：**

```
public Cart() {
       propertySupport = new PropertyChangeSupport ( this );
       lineItems = new java.util.Vector();
}
```

12. 右键单击 **[Cart Methods]** 节点，选择 **[** 增加方法 **]**，并如下所示定义 **findLineItem** 方法：

属性	值
名称	findLineItem
返回类型	int
参数类型	int
参数名称	pID

13. 在 **[** 源编辑器 **]** 中，将下列代码 （粗体）添加到 **findLineItem** 方法中：

```
public int findLineItem(int pID) {
       System.out.println("Entering Cart.findLineItem()");
       //Return the element number of the item in the cartItems
       //as specified by the passed ID.
       int cartSize = (lineItems == null) ? 0 : lineItems.size();
       int i;
       for (i = 0; i < cartSize; i++ ) {
           if  ( pID ==
               ((CartLineItem)lineItems.elementAt(i)).getId() )
           break;
       }
       if (i >= cartSize) {
           System.out.println("Couldn't find line item for ID: " +
               pID);
           return -1;
       }
       else
           return i;
}
```

14. 重复步骤 **12**，如下所示定义 **removeLineItem** 方法：

属性	值
名称	removeLineItem
返回类型	void
参数类型	int
参数名称	pID

15. **在 [源编辑器] 中将下列代码添加到 removeLineItem 方法中：**

```
public void removeLineItem(int pID) {
       System.out.println("Entering cart.removeLineItem()");
       int i = findLineItem(pID);
       if (i != -1) lineItems.remove(i);
       System.out.println("Leaving cart.removeLineItem()");
}
```

16. **选择 Cart bean ()，而不是类 ()，并按 F9 来编译 Cart bean。**

bean 编译时应当不会出现差错。现在创建 ShopCart JSP 页。

提示 – 此 bean 的已完成源代码可以在第 91 页的 “Cart Bean 源”中找到。

创建 [Shopping Cart] 页

现在创建该页，它将接收从 [CD Catalog List] 页传递来的参数并将其中某些参数 （id、title 和 price）显示为表中的行。此页还提供从表中删除项目、返回到目录表页及执行订购等机制。此页的标题是 [Shopping Cart]，其产生机制是 ShopCart JSP 页。

1. **右键单击 CDShopCart web 模块并选择 [新建] → [JSP]，创建 JSP 页。**
2. **将 JSP 页命名为 `ShopCart` 并单击 [完成]。**

ShopCart JSP 将显示在 [资源管理器] 和 [源编辑器] 中。

添加代码以从 Shopping Cart 表中添加或删除项目

现在添加创建购物车项目表的代码。要实例化 Cart 对象和 CartLineItem 对象，可使用指令导入 Cart bean、java.util 库 （CartLineItem 的类型是 Vector，包括在此库中）以及 CartLineItem。使用与在 ProductList JSP 页中用过的同样 core 标记，因此也添加一个指令来导入这些标记。

使用 scriptlet 代码创建购物车。然后添加代码，此代码可添加行项目 （此项目是用从 ProductList JSP 页传递来的参数创建的）。添加更多的代码，此代码在用户按下 [删除] 按钮时删除行项目。添加向 JSP 页转发操作的代码，如果表是空表，您需要在后续操作中创建该页 （EmptyCart JSP 页）。

对于 ProductList JSP 页，使用迭代标记来组织表数据。然后使用窗体创建表并为每一行添加一个 [Delete] 按钮。

1. 添加一个 **Page** 指令来导入 **java.util** 库和 **ShopCart** 包：

```
<%@page contentType="text/html" %>
<%@page import="java.util.*, ShopCart.*" %>
```

2. 将页标题更改为 **[Shopping Cart]** 并添加指令来导入 **core.jar** 库：

```
<head><title>Shopping Cart</title></head>
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

3. 在 **<body>** 标记下方，创建该页的 **[Shopping Cart]** 标题：

```
<body>
<h1>Shopping Cart</h1>
```

4. 在标题下，利用 **usebean** 标记来告诉 **JSP** 页使用 **Cart bean**：

```
<jsp:useBean id="myCart" scope="session" class="ShopCart.Cart" />
```

此代码实例化一个 Cart 对象并将其放到会话中。

现在指定当当前会话的操作是 [Add] 时所出现的结果。在用户单击 [ProductList] 页上的 [Add] 按钮时会发生这种情况。添加获取 cdId、 cdTitle 和 cdPrice 对象的代码并将其添加到 myCart 对象中。

5. 将获取当前操作及定义用户单击 **[Add]** 时将发生的情况作为代码的开始部分：

```
<%
String myOperation = request.getParameter("operation");
session.setAttribute("myLineItems", myCart.getLineItems());

if (myOperation.equals("Add")) {
CartLineItem lineItem = new CartLineItem();
lineItem.setId(request.getParameter("cdId"));
lineItem.setCdtitle(request.getParameter("cdTitle"));
lineItem.setPrice(request.getParameter("cdPrice"));
myCart.lineItems.addElement(lineItem);
}
```

接下来，指定会话操作是 [Delete] 时将发生的情况。这在用户单击 [Shopping Cart] 页上的 [Delete] 按钮时发生。

6. 使用下列代码指定当用户单击 **[Delete]** 时发生的情况：

```
if (myOperation.equals("Delete")) {
String s = request.getParameter("cdId");
System.out.println(s);
int idVal = Integer.parseInt(s);
myCart.removeLineItem(idVal);
}
```

最后，指定当 [Delete] 操作删除了 Cart CD 表中最后一行时将发生的情况。使用 JSP 将标记转发到 EmptyCart JSP 页 （很快就要创建该页）。从步骤 5 开始编写的脚本到此代码结束。必须要中断一下以便转发标记，然后继续进行直到最后完成 scriptlet。

7. 使用下列代码：

```
//If the last item is removed from the cart...
if (((Vector)session.getAttribute("myLineItems")).size() == 0) {
//End scriptlet temporarily so that you can use the JSP
// 攁orwardî tag to forward to the EmptyCart page.
%>
<jsp:forward page="EmptyCart.jsp" />
//Resume the scriptlet.
<%
}
%>
```

使用迭代标记填充 Cart 表

接下来使用 forEach 和 out 标记迭代传递的数据并获取每行的字段数据，就象在第 54 页的 “使用迭代标记显示数据”中所做的那样。

1. 通过创建表标题来开始一个购买候选人数据表：

```
<TABLE border=1>
<TR>
<TH>ID</TH>
<TH>CD Title</TH>
<TH>Price</TH>
</TR>
```

2. 使用 **forEach** 和 **out** 标记填充表：

```
<c:forEach var="item" items="${myLineItems}">
<TR>
<TD><c:out value="${item.id}"/></TD>
<TD><c:out value="${item.cdtitle}"/></TD>
<TD><c:out value="${item.price}"/></TD>
```

前面代码中的 out 标记从当前结果行中的命名字段里检索值。

3. 为每行创建一个 **[Delete]** 按钮，如第 72 页的 “向页添加按钮”所述：

```
<TD>
<form method=get action="ShopCart.jsp">
<input type=hidden name=cdId value="<c:out value="${item.id}"/>">
<input type=hidden name=cdTitle value="<c:out value=
"${item.cdtitle}"/>">
<input type=hidden name=cdPrice value="<c:out value=
"${item.price}"/>">
<input type=submit name=operation value="Delete">
</form>
</TD>
</TR>
</c:forEach>
</TABLE>
```

向页添加按钮

最后，在页的底部添加 [Resume Shopping]、 [Place Order] 和 [Cancel Order] 按钮。

- **将下列代码添加到 ShopCart JSP 页：**

```
<p>
<!--Create the three buttons.-->
<form method=get action="ProductList.jsp">
<input type=submit value="Resume Shopping">
</form>
<form method=get action="PlaceOrder.jsp">
<input type=submit value="Place Order">
</form>
<form method=get action="CancelOrder.jsp">
<input type=submit value="Cancel Order">
</form>
</p>
<!End the page.>
</body>
</html>
```

提示 – 要查看此页已完成的源代码，参见第 94 页的 “ShopCart.jsp 源”。

测试 [Shopping Cart] 页

您不需要直接测试 [ShopCart] 页。测试 [ProductList] 页并通过 [Add] 按钮导航至 [Shopping Cart] 页。首先，您必须重新部署 CDShopCart web 应用程序，因为它包含以前部署的版本中没有的组件。

注意 – 确保 Sun ONE 应用程序服务器 7 是 IDE 的缺省 web 服务器。有关信息，参见第 27 页的 “确认 Sun ONE 应用程序服务器 7 作为缺省服务器”。

---

1. **如果尚未启动，则通过选择 [工具] → [PointBase 网络服务器] → [启动服务器] 来启动 PointBase 网络服务器。**
2. **右键单击 CDShopCart web 模块并选择 [全部生成]。**

3. **再次右键单击 CDShopCart web 模块并选择 [部署]。**

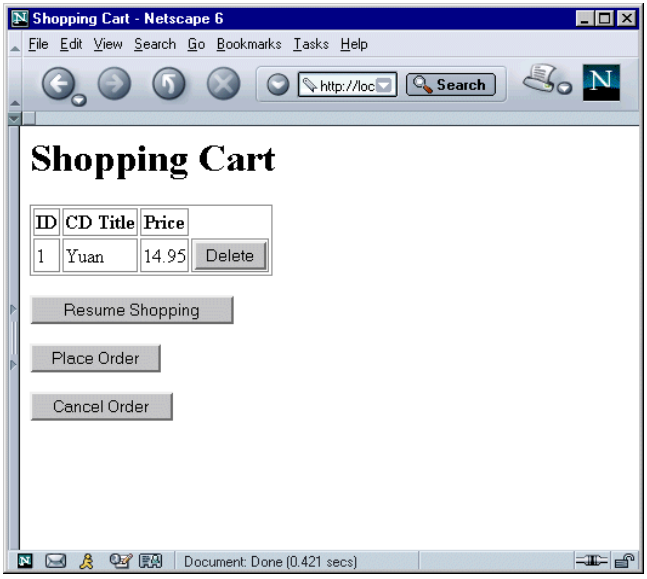
   IDE 将重新部署应用程序。一个进度窗口出现，显示部署过程的进度，部署完成后该窗口消失。

4. **右键单击 ProductList JSP 页并选择 [执行]。**

   IDE 启动缺省浏览器并显示 [CD Catalog List] 页。

5. **单击其中一个 [Add] 按钮导航至 [Shopping Cart] 页。**

   [Shopping Cart] 页看起来类似于下例：



6. **使用 [Resume Shopping] 按钮返回到 [CD Catalog List] 页。**

   CDShopCart 应用程序即将完成。现在剩下的工作只是创建 [EmptyCart] 和 [PlaceOrder] 页。

创建三个消息页

现在创建用户清空购物车时显示的 JSP 页，以及单击 [Place Order] 和 [Cancel Order] 按钮时显示的其它两个页。

创建 [Empty Cart] 页

当迭代器标记发现一个空 vector 时，将抛出一个异常而不是创建一个空表。您已通过测试对此情况进行了处理 （第 68 页的 “添加代码以从 Shopping Cart 表中添加或删除项目”中的步骤 7），然后您又通过显示 [Empty Cart] 页对该异常进行了处理。此页包含一个可使用户返回到产品列表页的 [Resume Shopping] 按钮。

图 3-3 [Empty Cart] 页

要创建 [Empty Cart] 页：

1. **右键单击 CDShopCart web 模块并选择 [新建] → [JSP]。**
2. **键入名称 `EmptyCart` 并单击 [完成]。**
   EmptyCart JSP 页会出现在 [资源管理器] 中，其源代码出现在 [源编辑器] 中。

3. 将页标题更改为 **[Empty Cart]** 并添加代码，如下所示：

```
<%@page contentType="text/html"%>
<html>
<head><title>Empty Cart</title></head>
<body>
<H1> Empty Cart </H1>
<!--Display a message-->
Your shopping cart is empty.
<P>
<!--Add a Resume Shopping button.-->
<form method=get action="ProductList.jsp">
<input type=submit value="Resume Shopping">
</FORM>
</P>
</body>
</html>
```

4. 通过选择 **[文件]** → **[保存]** 来保存 **[EmptyCart]** 页。

创建 [Place Order] 页

[Place Order] 和 [Cancel Order] 页非常简单，在您可以实现的许多操作中它仅显示了其中之一。由于编写的这些页所演示的 Sun ONE Studio 5 功能与您所看到的没有差别，因此指南将使用最简单的可能实现。

当用户单击 [Shopping Cart] 页上的 [Place Order] 按钮时，会出现 [Place Order] 页。显示此页后将结束会话。

图 **3-4** [Place Order] 页

要创建 [Place Order] 页：

1. 右键单击 **CDShopCart web** 模块并选择 **[** 新建 **]** → **[JSP]**。
2. 键入名称 **PlaceOrder** 并单击 **[** 完成 **]**。

3. 将页标题更改为 **[Place Order]** 并添加代码，如下所示：

```
<%@page contentType="text/html"%>
<html>
<head><title>Place Order</title></head>
<body>
<H1> Place Order </H1>
<!--Invalidate the session-->
<%
session.invalidate();
%>
Your order has been placed. Thank you for shopping.
<P>
<FORM method=get action="ProductList.jsp">
<INPUT type=submit value="Resume Shopping">
</FORM>
</P>
</body>
</html>
```

4. 通过选择 **[** 文件 **]** → **[** 保存 **]** 来保存 **[PlaceOrder]** 页。

创建 [Cancel Order] 页

当用户单击 [Shopping Cart] 页上的 [Cancel Order] 按钮时，会出现 [Cancel Order] 页。显示此页后将结束会话。该页类似于图 3-5。

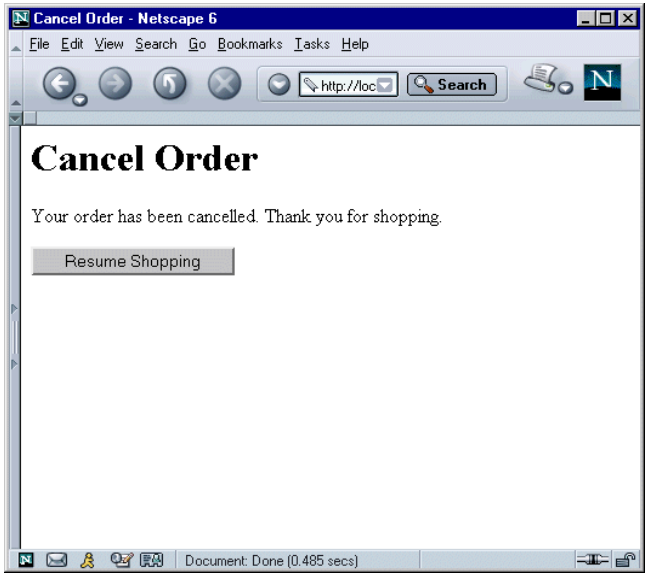


图 3-5 [Cancel Order] 页

要创建 [Cancel Order] 页：

1. 右键单击 **CDShopCart web** 模块并选择 **[** 新建 **]** → **[JSP & Servlet]** → **[JSP]**。
2. 键入名称 **CancelOrder** 并单击 **[** 完成 **]**。
3. 将页标题更改为 **[Cancel Order]** 并添加与 **[PlaceOrder]** 页类似的代码，如下所示：

```
<%@page contentType="text/html"%>
<html>
<head><title>Cancel Order</title></head>
<body>
<H1> Cancel Order </H1>
<%
session.invalidate();
%>
Your order has been cancelled. Thank you for shopping.
<P>
<FORM method=get action="ProductList.jsp">
<INPUT type=submit value="Resume Shopping">
</FORM>
</P>
</body>
</html>
```

4. 通过选择 **[** 文件 **]** → **[** 保存 **]** 来保存 **[CancelOrder]** 页。

测试三个消息页

如同 [Shopping Cart] 页的情况一样，通过运行 ProductList JSP 页来测试消息页。然后向 [Shopping Cart] 添加 CD 项并执行适当的操作以显示每一消息页。首先，您必须重新部署 CDShopCart web 应用程序，因为它包含以前部署的版本中没有的组件。

注意 – 确保 Sun ONE 应用程序服务器 7 是 IDE 的缺省 web 服务器。有关信息，参见第 27 页的 “确认 Sun ONE 应用程序服务器 7 作为缺省服务器”。

---

1. **如果尚未启动，则通过选择 [工具] → [PointBase 网络服务器] → [启动服务器] 来启动 PointBase 网络服务器。**
2. **右键单击 CDShopCart web 模块并选择 [全部生成]。**
3. **再次右键单击 CDShopCart web 模块并选择 [部署]。**

   IDE 将重新部署应用程序。一个进度窗口出现，显示部署过程的进度，部署完成后该窗口消失。
4. **右键单击 ProductList JSP 页并选择 [执行]。**

   IDE 启动缺省浏览器并显示 [CD Catalog List] 页。
5. **单击 [Add] 按钮导航至 [Shopping Cart] 页。**
6. **要测试 [Empty Cart] 页，可单击刚放入购物车中的项目上的 [Delete] 按钮。**

   [Empty Cart] 页出现。
7. **单击 [Resume Shopping] 按钮返回到 [CD Catalog List] 页。**
8. **向购物车添加一个或多个 CD。**
9. **通过单击 [Cancel Order] 按钮来测试 [Cancel Order] 页。**
10. **当该页出现时，单击 [Resume Shopping] 按钮返回到 [CD Catalog List] 页。**
11. **向购物车添加另一 CD。**

12. **当显示 [Shopping Cart] 页时，确保所添加的 CD 是购物车中唯一的一项。**
    购物车中应仅有一个 CD，因为 [Cancel Order] 操作 （步骤 9）终止了前一个会话。
13. **向购物车添加更多的 CD，然后测试 [Place Order] 按钮。**
14. **当 [Place Order] 页出现时，单击 [Resume Shopping] 按钮返回到 Catalog 页。**
15. **向购物车添加另一 CD。**
    [Place Order] 页已终止了会话。因此购物车中应仅有一个 CD。
16. **要停止应用程序，可将浏览器指向不同的 URL。**
    现在即完成了 CDShopCart 应用程序。

附录 A

CDShopCart 源文件

本附录显示下列 CDShopCart 组件的代码：

- 第 82 页的 “ProductList.jsp 源”
- 第 84 页的 “CartLineItem Bean 源”
- 第 87 页的 “CartLineItemBeanInfo 源”
- 第 91 页的 “Cart Bean 源”
- 第 94 页的 “ShopCart.jsp 源”
- 第 96 页的 “EmptyCart.jsp 源”
- 第 96 页的 “PlaceOrder.jsp 源”
- 第 97 页的 “CancelOrder.jsp 源”

此代码也在 CDShopCart 应用程序 zip 文件内作为源文件而提供，您可从下列地址中的 Sun ONE Studio 5 开发人员资源门户下载此文件：

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

提示 – 如果您使用这些文件将代码剪切并粘贴到 Sun ONE Studio 5 [源编辑器] 中，则所有的格式化均会丢失。要自动重新格式化此代码，请将光标放在 [源编辑器] 中，并按 Ctrl-Shift-F。

建议 Solaris 和 Linux 用户不要复制这些文件，因为 [源编辑器] 不会读取行末位置的回车。要查看源文件，请解压缩 CDShopCart 源 zip 文件，然后在 IDE 中安装解压缩的目录。

ProductList.jsp 源

```
<%@page contentType="text/html"%>
<html>

<head><title>CD Catalog List</title></head>
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
<body>
<h1> CD Catalog List </h1>

<sql:setDataSource var="productDS"
url="jdbc:pointbase:server://localhost/cdshopcart"
driver="com.pointbase.jdbc.jdbcUniversalDriver"
user="PBPUBLIC" password="PBPUBLIC" />

<%--<sql:setDataSource var="productDS"
url="jdbc:oracle:thin:@hostname:port#:SID"
driver="oracle.jdbc.driver.OracleDriver"
user="userid" password="password" /> --%>

<%--<sql:setDataSource var="productDS"
url="jdbc:weblogic:mssqlserver4:database@hostname:port#"
driver="weblogic.jdbc.mssqlserver4.Driver"
user="userid" password="password" /> --%>

<sql:query var="productQuery" dataSource="${productDS}" >
SELECT * FROM CD
</sql:query>

<TABLE border=1>
<TR>
<TH>ID</TH>
<TH>CD Title</TH>
<TH>Artist</TH>
<TH>Country</TH>
<TH>Price</TH>
</TR>
<c:forEach var="row" items="${productQuery.rows}" >
```

```
<TR>
<TD><c:out value="${row.ID}"/></TD>
<TD><c:out value="${row.CDTITLE}"/></TD>
<TD><c:out value="${row.ARTIST}"/></TD>
<TD><c:out value="${row.COUNTRY}"/></TD>
<TD><c:out value="${row.PRICE}"/></TD>
<TD>
<form method=get action="ShopCart.jsp">
<input type=hidden name=cdId value="<c:out value="${row.ID}"/>">
<input type=hidden name=cdTitle value="<c:out value="${row.CDTITLE}"/>">
<input type=hidden name=cdPrice value="<c:out value="${row.PRICE}"/>">
<input type=submit name=operation value=Add>
</form>
</TD>
</TR>
</c:forEach>
</TABLE>

</body>
</html>
```

CartLineItem Bean 源

```
/*
 * CartLineItem.java
 *
 * Created on April 11, 2003, 2:24 PM
 */

package ShopCart;

import java.beans.*;

public class CartLineItem extends Object implements java.io.Serializable {

    private static final String PROP_SAMPLE_PROPERTY = "SampleProperty";

    private String sampleProperty;

    private PropertyChangeSupport propertySupport;

    /** Holds value of property cdtitle. */
    private String cdtitle;

    /** Holds value of property id. */
    private int id;

    /** Holds value of property price. */
    private double price;

    /** Creates new CartLineItem */
    public CartLineItem() {
        propertySupport = new PropertyChangeSupport( this );
    }

    public String getSampleProperty() {
        return sampleProperty;
    }

    public void setSampleProperty(String value) {
```

```
        String oldValue = sampleProperty;
        sampleProperty = value;
        propertySupport.firePropertyChange(PROP_SAMPLE_PROPERTY, oldValue,
sampleProperty);
    }


    public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);
    }

    /** Getter for property cdtitle.
     * @return Value of property cdtitle.
     *
     */
    public String getCdtitle() {
        return this.cdtitle;
    }

    /** Setter for property cdtitle.
     * @param cdtitle New value of property cdtitle.
     *
     */
    public void setCdtitle(String cdtitle) {
        this.cdtitle = cdtitle;
    }

    /** Getter for property id.
     * @return Value of property id.
     *
     */
    public int getId() {
        return this.id;
    }

    /** Setter for property id.
     * @param id New value of property id.
```

```
     *
     */
    public void setId(int id) {
        this.id = id;
    }

    /** Getter for property price.
     * @return Value of property price.
     *
     */
    public double getPrice() {
        return this.price;
    }

    /** Setter for property price.
     * @param price New value of property price.
     *
     */
    public void setPrice(double price) {
        this.price = price;
    }

    public void setId(java.lang.String pId) {
        int val = Integer.parseInt(pId);
        this.setId(val);
    }

    public void setPrice(java.lang.String pPrice) {
        double val = Double.parseDouble(pPrice);
        this.setPrice(val);
    }

}
```

CartLineItemBeanInfo 源

```
package ShopCart;

import java.beans.*;

public class CartLineItemBeanInfo extends SimpleBeanInfo {

    // Bean descriptor information will be obtained from
introspection.//GEN-FIRST:BeanDescriptor
    private static BeanDescriptor beanDescriptor = null;
    private static BeanDescriptor getBdescriptor(){
        //GEN-HEADEREND:BeanDescriptor

        // Here you can add code for customizing the BeanDescriptor.

        return beanDescriptor;     } //GEN-LAST:BeanDescriptor


    // Properties information will be obtained from
introspection.//GEN-FIRST:Properties
    private static PropertyDescriptor[] properties = null;
    private static PropertyDescriptor[]
getPdescriptor(){//GEN-HEADEREND:Properties

        if (properties == null) {
            try {
                PropertyDescriptor[] props = {
                    new PropertyDescriptor("cdtitle",
                    CartLineItem.class),
                    new PropertyDescriptor("id", CartLineItem.class),
                    new PropertyDescriptor("price",
                    CartLineItem.class)};
                    properties = props;
            } catch (IntrospectionException ex) {
                return null;
            }
        }
```

```
        return properties;     } //GEN-LAST:Properties

    // Event set information will be obtained from
introspection.//GEN-FIRST:Events
    private static EventSetDescriptor[] eventSets = null;
   private static EventSetDescriptor[] getEdescriptor(){//GEN-HEADEREND:Events

        // Here you can add code for customizing the event sets array.

        return eventSets;     } //GEN-LAST:Events

   // Method information will be obtained from introspection.//GEN-FIRST:Methods
    private static MethodDescriptor[] methods = null;
    private static MethodDescriptor[] getMdescriptor(){//GEN-HEADEREND:Methods

        // Here you can add code for customizing the methods array.

        return methods;     } //GEN-LAST:Methods


    private static int defaultPropertyIndex = -1; //GEN-BEGIN:Idx
    private static int defaultEventIndex = -1; //GEN-END:Idx


    //GEN-FIRST:Superclass

    // Here you can add code for customizing the Superclass BeanInfo.

    //GEN-LAST:Superclass

    /**
     * Gets the bean's <code>BeanDescriptor</code>s.
     *
     * @return BeanDescriptor describing the editable
     * properties of this bean.  May return null if the
     * information should be obtained by automatic analysis.
     */
    public BeanDescriptor getBeanDescriptor() {
        return getBdescriptor();
    }
```

```
    /**
     * Gets the bean's <code>PropertyDescriptor</code>s.
     *
     * @return An array of PropertyDescriptors describing the editable
     * properties supported by this bean.  May return null if the
     * information should be obtained by automatic analysis.
     * <p>
     * If a property is indexed, then its entry in the result array will
     * belong to the IndexedPropertyDescriptor subclass of PropertyDescriptor.
     * A client of getPropertyDescriptors can use "instanceof" to check
     * if a given PropertyDescriptor is an IndexedPropertyDescriptor.
     */
    public PropertyDescriptor[] getPropertyDescriptors() {
        return getPdescriptor();
    }

    /**
     * Gets the bean's <code>EventSetDescriptor</code>s.
     *
     * @return  An array of EventSetDescriptors describing the kinds of
     * events fired by this bean.  May return null if the information
     * should be obtained by automatic analysis.
     */
    public EventSetDescriptor[] getEventSetDescriptors() {
        return getEdescriptor();
    }

    /**
     * Gets the bean's <code>MethodDescriptor</code>s.
     *
     * @return  An array of MethodDescriptors describing the methods
     * implemented by this bean.  May return null if the information
     * should be obtained by automatic analysis.
     */
    public MethodDescriptor[] getMethodDescriptors() {
        return getMdescriptor();
    }

    /**
     * A bean may have a "default" property that is the property that will
     * mostly commonly be initially chosen for update by human's who are
```

```
     * customizing the bean.
     * @return  Index of default property in the PropertyDescriptor array
     *    returned by getPropertyDescriptors.
     * <P>Returns -1 if there is no default property.
     */
    public int getDefaultPropertyIndex() {
        return defaultPropertyIndex;
    }

    /**
     * A bean may have a "default" event that is the event that will
     * mostly commonly be used by human's when using the bean.
     * @return Index of default event in the EventSetDescriptor array
     *    returned by getEventSetDescriptors.
     * <P>Returns -1 if there is no default event.
     */
    public int getDefaultEventIndex() {
        return defaultEventIndex;
    }
}
```

Cart Bean 源

```
/*
 * Cart.java
 *
 * Created on April 11, 2003, 2:36 PM
 */

package ShopCart;

import java.beans.*;

public class Cart extends Object implements java.io.Serializable {

    private static final String PROP_SAMPLE_PROPERTY = "SampleProperty";

    private String sampleProperty;

    private PropertyChangeSupport propertySupport;

    /** Holds value of property lineItems. */
    public java.util.Vector lineItems;

    /** Creates new Cart */
    public Cart() {
        propertySupport = new PropertyChangeSupport( this );
        lineItems = new java.util.Vector();
    }

    public String getSampleProperty() {
        return sampleProperty;
    }

    public void setSampleProperty(String value) {
        String oldValue = sampleProperty;
        sampleProperty = value;
        propertySupport.firePropertyChange(PROP_SAMPLE_PROPERTY, oldValue,
sampleProperty);
    }
```

```
public void addPropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.addPropertyChangeListener(listener);
}

public void removePropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.removePropertyChangeListener(listener);
}

/** Getter for property lineItems.
 * @return Value of property lineItems.
 *
 */
public java.util.Vector getLineItems() {
    return this.lineItems;
}

/** Setter for property lineItems.
 * @param lineItems New value of property lineItems.
 *
 */
public void setLineItems(java.util.Vector lineItems) {
    this.lineItems = lineItems;
}

public int findLineItem(int pID) {
    System.out.println("Entering Cart.findLineItem()");
    //Return the element number of the item in the cartItems
    //as specified by the passed ID.
    int cartSize = (lineItems == null) ? 0 : lineItems.size();
    int i;
    for (i = 0; i < cartSize; i++ ) {
        if ( pID == ((CartLineItem)lineItems.elementAt(i)).getId() )
            break;
    }
    if (i >= cartSize) {
        System.out.println("Couldn't find line item for ID: " + pID);
        return -1;
    }
    else
```

```
            return i;
    }

    public void removeLineItem(int pID) {
        System.out.println("Entering cart.removeLineItem()");
        int i = findLineItem(pID);
        if (i != -1) lineItems.remove(i);
        System.out.println("Leaving cart.removeLineItem()");
    }

}
```

ShopCart.jsp 源

```
<%@page contentType="text/html"%>
<%@page import="java.util.*, ShopCart.*" %>
<html>
<head><title>Shopping Cart</title></head>
<%@taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<body>
<h1> Shopping Cart </h1>
<jsp:useBean id="myCart" scope="session" class="ShopCart.Cart" />

<%
String myOperation = request.getParameter("operation");
session.setAttribute("myLineItems", myCart.getLineItems());

if (myOperation.equals("Add")) {
CartLineItem lineItem = new CartLineItem();
lineItem.setId(request.getParameter("cdId"));
lineItem.setCdtitle(request.getParameter("cdTitle"));
lineItem.setPrice(request.getParameter("cdPrice"));
myCart.lineItems.addElement(lineItem);
}
if (myOperation.equals("Delete")) {
String s = request.getParameter("cdId");
System.out.println(s);
int idVal = Integer.parseInt(s);
myCart.removeLineItem(idVal);
}
//If the last item is removed from the cart...
if (((Vector)session.getAttribute("myLineItems")).size() == 0) {
//End scriptlet temporarily so that you can use the JSP
//?forward? tag to forward to the EmptyCart page.
%>
<jsp:forward page="EmptyCart.jsp" />
//Resume the scriptlet.
<%
}
%>
```

```
<TABLE border=1>
<TR>
<TH>ID</TH>
<TH>CD Title</TH>
<TH>Price</TH>
</TR>
<c:forEach var="item" items="${myLineItems}">
<TR>
<TD><c:out value="${item.id}"/></TD>
<TD><c:out value="${item.cdtitle}"/></TD>
<TD><c:out value="${item.price}"/></TD>
<TD>
<form method=get action="ShopCart.jsp">
<input type=hidden name=cdId value="<c:out value="${item.id}"/>">
<input type=hidden name=cdTitle value="<c:out value="${item.id}"/>">
<input type=hidden name=cdPrice value="<c:out value="${item.price}"/>">
<input type=submit name=operation value="Delete">
</form>
</TD>
</TR>
</c:forEach>
</TABLE>

<p>
<!--Create the three buttons.-->
<form method=get action="ProductList.jsp">
<input type=submit value="Resume Shopping">
</form>
<form method=get action="PlaceOrder.jsp">
<input type=submit value="Place Order">
</form>
<form method=get action="CancelOrder.jsp">
<input type=submit value="Cancel Order">
</form>
</p>
<!End the page.>
</body>
</html>
```

EmptyCart.jsp 源

```
<%@page contentType="text/html"%>
<html>
<head><title>Empty Cart</title></head>
<body>
<H1> Empty Cart </H1>
<!--Display a message-->
Your shopping cart is empty.
<P>
<!--Add a Resume Shopping button.-->
<form method=get action="ProductList.jsp">
<input type=submit value="Resume Shopping">
</FORM>
</P>
</body>
</html>
```

PlaceOrder.jsp 源

```
<%@page contentType="text/html"%>
<html>
<head><title>Place Order</title></head>
<body>
<H1> Place Order </H1>
<!--Invalidate the session-->
<%
session.invalidate();
%>
Your order has been placed. Thank you for shopping.
<P>
<FORM method=get action="ProductList.jsp">
<INPUT type=submit value="Resume Shopping">
</FORM>
```

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

CancelOrder.jsp 源

```
<%@page contentType="text/html"%>
<html>
<head><title>Cancel Order</title></head>
<body>
<H1> Cancel Order </H1>
<%
session.invalidate();
%>
Your order has been cancelled. Thank you for shopping.
<P>
<FORM method=get action="ProductList.jsp">
<INPUT type=submit value="Resume Shopping">
</FORM>
</P>
</body>
</html>
```

附录 B

CDShopCart 数据库脚本

本附录显示 CDShopCart 教程的下列数据库脚本：

- 第 100 页的 “PointBase 数据库脚本”
- 第 101 页的 “Oracle 数据库脚本”
- 第 102 页的 “Microsoft SQLServer 数据库脚本”
- 第 103 页的 “IBM DB2 数据库脚本”

这些脚本也在 CDShopCart 应用程序 zip 文件中以文件形式提供，您可从 Sun ONE Studio 5 开发人员资源门户下载此 zip 文件：

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

PointBase 数据库脚本

将下列 SQL 脚本用于 PointBase 数据库。

```
drop table CD;

create table CD(
       id     int,
       cdtitle char(20),
       artist  char(20),
       country char(20),
       price   number(8,2),
       primary key(id));

insert into CD (id, cdtitle, artist, country, price)
  values (1, 'Yuan','The Guo Brothers', 'China',14.95);
insert into CD (id, cdtitle, artist, country, price)
  values (2, 'Drums of Passion','Babatunde Olatunji', 'Nigeria',16.95);
insert into CD (id, cdtitle, artist, country, price)
  values (3, 'Kaira','Tounami Diabate', 'Mali',13.95);
insert into CD (id, cdtitle, artist, country, price)
  values (4, 'The Lion is Loose','Eliades Ochoa', 'Cuba',12.95);
insert into CD (id, cdtitle, artist, country, price)
  values (5, 'Dance the Devil Away','Outback', 'Australia',14.95);

commit;
```

Oracle 数据库脚本

将下列 SQL 脚本用于 Oracle 数据库。

```
/* Run with: sqlplus tutorial/tutorial@dbname @scriptname */
drop table CD;

create table CD(
       id     int,
       cdtitle char(20),
       artist  char(20),
       country char(20),
       price   number(8,2),
       primary key(id));
grant all on CD to public;

insert into CD (id, cdtitle, artist, country, price)
  values (1, 'Yuan','The Guo Brothers', 'China',14.95);
insert into CD (id, cdtitle, artist, country, price)
  values (2, 'Drums of Passion','Babatunde Olatunji', 'Nigeria',16.95);
insert into CD (id, cdtitle, artist, country, price)
  values (3, 'Kaira','Tounami Diabate', 'Mali',13.95);
insert into CD (id, cdtitle, artist, country, price)
  values (4, 'The Lion is Loose','Eliades Ochoa', 'Cuba',12.95);
insert into CD (id, cdtitle, artist, country, price)
  values (5, 'Dance the Devil Away','Outback', 'Australia',14.95);
commit;
```

Microsoft SQLServer 数据库脚本

将下列 SQL 脚本用于 Microsoft SQLServer 数据库。

```
use cdcat
go
drop table CD
go

create table CD(
       id     int,
       cdtitle varchar(20) null,
       artist  varchar(20) null,
       country varchar(20) null,
       price   money null,
PRIMARY KEY(id))
go
grant all on CD to public
go

insert into CD (id, cdtitle, artist, country, price)
  values (1, 'Yuan','The Guo Brothers', 'China',14.95)
insert into CD (id, cdtitle, artist, country, price)
  values (2, 'Drums of Passion','Babatunde Olatunji', 'Nigeria',16.95)
insert into CD (id, cdtitle, artist, country, price)
  values (3, 'Kaira','Tounami Diabate', 'Mali',13.95)
insert into CD (id, cdtitle, artist, country, price)
  values (4, 'The Lion is Loose','Eliades Ochoa', 'Cuba',12.95)
insert into CD (id, cdtitle, artist, country, price)
  values (5, 'Dance the Devil Away','Outback', 'Australia',14.95)
go
```

IBM DB2 数据库脚本

将下列 SQL 脚本用于 IBM DB2 数据库。

```
drop table CD

create table CD(
       id     int not null primary key ,
       cdtitle char(20),
       artist  char(20),
       country char(20),
       price   num(8,2))

insert into CD (id, cdtitle, artist, country, price)
   values (1, 'Yuan','The Guo Brothers', 'China',14.95)
insert into CD (id, cdtitle, artist, country, price)
   values (2, 'Drums of Passion','Babatunde Olatunji', 'Nigeria',16.95)
insert into CD (id, cdtitle, artist, country, price)
   values (3, 'Kaira','Tounami Diabate', 'Mali',13.95)
insert into CD (id, cdtitle, artist, country, price)
   values (4, 'The Lion is Loose','Eliades Ochoa', 'Cuba',12.95)
insert into CD (id, cdtitle, artist, country, price)
   values (5, 'Dance the Devil Away','Outback', 'Australia',14.95)
```

附录 C

利用 Oracle 数据库创建教程

本附录介绍利用 Oracle 数据库创建并运行 CDShopCart 教程必须执行的步骤。包括的主题有：

- 第 106 页的 “将 IDE 连接到 Oracle 数据库”
- 第 108 页的 “创建数据库表”
- 第 110 页的 “利用 Oracle 数据库创建 CDShopCart 应用程序”

注意 – 本书有几处引用了 *CDShopCart 应用程序文件*。这些文件包括教程应用程序的已完成版本、一个说明如何运行已完成应用程序的自述文件，以及用于创建所需数据库表的 SQL 脚本文件。这些文件被压缩成一个 zip 文件，可以从 Sun ONE Studio 5 开发人员资源门户下载，网址为：
`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

将 IDE 连接到 Oracle 数据库

配置 Sun ONE Studio 5 软件以连接到 Oracle 数据库需要进行以下操作：

- 启用 Oracle JDBC 驱动程序
- 将 IDE 连接到 Oracle Server

启用 Oracle Type 4 JDBC 驱动程序

启用 JDBC 驱动程序也就是将驱动程序库放入 Sun ONE Studio 5 和 Sun ONE 应用程序服务器 7 类路径中。为此，需要 Oracle Type 4 JDBC 驱动程序库 （`classes12.zip` 文件）。您可以从 Oracle 门户下载此驱动程序。

要启用 Oracle Type 4 JDBC 驱动程序：

1. **将 Oracle Type 4 驱动程序库复制到** *s1studio-install-directory*`/lib/ext` **目录**。

   例如，将 `classes12.zip` 文件复制到 `c:\Sun\studio5_se\lib\ext`。Solaris 和 Linux 用户将具有不同的目标目录。

注意 – 您必须拥有根特权或管理员权限才能向 Sun ONE Studio 5 起始目录中写入。

---

2. **重新启动 IDE**。
3. **在 [资源管理器] 的 [运行环境] 窗格中，选择应用程序服务器实例**。

   它被标记为 *app-server-name* (*app-server-host*:*app-server-port*)。例如，缺省服务器为服务器 1 (localhost:4848)，或者，标准用户的服务器可能为 MyServer (localhost:4855)。

4. **显示应用程序服务器实例的属性**。

   属性窗口通常位于 [资源管理器] 窗口下面。选择节点可以在窗口中显示属性。或者，右键单击服务器实例节点，并选择 [属性]。

5. **打开 [类路径后缀] 属性的属性编辑器**。

   单击该属性的值字段，然后单击出现的省略号按钮。显示 [类路径后缀] 编辑器窗口。

6. **单击 [添加 JAR/ZIP] 按钮。**

   使用 [添加 JAR 文件] 文件查找器查找 `classes12.zip` 文件。

7. **选择 `classes12.zip` 文件，并单击 [确定]。**

8. **单击 [确定] 关闭属性编辑器窗口。**

将 IDE 连接到 Oracle Server

要在教程中执行各种操作，就必须将 IDE 连接到 Oracle 数据库。可在创建组件前或在创建过程中进行连接。以下是预先与数据库建立连接的方法：

1. **确保 Oracle 服务器正在运行。**

2. **在 [资源管理器] 的 [运行环境] 窗格中，展开 [数据库] 节点及其 [驱动程序] 子节点。**

   显示一个标记为 Oracle thin 的节点。

   如果此节点上有红色闪电符号穿过，则表明您未正确地启用 Oracle JDBC 驱动程序。按照第 106 页的 “启用 Oracle Type 4 JDBC 驱动程序”中的步骤进行操作。

3. **右键单击该节点并选择 [连接使用]。**

   显示 [新建数据库连接] 对话框。

4. **确保在名称字段中选择 Oracle thin。**

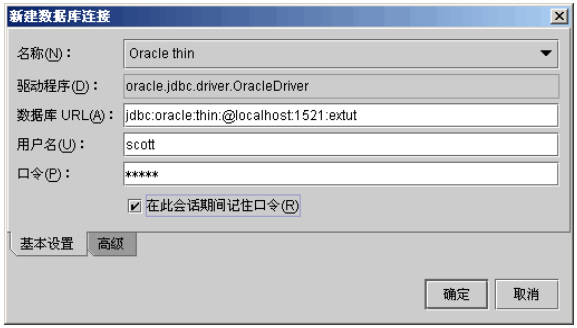
5. **填入 [URL]、 [用户] 和 [口令] 的属性值。**

   例如，下列值对于一个本地安装的 Oracle 数据库而言是正确的，该数据库的 SID 为 “extut”，缺省的 Oracle 登录内容：[用户] 为 “scott”， [口令] 为 “tiger”。（1521 是标准 Oracle 端口号。）

名称	值
URL	jdbc:oracle:thin:@localhost:1521:extut
用户	scott
口令	tiger

6. **启用 [在此会话期间记住口令] 选项。**

[新建数据库连接] 对话框显示如下：



7. **单击 [确定]**。

8. **关闭 [驱动程序] 节点**。

   显示新的 Oracle thin 驱动程序节点，它标记为 jdbc:oracle:thin:@*hostname*:1521:*sid* [*Username* 在 *Password* 之上]。

创建数据库表

CDShopCart 教程使用一个数据库表，您必须在 Oracle Server 数据库中创建此表。随后的说明将介绍如何使用提供的 SQL 脚本创建该表。 Microsoft Windows 用户可以复制并粘贴附录 B 中提供的 SQL 脚本以创建此表。 Solaris 和 Linux 用户可以使用脚本文件 `CDCatalog_ora.sql`，该文件在 CDShopCart 应用程序文件中提供。

要在 Microsoft Windows 系统的 Oracle 数据库中安装教程表：

1. **通过选择 [开始] → [程序] → Oracle（您的版本）→ [应用程序开发] → [SQL Plus] 打开 [Oracle 控制台]**。

2. **使用您的用户名和口令登录到 SQL Plus**。

3. **当出现 SQL 提示符时，从附录 B 中复制脚本并将其粘贴到提示符后**。

提示 – 初始的 DROP 语句 （该语句将引用一个还未创建的表）将生成一个无害错误。但是，如果您将来想再次运行此脚本来初始化该表，将会用到这条 DROP 语句。

要在 Solaris 或 Linux 环境中安装教程表：

1. **解压缩来自开发人员资源门户的** `CDShopCart.zip` **文件。**

   例如，将其解压缩到 `/MyZipFiles` 目录。

2. **在命令提示符处，键入：**

```
$ cd your-unzip-dir/CDShopCart/db
$ sqlplus db-userid/db-password@db-servicename @CDCatalog_ora.sql
```

例如，

```
$ cd /MyZipFiles/CDShopCart/db
$ sqlplus scott/tiger@MyDB @CDCatalog_ora.sql
```

DROP 语句将产生错误，但是错误无害。

在 IDE 中查看数据库表

执行此步骤前，将 IDE 连接到数据库，如第 107 页的 “将 IDE 连接到 Oracle Server” 所述。

要测试您已在连接到 IDE 的数据库中创建了 CD 表：

1. **在 [资源管理器] 的 [运行环境] 标签中，展开 [数据库] 节点、 [Oracle 连接] 节点及其 [表] 节点。**

   如果未看到 CD 表，右键单击 [表] 节点，并从上下文菜单中选择 [刷新]。

2. **右键单击 CD 表，并从上下文菜单中选择 [查看数据]。**

   显示 [命令编辑器]，其中显示表的数据。

利用 Oracle 数据库创建 CDShopCart 应用程序

要使用 Oracle 数据库创建应用程序，您只需对第 3 章进行一处更改：

在第 53 页的 “使用 setDataSource 标记连接到数据库”一节的第 2 步中，使用 setDataSource 语句 f 或 Oracle 而非 PointBase。针对您的连接为变量填入正确的值。例如，对于本地系统上的 Oracle 数据库，数据库名为 “MyOracleDB”，用户 id 为 “scott”，口令为 “tiger”：

```
<sql:setDataSource var="productDS"
url="jdbc:oracle:thin:@localhost:1521:MyOracleDB"
driver="oracle.jdbc.driver.OracleDriver"
user="scott" password="tiger" />
```

否则，照文中所述执行其它所有指令。

索引

A

安装此数据库表，27 至 29

B

bean 属性，增加，59
BeanInfo w/o 图标菜单项，64
编译菜单项，63

C

`CancelOrder` JSP 页
  创建，77 至 78
  说明，39
  显示，78
  源代码，97
`Cart` bean
  创建，65 至 68
  对构造函数编码，66
  说明，39
  添加 `findLineItem` 方法，67
  添加 `lineItems` 属性，65
  添加 `removeLineItem` 方法，67
  源代码，91
`CartLineItem` bean
  `setId` 和 `setPrice` 重载，61
  创建，59 至 63
  说明，39
  源代码，84
`CartLineItemBeanInfo` 组件
  创建，63 至 65
  说明，39
  源代码，87
CD 表，30
`CDShopCart` web 模块
  创建，44 至 46
  设置上下文根，46
CDShopCart 应用程序
  功能说明，31, 33
  结构，38
  压缩源文件，19, 105
  应用程序方案，32
CDShopCart 应用程序页
  Cancel Order，78
  CD Catalog List，51
  Empty Cart，74
  Place Order，76
  Shopping Cart，58
core JSP 标记
  `forEach`，54, 71
  `out`，54, 71
  使用，54
`core.tld` 文件
  导入，53
  描述，48
查询数据库 JSP 标记，54
重新格式化代码，63
[创建服务器实例] 菜单项，24

E

EmptyCart JSP page
  源代码 , 96
EmptyCart JSP 页
  创建 , 74
  说明 , 38

F

findLineItem 方法，创建 , 67
forEach JSP 标记 , 54, 71
方法，创建 , 61, 67

H

HTML 页
  查看源代码 , 74
  创建 , 74

I

IBM DB2 软件，教程数据库脚本源 , 103

J

JavaBean 组件
  添加方法 , 61, 67
  添加属性 , 65
Javadoc
  在 IDE 中使用 , 16
JDBC 驱动程序，启用 (Oracle), 106 至 107
JSP 标记库
  JSTL，*参见* JSTL 标记库
  描述 , 47
JSP 代码，类型 , 47
JSP 页
  测试 , 56 至 57
  创建 , 68
JSTL 标记库
  参见 core JSP 标记，数据库 JSP 标记
  JAR 文件的定义 , 48
  jstl.jar 文件 , 48
  standard.jar 文件 , 48
  导入到 web 模块 , 49
  和属性 , 64
  联机信息 , 47
  使用 var, 49
  示例 , 47
  在 JSP 页中导入 , 48
jstl.jar 文件 , 48

L

lineItems 属性，创建 , 65
类路径，与安装的文件系统相关 , 50

M

Microsoft SQLServer 软件
  jdbc 连接字符串 , 53
  教程数据库脚本源 , 102

N

Netscape 浏览器，支持版本 , 20

O

Oracle 数据库
  *参见*数据库
  JDBC 连接字符串 , 53, 110
  安装 type 4 JDBC 驱动程序 , 106
  教程数据库脚本源 , 101
  连接到 IDE, 107 至 108
  在 IDE 中查看数据 , 109
out JSP 标记 , 54, 71

P

PlaceOrder JSP 页
  创建 , 75
  说明 , 39

源代码，96
在 TP 前显示，76
PointBase 数据库
JDBC 连接字符串，53
安装数据库表，27 至 29
教程数据库脚本源，100
支持版本，20
ProductList JSP 页
测试，56 至 57
创建，51 至 56
连接到 Oracle，110
说明，38
用户视图，33
源代码，82
在浏览器中显示，51

R

removeLineItem 方法，创建，67
runide.sh 脚本，21

S

setDataSource 数据库 JSP 标记
Oracle 设置，110
PointBase 设置，53
setDataSource 数据库 JSP 标记
语法，49
setId 方法，重载，61
setPrice 方法，重载，63
ShopCart JSP 页
创建，68 至 72
创建按钮，72
对正文进行编码，68
说明，38
源代码，94
在浏览器中显示，58
执行测试，72
ShopCart 包，创建，59
sql.tld 文件
导入，53
描述，48
standard.jar 文件，48
Sun ONE Studio 5 IDE
类路径，50
连接到 Oracle 服务器，107 至 108
启动 IDE，21
设置数据库连接 (Oracle)，106 至 108
Sun ONE Studio 5, Standard Edition，获得地址，20
Sun ONE 应用程序服务器 7
启动管理服务器 （标准用户），23 至 25
启动管理服务器 （超级用户），22
启动应用程序服务器，26
确认为缺省服务器，27
上下文根属性，46
示例应用程序，下载位置，16
数据库
安装 Oracle JDBC 驱动程序，106
创建教程表 (Oracle)，108 至 109
创建教程表 (PointBase)，28 至 29
设置连接 (Oracle)，106 至 108
在 IDE 中查看数据，109
支持版本，20
数据库 JSP 标记
setDataSource，53, 110
查询，49, 54

T

taglib 指令，使用，48, 53

V

var，在 JSTL 标记语法中，49

W

WAR 文件，定义，44
web 服务器
确认缺省服务器，27
支持版本，20
web 模块
部件快照，46

部署，57
创建，43
何处查找详细信息，37
目录分层结构，44
说明，40
执行，57
执行全部生成，57
web 应用程序，37
web 组件，37
web 浏览器，支持版本，20
web.xml 文件，描述符，44
WEB-INF 目录，44

X

新建 Java Bean 菜单项，59
新建 Java 包菜单项，59
新建 JSP 菜单项，68
新建向导，打开，44

Z

增加 JSP 标记库菜单项，49
增加方法菜单项，61, 67
增加管理服务器菜单项，23
增加属性菜单项，59