构建 J2EE™ 应用程序

Sun™ ONE Studio 5 编程系列

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

部件号码 817-3290-10
2003 年 10 月，修订 A

关于本文档的建议请发到：docfeedback@sun.com

请回收

目录

图

开始之前

Sun Microsystems, Inc 支持的 Java Community Process 发展了用 Java™ 2 平台，Enterprise Edition （J2EE™ 平台）设计分布式企业应用程序的标准。第 10 页的 “阅读本书须知” 中所列的 J2EE 平台文档涵盖了应用程序设计和体系结构的这些标准。

本书介绍了如何使用 Sun™ ONE Studio 5, Standard Edition 开发人员工具来实现这些体系结构。其中包括使用 IDE 来组合组件并创建 J2EE 模块，确定所有组件都按照应用程序设计所指定的方式进行交互操作。此外还包括组合 J2EE 模块来创建 J2EE 应用程序，确定模块间的分布式交互操作按照应用程序设计所调用的方式进行。

不同平台中的屏幕截图会稍有不同。虽然几乎所有的过程都使用 Sun ONE Studio 5 软件界面，但有时也可能要求您在命令行输入命令。不同平台的命令行也会稍有不同，例如，在 Microsoft Windows 中的命令有可能是：

```
c:>cd MyWorkDir\MyPackage
```

而 UNIX 命令则可能是：

```
% cd MyWorkDir/MyPackage
```

阅读本书须知

本书旨在让使用 Sun ONE Studio 5 IDE 的读者能够组装、部署或执行 J2EE 应用程序。第一章简明扼要地介绍了在 J2EE 平台中进行组装和部署的概念，读者可以获得对组装和部署的初步理解。

阅读本书前，您必须熟悉以下内容：

- Java 编程语言
- J2EE 的概念
- Web 和应用程序服务器软件

本书要求具备 J2EE 概念的知识，如下列资源中所述：

- 《Java 2 平台，Enterprise Edition 蓝图》
  `http://java.sun.com/j2ee/blueprints`
- 《Java 2 平台，Enterprise Edition 规范》
  `http://java.sun.com/j2ee/download.html#platformspec`
- 《J2EE 教程》
  `http://java.sun.com/j2ee/tutorial`
- 《Java Servlet 规范 2.3 版》
  `http://java.sun.com/products/servlet/download.html#specs`
- 《JavaServer Pages 规范 1.2 版》
  `http://java.sun.com/products/jsp/download.html#specs`

若熟悉基于 XML 的 RPC 中的 Java API (JAX-RPC)，则对读者更有帮助。更多信息请参阅以下 web 页：

`http://java.sun.com/xml/jaxrpc`

注 – Sun 公司不对本文档所提及的第三方 web 站点的可用性负责，而且 Sun 公司不认可也不对以上站点或资源上的任何内容、广告、产品或其它资料承担责任。此外， Sun 公司也不会因您使用或依靠以上任何站点或资源上的 （或通过该站点或资源所获取的）内容、货物或服务所产生的 （或所谓产生的）任何损失承担责任。

本书的结构

J2EE 平台能够使用面向组件的方法来开发企业应用程序。应用程序开发人员将企业逻辑封装在 Enterprise JavaBeans™ (EJB™) 组件和 web 组件中。创建组件之后，开发人员将组件组装到模块中而模块是执行可识别业务任务的逻辑单元。组装模块之后，开发人员再将模块组装到 J2EE 应用程序中， J2EE 应用程序执行全部的业务处理。

本书介绍了如何使用 Sun ONE Studio 5 开发环境将组件组装到模块中，再将模块组装到应用程序中。本书通过一系列方案来展现这些信息。

第 1 章概述了组装和部署 J2EE 的概念。其中定义了模块和应用程序的 J2EE 单元，并且仔细分析了模块和应用程序的部署描述符。此外还解释了如何在 IDE 中组装模块和应用程序。特别要指出的是，该章还解释了如何使用模块和应用程序属性表来设置模块和应用程序部署描述符。

第 2 章提供了如何组装 web 模块的方案。此外该章还简要描述了用作 J2EE 应用程序前端的 web 模块，最后介绍了如何对 web 模块进行编程。

第 3 章提供了如何组装 EJB 模块的方案。该章简要描述了用于 J2EE 应用程序的 EJB 模块。最后介绍了如何对 EJB 模块进行编程。

第 4 章提供了如何通过组合 web 模块和 EJB 模块来组装 J2EE 应用程序的方案。该章概述了组合 web 模块和 EJB 模块的 J2EE 应用程序。在该章的最后，介绍了如何组装应用程序。该方案使用 Java RMI 在两个模块之间进行同步交互操作。

第 5 章提供了如何使用消息驱动企业 bean 或 MDB 建立模块间异步通信的方案。该章简要描述了用于企业应用程序的异步通信。最后，该章介绍了如何对应用程序的发送端和接收端进行编程。这一章的方案使用了与 EJB 模块进行通信的 web 模块，不过其中的示例也适用于其它模块组合。

第 6 章解释了如何使用 IDE 来编程容器管理的事务。

第 7 章解释了如何使用 IDE 保护 J2EE 应用程序中的资源。该章介绍了如何在模块级设置安全角色，如何使用安全角色来限制对 web 模块资源和企业 bean 方法的访问。此外，该章还介绍了将模块组装到应用程序时如何映射安全角色。

第 8 章解释了如何部署并执行已组装的应用程序。特别要指出的是，该章解释了如何在部署前使应用程序适应特定的服务器产品。

附录 A 说明 IDE 用于与 web 和应用程序服务器进行交互的机制，其中包括对部署过程的详细说明。

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 `.cvspass` 文件。 使用 `DIR` 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	`> login` `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*保存更改。
AaBbCc123	命令行变量，用实际的名称或值替换	要删除文件，请输入 **DEL** *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, Mobile 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）

  这些教程演示了如何使用 Sun ONE Studio 4 各版本的主要功能。

  - 《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 功能以及构建 J2EE 应用程序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。启动此 IDE 时，您可以在资源管理器的 [Javadoc] 窗格中访问到此 Javadoc 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，位于 `http://docs.sun.com`
迷你教程	HTML，位于 `http://forte.sun.com/ffj/tutorialsandexamples.html`
集成示例自述文件	HTML，位于 *s1studio-install-directory*`/examples` 的示例子目录中
发行说明	HTML，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 (817-3290-10)。

第 1 章

组装、部署和执行的基础知识

使用 J2EE 平台进行开发的模块化特性是将较小的单元进行组合以创建较大的单元。您可以组合组件以创建模块，然后组合模块以创建应用程序。这种组合较小的 J2EE 软件单元来创建较大单元的方法被称为组装。

组装的模块和应用程序需要运行环境服务，例如，由 J2EE 平台提供的容器管理持久性、容器管理事务和容器管理安全性验证。组装模块或应用程序时必须决定所需的运行环境服务，然后必须在 J2EE 部署描述符中指定这些服务。

本章描述了 J2EE 模块和应用程序的一些基本特性，这些特性会影响组装过程。此外还介绍了使用 Sun ONE Studio 5 开发人员工具进行组装的基础知识。

组装的基础知识

J2EE 组装是一个包括许多独立任务的过程。只有正确组装应用程序或模块，才能够将应用程序或模块部署到 J2EE 应用程序服务器，并执行该模块或应用程序。

成功组装的最大阻碍是组装过程的多变性。因此，组装的每个模块或应用程序都需要不同运行环境服务的组合和一组不同的组装任务，组装模块和应用程序没有标准过程。在组装过程开始前，必须了解什么是正确组装的模块或应用程序。本节提供了在 J2EE 平台上的背景知识，能够帮助您识别什么是正确组装的模块或应用程序。

J2EE 应用程序是模块化的

J2EE 应用程序是一组模块，应用程序中的模块是多组的组件。在 J2EE 平台上，将组件组合为模块并将模块组合为应用程序的机制是部署描述符。部署描述符是对模块或应用程序的 “成分列表”。

- 应用程序的部署描述符列出了应用程序中的模块。
- 模块的部署描述符列出了模块中的组件。

要了解 J2EE 平台使用部署描述符的原因，请考虑如何部署并执行应用程序的源代码。在开发阶段，组件以许多源文件的形式存在于开发环境中。只有将这些源文件部署到 J2EE 应用程序服务器时，才能够执行它们。这些组件必须在由应用程序服务器提供的运行环境下执行。

部署应用程序时将编译应用程序部署描述符中列出的源文件，然后将编译后的文件安装在应用程序服务器管理的目录中。在部署时，源文件实际上是被编译到 J2EE 应用程序中。部署完成后，就可以在应用程序服务器的环境中执行部署的应用程序。

部署描述符是开发阶段使用的一种机制，其中列出了将同时作为模块或应用程序部署的文件。在开发阶段组装模块或应用程序时，实际上并没有修改源文件，您只需准备一个描述部署过程的模块或应用程序的部署描述符。

部署描述符是 XML 文件。它们使用特定的 XML 标记来标识应用程序和组成应用程序的模块 （或标识模块和组成模块的组件）。编码范例 1-1 显示了名为 `CatalogApp` 的 J2EE 应用程序中的部署描述符。该部署描述符列出了 `CatalogApp` 应用程序中的模块。

编码范例 1-1 `CatlaogApp` 的部署描述符

```
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application
      1.3//EN" "http://java.sun.com/dtd/application_1_3.dtd">
<application>
 <?xml version="1.0" encoding="UTF-8"?>
 <display-name>CatalogApp</display-name>
 <description>J2EE Application CatalogApp</description>
 <module>
  <ejb>CatalogData.jar</ejb>
  <alt-dd>CatalogData.xml</alt-dd>
 </module>
 <module>
  <web>
   <web-uri>CatalogWebModule.war</web-uri>
   <context-root>catalog</context-root>
  </web>
  <alt-dd>CatalogWebModule.xml</alt-dd>
 </module>
</application>
```

`CatalogApp` 应用程序包含两个模块：`CatalogData` 和 `CatalogWebModule`。该部署描述符使用 `<module>` 标记来标识模块。

部署 `CatalogApp` 应用程序时，应用程序服务器读取应用程序的部署描述符。`CatalogApp` 部署描述符所列出的每个模块都有其自身模块级的部署描述符。应用程序服务器继续读取这两个模块的部署描述符。这些部署描述符标识两个模块中 J2EE 组件的源文件。编码范例 1-2 和编码范例 1-3 显示模块级的部署描述符。

使用 Sun ONE Studio 5 IDE 时，IDE 为您准备了部署描述符。您不必书写部署描述符标记，但是应该了解在 IDE 中工作时 IDE 会为您准备部署描述符。

J2EE 应用程序由 J2EE 运行环境支持

在运行环境下，J2EE 应用程序使用由 J2EE 应用程序服务器所提供的服务。这些服务包括容器管理持久性、容器管理事务和容器管理安全性验证。

应用程序和其中的模块必须让应用程序服务器知道它们所需的服务。部署描述符就是告诉应用程序服务器哪些是所需服务的机制。

例如，考虑 J2EE 容器管理事务。要使用容器管理事务服务，就必须告诉 J2EE 应用程序服务器需要哪些事务服务。将企业 bean 组装到 EJB 模块时，您要通过设置每个企业 bean 的事务特性属性来定义事务边界。 IDE 在部署描述符中包括每个企业 bean 事务属性的值。

编码范例 1-2 显示名为 `CatalogData` 的 EJB 模块 （`CatalogData` 模块是编码范例 1-1 中所列的两个模块之一）中的部署描述符。具有事务属性值的标记出现在部署描述符末端。

编码范例 1-2 EJB 模块部署描述符

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
       2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
 <display-name>CatalogData</display-name>
 <enterprise-beans>
  <session>
   <display-name>CatalogManagerBean</display-name>
   <ejb-name>CatalogManagerBean</ejb-name>
   <home>CatalogBeans.CatalogManagerBeanHome</home>
   <remote>CatalogBeans.CatalogManagerBean</remote>
   <ejb-class>CatalogBeans.CatalogManagerBeanEJB</ejb-class>
   <session-type>Stateful</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-local-ref>
    <ejb-ref-name>ejb/ItemBean</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
    <local-home>CatalogBeans.ItemBeanLocalHome</local-home>
    <local>CatalogBeans.ItemBeanLocal</local>
    <ejb-link>ItemBean</ejb-link>
   </ejb-local-ref>
   <ejb-local-ref>
    <ejb-ref-name>ejb/ItemDetailBean</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
    <local-home>CatalogBeans.ItemDetailBeanLocalHome</local-home>
    <local>CatalogBeans.ItemDetailBeanLocal</local>
    <ejb-link>ItemDetailBean</ejb-link>
   </ejb-local-ref>
  </session>
```

编码范例 **1-2** EJB 模块部署描述符 *（后续）*

```
  <entity>
   <display-name>ItemBean</display-name>
   <ejb-name>ItemBean</ejb-name>
   <local-home>CatalogBeans.ItemBeanLocalHome</local-home>
   <local>CatalogBeans.ItemBeanLocal</local>
   <ejb-class>CatalogBeans.ItemBeanEJB</ejb-class>
   <persistence-type>Container</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <reentrant>False</reentrant>
   <abstract-schema-name>ItemBean</abstract-schema-name>
   <cmp-field>
    <field-name>itemsku</field-name>
   </cmp-field>
   <cmp-field>
    <field-name>itemname</field-name>
   </cmp-field>
   <primkey-field>itemsku</primkey-field>
   <query>
    <query-method>
     <method-name>findAll</method-name>
     <method-params/>
    </query-method>
    <ejb-ql>SELECT Object (I) FROM ItemBean AS I</ejb-ql>
   </query>
  </entity>
  <entity>
   <display-name>ItemDetailBean</display-name>
   <ejb-name>ItemDetailBean</ejb-name>
   <local-home>CatalogBeans.ItemDetailBeanLocalHome</local-home>
   <local>CatalogBeans.ItemDetailBeanLocal</local>
   <ejb-class>CatalogBeans.ItemDetailBeanEJB</ejb-class>
   <persistence-type>Container</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <reentrant>False</reentrant>
   <abstract-schema-name>ItemDetailBean</abstract-schema-name>
    <cmp-field>
     <field-name>itemsku</field-name>
    </cmp-field>
    <cmp-field>
     <field-name>description</field-name>
    </cmp-field>
    <primkey-field>itemsku</primkey-field>
   </entity>
 </enterprise-beans>
 <assembly-descriptor>
  <container-transaction>
   <description>This value was set as a default by Sun ONE Studio.</description>
```

编码范例 1-2 EJB 模块部署描述符 *（后续）*

```
    <method>
     <ejb-name>CatalogManagerBean</ejb-name>
     <method-name>*</method-name>
     </method>
     <trans-attribute>Required</trans-attribute>
    </container-transaction>
    <container-transaction>
    <description>This value was set as a default by Sun ONE Studio.</description>
     <method>
      <ejb-name>ItemBean</ejb-name>
      <method-name>*</method-name>
     </method>
     <trans-attribute>Required</trans-attribute>
    </container-transaction>
    <container-transaction>
    <description>This value was set as a default by Sun ONE Studio.</description>
     <method>
     <ejb-name>ItemDetailBean</ejb-name>
     <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
   </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

当部署并执行包含该 EJB 模块的应用程序时，应用程序服务器识别在部署描述符中指定的事务边界。应用程序服务器在指定点打开并提交事务 （或将事务转返）。

其它运行环境服务的处理与容器管理事务类似。每个服务都有自身的部署描述符标记，可以明确指出所需的服务。 IDE 自动为您写入这些标记，所以不需要了解用在部署描述符中的标记。

J2EE 应用程序是分布式的

J2EE 应用程序不仅是模块化和使用 J2EE 平台的运行环境服务，此外它还是分布式的。应用程序中的每个模块都可以部署到不同的计算机来创建分布式应用程序。图 1-1 显示了由两个模块所组成的应用程序。该应用程序实现了典型的多层应用程序体系结构。

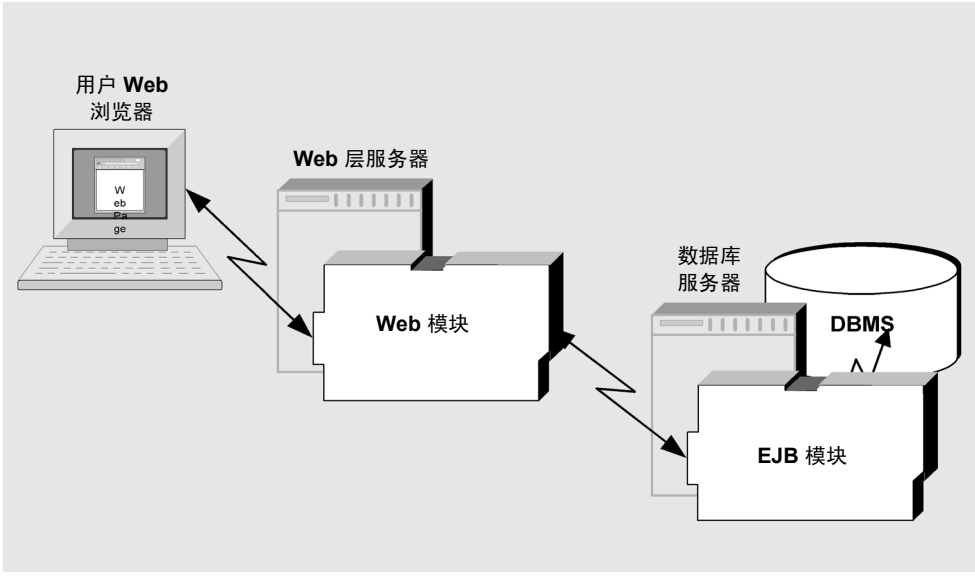

图 1-1 使用 J2EE 组件和模块的多层应用程序

Web 模块被部署到专门与用户进行 HTTP 交互的计算机上。应用程序服务器提供 HTTP 连接，使用该 HTTP 连接，应用程序服务器将 web 模块中定义的 web 页发送到在用户桌面计算机上运行的浏览器。

EJB 模块部署到专门用于数据库操作的另一台计算机上。 Web 模块使用 Java RMI 与 EJB 模块进行通讯。而应用程序服务器提供对 Java RMI 交互操作的运行环境支持。

J2EE 平台提供多种技术支持模块间的分布式交互操作， J2EE 平台的分布式技术包括：

- **通过 HTTP 连接的基于 web 的通讯。**这种通讯通常在用户和应用程序之间使用。
- **使用 Java RMI-IIOP 的同步方法调用。**这种方法调用主要用于调用企业 bean 方法。
- **使用 Java 消息服务 (JMS) 的异步消息传递。**消息可以送至队列或主题中。

J2EE 平台提供支持不同种交互操作的不同种组件。例如，J2EE 平台使用消息驱动企业 bean 来支持模块间的异步消息传递。

在应用程序中，为分布式交互操作使用哪种技术是应用程序设计的一项任务。决定使用哪种组件也是一项设计任务。在组装应用程序时，需要知道已设计了哪种交互操作，以及如何实现该操作。您要通过执行设置 EJB 引用来完成 Java RMI 交互操作，设置队列来实现 JMS 消息传递交互操作这样的组装任务来实现交互操作。

J2EE 平台还支持 J2EE 模块与外部资源 （如数据源）间的交互操作。支持这些交互操作的 J2EE 技术包括下列内容：

- JDBC™ 技术
- 容器管理持久性

组装应用程序时，请确保已经标识了由应用程序使用的任何外部资源。标识外部资源的开发阶段机制是部署描述符。

编码范例 1-3 显示了名为 `CatalogWebModule` 的 web 模块的部署描述符。该模块和名为 `CatalogData` 的 EJB 模块被组装到 J2EE 应用程序中。在两个模块间进行交互操作的技术是 Java RMI。Java RMI 交互操作需要一个远程 EJB 引用，该远程 EJB 引用由编码范例 1-3 底部的 `<ejb-ref>` 标记声明。

编码范例 1-3 Web 模块部署描述符

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
    2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
 <servlet>
  <servlet-name>AllItemsServlet</servlet-name>
  <servlet-class>AllItemsServlet</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>DetailServlet</servlet-name>
  <servlet-class>DetailServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>AllItemsServlet</servlet-name>
  <url-pattern>/servlet/AllItemsServlet</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>DetailServlet</servlet-name>
  <url-pattern>/servlet/DetailServlet</url-pattern>
 </servlet-mapping>
 <session-config>
  <session-timeout>
   30
  </session-timeout>
 </session-config>
 <welcome-file-list>
  <welcome-file>
   index.jsp
   </welcome-file>
   <welcome-file>
    index.html
   </welcome-file>
```

编码范例 1-3 Web 模块部署描述符 *（后续）*

```
    <welcome-file>
     index.htm
    </welcome-file>
   </welcome-file-list>
   <ejb-ref>
    <ejb-ref-name>ejb/CatalogManagerBean</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>CatalogBeans.CatalogManagerBeanHome</home>
    <remote>CatalogBeans.CatalogManagerBean</remote>
    <ejb-link>CatalogManagerBean</ejb-link>
   </ejb-ref>
</web-app>
```

在 Sun ONE Studio 5 IDE 中工作时，您无须编码模块和应用程序中的部署描述符，而是使用 IDE 中组件、模块和应用程序的可视化表示。 IDE 会为您准备部署描述符。

模块和应用程序的可视化表示

J2EE 组装的多数说明讨论了部署描述符文件的内容，例如在编码范例 1-1、编码范例 1-2 和编码范例 1-3 中所示的文件。这些说明告诉您如何为 XML 编码。Sun ONE Studio 5 IDE 提供组件、模块和应用程序的可视化表示。所以您无需编码部署描述符文件，只需使用表示组件、模块和应用程序的资源管理器窗口节点。

在 IDE 中进行组装时，资源管理器为要创建的模块或应用程序创建可视化表示。表示应用程序的节点包含应用程序中模块的子节点，表示模块的节点包含模块中组件的子节点。在使用可视化表示时， IDE 会创建一个与其匹配的部署描述符。

每个节点都有一个属性表单，允许您配置节点所表示的组件、模块或应用程序。多数属性会映射到部署描述符标记。（不过属性的数量多过部署描述符标记。）通过设置部署的属性，您可以为其配置组件、模块或应用程序，而 IDE 会将标记增加到它的部署描述符中。这些标记标识应用程序服务器所需的服务。

以下几节介绍了模块和应用程序的 IDE 可视化表示。

Web 模块

Web 模块有标准的目录结构 （更多信息请参阅 《构造 Web 组件》），该结构在资源管理器窗口中表示。图 1-2 显示了资源管理器中的 web 模块，web 模块的节点和子节点表示模块中单独的目录和文件。

Web 模块的顶层节点表示 web 模块的顶层目录。要使 IDE 能够将目录识别为 web 模块，您就必须将目录安装为资源管理器窗口文件系统。如果将资源管理器中的 web 模块目录直接安装为另一文件系统的子目录，IDE 就不会将 web 模块目录识别为 web 模块。

顶层节点有 WEB-INF 目录的子节点，WEB-INF 目录有库子目录和类子目录的子节点，其中库子目录用于具有 JAR 文件格式的 web 组件，而类子目录用于任何具有 `.java` 文件格式的 web 组件。 WEB-INF 节点还有表示模块部署描述符文件的 web 子节点。以上就是 web 模块的标准目录结构。

Web 模块还有开发人员所增加的组件和资源的节点。图 1-2 显示了名为 `index.html` 的 HTML 页的节点。 `classes` 目录包含两个 servlet 类（即 `AllItemsServlet` 和 `ItemDetailServlet`）的节点。

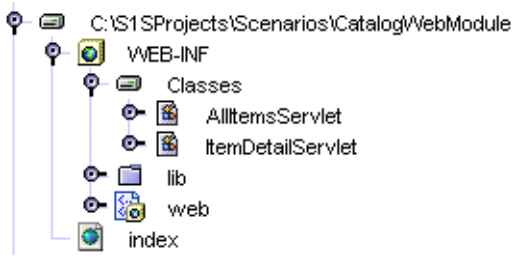

图 1-2 Web 模块的节点和子节点

注意， web 模块的这种表示与特定的目录及其内容相对应。该部署描述符是名为 `web.xml` 的文件，在资源管理器中通过 web 节点表示。而 `web.xml` 文件是 web 模块源代码的一部分。

EJB 模块

EJB 模块的表示方法与 web 模块不同。 EJB 模块的顶层节点不表示特定的目录及其内容。相反， EJB 模块节点却表示模块的部署描述符。 EJB 模块节点的作用是用作企业 bean 列表。这些企业 bean 可以在一个目录下或在不同文件系统的多个目录下，表示部署描述符的顶层节点追踪组件的源代码所在的位置。

用逻辑节点表示 EJB 模块，您就能够将不同目录下的企业 bean 组合到一个 EJB 模块中，而且还可以将部署描述符中的配置信息和源代码分开。在部署 EJB 模块时生成部署描述符 XML 文件，而且在部署描述符中标识的组件源文件被编译到 EJB JAR 文件。

图 1-3 显示了资源管理器窗口中的 EJB 模块。该模块有包括在模块中的三个企业 bean 的子节点。每个企业 bean 都可以在不同的目录中，甚至一个单独的企业 bean 也可能在多个不同的目录中。例如，企业 bean 接口的源代码可能在一个目录下，而实现这些接口的类却可能在另一个不同的目录下。

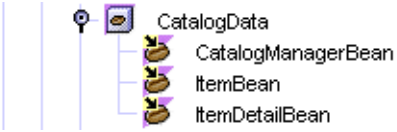

图 1-3 EJB 模块的节点和子节点

J2EE 应用程序

J2EE 应用程序也由逻辑节点表示。像 EJB 模块节点一样，顶层 J2EE 应用程序节点不表示单一的目录或文件系统。相反，应用程序节点表示应用程序的部署描述符，并作为组成应用程序的模块列表。这些模块的源代码可以位于一个以上的目录或文件系统中。

IDE 维护与源代码分开的应用程序级部署描述符。可以在一个以上的 J2EE 应用程序中包含相同的源代码。部署过程将编译所有的源文件，并将编译的版本与 EAR 文件中的部署描述符相关联。

图 1-4 显示了资源管理器中的 J2EE 应用程序。图 1-2 和图 1-3 所示的模块已经增加到应用程序中，这些模块由应用程序节点的子节点表示。

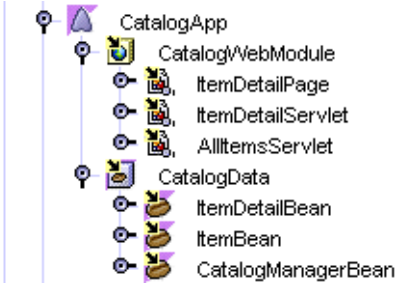

图 1-4 J2EE 应用程序节点及其子节点

属性表单

每个表示 J2EE 模块或应用程序的节点都有属性表单，属性表单包含的某些属性使您能够指定模块或应用程序所需的应用程序服务器中的服务。这些属性与出现在模块或应用程序部署描述符中的标记相对应。在设置属性值时需要提供将在部署描述符中使用的信息。不过您只需使用该属性表单，而不必使用文本编辑器编辑和格式化 XML 部署描述符。

- 对于 web 模块，部署描述符以文件的形式存在，并且该文件出现在资源管理器窗口。请参阅图 1-2，您在 web 模块部署描述符中所指定的值是与源文件相关联的。
- 对于 EJB 模块或 J2EE 应用程序，在部署模块或应用程序之前不会生成部署描述符文件，部署编译源文件并生成一个归档文件和一个 EJB Jar 或 EAR 文件。部署描述符文件就包括在该归档文件中，部署描述符与源文件的一个特定归档副本相关联。

属性具有能够帮助您选择正确值的属性编辑器。打开属性表单时，通过单击属性并单击其省略号 (...) 按钮，打开编辑器中的各个属性。而使用属性编辑器的过程大不相同：有些用于浏览 IDE 节点，有些却用于打开其它级别的对话框，还有很多其它情况。要了解更复杂的属性编辑器，请使用联机帮助。

属性表单有许多标签，名为 [属性] 的标签列出了由 J2EE 规范定义的标准属性。其它标签的名称是应用程序服务器产品的名称。这些标签搜集了 J2EE 规范定义的信息之外的其它信息，但这些信息却是特定应用程序服务器产品所必须的。这些标签是属性表单的特定服务器标签。组装模块或应用程序时，需要使用标准属性和使用的服务器产品的特定服务器属性。

部署的基础知识

部署是编译组成 J2EE 应用程序的源文件并将编译的文件安装到由 J2EE 应用程序服务器管理的目录中的过程。部署是由应用程序服务器或与服务器一起发布的软件 （如部署工具或管理工具）来执行的。

IDE 与应用程序服务器进行通讯。它提供将模块或应用程序部署到应用程序服务器并执行模块或应用程序的命令。您可以在不离开 IDE 的情况下进行部署和执行。部署由插件或应用程序服务器软件执行，但是执行是发生在应用程序服务器的环境下，而对部署和执行的管理却是在 IDE 中。

在执行并测试应用程序之后，您就可以修改源代码，重新部署应用程序和执行新的版本。该过程能够一直持续，直到您不再需要进行为止。

部署已组装的应用程序的实际过程是非常简单的。右键单击应用程序节点并选择 [部署] 命令。但是，在部署之前，必须设置 IDE 使其与应用程序服务器一起工作。下表概括了设置应用程序服务器的步骤：

1. 安装应用程序服务器。
2. 安装能够实现 IDE 和应用程序服务器之间通讯的服务器插件， Sun ONE 应用程序服务器 7， Standard Edition 的插件随 IDE 一起安装。插件也可用于其它广泛使用的应用程序服务器产品。安装插件会在资源管理器 [运行环境] 标签下创建应用程序服务器节点。（要了解关于插件的更多信息，请参阅附录 A。）
3. 使用应用程序服务器节点的菜单命令，建立 IDE 与应用程序服务器之间的通讯。
4. 使用应用程序的属性表单，指定要部署的应用程序的目标应用程序服务器。
5. 右键单击应用程序节点并选择 [部署] 命令。部署软件读取由应用程序节点所表示的部署描述符，然后部署在部署描述符中列出的源文件。

IDE 的大多数安装包括了 Sun ONE 应用程序服务器 7，安装程序会自动配置 IDE 以便使其与应用程序服务器一起工作。 *Sun ONE Studio 5, Standard Edition Getting Started Guide* 对安装进行了详细的解释。

执行的基础知识

部署完应用程序后，您就可以执行该应用程序了。 IDE 将表示应用程序的资源管理器节点与应用程序的部署副本相关联。可以右键单击该节点并选择 [执行] 命令，然后 IDE 就会指示应用程序服务器执行应用程序的部署副本。

许多 IDE 节点，其中包括 J2EE 应用程序节点，都有 [执行] 命令。如果需要，右键单击一个节点并选择 [执行]，IDE 会部署并执行应用程序。[执行] 命令的结果取决于应用程序的类型。例如，如果应用程序包括 web 模块，那么 [执行] 命令就会启动 web 浏览器并打开该应用程序的 URL。

您还可以完全在应用程序服务器环境下执行部署的应用程序而无需使用 IDE。例如，要执行一个包括 web 模块的已部署应用程序，请启动 web 浏览器并打开应用程序中的一个 web 页。

使用本书

Sun Microsystems, Inc 支持的 Java Community Process 发展了用 J2EE 组件设计分布式企业应用程序的标准。第 10 页的 “阅读本书须知”中所列的 J2EE 平台文档涵盖了应用程序设计和体系结构的这些标准。

本书介绍了如何使用 Sun ONE Studio 5 IDE 来实现这些体系结构。使用 IDE 来组合组件并创建 J2EE 模块，确定所有组件都按照应用程序设计所指定的方式进行交互操作。此外，还组合 J2EE 模块来创建 J2EE 应用程序，确定模块间的分布式交互操作按照应用程序设计所调用的方式进行。

本书通过展示多个示例或方案介绍了组装。每个方案都展示了组件或模块的真实组合，并示范了将它们组装到模块或应用程序中的方法。在方案中所描述的业务问题都是真实的，但这不意味着本书是设计 J2EE 应用程序的完全指南。

这些方案的真正目的是说明如何对组件和模块之间的特定类型的交互操作进行编程。一旦您决定了应用程序的设计，就可以使用本书中的方案来帮助您对组件和模块之间的交互操作进行编程。

您可能无法在某一个方案中找到所需的全部内容，因为每个方案只着眼于一或两种 J2EE 交互操作，而实际的 J2EE 应用程序却可能包括大量组件和交互操作。不过，您却可以在本书中找到每种交互操作的示例。

例如，要编程一个具有 web 模块和 EJB 模块的普通类型的 J2EE 应用程序，您就可以查看第 2 章（包含 web 模块的组装），第 3 章（包含 EJB 模块的组装）和第 4 章（包含将 web 模块和 EJB 模块组装到 J2EE 应用程序中的内容）。如何设置事务请阅读第 6 章，安全性的内容则请阅读第 7 章。

本书可以作为使用 Sun ONE Studio 5 开发环境开发分布式企业应用程序的指南。并且说明了如何开发 J2EE 模块，如何为不同类型的交互操作编程模块，如何从 J2EE 平台请求企业服务 （例如安全性检查和事务管理）。

第 2 章

方案：Web 模块

图 2-1 显示了 web 模块和该模块参与的交互操作。该模块是 J2EE 应用程序的一部分，图中所示的交互操作是 J2EE 应用程序中 web 模块的典型操作。该 web 模块与 HTTP 连接 （图中通过标为 1 的箭头表示）上的用户和 EJB 模块 （通过标为 3 的箭头表示）所提供的中间层服务进行交互。在该 web 模块中， web 组件相互间进行交互 （通过标为 2 箭头的表示）。本章描述了该 web 模块，并解释了如何编程图 2-1 所示的交互操作。

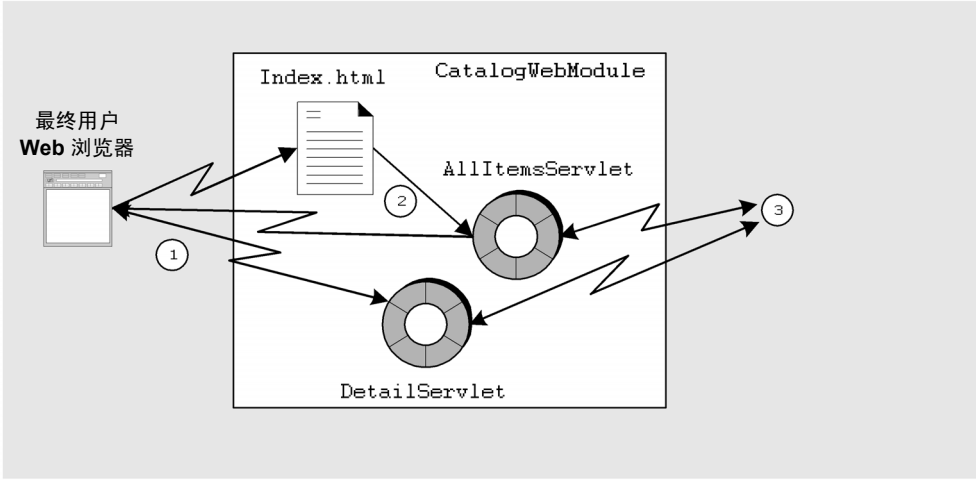

图 2-1 CatalogWebModule web 模块

本模块中的交互操作

本方案描述了图 2-1 中所示的 web 模块和交互操作的用法之一。在本方案中，该模块是支持某零售 web 站点的 J2EE 应用程序的前端。该 web 模块包含访问该站点的联机顾客所见到的 web 页面，对这些页面中的顾客输入信息进行处理。显示页面并处理输入信息是 J2EE 应用程序中 web 模块的典型任务。

从顾客的观点来看，该应用程序是一系列的 web 页面。顾客使用 web 浏览器来查看应用程序的主页，并使用出现在 web 页面上的文本字段、按钮和其它控件来进行输入。从开发人员的观点看，应用程序是一组 web 组件，它们接收 HTTP 请求并返回 HTTP 和响应。

本方案中的 web 模块仅包含了几个组件，且只处理两个不同的请求。尽管很简单，但是这些组件和请求说明了 web 组件如何提供用户、 web 模块和 EJB 模块间所需的交互操作。在本方案中涵盖的特定交互操作概述如下：

1. 联机顾客启动 web 浏览器并打开应用程序的根 URL，然后打开到应用程序的连接。该操作将打开应用程序的主页。实际的购物站点主页会显示出许多选项，其中包括请求按种类显示项目、请求关键字搜索、请求实时客户服务信息等等。而在这个简单的示例中，主页仅显示一个选项，该选项是到另一个显示全部目录的页面的链接。
2. 顾客单击链接。该操作生成一个请求，该请求由其中名为 `AllItemsServlet` 的 servlet 处理。 `AllItemsServlet` 通过调用 EJB 模块的业务方法（`getAllItems` 方法）来处理该请求，该方法返回数据。
3. `AllItemsServlet` 将从 EJB 模块返回的数据准备好并显示在顾客的浏览器中。Servlet 将从 EJB 模块返回的独立字段值与格式化该字段值的 HTML 标记进行组合。`AllItemsServlet` 将这个字段值和 HTML 标记的组合写入返回到用户 web 浏览器的输出流。顾客的 web 浏览器处理 HTML 输出并显示目录。在目录的显示中，每个项目的名称都显示为到项目详细信息的 HTML 链接。
4. 顾客浏览显示的目录并单击其中一个链接。该请求由另一个名为 `DetailServlet` 的 servlet 处理。 `DetailServlet` 调用 EJB 模块的另一个业务方法（`getOneItemDetail`）来获取该项目的详细信息。
5. `DetailServlet` 处理从 EJB 模块返回的数据，用于显示在顾客的浏览器中。如同 `AllItemsServlet`， `DetailServlet` 准备一个组合字段值和 HTML 标记的输出流。顾客的 web 浏览器处理该输出流并将其显示为另一个 web 页面。

由 `AllItemsServlet` 和 `DetailServlet` 写入的 HTML 输出只含有文本。尽管很简单，但是这些示例说明了 web 组件是如何处理 HTTP 请求的：通过使用远程方法调用从 EJB 模块获取数据，再将数据写入 HTML 输出流。一个经验丰富的 web 设计人员或 web 程序员能够使用这种操作编写更复杂的输出。

Web 模块和 EJB 模块之间的交互操作是通过 Java RMI 方法调用实现的，而 Java RMI 是 EJB 模块设计所必须的。关于 EJB 模块设计的更多信息，请参阅第 52 页的 “本模块中的交互操作”。

关于创建 web 组件、在 web 组件中编写企业业务逻辑以及类似任务的指示，请参阅《构造 Web 组件》。

本模块的编程

表 2-1 总结了用于创建上节所述并在图 2-1 中图示的 web 模块的编程方法。

表 2-1 本方案所需的编程方法

应用程序元素	所需的编程方法
应用程序服务器	无。
Web 模块	创建 web 模块。 创建欢迎页面：`index.html`。该页面包括一个执行 servlet `AllItemsServlet` 的 HTML 链接。 创建两个 servlet：`AllItemsServlet` 和 `ItemDetailServlet`。 编码处理 HTTP 请求并生成 HTTP 响应的 `processRequest` 方法。这些 `processRequest` 方法是： 1. 执行 “Java 命名和目录接口” (JNDI) 查找，以调用 EJB 模块业务方法。 2. 将从 EJB 模块获取的数据写入 servlet 的响应中。 由 `AllItemsServlet` 输出的 HTML 页面包含一个到 `ItemDetailServlet` 的 HTML 链接。 为每个 servlet 设置 URL 模式。
J2EE 应用程序	关于将 web 模块增加到 J2EE 应用程序的指示，请参阅第 4 章。

后续的几节说明了如何执行这些编程任务中的大部分任务。其中方法签名指定了每个交互操作的输入和输出。关于过程的几节说明了如何将这些输入和输出连接到其它组件和模块，但是不包括关于创建 web 模块或将 web 组件增加到模块的指示。（要了解这些任务的信息，请参阅联机帮助或 《构造 Web 组件》。）

创建欢迎页面

零售 web 站点的设计要求顾客在主页上开始与站点的交互操作。主页是一个典型的 web 站点功能，它为用户提供了入口点，该入口点可以标识站点并向客户显示可用的选项。顾客查看主页并选择要使用的功能。

首先顾客要启动浏览器并打开购物站点的 URL。在 web 模块内部，该 URL 已经被映射到应用程序的上下文根。如果需要在用户打开站点的 URL 时应用程序始终显示该主页，那么请将该主页标识为欢迎页面。

创建 HTML 页面

在资源管理器窗口的节点分层结构中，欢迎页面的 HTML 文件必须与 web 模块的 WEB_INF 节点在同一级别。请参阅图 1-2 中的示例。

要在正确的级别创建 HTML 文件：

1. **右键单击包含绿色 WEB-INF 节点的文件系统节点，并选择 [新建] →[所有模板]。**

   [新建] 向导的 [选择模板] 页打开。

2. **选择 [HTML 文件] 模板。**

   a. **在 [选择模板] 字段，展开 JSP 和 Servlet 节点并选择 [HTML 文件] 节点。**

   b. **单击 [下一步]**

      [新建] 向导的 [新建对象名称] 页面打开。

3. **在名称字段中输入 `index`，然后单击 [完成]。**

   IDE 创建新的 HTML 文件并在资源管理器中使用名为 `index` 的新节点表示。这时 [源编辑器] 打开，同时新文件中出现光标。

4. **将 HTML 代码输入欢迎页面。**

   编码范例 2-1 显示了本方案中使用的简单欢迎页面的 HTML 代码。

编码范例 2-1 目录显示模块的欢迎页面

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
  <head>
    <title>Online Catalog</title>
  </head>
  <body>
<h2>
Inventory List Application
</h2>

<p>
<a href="allItems">Display the Catalog
</a>

</body>
</html>
```

该欢迎页面只向用户显示一个选项，即名为 `Display the Catalog` 的文本链接。该链接使用 URL 模式来指定模块中的其中一个 servlet：`AllItemsServlet`。用户单击该链接时，浏览器向 web 模块发送另一个请求。该请求通过执行 `AllItemsServlet` 处理。用户看到的下一页是由 `AllItemsServlet` 输出的页面。要查看这个由 `AllItemsServlet` 输出的页面，请参考编码范例 2-2。

虽然实际 web 站点的欢迎页面包括许多具有不同功能的链接，但是每个链接都遵从在该示例中演示的原则。显示给用户的页面包含生成 HTTP 请求的链接或操作。每个请求都由 web 模块中的某些组件处理，web 模块通过写入另一个供用户查看的页面进行响应。

在该示例中，链接指定了一个 servlet，而没有指定 servlet 的方法。当请求只指定一个 servlet 名称时，缺省操作是执行 servlet 的 `doGet` 方法。您也可以编写指定 servlet 其它方法之一的链接，如 `doPost`。关于 servlet 方法的更多信息，请参阅《构造 Web 组件》。

将页面标识为模块的欢迎页面

在 IDE 中创建 web 模块时，模块具有列出欢迎页面文件中缺省名称的 [欢迎文件] 属性。图 2-2 显示了具有缺省名称的 [欢迎文件] 属性编辑器。用户访问应用程序的根 URL 时，应用程序服务器在模块目录中搜索具有这些缺省名称的文件。搜索到的第一个文件被显示为欢迎页面。

为模块创建欢迎文件的最简单方法是创建一个具有其中一个缺省名称的文件。例如，在本方案中，您就创建了一个名为 `index.html` 的文件。

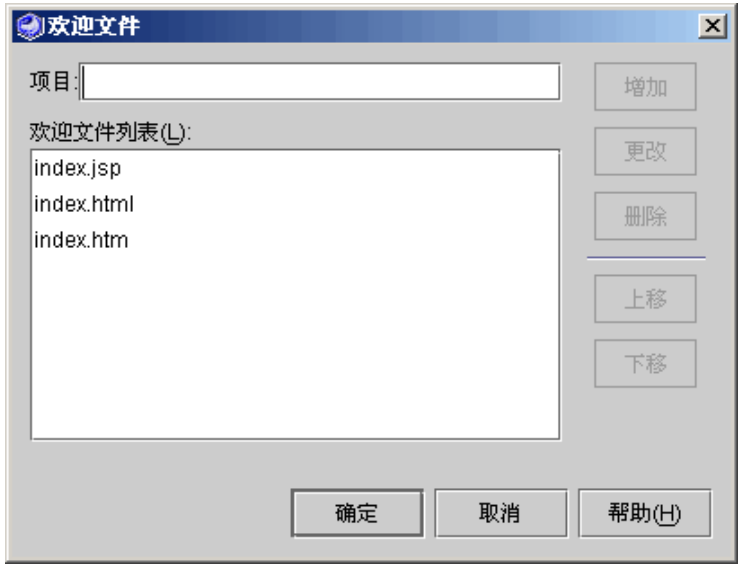

图 2-2 [欢迎文件] 属性编辑器

如果要使模块的欢迎页面使用一个不同名称的文件，请打开 [欢迎文件] 属性编辑器，增加要使用的文件的名称。

Servlet 方法的编程

本方案中的 servlet 通过调用企业 bean 方法从 EJB 模块获取数据。对于所调用的方法，有两个编程任务：在调用 servlet 时，为调用的企业 bean 编码调用方法并设置引用声明。下面首先说明方法体代码的编码。

注 – 该示例中的 servlet 是由 IDE 的 servlet 模板创建的。HttpServlet 是由该模板创建的 servlet，它们包含名为 `processRequest` 的方法。 `doGet` 和 `doPost` 方法都调用 `processRequest`，因此您可以将处理请求的代码增加到 `processRequest` 方法。

方法体

编码范例 2-2 显示了如何在 `AllItemsServlet` 中实现 `processRequest` 方法。当用户单击出现在欢迎页面上的 `Display the Catalog` 链接时，该方法开始执行。

欢迎页面上的 URL 模式将 servlet 命名为 `AllItemsServlet`，但是 URL 模式不指定方法。在这种情况下，应用程序服务器执行缺省操作和 servlet 的 `doGet` 方法。`doGet` 方法调用 `processRequest` 方法。

编码范例 2-2 显示了 `AllItemsServlet` 是如何从 EJB 模块获取目录数据并将其显示给用户的，其中有三个步骤：

1. `AllItemsServlet` 使用 JNDI 查找来获取对 `CatalogData` EJB 模块中的会话企业 bean 的远程引用。关于 `CatalogData` 模块的更多信息，请参阅第 3 章。
2. `AllItemsServlet` 调用会话 bean 的业务方法 （`getAllItems` 方法）。
3. Servlet 将远程方法调用返回的数据写入 HTML 输出流中，然后输出流返回到用户的浏览器窗口。

编码范例 **2-2** Servlet `AllItemsServlet` 的 `processRequest` 方法

```
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
                              throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  /* 在此处输出页面 */
  out.println("<html>");
  out.println("<head>");
  out.println("<title>AllItemsServlet</title>");
  out.println("</head>");
  out.println("<body>");

  out.println("<h2>The Inventory List</h2>");

  out.println("<table>");
  out.println("<tr>");
  out.println("<td>Item");
  out.println("<td>Item SKU");
  out.println("<td>Detail");

  CatalogBeans.CatalogManagerBeanHome catHome;
  CatalogBeans.CatalogManagerBean catRemote;

  try {
    InitialContext ic = new InitialContext();
    Object objref  = ic.lookup("java:comp/env/ejb/CatalogManagerBean");
    catHome = (CatalogBeans.CatalogManagerBeanHome) objref;
    catRemote = catHome.create();

    java.util.Vector allItems = catRemote.getAllItems();

    Iterator i = allItems.iterator();
    while (i.hasNext()) {

      CatalogBeans.iDetail itemDetail = (CatalogBeans.iDetail)i.next();
      out.println("<tr>" +
        "<td>" +
        itemDetail.getItemname() +

        "<td>" +
        itemDetail.getItemsku() +

        "<td>" +
        "<a href=\"" + response.encodeURL("itemDetail?sku=" +
        itemDetail.getItemsku()) +
        "\"> " +
        "Get Item Detail" +
```

编码范例 **2-2** Servlet `AllItemsServlet` 的 `processRequest` 方法 *（后续）*

```
          "</a>");
      }
    }
    catch (javax.naming.NamingException nE) {
      System.out.println("Naming Exception on Lookup" + nE.toString());
    }
    catch (javax.ejb.CreateException cE) {
      System.out.println("CreateException" + cE.toString());
    }
    catch (java.rmi.RemoteException rE) {
      System.out.println("RemoteException" + rE.toString());
    }
    catch(Exception e) {
      System.out.println(e.toString());
    }

    out.println("</table>");


    out.println("</body>");
    out.println("</html>");

    out.close();
  }
```

`lookup` 语句指定 `CatalogManagerBean`，但实际上该字符串是引用的名称，而不是所引用的企业 bean。为了便于记住所指的是哪个 bean，企业 bean 名称经常作为引用名称使用。实际上，引用的企业 bean 在引用声明中指定，该内容将在下一节提及。

EJB 引用的引用声明

如同 `AllItemsServlet`， web 组件通过 EJB 引用在 EJB 模块中调用企业 bean 的方法。下面列出了 EJB 引用的两个部分：

- **JNDI 查找代码**，使用 JNDI 命名工具来获取对已命名的企业 bean 的远程引用。
- **引用声明**，部署描述符的一部分。引用声明告诉应用程序服务器在查看代码中引用了哪个特定的 bean。

在本方案中，查看代码位于 `AllItemsServlet` 的 `processRequest` 方法中。（请参阅编码范例 2-2。）您可以编译该代码，但是没有引用声明就无法在运行环境返回引用。引用声明将用于查找语句中的引用名称映射到企业 bean 的实际名称中。

要设置 web 模块的 EJB 引用声明：

1. **右键单击 web 模块的 web 节点并选择 [属性] →[引用] 标签 →[EJB 引用] → 省略号 (...) 按钮。**

   [EJB 引用] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加 EJB 引用] 对话框打开。使用该对话框来设置引用声明。

3. **要设置引用声明，请输入用于 `lookup` 语句的引用名称和用于方法调用的起始接口和远程接口的名称。**

   图 2-3 显示了在字段中具有这些值的 [增加 EJB 引用] 对话框。如果要使引用在运行环境下工作，或使 JNDI 查找将远程引用返回到企业 bean，就必须将引用链接到同一应用程序的特定企业 bean 中。在部署并执行应用程序之前必须链接该引用，但此刻是不需要链接的。

   在开发过程中遇到此情况时，您可以先保持引用的未链接状态，然后在将 web 模块组装到 J2EE 应用程序之后再进行链接。在某些情况下可能会选择在开发的这个阶段解决引用，您应该考虑以下几个条件：

   - 如果企业 bean 不在您的开发环境中，那么就不能链接该引用。提供接口的名称并单击 [确定]。直到您具有了开发环境中出现的接口的副本，才能够编译 JNDI 查找代码。

     图 2-3 显示了设置未链接引用的 [增加 EJB 引用] 对话框。 [起始接口] 和 [远程接口] 被指定，但 [引用的 EJB 名称] 字段为空。以后将在应用程序属性表单上链接该引用。

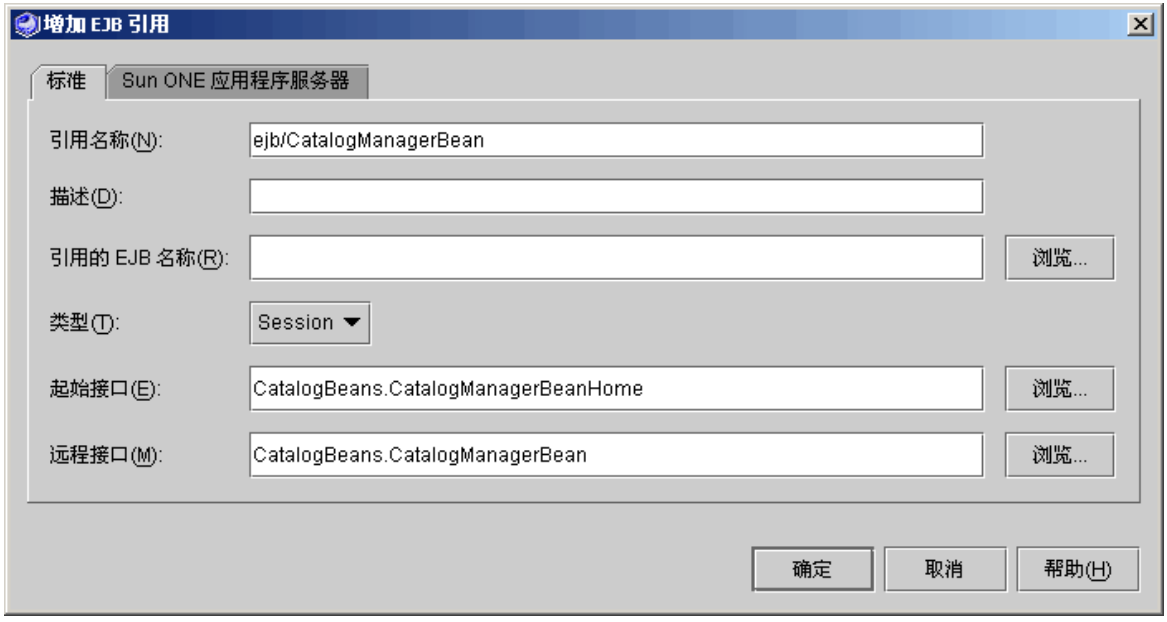

图 2-3 具有未链接引用的 [EJB 引用] 属性编辑器

- 如果企业 bean 在您的开发环境中可用，那么此时就能够链接引用。单击 [引用的 EJB 名称] 字段旁的 [浏览] 按钮。使用出现的对话框来选择调用的企业 bean，最后单击 [确定]。

  图 2-4 显示了链接到 `CatalogManagerBean` 的名为 `ejb/CatalogManagerBean` 的引用。

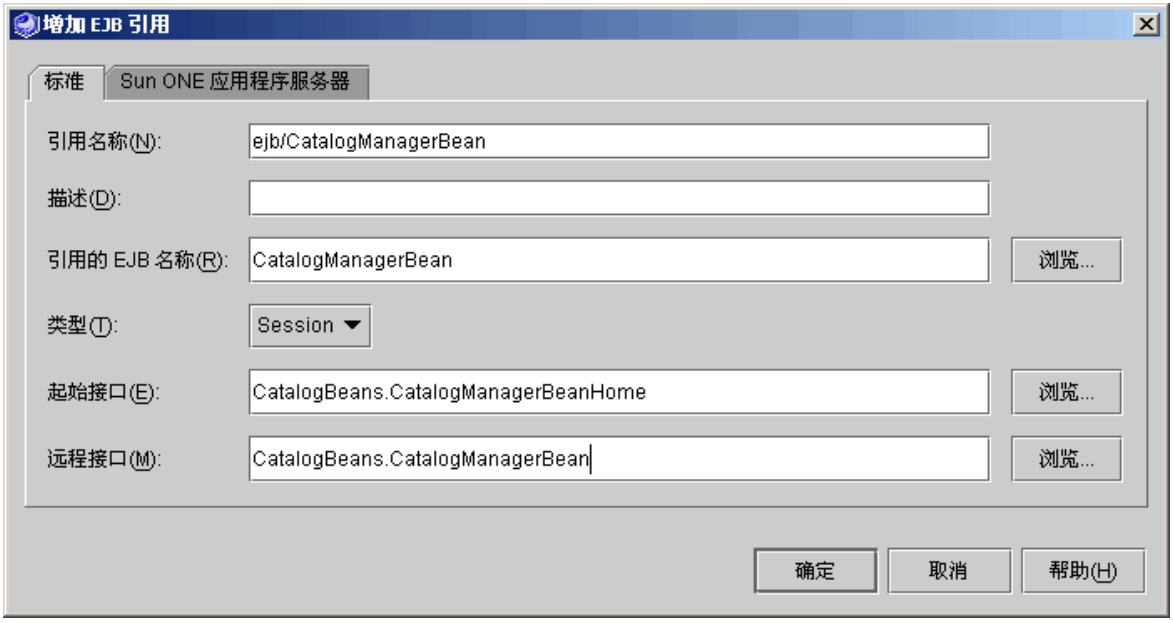

图 2-4 具有链接引用的 [EJB 引用] 属性编辑器

即使调用的企业 bean 可用，您也可能选择此时不链接引用。这其中有很多原因，例如，如果 web 组件有机会用于多个应用程序，则不需要在 web 模块的属性表单上链接该引用。开发人员重新使用 web 模块，能够将该引用重新链接到可以实现相同接口的某些其它企业 bean。重新链接该引用会改变 `web.xml` 部署描述符文件中的值，同时也影响源代码的使用。

如图 2-3 所示，在本方案中引用保持未链接状态。在创建了 J2EE 应用程序后，引用在应用程序节点的属性表单中链接。请参阅第 72 页的 “链接 EJB 引用”。

将 URL 映射到 Servlet

在欢迎页面上设置的链接使用以下 HTML 标记格式化：

```
<a href="allItems">Display the Catalog</a>
```

当顾客单击该链接时，应用程序服务器将 HTML 标记中的 URL 模式附加到 URL 路径，然后执行所产生的 URL。要使链接正确工作，应用程序服务器所生成的 URL 必须映射到要执行的 servlet。下一节解释如何将 URL 映射到 servlet。

了解 Servlet 映射

在部署的 web 模块中， servlet 的 URL 和 web 模块中的其它 web 资源是将名称附加到 URL 路径的结果。对于部署到 Sun ONE 应用程序服务器 7 的模块，URL 路径有以下的通用形式：

`http://`*hostname*`:`*port*`/`*web-context*`/`*URL-pattern*

该路径中的元素按以下方式决定：

- hostname 是正在运行应用程序服务器的机器名， port 是为该服务器示例的 HTTP 请求指定的端口。端口号在安装应用程序服务器时分配。
- Web 上下文是在将模块增加到 J2EE 应用程序*之后*，指定为 web 模块的属性的字符串。 Web 上下文限定模块中所有的 web 资源。
- URL 模式是标识特定 servlet 或 JSP 页面的字符串。在 web 模块属性表单上设置 URL 模式。将 web 模块组装到 J2EE 应用程序中前可以分配 URL 模式。

换句话说，在 web 模块中分配的 URL 模式与以后将模块增加到 J2EE 应用程序时所要分配的 web 上下文是相对的。本方案中的 URL 使用了使它们与 web 上下文相对的格式。所以无论您在组装应用程序时提供什么样的 web 上下文，这些链接还是会在应用程序执行时正确工作。关于设置 web 上下文的信息，请参阅第 71 页的 “设置 web 模块的 web 上下文”。

检查缺省 Servlet 映射

在缺省情况下创建 servlet 时，Servlet 向导会使用您在 Servlet 向导首页为 servlet 名称所提供的类名，并将 servlet 名称映射到包括该 servlet 名称的 URL 模式中。图 2-5 显示了具有 `AllItemsServlet` 缺省设置的 web 模块的 [Servlet 映射] 属性编辑器。

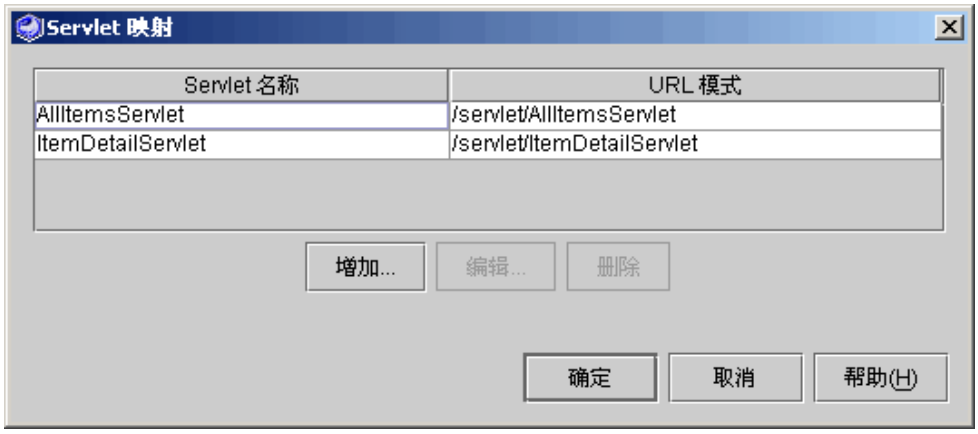

图 2-5 [Servlet 映射] 属性编辑器

如果使用该 servlet 映射部署 web 模块， `AllItemsServlet` 就映射到以下 URL：

`http://`*hostname*`:`*port*`/`*web-context*`/servlet/AllItemsServlet`

编辑 Servlet 映射

如果要将一个不同的 URL 映射到 servlet，那么请使用 servlet 映射属性编辑器编辑该映射。在本方案中，您将 URL 模式从缺省值更改为更具有特定意义的值。

要编辑 servlet 映射：

1. **右键单击 web 模块的 web 节点，并选择 [属性] →[Servlet 映射] →省略号 (...) 按钮。**

   [Servlet 映射] 属性编辑器打开。[Servlet 映射] 属性编辑器列出模块中的所有 servlet 和为它们设置的映射。

2. **选择 `allItemsServlet` 的当前映射，然后单击 [编辑] 按钮。**

   [编辑 Servlet 映射] 对话框打开。

3. **在 [URL 模式] 字段中，输入 `/allItems`，单击 [确定]。**

图 2-6 显示了具有 `AllItemsServlet` 和 `DetailServlet` 新映射的 [Servlet 映射] 属性编辑器。

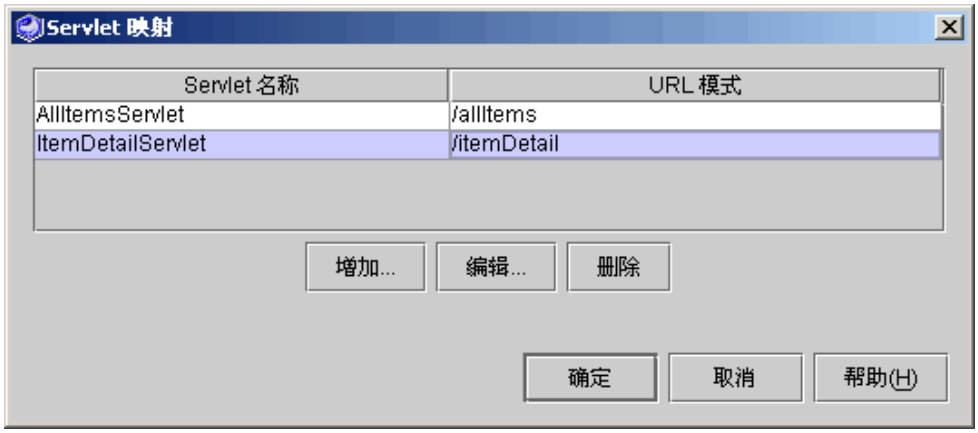

图 2-6 [Servlet 映射] 属性编辑器

编辑该 servlet 映射之后，使用以下 URL 就可以执行 `AllItemsServlet`：

`http://`*hostname*`:`*port*`/`*web-context*`/allItems`

注意，新的 URL 模式是在 HTML 标记中所用的字符串，该 HTML 标记创建与欢迎页面的链接 （请参阅编码范例 2-1）。单击链接，立即执行 `AllItemsServlet`。

其它组装任务

上述几节包含了组装 `CatalogWebModule` 所需的组装任务。本节包含 `CatalogWebModule` 方案中不需要的 web 模块组装任务。

Web 模块可能需要您执行这些其它组装任务中的某些任务。本节包含了可以执行的多个 web 模块组装任务。

设置错误页面

如果要为 web 模块指定错误页面，那么就需要在模块的部署描述符中标识该错误页面。在 [错误页面] 属性编辑器中操作。

要为 web 模块设置错误页面：

1. **右键单击 web 节点，并选择 [属性] →[部署] 标签 →[错误页面] →省略号 (...) 按钮。**
   [错误页面] 属性编辑器打开。
2. **使用 HTTP 错误代码或 Java 异常类标识错误，并将它映射到特定的错误页面。**
   您可以使用 HTTP 错误页面或 Java 异常类来标识错误。注意，编辑器有两个 [增加] 按钮，每个按钮对应一种错误。无论是哪种类型，您都要指定错误并将错误映射到页面。图 2-7 显示了 HTTP 错误代码 404 被映射到特定的错误页面之后的属性编辑器。

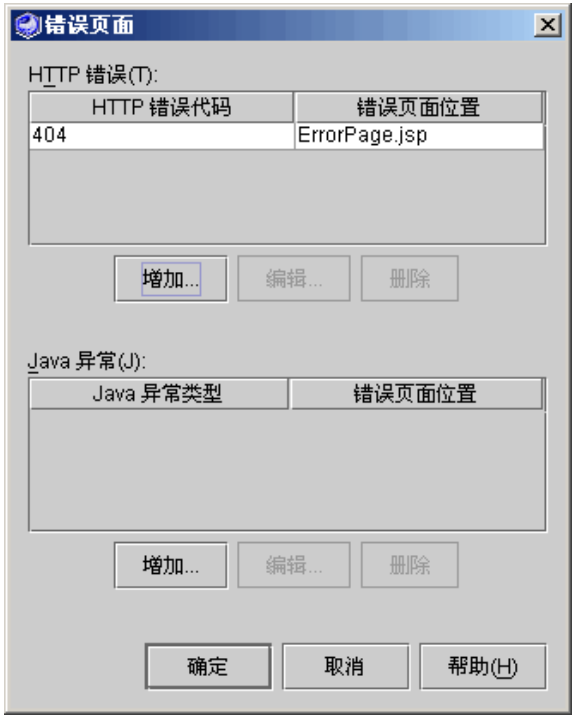

图 2-7 [错误页面] 属性编辑器

设置 JSP 页面

如果您要组装的 web 模块包含 JSP 页面组件，那么有多种方法执行这些组件。如果您创建的是名为 `myJsp` 的新 JSP 页面，那么可以使用以下任何一种方法来执行该页面。

使用 HTML 链接执行 JSP 页面

如果要从 HTML 链接执行 JSP 页面，请按照以下示例设置链接：

```
<a href="myJsp.jsp">Execute myJsp</a>
```

通过程序执行 JSP 页面

如果您使用 IDE 创建 JSP 页面，那么就没有为它创建应用程序描述符条目。而如果业务逻辑通过程序访问 JSP 页面，就不需要部署描述符条目。例如，下面的代码位于执行 myJsp 的 servlet 中。注意，该代码通过提供其实际文件名称（myJsp.jsp）来标识要执行的 JSP 页面。

编码范例 2-3 通过程序执行 JSP 页面的代码

```
...
response.setContentType("text/html");
        RequestDispatcher dispatcher;
        dispatcher = getServletContext().getRequestDispatcher ("/myJsp.jsp");
        dispatcher.include(request, response);
...
```

使用 URL 到 JSP 的映射

前面的示例是通过指定实际文件名称（myJsp.jsp）来执行 JSP 页面的。也可以将 URL 模式映射到 JSP 页面，然后引用到该页面的 URL 模式来执行页面。完成该过程需要两个步骤，首先是为 JSP 文件设置 servlet 名称。

要设置 JSP 页面的 URL 映射：

1. **右键单击 web 节点，并选择 [属性] →[部署] 标签 →[JSP 文件] →省略号 (...) 按钮。**

   [JSP 文件] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加 JSP 文件] 对话框打开。

   a. 在 [JSP 文件] 字段中，输入要设置的 JSP 文件的文件名称。

   b. 在 [Servlet 名称] 字段中，输入要映射到 JSP 文件的 servlet 名称。

c. 单击 **[确定]**。 **[增加 JSP 文件]** 对话框关闭。

图 2-8 显示了将名为 `itemDetailPage` 的 servlet 映射到 `myJsp.jsp` 文件之后的 [JSP 文件] 属性编辑器。

图 2-8 [JSP 文件] 属性编辑器

3. 再次单击 **[确定]**，关闭 **[JSP 文件]** 属性编辑器并返回属性表单。
4. 仍在属性表单中，右键单击 **[Servlet 映射]** 属性并单击省略号 **(...)** 按钮。

   [Servlet 映射] 属性编辑器打开。
5. 单击 **[增加]** 按钮。

   [增加 Servlet 映射] 对话框打开。
6. 在 **[增加 Servlet 映射]** 对话框中，将 **URL** 模式映射到新 **servlet** 名称中。

   a. 在 **[Servlet 名称]** 字段中，输入已映射到 **JSP** 文件的 **servlet** 名称。

   b. 在 **[URL 模式]** 字段中，输入要映射到 **servlet** 名称，最后是 **JSP** 文件的 **URL** 模式。

c. 单击 [确定]。 [增加 Servlet 映射] 对话框关闭。

图 2-9 显示了 [Servlet 映射] 属性编辑器，其中具有映射到 servlet 名称 `ItemDetailPage` 的 URL 模式 `ItemDetail`。 Servlet 名称 `ItemDetailPage` 已被映射到 `myJsp.jsp` JSP 文件中。

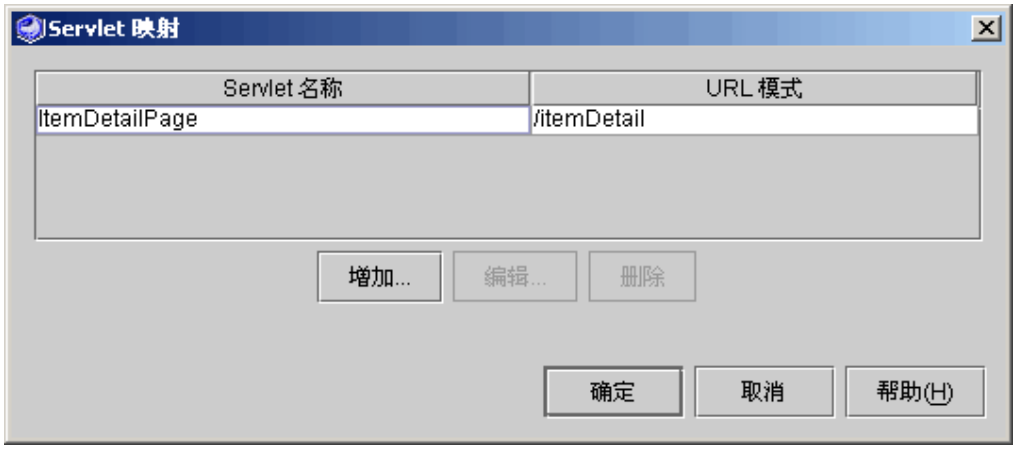

图 2-9 [Servlet 映射] 属性编辑器

d. 再次单击 [确定]，关闭 [Servlet 映射] 属性编辑器。

映射完之后，使用以下 URL 就可以执行由 `myJSP.jsp` 文件定义的 JSP 页面：

`http://`*hostname*`:`*port*`/`*web-context*`/ItemDetail`

设置环境条目引用

环境条目分为两个部分：

- **JNDI 查找。**使用环境条目的 web 组件通过 JNDI 名称工具来查找条目的值。
- **在 web 模块部署描述符中的引用声明。**引用声明向应用程序服务器声明该引用，部署描述符还包含环境条目的值。

环境条目引用的 JNDI 查找

使用环境条目值的 web 组件需要如编码范例 2-4 所示代码的查找代码。

编码范例 2-4 环境条目的查找语句

```
try {
       // 获取初始上下文 -- 打开与 JNDI 命名的通讯:
       Context ic = new InitialContext();
       // 请求名为 “Cache Size” 环境条目的查找:
       Integer cacheSize = (Integer)
        ic.lookup( ò java:comp/env/NumberOfRecordsCached ò );
}
catch(Exception e) {
       System.out.println(e.toString());
       e.printStackTrace();
       return;
}
```

代码中的注释解释每行的用途。

环境条目的引用声明

要设置环境条目的引用声明：

1. **右键单击 web 节点并选择 [属性] →[引用] 标签 →[环境条目] →省略号 (...) 按钮。**
   [环境条目] 属性编辑器打开。
2. **单击 [增加] 按钮。**
   [增加环境条目] 对话框打开。
3. **声明该环境条目引用。**
   a. **在 [名称] 字段中，输入用于** `lookup` **语句的引用名称。**
   b. **在 [类型] 字段中，选择该环境条目的数据类型。**
   c. **在 [值] 字段中，输入环境条目的初始值。**
      图 2-10 显示了 [增加环境条目] 对话框，该对话框具有与编码范例 2-4 中的 `lookup` 语句相匹配的值。

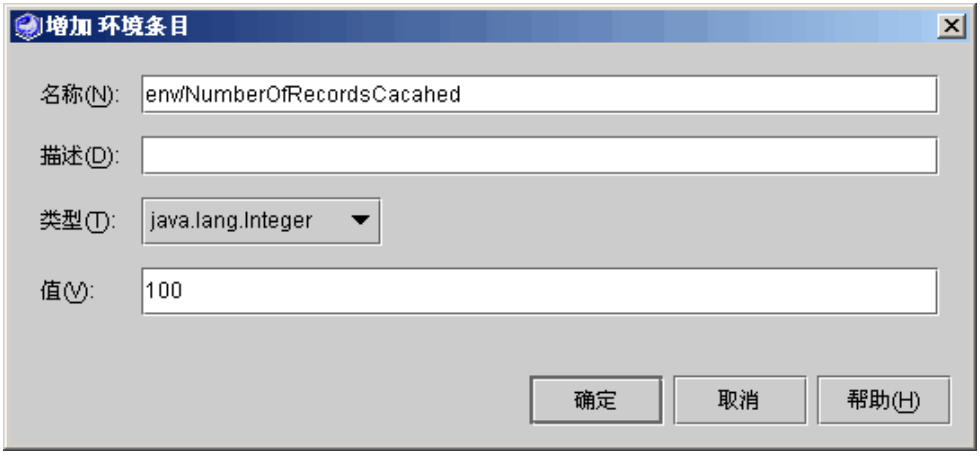

图 2-10 [增加环境条目] 对话框

4. **单击 [确定] 来关闭对话框并处理您的环境条目定义。**

第 3 章

方案：EJB 模块

图 3-1 显示了 EJB 模块和该模块参与的交互操作。该模块是 J2EE 应用程序的一部分，图中的交互操作是 J2EE 应用程序中 EJB 模块的典型操作。EJB 模块与客户端模块交互（该交互操作在图中通过标为 1 的箭头表示），此外 EJB 模块也与外部资源交互，在本例中为关系型数据库管理系统 （该交互操作通过标为 3 的箭头表示）。该 EJB 模块类似于大多数 EJB 模块，包含多个企业 bean，并且这些企业 bean 彼此进行交互（这些交互操作通过标为 2 的箭头表示）。

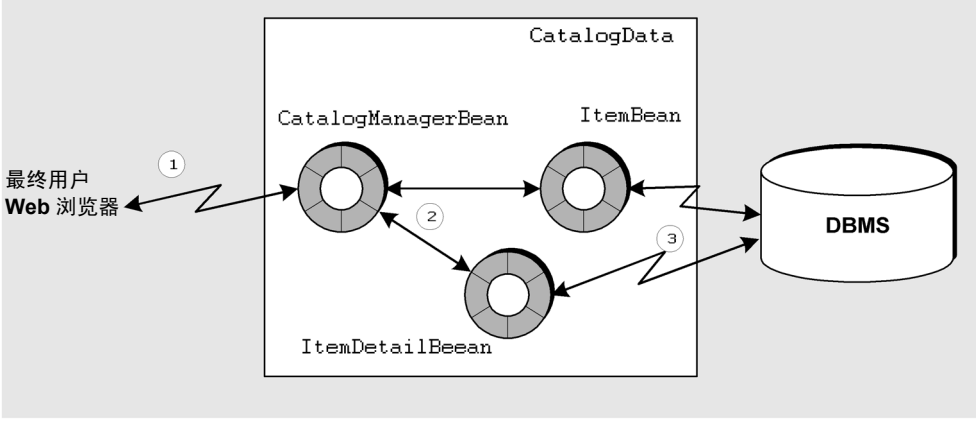

图 3-1 CatalogData EJB 模块

本模块中的交互操作

本方案描述了图 3-1 中所示的 EJB 模块和交互操作的用法之一。在本方案中，EJB 模块是支持一个零售 web 站点的 J2EE 应用程序的后端。该 EJB 模块包含与目录数据库交互的一组企业 bean。作为应用程序前端的 web 模块调用本 EJB 模块的方法，获取向用户显示的数据。在 J2EE 应用程序中，EJB 模块的典型任务是处理方法调用并与数据库交互。

应用程序中 web 模块的功能是作为 `CatalogData` EJB 模块的客户端。顾客使用由 web 模块生成的 web 页面来请求他们要查看的数据，而 web 模块负责调用 `CatalogData` 模块来获取数据，然后格式化数据并向顾客显示该数据。

EJB 模块负责从数据库获取请求的数据并将该数据传递到客户端模块。所以 EJB 模块必须能够处理数据的请求并返回正确数据。本方案中的 EJB 模块仅包含了三个企业 bean 且仅提供了两个业务方法。这些业务方法只能处理两种请求。尽管很简单，但是这些组件和交互操作显示了 EJB 模块如何提供客户端模块和数据库之间所需的交互操作。在本方案中涵盖的特定交互操作概述如下：

1. 客户端模块通过调用 `CatalogData` EJB 模块的方法来请求某些数据。在这个简单的方案中，客户端能够请求目录中所有项的列表，或请求某一项的详细信息。
2. 目录数据 EJB 模块通过生成数据库查询来处理请求，该数据库查询将获取请求的数据。
3. 目录数据 EJB 模块执行查询并将数据返回到客户端模块，然后客户端模块格式化数据并向用户显示该数据。

在客户端模块和 `CatalogData` EJB 模块之间的交互操作决定了选择实现该操作的 J2EE 技术和 `CatalogData` EJB 模块的内部体系结构。您应该考虑以下几点：

- 在客户端模块和 `CatalogData` 模块之间的交互操作是同步的。询问查看目录的联机顾客等待应用程序显示该目录。可以使用 Java RMI 来实现该交互操作以提供同步的交互操作。这意味着客户端模块将通过调用 `CatalogData` EJB 模块的方法来请求目录数据。
- EJB 模块的内部设计由客户端模块和 `CatalogData` EJB 模块之间的交互操作的面向会话的特性决定。联机顾客可以查看整个目录，并且可以请求某一项的详细资料。当多个顾客同时查看目录时，EJB 模块必须将请求匹配到用户会话并将正确数据返回到每个会话。该模块使用单一有态会话企业 bean 来满足这种要求，会话 bean 则管理通过客户端模块提交的所有请求。它通过调用生成数据库查询的实体 bean 的方法来处理请求。

这是 EJB 模块的通用体系结构。（关于用会话 bean 对会话建模的更多信息，请参阅《构建 Enterprise JavaBeans 组件》。）

为了生成数据库查询，J2EE 平台提供了一种称作实体企业 bean 的企业 bean。实体 bean 对数据库表建模，并且具有执行查询的方法。在本方案中，目录数据存储在两个表中，并且目录数据模块具有两个实体 bean，每个实体 bean 对一个表建模。实体 bean 处理 `CatalogData` EJB 模块与数据库交互操作的连接、查询语言和其它方面。

本模块的编程

表 3-1 概述了用于创建上节所述并在图 3-1 中图示的 CatalogData EJB 模块的编程方法。

表 3-1 本方案所需的编程方法

应用程序元素	所需的编程方法
应用程序服务器	无。
EJB 模块	创建具有远程接口的会话企业 bean（CatalogManagerBean）。远程接口适合于通过其它模块远程调用的方法。将业务方法增加到将目录数据返回到调用方的会话 bean。这些业务方法之一返回目录中的所有项，而其它业务方法返回由客户端指定的任何项的详细资料。 创建两个实体企业 bean （ItemBean 和 ItemDetailBean）来表示包含目录数据的两个数据库表。然后创建这些实体企业 bean 的本地接口。本地接口适合于由同一模块中其它企业 bean 调用的方法，将方法增加到返回目录中所有项的 ItemBean。 ItemDetailBean 具有自动生成的 findByPrimayKey 方法，该方法返回特定项的详细资料。最后会话 bean 调用这些方法来获取目录数据。 创建两个详细资料类，一个用于项目数据，一个用于 ItemDetail 数据。CatalogData EJB 模块将这些类的实例返回到调用方。 创建 CatalogData EJB 模块，该模块在资源管理器中通过 EJB 模块节点表示。下一步将三个企业 bean 增加到该模块，使用 CatalogData EJB 模块的属性表单来指定目录数据的数据源。
J2EE 应用程序	要查看如何将 EJB 模块增加到 J2EE 应用程序，请参阅第 4 章。

后续的几节说明了如何执行这些编程任务中的大部分任务。在接口中的方法签名指定了每个交互操作的输入和输出。关于过程的几节说明了如何将这些输入和输出连接到其它组件、其它模块和目录数据库。使用 Java 代码创建企业 bean 不包括在内，增加业务方法和实现业务方法的说明。（要了解这些任务的信息，请参阅联机帮助或 《构建 Enterprise JavaBeans 组件》。）

创建会话企业 bean 的远程接口

CatalogData EJB 模块的设计通过生成对有态会话 bean 方法的远程调用来调用客户端模块以获取目录数据。要使远程调用成为可能，则会话 bean 必须具有远程接口，可以在使用 [会话 Bean] 向导创建会话 bean 时生成远程接口。编码范例 3-1 显示了有态会话 bean 的已完成起始接口和远程接口。

编码范例 3-1 会话 bean 的起始和远程接口

```
public interface CatalogManagerBeanHome extends javax.ejb.EJBHome {
  public CatalogBeans.CatalogManagerBean create()
    throws javax.ejb.CreateException, java.rmi.RemoteException;
}


public interface CatalogManagerBean extends javax.ejb.EJBObject {
  public java.util.Vector getAllItems() throws java.rmi.RemoteException;
  public CatalogBeans.idDetail getOneItemDetail(java.lang.String sku)
    throws java.rmi.RemoteException;
}
```

这些接口定义了 CatalogData EJB 模块和客户端模块之间的两个交互操作。客户端通过调用 getAllItems 方法能够获取目录中所有项的列表，客户端也可以通过调用 getOneItemDetail 方法来指定某一项并获取该项的详细信息。

大多数实际的购物应用程序提供比这更多的功能。这些实际的应用程序在接口中具有两个以上的业务方法。

当 EJB 模块将引用返回到实体 bean 实例时，该应用程序服务器跟踪客户端模块对实体 bean 实例所做的任何更改，并自动生成数据库更新。将远程引用传递到客户端会消耗网络资源，并且只有客户端能更新数据时才能传递该引用。在本方案中，客户端仅显示数据，不需要使用该功能。CatalogData 模块返回普通 Java 类的实例。这些称为详细资料类的类具有与实体 bean 相同的字段。 CatalogData 模块从实体 bean 实例将数据复制到详细资料类实例，并将详细资料类实例返回到客户端用于显示。关于与实体企业 bean 一起使用详细资料类的更多信息，请参阅 《构建 Enterprise JavaBeans 组件》。

关于客户端模块使用远程接口来调用 CatalogData 模块的方法的更多信息，请参阅第 36 页的 “Servlet 方法的编程”。

getAllItems 和 getOneItemDetail 方法使用对实体 bean 方法的调用来实现。实体 bean 生成合适的查询并将请求的数据作为实体 bean 实例返回。要查看 getAllItems 方法的实现，请参阅编码范例 3-3。

创建实体企业 bean 的本地接口

要获取客户端请求的数据， CatalogManagerBean 需调用两个实体 bean 的方法。因为这些调用都位于模块中，所以实体 bean 不需要消耗资源的远程接口。相反，可以生成实体 bean 的本地接口。本地接口比远程接口更快速、更有效。只要不需要远程接口，您就应该使用本地接口。

使用 [实体 Bean] 向导创建实体 bean 时会生成本地接口。编码范例 3-2 显示了 Item 实体 bean 的已完成本地接口。 ItemDetail bean 的各个接口都是相似的。

编码范例 3-2 ItemBean 企业 bean 的本地起始和本地接口

```
public interface LocalItemBeanHome extends javax.ejb.EJBLocalHome {
  public CatalogBeans.LocalItemBean findByPrimaryKey(java.lang.Integer aKey)
    throws javax.ejb.FinderException;
  public java.util.Collection findAll() throws javax.ejb.FinderException;
}


public interface LocalItemBean extends javax.ejb.EJBLocalObject {
  public CatalogBeans.iDetail getIDetail();
}
```

CatalogManagerBean 调用 findAll 方法来获取目录中所有项的列表。

注 – 如果计划用 IDE 的测试应用程序功能来测试独立的企业 bean，那么您需要同时生成远程和本地接口。测试应用程序功能将生成操作企业 bean 的方法的 web 模块，并且该 web 模块客户端需要远程接口。

在会话企业 bean 中使用本地接口

客户端模块通过调用 CatalogManagerBean 业务方法之一来请求目录数据。这些是编码范例 3-1 中显示的 CatalogManagerBean 接口中声明的方法，业务方法的实现要调用实体 bean 方法。要进行这些对实体 bean 的调用， CatalogManagerBean 需要对创建的实体 bean 的本地接口的本地 EJB 引用。您必须编程这些本地 EJB 引用中的两个独立部分。

- **JNDI 查找代码。**两个会话 bean 业务方法都包括代码，该代码使用 JNDI 命名工具来获取对实体 bean 的 LocalHome 的引用。
- **引用的声明。**它由运行环境使用，通过查找代码来标识引用的特定 bean。

本地 EJB 引用的 JNDI 查找代码

编码范例 3-3 显示了会话 bean `getAllItems` 方法的实现。`CatalogData` EJB 模块的客户端调用本方法来获取联机目录中所有项的列表。该方法实现说明了 `CatalogManagerBean` 如何通过与实体 bean 交互来获取目录数据。在本代码中有三个步骤：

1. `CatalogManagerBean` 使用 JNDI 查找来获取对 `ItemBean` 本地起始接口的本地引用。
2. `CatalogManagerBean` 调用 `ItemBean.findAll` 方法。
3. `CatalogManagerBean` 将目录数据从实体 bean 实例复制到详细资料类实例，然后将详细资料类实例增加到向量，最后将向量返回到客户端。

在编码范例 3-3 中的注释标识了这些步骤

编码范例 3-3 `CatalogManagerBean` 的 `getAllItems` 方法

```
public java.util.Vector getAllItems() {
  java.util.Vector itemsVector = new java.util.Vector();
  try {
    if (this.itemHome == null) {
      try {
         // 使用 JNDI 查找来获取对实体的引用
         // Bean LocalHome。
        InitialContext iC = new InitialContext();
        Object objref = iC.lookup("java:comp/env/ejb/ItemBean");
        itemHome = (LocalItemBeanHome) objref;
      }
      catch(Exception e) {
        System.out.println("lookup problem" + e);
      }
    }
    // 使用本地引用来调用 findAll();
    java.util.Collection itemsColl = itemHome.findAll();
    if (itemsColl == null) {
      itemsVector = null;
    }
    else {
      // 将数据复制到详细资料类实例。
      java.util.Iterator iter = itemsColl.iterator();
      while (iter.hasNext()) {
        iDetail detail;
        detail=((CatalogBeans.LocalItemBean)iter.next()).getIDetail();
        itemsVector.addElement(detail);
      }
    }
  }
```

编码范例 **3-3** CatalogManagerBean 的 getAllItems 方法 *（后续）*

```
    catch(Exception e) {
      System.out.println(e);
    }
    return itemsVector;
  }
```

在 lookup 语句中指定的字符串是引用的名称，不是引用的企业 bean 的名称。为了便于记住所指的是哪个企业 bean，企业 bean 名称经常作为引用名称使用。对企业 bean 引用名称的实际映射由引用声明完成 （包含在下一步中）。

本地 EJB 引用的引用声明

写好 JNDI 查找代码后，您需要设置本地 EJB 引用声明。本地 EJB 引用声明将在 lookup 语句中使用的引用名称映射到实际企业 bean 名称。引用的企业 bean 必须与调用的 bean 在同一模块中。

要设置本地 EJB 引用声明：

1. **右键单击调用的 bean 的逻辑节点并选择 [属性] →[引用] 标签 →[EJB 本地引用] 属性 →省略号 (...) 按钮 →[增加] 按钮。**

   [增加 EJB 本地引用] 对话框打开。

2. **在对话框中定义引用。**

   a. **在 [引用名称] 字段，键入在 lookup 语句中使用的引用名称。**

      图 3-2 显示用于编码范例 3-3 中的引用名称。

   b. **将引用名称映射到企业 bean。**

      单击 [引用 EJB 名称] 字段旁的 [浏览] 按钮。这样将打开选择匹配企业 bean 的浏览器。图 3-2 显示该字段选择了 ItemBean。

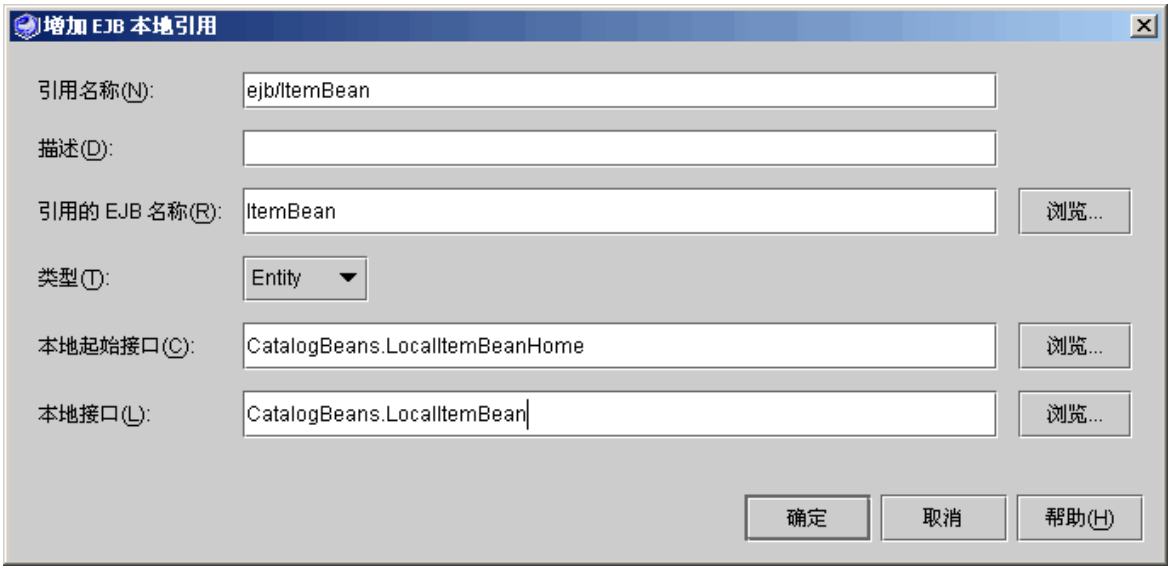

图 3-2 [增加 EJB 引用] 对话框

组装 EJB 模块

创建企业 bean 之后，再创建 `CatalogData` EJB 模块并增加企业 bean。此外，通过选择属性值来配置 EJB 模块，该属性值从应用程序服务器请求特定的运行环境服务。

不同应用程序服务器产品需要不同的部署信息，并且 EJB 模块属性包括某些特定服务器属性。本方案使用 Sun ONE 应用程序服务器的某些特定服务器属性。

创建 EJB 模块

在 IDE 中有两种创建 EJB 模块的方法，两个过程都在您所指定的位置创建 EJB 模块节点。

EJB 模块节点表示了模块的部署描述符。增加到模块的企业 bean 显示为模块节点的子节点，但是 IDE 不将企业 bean 的源文件复制到保存模块节点的目录。当决定放置 EJB 模块的位置时，请记住这一点。

- 如果要把模块的所有源码保存在单一文件系统中，则要考虑将 EJB 模块节点放置在该文件系统的顶层。
- 如果模块的源码位于不同的文件系统中，例如属于不同开发人员的文件系统，则要考虑创建仅包含模块和 J2EE 应用程序的一组目录。将这些目录与包含源码的目录分隔开。

要创建以企业 bean 节点开头的 EJB 模块：

1. **右键单击企业 bean 节点并选择 [创建新的 EJB 模块]。**

   [新建 EJB 模块] 对话框打开。

2. **命名模块并在文件系统中选择该模块的位置。**

   a. **在 [名称] 字段，输入新模块的名称。在本方案中， EJB 模块命名为** `CatalogData`。

   b. **在新的 EJB 模块字段的 [选择软件包位置]，选择要创建新模块的文件系统、目录或软件包。**

   c. **单击 [确定]。**

      表示新模块的节点在选定的文件系统、目录或节点下创建。在步骤 1 中右键单击的企业 bean 自动被包括在新模块中。

   可以将更多企业 bean 增加到模块。请参阅第 60 页的 “将企业 bean 和其它资源增加到模块”。

要从文件系统、软件包或目录节点创建 EJB 模块：

1. **右键单击任何资源管理器窗口节点，然后选择 [新建] →[所有模板]。**

   [新建] 向导的 [选择模板] 页打开。

2. **选择 [EJB 模块] 模板。**

   a. **在 [选择一个模板] 字段，展开 J2EE 节点并选择 EJB 模块节点。**

   b. **单击 [下一步]**

      [新建对象名称] 页面打开。

3. **定义新的 EJB 模块。**

   a. **在 [名称] 字段，输入模块的名称。**

      在本方案中，模块命名为 `CatalogData`。

   b. **单击 [完成]。**

      IDE 创建该模块并与步骤 1 中选定的文件系统、软件包或目录下的 EJB 模块一起表示在资源管理器中。

这两个过程都在选择的位置中创建 EJB 模块节点。请记住， IDE 在本目录中存储关于模块的部署描述符信息，但是不将模块中组件的源码复制到本目录中。

将企业 bean 和其它资源增加到模块

创建模块之后，就可以将企业 bean 增加到该模块。

要将企业 bean 增加到 EJB 模块：

1. **右键单击模块节点并选择 [增加 EJB]**。

   这将打开 [将 EJB 增加到 EJB 模块] 对话框。

2. **选择一个或多个企业 bean**。

   a. **在 [选择要增加到此 EJB 模块的 EJB] 字段，选择您要增加到该模块的企业 bean**。

   b. **单击 [确定]**。

      IDE 将选定的企业 bean 增加到模块。在资源管理器中， IDE 在模块节点下增加表示选定 bean 的节点。

将企业 bean 增加到模块时， IDE 管理企业 bean 在其它种类资源 （Java 类、图像文件等等）上的任何相关性。 IDE 自动将这些资源增加到模块定义。

关于其少数异常情况，请参阅第 65 页的 “标识额外文件”。

指定实体企业 bean 的数据源

在本方案中，`CatalogData` EJB 模块访问数据库来获取目录数据。J2EE 应用程序使用由应用程序服务器提供的连接池和其它服务，通过 J2EE 应用程序服务器访问数据库。必须将该数据库设置为具有 JNDI 名称的应用程序服务器资源。将数据库定义为应用程序服务器资源来将数据库 JNDI 名称映射到实际数据库 URL 中。

访问数据库的 J2EE 应用程序必须由包含数据库 JNDI 名称的资源引用来配置。设置资源引用的方法取决于您正在使用的实体 bean 的类型：

- 如果您正在使用容器管理的实体 bean，则应用程序服务器决定如何标识数据库。对于包括 Sun ONE 应用程序服务器在内的大多数应用程序服务器，您使用模块的特定服务器属性通过 JNDI 名称来标识数据源。
- 如果正在使用 bean 管理的实体 bean，则必须编写 JNDI 查找代码并设置将查找名称映射到 JNDI 名称的匹配资源引用声明。

注 – 在开发期间，您可以在 IDE 中打开到数据库的实时连接并使用 [实体 bean] 向导来创建建模数据库表的实体 bean。这种连接在*构建 Enterprise JavaBeans 组件*中说明。本节说明了如何指定在运行环境中使用的数据库。

使用容器管理实体 bean 的特定服务器标签

对于容器管理实体 bean，使用 EJB 模块的 CMP 资源属性通过 JNDI 名称来指定数据源，IDE 会自动创建资源引用。 CMP 资源属性包含缺省 Pointbase 数据库的 JNDI 名称。

要配置容器管理实体 bean 以使用数据库：

1. **右键单击 EJB 模块节点并选择 [属性] →[Sun ONE AS] 标签 →[CMP 资源] → 省略号 (...) 按钮。**

   [CMP 资源] 属性编辑器打开。

2. **输入标识 EJB 模块的数据库及其实体 bean 的值。**

   图 3-3 显示带有值的 [CMP 资源] 属性编辑器，该值使 `CatalogData` EJB 模块可以访问命名为 `sample` 的 Pointbase 数据库。

图 3-3 EJB 模块 CMP 资源属性编辑器

注 – IDE 的大多数安装情况均包括 Sun ONE 应用程序服务器和 PointBase 数据库服务器。安装 IDE 时将名为 `sample` 的 PointBase 数据库设置为应用程序服务器资源，JNDI 名称为 `jdo/PointbasePM`。在本方案中， `CatalogData` EJB 模块配置为使用 `sample` 数据库。

该过程的目的是与 Sun ONE 应用程序服务器和已安装的 PointBase 数据库一起使用 IDE。如果正在使用其它数据库产品或其它应用程序服务器产品，则必须修改该过程。

- 如果正在与 Sun ONE 应用程序服务器和非 PointBase 的数据库产品一起使用 IDE，则需要使用应用程序服务器的管理工具来将应用程序服务器的数据库定义为具有 JNDI 名称的 JDBC 数据源。然后，您可以与数据源 JNDI 名称一起使用该过程来配置 EJB 模块。

- 如果正在使用的应用程序服务器不是 Sun ONE 应用程序服务器，则数据库必须设置为应用程序服务器的数据源资源并给出 JNDI 名称。这些过程取决于应用程序服务器产品。用 JNDI 名称设置数据库后，您可以使用 JNDI 名称来配置 EJB 模块。在 [EJB 模块] 属性表单，使用正在使用的应用程序服务器产品的特定服务器标签。属性名称可以与 [CMP 资源] 不同。

如果要在已管理的测试环境或生产环境中部署应用程序，那么系统管理将可能负责将数据库设置为应用程序服务器资源并分配 JNDI 名称。在这种情况下，您只需从系统管理获取数据源的 JNDI 名称。

显式创建 bean 管理实体 bean 的资源工厂引用

如果 EJB 模块包含使用 bean 管理持久性的实体 bean 而不是容器管理持久性的实体 bean，则该实体 bean 已经包括准备并执行数据库查询的 JDBC 和 JTA 代码。这一部分包含在*构建 Enterprise JavaBeans 组件*中。这些代码必须包括获取对数据源的资源工厂引用的 JNDI 查找，该查找使用数据源的 JNDI 名称来获取引用。如同其它种类的 J2EE 引用，本类引用具有两部分：

- **JNDI 查找。**每个实体 bean 都包括使用 JNDI 查找语句来获取对已命名资源的引用的代码。
- **引用的声明。**该声明将用于查找语句中的引用名称映射到数据源的 JNDI 名称。

用显式查找指定的任何数据源必须是具有已分配 JNDI 名称的应用程序服务器的已命名资源。该应用程序服务器具有将 JNDI 名称映射到实际数据库 URL 的信息。

资源工厂引用的 JNDI 查找代码

编码范例 3-4 显示了用于 BMP 实体企业 bean 来查找数据源的代码：

编码范例 3-4 数据库的 JNDI 查找

```
try {
  // 获取初始上下文 -- 打开与 JNDI 命名的通讯：
  Context ic = new InitialContext();
  // 请求资源的查找 -- 在本示例中为 JDBC 数据源：
  javax.sql.DataSource hrDB = (javax.sql.DataSource)
                                ic.lookup( ò java:comp/env/jdbc/Local_HR_DB ò );
}
catch(Exception e) {
  System.out.println(e.toString());
  e.printStackTrace();
  return;
}
```

资源引用的引用声明

除了编写查找语句，还必须设置数据源资源的引用声明。

要设置数据源引用的引用声明：

1. **右键单击企业 bean 的逻辑节点并选择 [属性] →[引用] 标签 →[资源引用] → 省略号 (...) 按钮 →[增加] 按钮。**

   [增加资源引用] 对话框打开。

2. **提供标识数据源的信息。**

   a. **在 [名称] 字段中，输入用于 `lookup` 语句中的引用名称。**

      图 3-4 显示用于编码范例 3-4 中的引用名称。

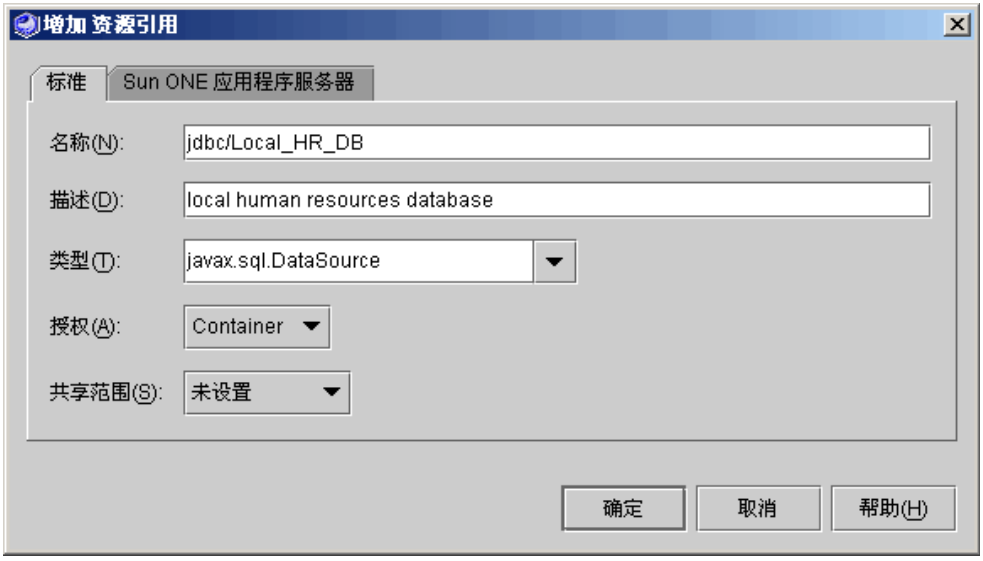

图 3-4 [增加资源引用] 对话框

b. 单击 [Sun ONE 应用程序服务器] 标签。

c. 将引用名称映射到数据源。

图 3-5 显示 [JNDI 名称] 字段，值为 `jdbc/jdbc-pointbase`。这是包括在大多数 IDE 安装中的 PointBase 数据库的 JNDI 名称的缺省名称。

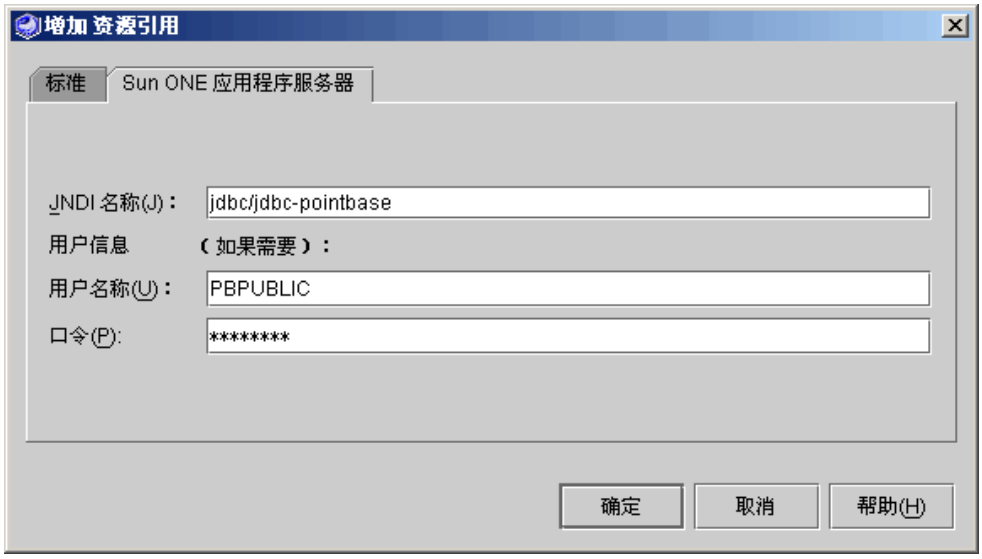

图 3-5 [增加资源引用] 对话框，特定服务器标签

该过程的目的是与 Sun ONE 应用程序服务器和已安装的 PointBase 数据库一起使用 IDE。如果正在使用其它数据库产品或其它应用程序服务器产品，则必须修改该过程。请按以下方法修改步骤：

- 如果正在与 Sun ONE 应用程序服务器和非 PointBase 的数据库产品一起使用 IDE，则需要使用应用程序服务器的管理工具来将应用程序服务器的数据库定义为具有 JNDI 名称的 JDBC 数据源。然后，您可以与数据源 JNDI 名称一起使用该过程来配置 EJB 模块。
- 如果正在使用的应用程序服务器不是 Sun ONE 应用程序服务器，那么使用的数据库必须用应用程序服务器设置为具有 JNDI 名称的数据源。这些过程取决于应用程序服务器产品。用 JNDI 名称设置数据库后，您可以使用 JNDI 名称来配置 EJB 模块。在 [增加资源引用] 对话框中，使用正在使用的应用程序服务器产品的特定服务器标签。

其它模块组装任务

上述几节涵盖了组装 CatalogData EJB 模块所需的组装任务。本节描述了在 CatalogData 方案中没有包含的某些 EJB 模块组装任务。

EJB 模块可能需要您执行这些其它组装任务中的某些任务。本节包含了可以执行的多个 web 模块组装任务。

对于每个模块，您需要决定组装任务所需的内容。关于模块可能提出的问题包括：

- 是否模块中所有引用都已经链接？您的模块可能包含对其它模块中组件的某些引用。将模块组装到 J2EE 应用程序后才能链接这些引用。
- 是否已设置模块的通用安全角色？是否已将在模块的企业 bean 中的任何安全角色引用都链接到这些通用安全角色？是否已将方法权限映射到这些通用安全角色？关于这些问题的更多信息，请参阅第 7 章。
- 是否已定义容器管理的事务？关于该问题的更多信息，请参阅第 6 章。

在 CatalogData EJB 模块方案中没有包含的某些 EJB 模块任务在后面的几节中描述。

标识额外文件

在大多数情况下，IDE 识别 EJB 模块中的企业 bean 的相关性，并包括 EJB JAR 文件中所有需要的文件，该 JAR 文件在部署时生成。但是， IDE 不能识别某些相关性。这些不能识别的相关性包括：

- 模块中使用帮助文件而不直接进行调用的企业 bean
- 模块中动态访问类并仅作为字符串而不作为类声明来使用类名称的企业 bean

在这些情况和相似的情况下， IDE 在它创建的归档文件中不能识别企业 bean 的相关性且不能包括帮助文件或类文件。您需要标识这些文件以便将它们包括进归档中并且在运行环境中可用。

要标识额外文件：

1. **右键单击 EJB 模块节点并选择 [属性] →[额外文件] →省略号 (...) 按钮。**

   [额外文件] 属性编辑器打开。

2. **选择应该被部署或归档为本模块的一部分的额外文件。**

   a. **在 [源] 字段，选择属于模块中的额外文件。**

   b. **单击 [增加] 按钮来将选定的文件增加到额外文件列表。**

3. **单击 [确定] 来关闭编辑器并处理您的选择。**

排除重复 JAR 文件

在某些情况下，您要防止 IDE 使用它所识别的某些文件相关性。例如，模块中的企业 bean 具有最常使用的 JAR 文件的相关性，而您知道该文件将出现在运行环境中。可以通过将该 JAR 文件标识为重复的 JAR 文件，防止 IDE 增加常用 JAR 文件的不必要的副本。

要排除重复 JAR 文件：

1. **右键单击模块节点并选择 [属性] →[要排除的库 Jar] →省略号 (...) 按钮。**

   [要排除的库 Jar] 属性编辑器打开。它显示了已安装 JAR 文件的列表。

2. **选择应该排除的 JAR 文件并单击 [增加] 按钮来将它们移动到要排除的库 JAR 的列表。**

3. **单击 [确定] 来关闭属性编辑器并处理您的选择。**

第 4 章

方案：Web 模块和 EJB 模块

图 4-1 显示组装在 J2EE 应用程序中的 web 模块和 EJB 模块。出现在图中的大多数交互操作包含在第 2 章和第 3 章中。例如，在用户的 web 浏览器和 web 模块之间的 HTTP 请求和响应包含在第 2 章中。模块间新的交互操作在图中通过标为 1 的箭头表示。

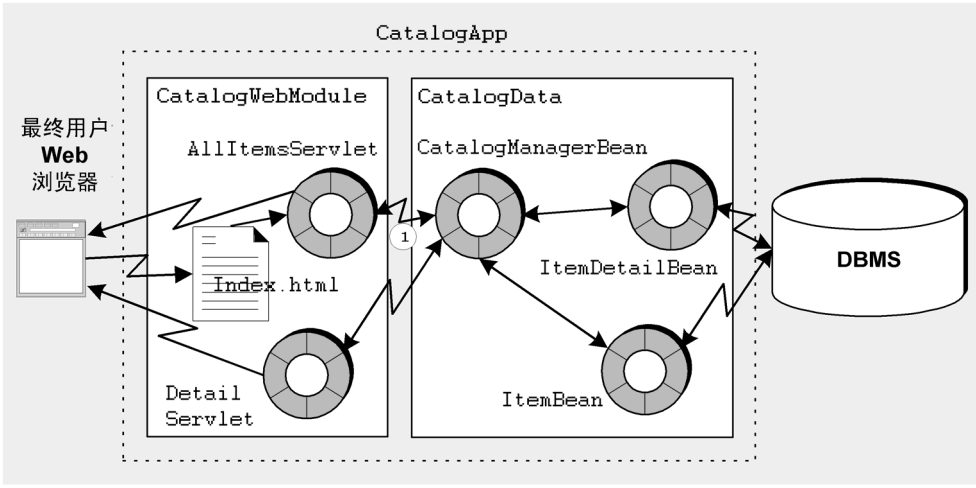

图 4-1 CatalogApp J2EE 应用程序

本应用程序中的交互操作

本方案描述了图 4-1 中所示的 J2EE 应用程序和交互操作的用法之一。本方案继续第 2 章和第 3 章中所述的联机购物应用程序。HTTP 与用户交互所需的编程已经在 web 模块中完成，与数据库交互所需的编程已经在 EJB 模块中完成。本方案解释如何将两个模块组合为执行端到端交互操作的 J2EE 应用程序。

在需要组装工作的模块之间有一个交互操作。该交互操作是从 web 模块到 EJB 模块的远程方法调用。本交互操作所需的大多数代码已经位于 web 模块或 EJB 模块中。已位于模块中的代码概述如下：

- Web 模块包含 JNDI 查找代码和 EJB 引用声明。关于该代码的更多信息，请参阅第 38 页的 “EJB 引用的引用声明”。
- EJB 模块包含支持远程方法调用的远程接口。关于该代码的更多信息，请参阅第 54 页的 “创建会话企业 bean 的远程接口”。

本章中的过程说明如何将两个模块组装到 J2EE 应用程序中，以及如何在模块间配置交互操作。组装应用程序后，您可以部署并执行应用程序。

本应用程序的编程

表 4-1 概述了创建和组装本方案中所述的 J2EE 应用程序需要的编程。

表 4-1 本方案所需的编程方法

应用程序元素	所需的编程方法
应用程序服务器	无。
Web 模块	请参阅第 2 章。
EJB 模块	请参阅第 3 章。
J2EE 应用程序	创建 `CatalogApp` J2EE 应用程序。该操作将在 Sun ONE Studio 5 资源管理器中创建 J2EE 应用程序节点，然后将 web 模块和 EJB 模块增加到应用程序。 指定 web 模块的 web 上下文。 确保 web 模块的 EJB 引用正确链接到 EJB 模块中的企业 bean。

以下几节说明这些编程任务。

创建 J2EE 应用程序

在 IDE 中有两种创建 J2EE 应用程序的方法，两个过程都在您所指定的位置中创建应用程序节点。

J2EE 应用程序节点表示应用程序的部署描述符。增加到应用程序的模块作为应用程序的子节点显示，但是 IDE 不将这些模块的源文件复制到保存应用程序节点的目录。决定放置 J2EE 应用程序节点的位置时要记住这一点。

- 如果要将应用程序的所有源码保存到单一文件系统中，则要考虑将应用程序放置在该文件系统的顶层。
- 如果应用程序的源码位于不同的文件系统中，例如属于不同开发人员的文件系统，则要考虑创建仅包含 J2EE 应用程序和模块的一组目录。将这些目录与包含源码的文件系统分隔开。

要从模块节点创建 J2EE 应用程序：

1. **右键单击 EJB 模块节点并选择 [新建应用程序]。**

   [新建应用程序] 对话框打开。

2. **命名 J2EE 应用程序并在文件系统中为它选择位置。**

   a. **在 [名称] 字段中，输入新的应用程序名。**

      在本方案中，新的 J2EE 应用程序命名为 `CatalogApp`。

   b. **在 [为新建的应用程序选择软件包的位置] 字段中，选择要创建新应用程序的文件系统、目录或软件包。**

   c. **单击 [确定]。**

      表示新应用程序的节点在选定的文件系统、目录或节点下创建。在步骤 1 中右键单击的模块自动包括在新的应用程序中。

   可以将更多模块增加到应用程序。关于这些过程的信息，请参阅第 71 页的 “将模块增加到 J2EE 应用程序”。

要从文件系统、软件包或目录节点创建 EJB 模块：

1. **右键单击任何资源管理器窗口节点并选择 [新建] →[所有模板]。**

   [新建] 向导的 [选择模板] 页打开。

2. **选择 [J2EE 应用程序] 模板。**

   a. **在 [选择模板] 字段中，展开 J2EE 节点并选择应用程序节点。**

   b. **单击 [下一步]**

      [新建对象名称] 页面打开。

3. **定义新的 J2EE 应用程序。**

   a. **在 [名称] 字段中，输入新的应用程序名。**

      在本方案中，应用程序命名为 `CatalogApp`。

   b. **单击 [完成]。**

      IDE 创建该应用程序并与步骤 1 中选定的文件系统、软件包或目录下的 J2EE 应用程序节点一起表示在资源管理器中。

这两个过程都在选择的位置中创建应用程序节点。请记住， IDE 在本目录中存储关于应用程序的部署描述符信息，但是不将应用程序模块中组件的源码复制到本目录中。

将模块增加到 J2EE 应用程序

创建 J2EE 应用程序之后，就可以向其增加模块。要将模块增加到应用程序：

1. **右键单击应用程序并选择 [增加模块]。**

   [将模块增加到应用程序] 对话框打开。

2. **选择一个或多个模块。**

   a. **在 [选择要增加到此应用程序的模块] 字段，选择您要增加到应用程序的模块。**

      - 要增加 web 模块，请选择模块的 WEB-INF 节点。
      - 要增加 EJB 模块，请选择模块节点。

   b. **单击 [确定]。**

      IDE 将选定的模块增加到 J2EE 应用程序。在资源管理器中， IDE 在应用程序节点下增加表示选定模块的节点。

将模块增加到 J2EE 应用程序后， IDE 记录模块对其它种类资源 （Java 类、图像文件等等）的任何相关性并自动将那些资源包括进应用程序中。

设置 web 模块的 web 上下文

将 J2EE 应用程序部署到 J2EE 应用程序服务器后， URL 映射到 web 模块中的 web 资源，该映射是通过将 URL 模式附加到 URL 路径来完成的。对于 Sun ONE 应用程序服务器， URL 路径一般具有以下形式：

`http://`*`hostname`*`:`*`port`*`/`*`web-context`*`/`*`URL-pattern`*

该路径中的元素按以下方式决定：

- hostname 是正在运行应用程序服务器的机器名， port 是为该服务器实例的 HTTP 请求指定的端口号。端口号在安装应用程序服务器时分配。
- Web 上下文是本过程中指定的字符串。 Web 上下文限定模块中所有的 web 资源。
- URL 模式是标识特定 servlet 或 JSP 页面的字符串。在 web 模块属性表单上分配 URL 模式。将 web 模块组装到 J2EE 应用程序中前，可以分配 URL 模式。关于该过程的信息，请参阅第 41 页的 “将 URL 映射到 Servlet”

换句话说，在 web 模块属性表单中分配的 URL 模式与本过程中分配的 web 上下文相关。

要设置 web 上下文：

1. **右键单击包括的 web 模块节点 （这是 J2EE 应用程序下的 web 模块节点）并选择 [属性]。**

2. **单击 [Web 上下文] 属性并输入您要使用的字符串。**

图 4-2 显示本方案中描述的 CatalogWebModule 的属性表单。 Web 上下文属性设为 catalog。

图 4-2 CatalogWebModule 的属性表单

Web 上下文属性设为 catalog 时，在 CatalogApp J2EE 应用程序中 web 资源的 URL 将具有以下通用格式：

http://*hostname*:*port*/catalog/*URL-pattern*

如果不提供 web 上下文，则它缺省为空白。如果让 CatalogApp J2EE 应用程序的 web 上下文缺省为空白，则 web 资源的 URL 将具有以下通用格式：

http://*hostname*:*port*/*URL-pattern*

链接 EJB 引用

CatalogWebModule 包含 JNDI 查找代码和 EJB 引用声明。 CatalogWebModule 代码和部署描述符指定类型 CatalogManagerBeanHome 和 CatalogManagerBean 的接口。关于如何设置查找代码和引用声明的信息，请参阅第 38 页的 “EJB 引用的引用声明”。

CatalogData EJB 模块包含类型 CatalogManagerBeanHome 和 CatalogManagerBean 的远程接口，也包含实现该接口名为 CatalogManagerBeanBean 的 bean 类。关于创建接口的信息，请参阅第 54 页的 “创建会话企业 bean 的远程接口”。

执行 CatalogApp J2EE 应用程序前，您必须将 web 模块中的 EJB 引用链接到 EJB 模块中的企业 bean。

在某些情况下，创建应用程序前在 web 模块的属性表单上链接引用。CatalogWebModule 中的引用不在 web 模块属性表单中链接。在本方案中，在 CatalogApp 属性表单上链接引用。

要在应用程序节点属性表单上链接 EJB 引用：

1. **右键单击应用程序节点并选择 [属性] →[EJB 引用] →省略号 (...) 按钮。**

   [EJB 引用] 属性编辑器打开。

2. **检查 EJB 引用的状态。**

   该编辑器说明所有引用已经在应用程序中声明。引用通过模块和引用名称来标识。

   图 4-3 显示了 CatalogApp J2EE 应用程序的 [EJB 引用] 属性编辑器。有一个在 CatalogWebModule 中声明的 EJB 引用。该引用命名为 ejb/CatalogManagerBean，且不解决该引用。

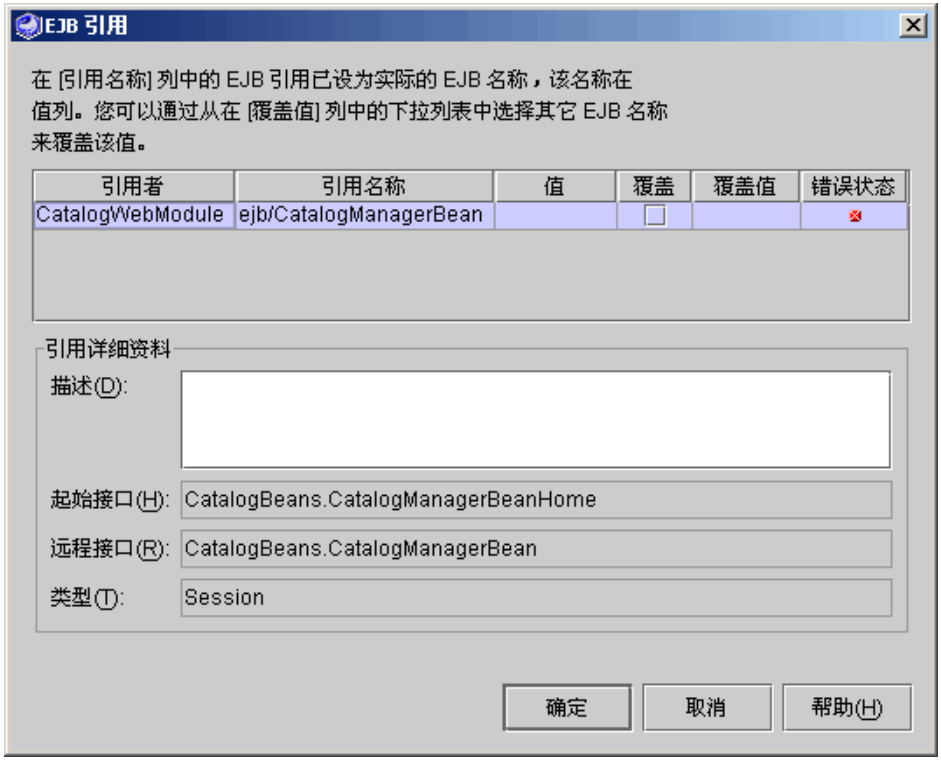

图 4-3 取消链接的 EJB 引用

如果不解决引用，则 [值] 字段为空并且 [错误状态] 字段显示表示错误的图标。

3. **链接任何未解决的引用。**

   要链接该引用，请单击 [覆盖值] 字段。该字段在实现引用中指定的接口的应用程序中显示了企业 bean 的列表，选择这些企业 bean 之一。

   图 4-4 显示了与图 4-3 相同的 EJB 引用，但是引用已与覆盖链接。应用程序执行时，在 `CatalogWebModule` 中编码的方法调用将调用在 [覆盖值] 字段中指定的企业 bean。在图 4-4 中，[覆盖值] 字段指定了 `CatalogData` EJB 模块中名为 `CatalogBeans.CatalogManagerBean` 的企业 bean。

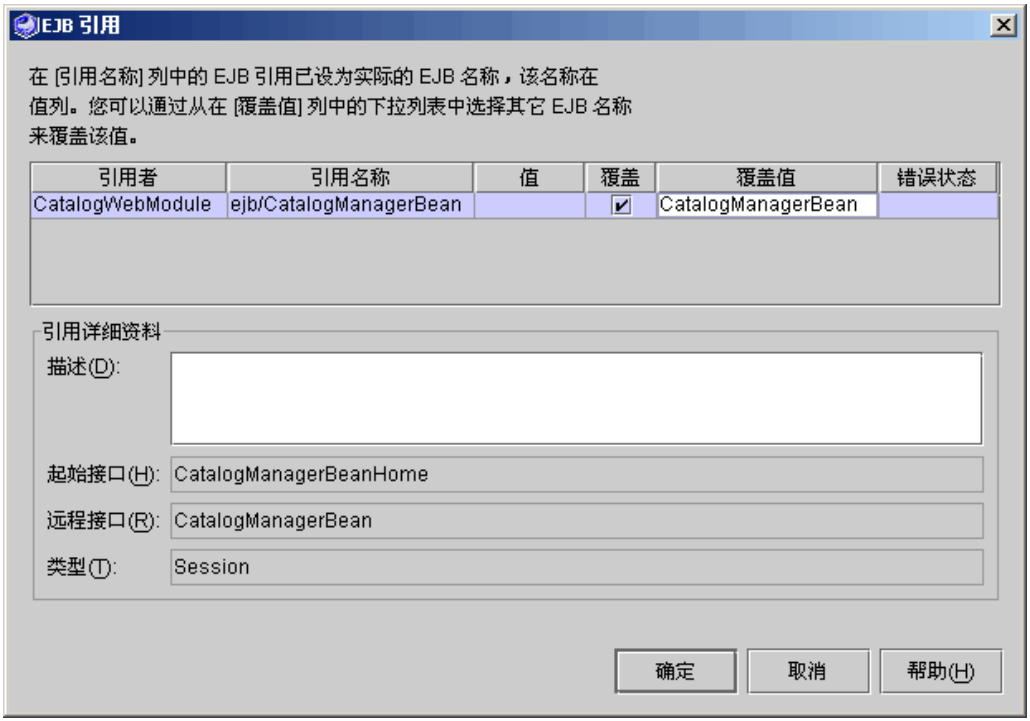

图 4-4 通过覆盖链接的 EJB 引用

解决引用后，[覆盖值] 字段显示已链接企业 bean 的名称，并且 [错误状态] 字段为空。

附加组装任务

上述几节包含了组装 CatalogApp J2EE 应用程序所需的组装任务。本节包含 CatalogApp 方案中不需要的其它 J2EE 应用程序组装任务。

J2EE 应用程序可能需要这些其它组装任务中的某些任务。本节包含了可以执行的一个应用程序组装任务。

覆盖环境条目

如果应用程序包含环境条目，则您可能需要覆盖在模块属性表单上设置的值。在应用程序的 [环境条目] 属性编辑器上覆盖该值。

要覆盖环境条目值：

1. **右键单击应用程序节点并选择 [属性] →[环境条目] →省略号 (...) 按钮。**

   [环境条目] 属性编辑器打开，该编辑器说明了已在应用程序中声明的所有环境条目。环境条目通过引用名称和模块来标识。

2. **研究应用程序中的环境条目。**

   图 4-5 显示了 CatalogApp 应用程序的 [环境条目] 属性编辑器。属性编辑器显示了 web 模块中声明的环境条目。（要了解如何在 web 模块中声明环境条目，请参阅第 47 页的 “设置环境条目引用”。）

   [值] 字段显示 100，它是在 web 模块的属性表单上设置的初始值。

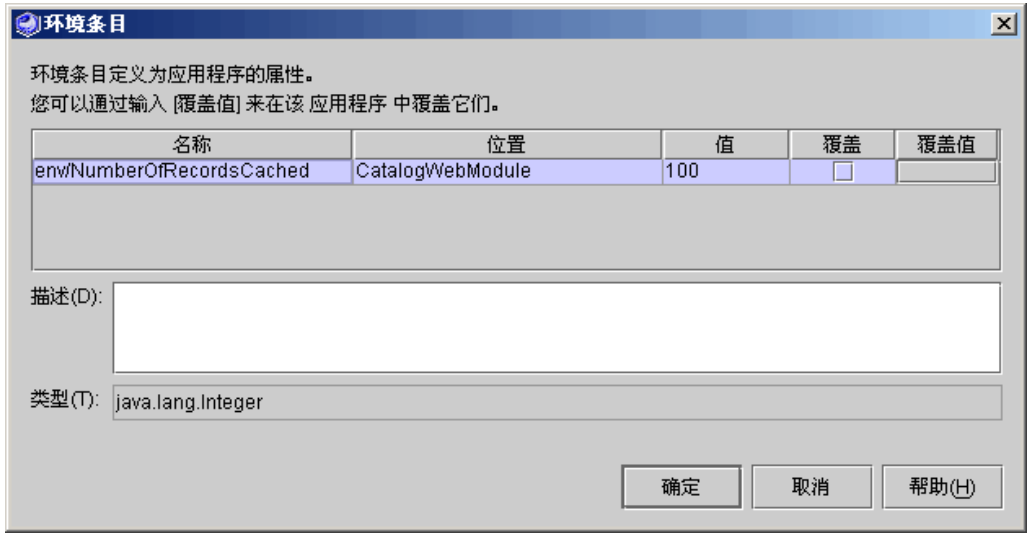

图 4-5 J2EE 应用程序 [环境条目] 属性编辑器

3. **当初始值不适合于组装的应用程序时，请覆盖这些值。**

   a. **要覆盖环境条目，请单击 [覆盖值] 字段和省略号 (...) 按钮。**

      [覆盖值] 对话框打开。

   b. **在 [值] 字段中，输入覆盖值。单击 [确定] 关闭此对话框，然后返回属性编辑器。**

      图 4-6 显示了覆盖字段中有一个值的 [环境条目] 属性编辑器。

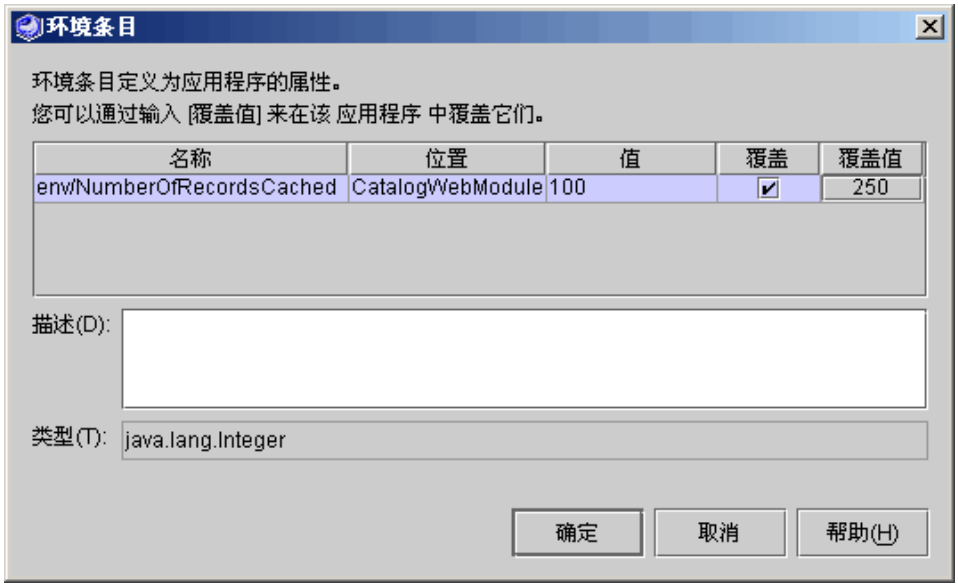

图 4-6 覆盖环境条目值

如果在开发环境中具有 web 模块的源文件，则可以在 web 模块属性表单上更改该环境条目值。但是，如果 web 模块有机会用于多个应用程序，则最好在应用程序属性表单上覆盖该值。如果其它开发人员在其它应用程序中重新使用 web 模块并在 web 模块属性表单上更改环境条目的值，那么您重新部署时将更改此应用程序中的值。

查看并编辑部署描述符

一般而言，您应该通过使用模块和应用程序属性表单来控制部署描述符的内容。通过设置这些属性，您可以控制部署描述符的内容。IDE 允许您查看模块和应用程序的实际 XML 部署描述符。

查看模块部署描述符

您可以查看包括 web 模块和 EJB 模块的 J2EE 应用程序的部署描述符。要查看部署描述符：

- **右键单击包括 EJB 模块节点 （EJB 模块节点位于 J2EE 应用程序节点下）或包括 web 模块的 J2EE 应用程序节点，然后选择 [查看部署描述符]。**

  IDE 在 [源编辑器] 中以只读模式打开部署描述符。

编辑模块部署描述符

要编辑 EJB 模块的部署描述符：

- **右键单击 EJB 模块节点并选择 [部署描述符最终编辑]。**

  IDE 在 [源编辑器] 中打开部署描述符。

要编辑 web 模块的部署描述符：

- **右键单击 web 模块的 web 节点然后选择 [编辑]。**

  IDE 在 [源编辑器] 中打开部署描述符。

第5章

方案：Web 模块和队列模式消息驱动 Bean

图 5-1 显示组装在 J2EE 应用程序中的 web 模块和 EJB 模块，模块之间的交互操作是异步消息传送。 Web 模块将消息发送到队列，而 EJB 模块中消息驱动的企业 bean 从队列中读取消息。将消息发送到队列在图中通过标为 1 的箭头表示，从队列读取消息通过标为 2 的箭头表示。在模块中，消息驱动 bean 读取消息，然后通过调用其它企业 bean 的方法初始化过程。

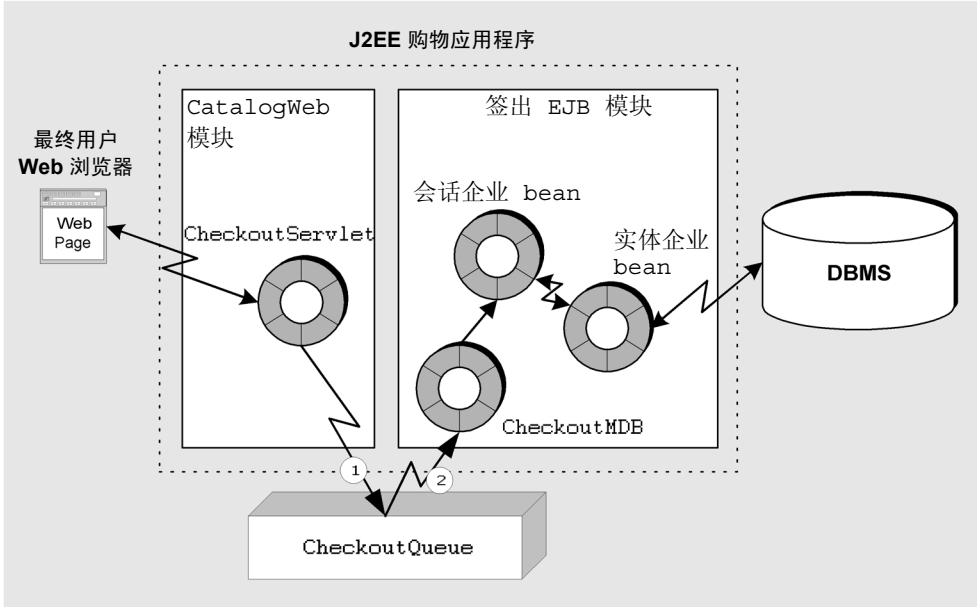

图 5-1 具有队列模式消息驱动 bean 的 J2EE 应用程序

本应用程序中的交互操作

本方案描述了图 5-1 中所示的 J2EE 应用程序和交互操作的用法之一。本方案继续说明零售 web 站点应用程序，不过讨论在不同 EJB 模块中实现的另一种交互操作。

在本方案中顾客会买单。买单之前，顾客与在 web 模块中定义的 web 页面交互，选择商品并将商品放入购物车。某些顾客的操作调用 EJB 模块的业务方法以便于选择数据。应用程序同步处理这些操作 —顾客请求某些信息，然后继续下一步之前等待应用程序响应。第 2 章、第 3 章和第 4 章说明了在 J2EE 应用程序中如何实现这种逻辑。

最后，顾客准备买单。顾客查看购物车的内容，选择发货方式，确认总额并提供信用卡号。顾客查看并确认订单，然后离开站点。一段时间后，应用程序处理订单并通过电子邮件通知顾客。本买单方案中包含的特定交互操作概述如下：

1. Web 模块显示一个页面，该页面显示定购的商品、交货地址、发货方式和付款方式。顾客确认订单并离开站点。订单的详细资料保存在数据库中。
2. 顾客确认订单后， web 模块将消息发送到消息队列。该消息标识要处理的订单。
3. 该消息队列位于应用程序之外。它由应用程序服务器维护。
4. 队列将消息传递到执行订单处理的 EJB 模块的消息驱动企业 bean 中。该队列通过调用消息驱动 bean 的 `onMessage` 方法，将消息作为参数来传递。
5. 该消息驱动 bean 不包含处理订单的业务逻辑，它仅检查消息并初始化订单完成过程。该消息驱动 bean 通过调用模块中其它企业 bean 的业务方法来初始化订单完成过程，这是使用消息驱动 bean 的典型方法。
6. 处理订单后，应用程序发送电子邮件来通知顾客。

本章中的过程说明了如何设置消息队列和队列连接工厂，如何配置发送模块和接收模块来使用队列。

为消息驱动通讯编程

表 5-1 概述了本方案中描述的和在图 5-1 中所示的实现消息驱动交互操作所需的编程方法。

表 5-1 本应用程序所需的编程方法

应用程序元素	需要的设置
应用程序服务器	设置队列名 `CheckoutQueue` 和队列连接工厂名 `CheckoutQCF`。在 IDE 中使用应用程序服务器的管理工具进行设置。
Web 模块	创建发送消息的 servlet `CheckoutServlet`。将代码增加到 `CheckoutServlet` 的 `processRequest` 方法： 1. 使用 JNDI 查找来获取对 `CheckoutQueue` 和 `CheckoutQCF` 的引用。 2. 调用 `CheckoutQueue` 方法格式化并发送消息。
EJB 模块	创建消息驱动企业 bean `CheckoutMDB`。 使用 `CheckoutMDB` 属性表单，将 `CheckoutMDB` 配置为 `CheckoutQueue` 的消息目标。 编码 `CheckoutMDB` 的 `onMessage` 方法。

后续的几节说明了如何执行这些编程任务。

为完整应用程序编程还需要其它编程任务。这些任务包括创建 web 组件和 web 模块，创建会话和实体企业 bean 以及创建 EJB 模块，这些任务包含在讨论这些问题的其它章节中。本章主要讨论消息驱动交互操作。

设置应用程序服务器

买单交互操作的设计调用 web 模块将消息发送到队列，调用 EJB 模块从队列读取消息然后处理消息中标识的订单。本交互操作需要队列和队列连接工厂，队列和队列连接工厂是在 IDE 之外创建的应用程序服务器的资源。

- 如果您正在独立开发环境中工作，则需要自己管理队列和队列连接工厂。
- 如果您正在管理的测试环境或生产环境中工作，则系统管理将可能定义、配置并管理队列和队列连接工厂。这种情况中，您只需获取队列和队列连接工厂的 JNDI 名称。此外，阅读设置队列和队列连接工厂的下列说明将帮助您理解应用程序如何使用这些资源。

在完成本节中的过程前，您需要具有应用程序服务器和安装好的 IDE 应用程序服务器插件，还需要创建应用程序服务器实例。应用程序服务器插件和服务器实例通过显示在资源管理器的运行环境标签中的节点表示。关于应用程序服务器插件和应用程序服务器实例节点的更多信息，请参阅第 119 页的 “服务器产品节点”。

设置队列

本节解释如何设置 Sun ONE 应用程序服务器的消息队列。其它应用程序服务器的过程也是相似的。

要将队列增加到 Sun ONE 应用程序服务器：

1. **单击资源管理器的 [运行环境] 标签。**
2. **展开 Sun ONE 应用程序服务器 7 节点。**
3. **右键单击 [未注册的 JMS 资源] 节点，然后选择 [增加新的 JMS 资源]。**
   [新建] 向导的 [JMS 资源] 页面打开。
4. **定义队列：**
   a. **在 [JNDI 名称] 字段中，输入 `jms/CheckoutQueue`。**
   b. **确保 [资源类型] 字段设为 `javax.jms.Queue`。**
   c. **单击 [下一步]。**
      [新建] 向导的 [属性] 页面打开。
5. **定义 `imqDestinationName` 属性。**
   a. **单击 [增加]。**
      第一行属性被激活。
   b. **在 [名称] 字段中，选择 `imqDestinationName`。**
   c. **在 [值] 字段中，输入 `Checkout`。**
      `Checkout` 是要创建的物理队列的名称。 J2EE 应用程序将使用您分配的 JNDI 名称 `CheckoutQueue` 来访问名为 `Checkout` 的队列。
   d. **单击 [完成]。**
      [您要继续注册吗] 对话框打开。
6. **注册队列：**
   a. **单击 [注册]。**
      [Java 邮件会话注册] 对话框打开。
   b. **在 [服务器实例] 字段中，选择要注册队列的应用程序服务器实例。**
      选择将 J2EE 应用程序部署到的应用程序服务器。
   c. **单击 [注册]。**
      [资源已注册成功] 消息出现。
   d. **单击 [关闭]。**

设置 QueueConnectionFactory

本节解释如何设置 Sun ONE 应用程序服务器的队列连接工厂。其它应用程序服务器的过程也是相似的。

要将队列连接工厂增加到 Sun ONE 应用程序服务器：

1. **右键单击 [未注册的 JMS 资源] 节点，然后选择 [增加新的 JMS 资源]。**
   [新建] 向导的 [JMS 资源] 页面打开。
2. **定义队列连接工厂：**
   a. **在 [JNDI 名称] 字段中，输入 `jms/CheckoutQCF`。**
   b. **确保 [资源类型] 字段设为** `javax.jms.QueueConnectionFactory`。
   c. **单击 [完成]。**
      [您要继续注册吗] 对话框打开。
3. **注册队列连接工厂：**
   a. **单击 [注册]。**
      [Java 邮件会话注册] 对话框打开。
   b. **在 [服务器实例] 字段中，选择您要注册队列连接工厂所用的应用程序服务器实例。**
      选择创建队列时您所选定的应用程序服务器实例。
   c. **单击 [注册]。**
      [资源已注册成功] 消息出现。
   d. **单击 [关闭]。**

Web 模块编程

在本方案中， `CheckoutServlet` 发送请求订单最终处理的消息。该消息标识要处理的订单， `CheckoutServlet` 调用队列连接工厂和队列的方法来发送消息。

CheckoutServlet 需要到队列和队列连接工厂的引用来调用队列和队列连接工厂方法。`CheckoutServlet` 使用 JNDI 查找来从应用程序服务器环境获取队列和队列连接工厂引用。

类似于大多数 J2EE 引用查找，队列和队列连接工厂引用查找具有两个部分：

- **JNDI 查找代码。**Servlet 包括使用 JNDI 命名工具来获取到队列或队列连接工厂的引用的代码。
- **引用的声明。**该声明将 JNDI 查找语句中的名称映射到队列或队列连接工厂的实际 JNDI 名称。

队列和队列连接工厂是应用程序服务器的命名资源。应用程序组件使用 JNDI 名称来获取引用。要了解如何将 JNDI 名称分配到队列和队列连接工厂，请参阅第 81 页的 “设置应用程序服务器”。

JNDI 查找代码

编码范例 5-1 显示了 CheckoutServlet 的 processRequest 方法，该方法执行 JNDI 查找。获取队列和队列连接工厂后，processRequest 调用队列和队列连接工厂的方法来创建和发送消息。代码示例包含标识这些操作的注释。

编码范例 5-1 源于 servlet，但是任何类型的 J2EE 组件都能使用相似的代码来发送消息。在用作消息发送器的应用程序客户端或企业 bean 中可以重用本代码。

编码范例 5-1 CheckoutServlet 的 processRequest 方法

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.jms.*;
import javax.naming.*;

// ...

protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
                              throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  // 在此处输出页面

  // 删除缺省方法正文并插入下列行：
  Context jndiContext = null;
  javax.jms.TextMessage msg = null;
  QueueConnectionFactory queueConnectionFactory = null;
  QueueConnection queueConnection = null;
  QueueSession queueSession = null;
  Queue queue = null;
  QueueSender queueSender = null;
  TextMessage message = null;

  out.println("<html>");
  out.println("<head>");
  out.println("<title>Servlet</title>");
  out.println("</head>");
  out.println("<body>");

  try {
    // 连接到缺省命名服务 -- 由应用程序服务器管理
```

编码范例 5-1 CheckoutServlet 的 processRequest 方法 *（后续）*

```
    jndiContext = new InitialContext();
  }
  catch (NamingException e) {
    out.println("Could not create JNDI " + "context: " +
    e.toString());
  }

  try {
    // 执行本方案中创建的缺省 QueueConnectionFactory 和队列
    // 的 JNDI 查找。
    queueConnectionFactory = (QueueConnectionFactory)
                 jndiContext.lookup("java:comp/env/jms/CheckoutQCF");
    queue = (Queue)
                jndiContext.lookup("java:comp/env/jms/CheckoutQueue");
  }
  catch (NamingException e) {
    out.println("JNDI lookup failed:" + e.toString());
  }

  try {
    // 使用引用来连接到队列并发送消息。
    queueConnection =
                       queueConnectionFactory.createQueueConnection();
    queueSession = queueConnection.createQueueSession(false,
                                             Session.AUTO_ACKNOWLEDGE);
    queueSender = queueSession.createSender(queue);
    message = queueSession.createTextMessage();
    message.setText("Order #33454344");
    queueSender.send(message);
  }
  catch (JMSException e) {
    out.println("Exception occurred:" + e.toString());
  }
  finally {
    if (queueConnection != null) {
      try {
        queueConnection.close();
      }
      catch (JMSException e) {}
    } // if 语句结束点
  } // 最后结束点
  // 和要插入代码的结束点

  out.close();
}
```

关于创建和发送消息的更多信息，请参阅 《构建 Enterprise JavaBeans 组件》。

队列的引用声明

引用声明出现在模块的部署描述符中。引用声明将用于查找语句中的引用名映射到应用程序服务器环境中的 JNDI 名称。

要设置队列的引用声明：

1. **右键单击 web 模块的 web 节点并选择 [属性] →[引用] 标签 →[资源环境引用] → 省略号 (...) 按钮。**

   [资源环境引用] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加资源环境引用] 对话框打开。

3. **声明 [资源环境引用]。**

   a. **在 [名称] 字段中，输入显示在 `lookup` 语句中的引用名称。**

      图 5-1 显示 [名称] 字段中的值 `jms/CheckoutQueue`。这是用于编码范例 5-1 中的引用名称。

   b. **在 [类型] 字段中，选择 `javax.jms.Queue`。**

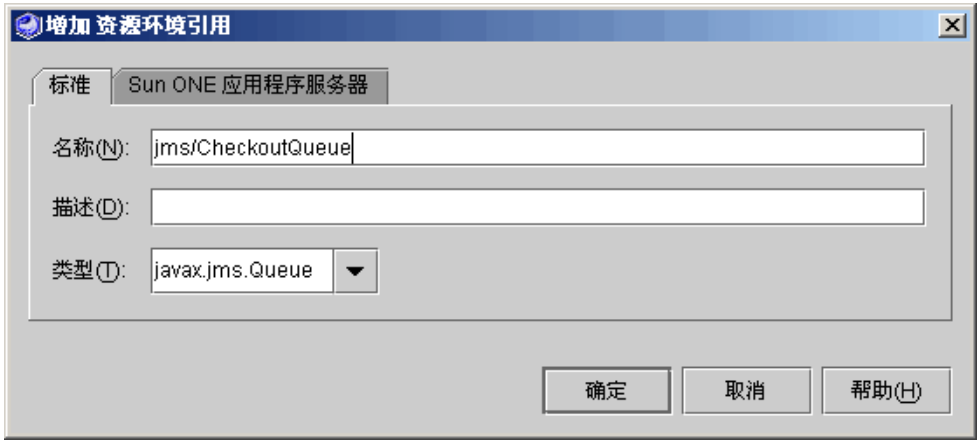

图 5-2 增加 `CheckoutQueue` 的资源环境引用

4. **将引用名称映射到 JNDI 名称。**

   a. **单击 [增加] 对话框的 [Sun ONE 应用程序服务器] 标签。**

   b. **在 [JNDI 名称] 字段，输入队列的 JNDI 名称。**

      图 5-3 显示 [JNDI 名称] 字段中的值 `jms/CheckoutQueue`。该值将 [标准] 标签上的引用名称映射到名为 `CheckoutQueue` 的队列。要了解如何命名队列，请参阅第 81 页的 “设置应用程序服务器”。

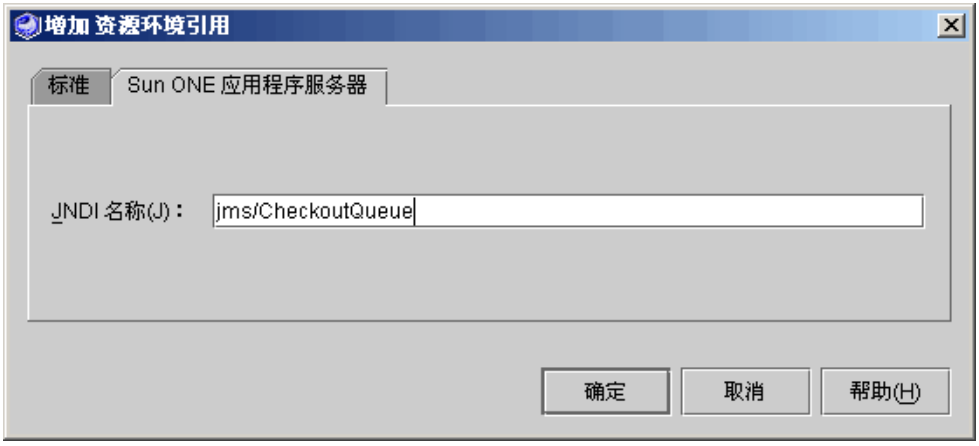

图 5-3 提供队列引用的 JNDI 名称

QueueConnectionFactory 的引用声明

要设置队列连接工厂的引用声明：

1. **右键单击 web 模块的 web 节点并选择 [属性] →[引用] 标签 →[资源引用] → 省略号 (...) 按钮。**

   [资源引用] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加资源引用] 对话框打开。

3. 声明资源引用。

   a. 在 [名称] 字段中，输入显示在 `lookup` 语句中的引用名称。

      图 5-4 在 [名称] 字段中显示值 `jms/CheckoutQueue`。这是用于编码范例 5-1 中的引用名称。

   b. 在 [类型] 字段中，选择 `javax.jms.QueueConnectionFactory`。

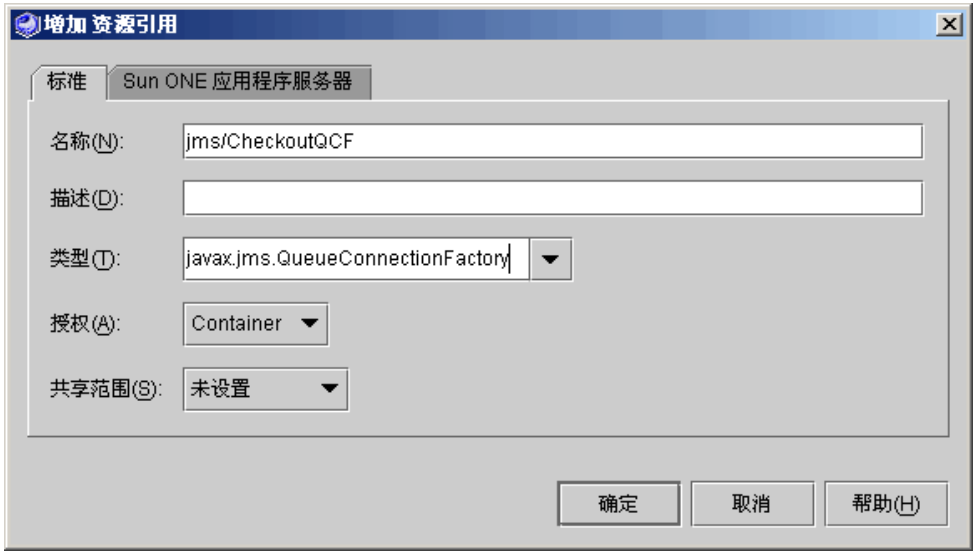

图 5-4 QueueConnectionFactory 的资源引用

4. **将引用名称映射到 JNDI 名称。**

   a. **单击 [增加] 对话框的 [Sun ONE 应用程序服务器] 标签。**

   b. **在 [JNDI 名称] 字段，输入队列连接工厂的 JNDI 名称。**

      图 5-5 在 [JNDI 名称] 字段中显示值 `jms/CheckoutQueue`。该值将 [标准] 标签上的引用名称映射到名为 `CheckoutQCF` 的队列连接工厂。要了解如何命名队列连接工厂，请参阅第 81 页的 “设置应用程序服务器”。

      关于其它授权类型的信息，请参阅 《构建 Enterprise JavaBeans 组件》中消息驱动 bean 所包含的部分。

图 5-5 QueueConnectionFactory 引用的 JNDI 名称

EJB 模块编程

在本买单方案中，处理顾客买单请求的业务逻辑位于 `Checkout` EJB 模块中。第 83 页的 “Web 模块编程”显示 web 模块如何查找队列引用和队列连接工厂引用并发送消息。要完成本交互操作， EJB 模块中的消息驱动 bean 需要从队列接收消息。

消息驱动 bean 不使用程序引用，也不需要 JNDI 查找代码。可以使用消息驱动 bean 的属性表单来指定应使用的队列和队列连接工厂，在部署描述符中设置这些属性以设置标记。设为配置消息驱动 bean 的属性在下面列出：

- **[消息驱动目标] 属性。**该属性指定了消息驱动 bean 使用的目标类型。
- **[Mdb 连接工厂] 属性。**该属性指定了队列连接工厂。
- **[JNDI 名称] 属性。**该属性指定了队列，是 Sun ONE 应用程序服务器的特定属性。其它应用程序服务器将使用不同的属性来指定队列。

部署应用程序时，应用程序服务器自动使用在部署描述符中指定的队列连接工厂来打开从消息驱动 bean 到在部署描述符中指定的队列的连接。

配置 [消息驱动目标] 属性

指定队列和队列连接工厂前，您需要将消息驱动 bean 配置为队列用户。

要将消息驱动 bean 配置为队列用户：

1. **右键单击消息驱动 bean 的逻辑节点并选择 [属性] →[消息驱动目标] →省略号 (...) 按钮。**

   [消息驱动目标] 属性编辑器打开。

2. **将消息驱动 bean 标识为队列用户。**

   a. **在 [目标类型] 字段中，选择**队列。

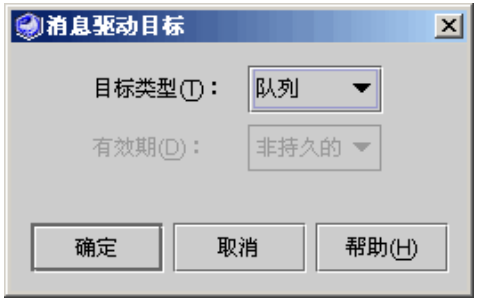

图 5-6 消息驱动 bean 属性表单

   b. **单击 [确定]。**

指定连接工厂

要配置队列连接工厂的消息驱动 bean：

1. **右键单击消息驱动 bean 的逻辑节点并选择 [属性] →[Sun ONE AS] 标签 →[Mdb 连接工厂] →省略号 (...) 按钮。**

   [Mdb 连接工厂] 属性编辑器打开。

2. **指定队列连接工厂：**

   a. **在 [Jndi 名称] 字段中，输入队列连接工厂的 JNDI 名称。**

      图 5-7 显示在 [Jndi 名称] 字段中的值 `jms/CheckoutQCF`。`jms/CheckoutQCF` 是在发送 web 模块中指定的队列连接工厂。

   b. **如果需要用户名和口令，则在 [名称] 和 [口令] 字段中输入。**

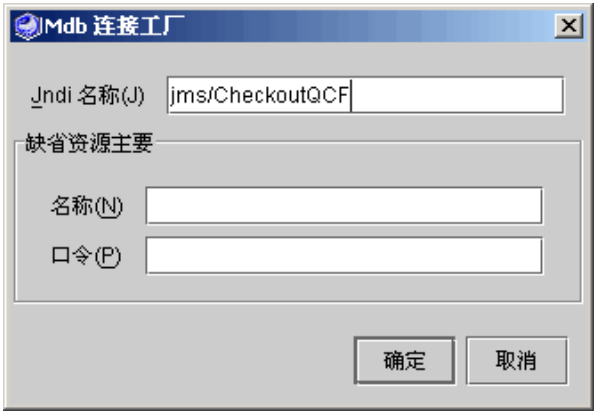

图 5-7 消息驱动 bean 的 [连接工厂] 属性编辑器

   c. **单击 [确定]。**

指定队列

要配置队列的消息驱动 bean：

1. **右键单击消息驱动 bean 的本地节点并选择 [属性] →[Sun ONE AS] 标签 →[JNDI 名称] →省略号 (...) 按钮。**

   [JNDI 名称] 属性编辑器打开。

2. **指定队列。**

   在本方案中，使用 `jms/CheckoutQueue`。`jms/CheckoutQueue` 是在发送 web 模块中指定的队列。

`onMessage` 方法的编码

应用程序服务器通过调用 bean 的 `onMessage` 方法将消息传递到消息驱动 bean。消息作为 `onMessage` 方法的参数传递。编码范例 5-2 显示了 `onMessage` 方法。您可以查看到作为参数传送的消息和增加消息处理代码的位置。

编码范例 5-2 `onMessage` 方法

```
public void onMessage(javax.jms.Message aMessage) {
        // 在此处处理消息。
    }
```

在本方案中，如同图 5-1 中所示，该消息驱动 bean 在同一 EJB 模块中立即调用会话 bean 的业务方法。该会话 bean 控制订单的处理，这是典型的 `onMessage` 行为。更多关于编写 `onMessage` 方法的信息，请参阅 《构建 Enterprise JavaBeans 组件》。

消息驱动 bean 使用 EJB 本地引用调用会话 bean。关于如何使用 EJB 本地引用实现方法调用的更多信息，请参阅第 56 页的 “本地 EJB 引用的 JNDI 查找代码”和第 57 页的 “本地 EJB 引用的引用声明”。

组装 J2EE 应用程序

图 5-1 显示组装在 J2EE 应用程序中发送消息的 web 模块和接收消息的 EJB 模块，如同本章所述对这些模块进行编程。创建应用程序并将两个模块增加到该应用程序，两个模块已配置好来使用 `CheckoutQueue` 和 `CheckoutQCF`。对于消息驱动交互操作，不需要打开 J2EE 应用程序属性表单并执行任何附加组装工作。

关于创建应用程序和增加模块的信息，请参阅第 69 页的 “创建 J2EE 应用程序”。

第 6 章

事务

本章主要介绍了使用 EJB 模块属性表单对容器管理事务进行编程。关于 bean 管理事务，请参阅 《构建 Enterprise JavaBeans 组件》。

缺省事务边界

事务边界是由事务相关的企业 bean 的 [事务属性] 属性来决定的。

当您使用 IDE 的 EJB 向导创建企业 bean 并选择容器管理事务时，向导创建的企业 bean 的 [事务属性] 属性将具有缺省值。本节将介绍如何查看并解释缺省的事务属性设置。

要打开 [事务设置] 属性编辑器并检查缺省设置：

- **右键单击 EJB 模块节点并选择 [属性] →[事务设置] 属性 →省略号 (...) 按钮。**

  [事务设置] 属性编辑器打开。

图 6-1 显示了第 3 章中说明的 CatalogData EJB 模块的 [事务设置] 属性编辑器。图 6-1 中显示的值都是缺省的事务属性设置。

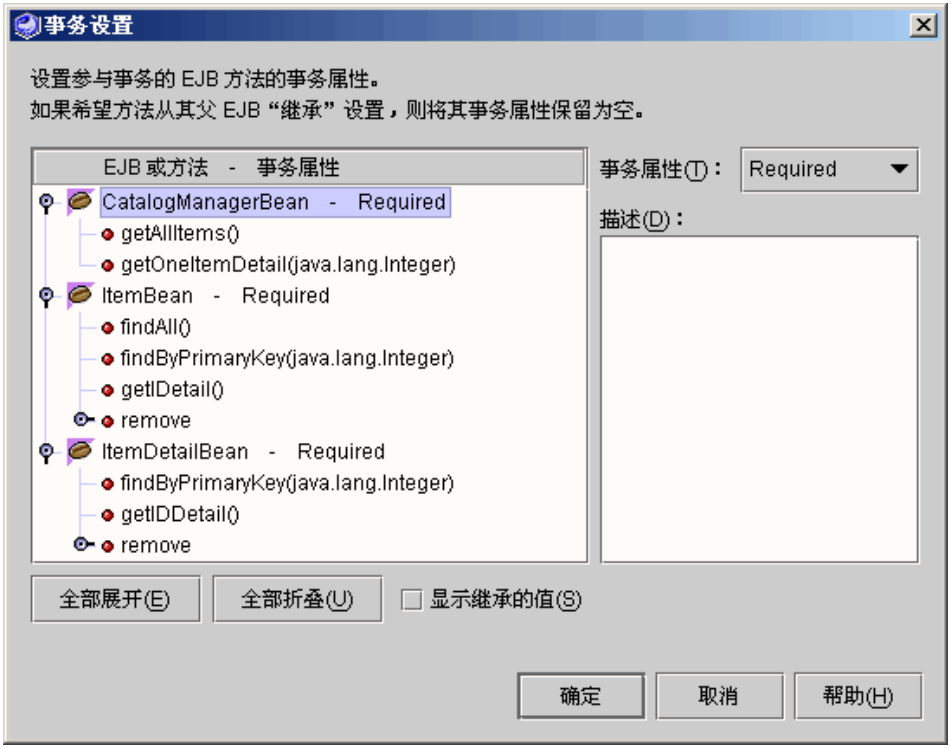

图 6-1 缺省的事务属性设置

缺省的事务属性设置显示在 [EJB 或方法 —事务属性] 字段中。要理解此处显示的内容，请注意其中的下列特性：

- 模块中的每一个企业 bean 都出现在其中。每个企业 bean 都使用一个节点来表示。
- 每一个企业 bean 的名称后都跟有它的事务属性。例如，图 6-1 中的第一个节点 CatalogManagerBean 后跟有单词 [必需的]。[必需的] 是 [事务属性] 属性的缺省设置。在图 6-1 中，所有的企业 bean 都具有缺省设置。
- 代表企业 bean 的节点可以被展开，用于显示代表每个企业 bean 的方法的节点。在图 6-1 中，所有的企业 bean 节点都已展开。
- 方法有它们自己的事务属性值，该值出现在方法名称的后面。如果事务属性值为空，则方法将继承其所属的企业 bean 的事务属性值。
- 在图 6-1 中，所有的方法节点都没有显示事务属性值。因此，所有这些方法都将继承值 [必需的]。这就是方法的事务属性的缺省设置。

[必需的] 是 《Java 2 平台， Enterprise Edition 规范》中定义的事务属性值之一。以下是具有 [必需的] 事务属性的方法的规则：

- 如果在没有活动事务的情况下调用具有必需的属性的方法，应用程序服务器将开始一个新的事务。
- 如果当活动事务正在进行时调用具有 [必需的] 属性的方法，应用程序服务器将在活动事务内执行该方法。

这是企业 bean 方法的缺省行为。

重定义事务边界

EJB 模块中的业务事务通常跨越多个企业 bean。企业 bean 的业务逻辑的通用体系结构包括一个会话 bean 和多个实体 bean。客户端调用会话 bean，然后会话 bean 调用实体 bean 的方法。

例如，客户端上存在两条相关的新数据库记录所对应的数据。会话 bean 通过调用两个实体 bean 的 create 方法来创建数据库插入，这两项数据库插入操作必须处于同一个事务中。图 6-2 中显示了这种类型的事务。

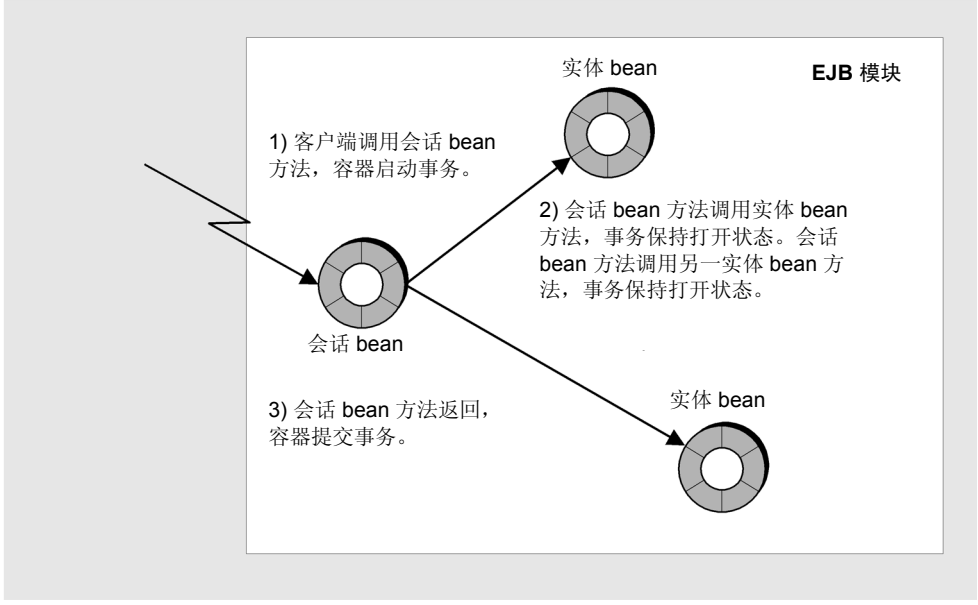

图 6-2 复杂事务

当您将这些实体 bean 组装到 EJB 模块中时，需要应用程序服务器能够识别业务事务的边界。您需要应用程序服务器能够包含客户端调用单个数据库事务后 EJB 模块需要执行的所有操作，此外还需要应用程序服务器能够执行以下操作：

- 当客户端调用会话 bean 时，您需要应用程序服务器打开一个事务。
- 当会话 bean 调用实体 bean 时，您需要应用程序服务器能确保事务处于打开状态。
- 您需要当会话 bean 对于实体 bean 的最后一次调用返回时，应用程序关闭事务并结束客户端调用的会话 bean。

如果应用程序服务器能够识别这些事务边界，EJB 模块需要执行的所有任务将被一起提交或转返。

要配置事务边界，您需要打开 [事务设置] 属性编辑器并修改事务相关的企业 bean 的事务属性。

下列步骤显示了如何编辑事务属性设置。此过程将图 6-1 中显示的缺省事务设置更改为图 6-2 中显示的事务边界所需的事务属性设置。

要更改事务属性：

1. **右键单击 EJB 模块节点并选择 [属性] →[事务设置] 属性 →省略号 (...) 按钮。**

   [事务设置] 属性编辑器打开。

2. **将会话 bean 方法的事务属性更改为** `RequiresNew`。

   您可以更改 `CatalogManagerBean` 方法的行为，这样对它的任何一个业务方法（`getAllItems` 和 `getOneItemDetail`）的调用都将开始一个新事务。要实现这一目的，将事务属性更改为 `RequiresNew`。

   a. **单击会话 bean 方法以选择它。**

   b. **将 [事务属性] 字段的值更改为** `RequiresNew`。

      注意，新的事务属性值处于方法节点的后面。

3. **将实体 bean 方法的事务属性更改为** `Mandatory`。

   您需要更改实体 bean 方法的行为，使这些方法在会话 bean 方法打开的事务的边界内执行。要实现这一目的，将事务属性更改为 `Mandatory`。这样应用程序服务器就知道了只有当事务正在进行时，才能执行这些方法。

   a. **单击实体 bean 方法以选择它。**

   b. **将 [事务属性] 字段的值更改为** `Mandatory`。

      注意，新的事务属性值处于方法节点的后面。

4. 单击 [确定] 用于关闭编辑器并保存所做的更改。

图 6-3 显示了 [事务设置] 属性编辑器中包含了新的事务属性设置。

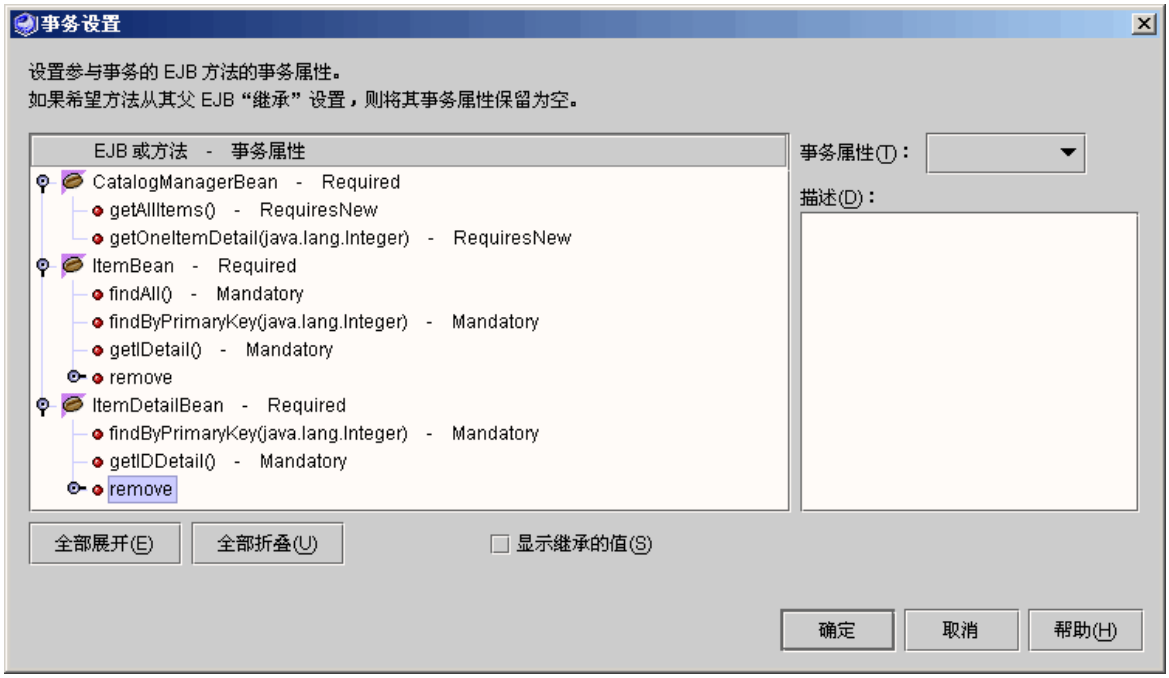

图 6-3 更改后的事务设置

现在事务边界就与业务事务相匹配了，下表描述属性编辑器中的可见更改：

- 会话 bean 方法的事务属性被设置为 `RequiresNew`，因为您需要容器在客户端调用这些方法中的任何一个时能够打开一个新的事务。
- 实体 bean 业务方法的事务属性被设置为 `Mandatory`，这也就是说这些方法必须当事务正在进行时才能被调用。需要让这些方法在会话 bean 打开事务之后被调用，并且希望这些方法在会话 bean 的事务的边界内执行。
- 注意，我们是在方法级别对事务属性进行的修改，现在方法节点在方法名称的后面显示了新的事务属性值。

对于您使用的每个 EJB 模块，必须分析模块的业务逻辑并确定逻辑所暗示的事务边界。然后使用 [事务属性] 属性编辑器来实现那些事务边界。可以设置企业 bean 或其中单个方法的事务属性用于指定相关事务的事务边界。

第 7 章

安全性

应用程序的安全性是通过对程序的特性和功能的使用加以限制来实现的。而对于程序的特性和功能的使用限制又是通过对于应用程序管理的数据的访问进行限制实现的。

J2EE 应用程序的安全性通过应用程序资源和用户角色的共同作用来实现。下面对这两个概念进行定义：

- 资源是应用程序中的可见或可调用特性。对于 EJB 模块，资源是在起始接口或远程接口声明的公共 EJB 方法。对于 web 模块，资源指的是映射到 JSP 页面、 servlet 方法以及其它组件的 URL 模式。
- 角色是指与用户相关联的名称。

J2EE 应用程序的安全性通过将资源映射到角色来实现。当资源被调用时，调用方必须提供已拥有访问该资源权限的角色名称。如果调用方不能提供授权角色，那么调用将被拒绝。在 J2EE 应用程序中，在允许调用方执行资源之前，应用程序服务器将会对调用方的角色进行检验。

角色和资源的授权组合在部署描述符中声明。应用程序服务器从部署描述符中读取并应用这些授权组合，这也被称为声明性安全性。

下表描述为了确保 J2EE 应用程序安全性而需要执行的任务：

- 声明角色。
- 指定允许访问资源的角色。

例如，假设您正在开发使用人力资源数据的 web 模块。该模块的说明将指出哪些 web 资源是所有的雇员都可以访问，雇员可以用它们来维护人员信息；而哪些 web 资源只有人力资源办事员角色、人力资源管理角色、审核角色才能访问的等等。然后声明代表这些类型用户的安全角色，并将 web 模块中的资源映射到这些角色。

EJB 模块的安全性与此类似。模块的说明中将列出不同类型的用户，并告诉您哪些类型的用户被获准使用企业 bean 方法返回的数据。然后需要声明代表这些组用户的安全角色，并将模块中的 EJB 方法映射到合适的角色。

一般而言，在模块的属性表单上设置 J2EE 的安全性。需要为模块声明一组角色，然后将模块资源映射到模块中声明过的角色集合。

当模块被组装成应用程序并部署后，您需要将在模块中声明的角色映射到应用程序服务器环境中的实际用户名称和组名称。在 J2EE 应用程序的属性表单中，将角色映射到用户和组。当应用程序部署到环境 （如拥有声明过的用户和组的生产环境）时，需要执行该映射。

后续的几节将说明如何使用 IDE 来为 web 模块 （EJB 模块）设置安全性，以及在将模块组装到 J2EE 应用程序以后如何合并不同类的安全声明。

Web 模块的安全性

Web 模块有多个属性编辑器和对话框用于声明安全角色并将其映射到 web 资源。本节涵盖了这两个任务中的属性编辑器和对话框：

- 为 web 模块声明安全角色
- 定义需要确保其安全性的 web 资源，并将它们映射到安全角色。

每一个任务都在相应的节中说明。

为 web 模块声明安全角色

要为 web 模块声明安全角色：

1. **右键单击 web 模块的 web 节点，选择 [属性] →[安全性] 标签 →[安全角色] → 省略号 (...) 按钮。**

   [安全角色] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加安全角色] 对话框打开。

3. **定义新的安全角色。**

   a. **在 [角色名称] 字段中，为新的安全角色输入名称。**

   b. **在 [描述] 字段中，为该角色输入简短描述。**

      当您需要合并同一应用程序中组装的各个模块的安全角色时，安全角色的描述就显得尤为有效。当您将安全角色映射到应用程序服务器环境中的用户和组时，该描述也是十分有用的。

c. **单击 [确定] 用于关闭对话框。**

图 7-1 显示声明 Me 和 EveryoneElse 这两个角色以后的 [安全角色属性] 编辑器。

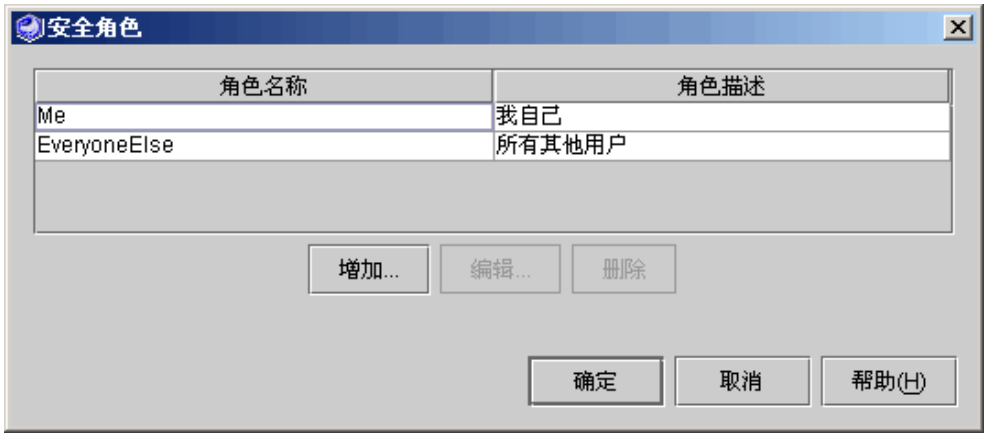

图 7-1 为 web 模块声明的安全角色 Me 和 EveryoneElse

4. **再次单击 [确定] 关闭属性编辑器。**

定义 web 资源并将其映射到安全角色

要定义 web 资源并设置安全性：

1. **右键单击 web 模块的 web 节点，选择 [属性] →[安全性] 标签 →[安全限制] → 省略号 (...) 按钮。**

   [安全限制] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加安全限制] 对话框打开。

3. **单击 [增加安全限制] 对话框上的 [增加] 按钮。**

   [增加 web 资源集合] 对话框打开。

4. **定义 web 资源。**

   a. **在 [资源名称] 字段中，为资源输入名称。**

   b. **在 [URL 模式] 字段中，输入 URL 模式。**

   URL 模式必须在模块中预定义。图 7-2 显示了将 URL 模式 `/allItems` 定义为 web 资源时显示的字段值。该 URL 模式已被映射到 servlet `AllItemsServlet`。图 7-2 中的值设置了执行 `AllItemsServlet` 所需的 web 资源。要查看 URL 模式 `/allItems` 是如何被映射到 servlet `AllItemsServlet` 的，请参阅第 41 页的“将 URL 映射到 Servlet”和 第 44 页的 “设置 JSP 页面”。

图 7-2 定义名为 `allItems` 的 web 资源

注意，您可以定义 web 资源，使其应用到全体与 URL 模式相关联的 HTTP 方法或其中的一部分。

c. 单击 [确定] 关闭此对话框，然后返回 [增加安全限制] 对话框。

图 7-3 显示了设置名为 `AllItems` 和 `ItemDetail` 两项 web 资源以后的 [增加安全限制] 对话框。

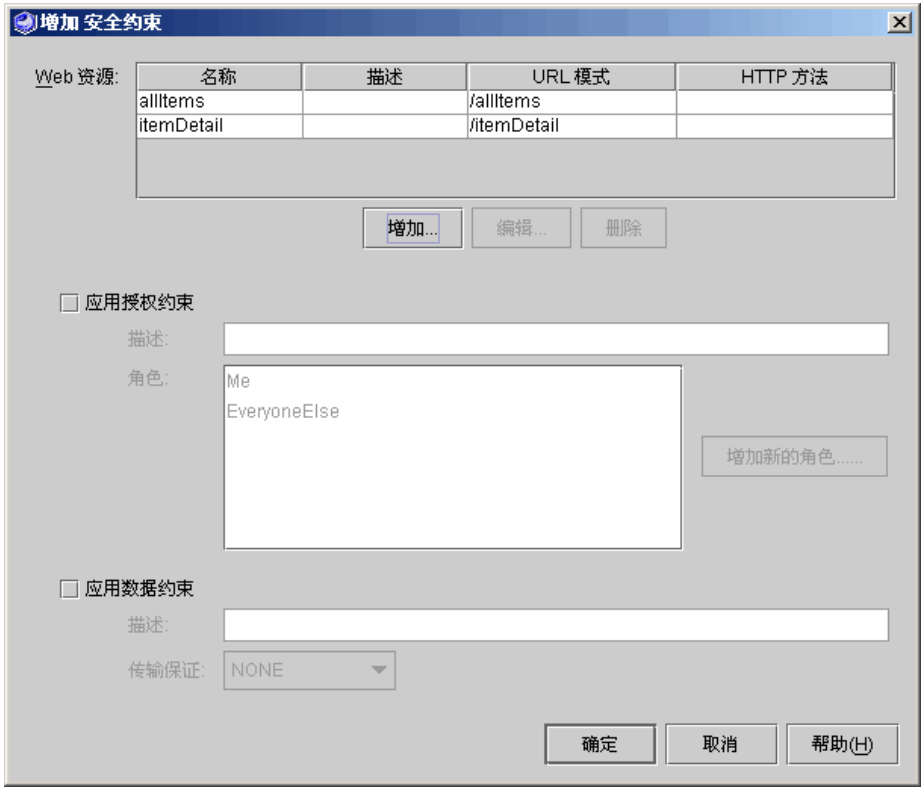

图 7-3 [增加安全限制] 对话框中的 `allItems` 资源

5. 通过指定访问资源的限制来确保资源的安全性。

   a. 单击资源以选中它。

   b. 使用 [增加安全限制] 对话框中的字段来描述访问资源的限制。

c. 要使用安全角色作为限制，请单击 [应用授权限制] 复选框，然后选择 [角色] 字段中的一个或多个角色。

在图 7-4 中， `allItems` 资源已被选定，同时 `EveryoneElse` 角色也被选定。这些选项指定 `allItems` 资源必须由 `EveryoneElse` 角色来调用。使用其他角色的调用方将遭到拒绝。

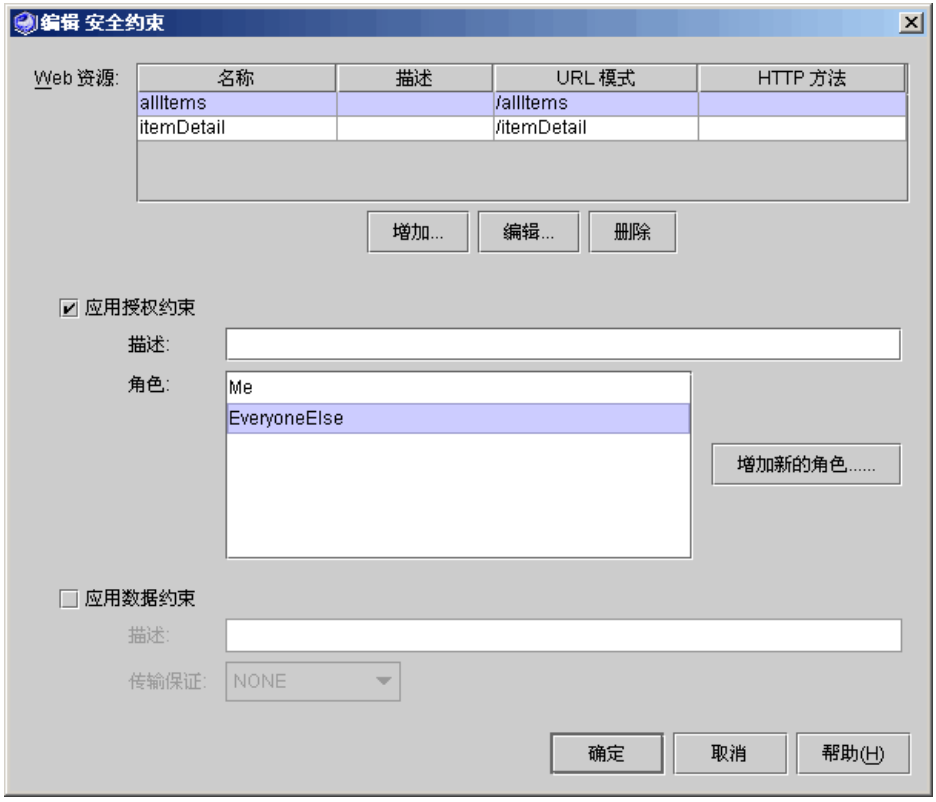

图 7-4 为名为 `allItems` 的 web 资源指定限制

d. 单击 [确定] 保存指定的限制，然后返回 [安全限制] 属性编辑器。

6. 再次单击 [确定] 保存刚才的操作，然后返回属性表单。

Web 模块的可编程安全性

如果模块中的任何 web 组件需要使用可编程安全性，您需要将方法代码中的安全角色引用映射到模块属性表单中声明的安全角色。

使用可编程安全性特性的 web 组件包含直接访问调用方的凭证的代码，并在应用程序服务器运行声明性安全性机制之前就对其进行验证。编码范例 7-1 显示了方法代码中的少数几行，该代码使用了名为 `roleRefMe` 的安全角色引用。

编码范例 7-1 使用 `roleRefMe` 安全角色引用的方法代码

```
...
context.isCallerInRole(roleRefMe);
...
```

`roleRefMe` 这样的安全角色引用是实际引用名称的占位符。在模块级声明角色以前就已完成了代码的编写，然而那时实际角色名称仍是未知的。组装模块时将声明安全角色引用，并将其映射到已声明的安全角色。

要声明安全角色引用并将其映射到安全角色：

1. **右键单击 web 模块的 web 节点，选择 [属性] →[部署] 标签 →[Servlet] → 省略号 (...) 按钮。**

   [Servlet] 属性编辑器打开。

2. **选择包含安全角色引用的 servlet，单击 [编辑] 按钮。**

   [编辑 Servlet] 对话框打开。

3. **单击 [安全角色引用] 字段，然后单击 [增加]。**

   [增加安全角色引用] 对话框打开。

4. **声明安全角色引用，并将其映射到角色。**

   a. **在 [角色引用名称] 字段中，输入方法代码中使用的角色引用名称。**

   b. **在 [角色引用链接] 字段中，输入即将与角色引用名称相链接的现有安全角色名称。**

   c. **单击 [确定] 关闭此对话框，然后返回 [编辑 Servlet] 对话框。**

图 7-5 显示了 [编辑 Servlet] 对话框。名为 roleRefMe 的安全角色引用被映射到名为 Me 的角色。

图 7-5 映射到角色 Me 的名为 roleRefMe 的安全角色引用

当执行方法代码时，会将角色引用映射到角色 Me 并测试调用方的凭证。安全角色必须在执行此类映射之前在模块属性中声明。

EJB 模块的安全性

EJB 模块有多个属性编辑器和对话框用于声明安全角色并将其映射到企业 bean 方法。本节涵盖了这两个任务中的属性编辑器和对话框：

- 为 EJB 模块声明安全角色
- 将企业 bean 方法映射到安全角色

每一个任务都在相应的节中说明。

声明 EJB 模块安全角色

为 EJB 模块声明安全角色 :

1. **右键单击 EJB 模块节点并选择 [属性] →[安全角色] →省略号 (...) 按钮。**

   [安全角色] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加安全角色] 对话框打开。

3. **输入定义角色的值。**

   a. **在 [名称] 字段中，输入角色名称。**

   b. **在 [描述] 字段中，输入对于该角色的描述。**

      当您需要合并同一应用程序中组装的各个模块的安全角色时，安全角色的描述就显得尤为有效。当您将安全角色映射到部署环境中的用户和组时，该描述也是十分有用的。

c. **单击 [确定]。**

[增加] 对话框关闭，并返回到 [安全角色] 属性编辑器。图 7-6 显示声明 `Me` 和 `EveryoneElse` 这两个角色以后的 [安全角色] 属性编辑器。

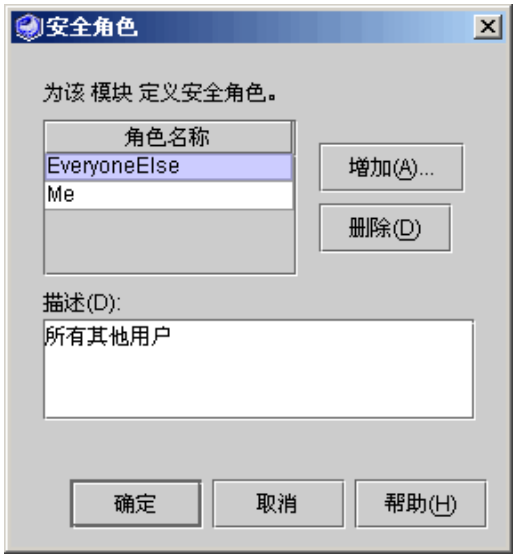

图 7-6 EJB 模块的 [安全角色] 属性编辑器

将安全角色映射到方法权限

在为模块声明了安全角色以后，可以使用这些角色来限制模块中的企业 bean 方法的访问。

注 – 要映射*已包含的 EJB 节点*的安全角色，它们作为 EJB 模块中的子节点，代表了模块中的企业 bean。

要将安全角色映射到方法权限：

- **右键单击已包含的 EJB 节点并选择 [属性] →[方法权限] →省略号 (...) 按钮。**

  [方法权限] 属性编辑器打开。

  [方法权限] 属性编辑器是一个表，其中的行是企业 bean 方法，列是模块中声明过的安全角色。图 7-7 显示了 `CatalogManagerBean` 的属性编辑器。在 `CatalogData` EJB 模块中声明的两个安全角色 `EveryoneElse` 和 Me 分别以列的形式表示。

图 7-7 EJB [方法权限] 属性编辑器

可以有许多不同的方法来使用 [方法权限] 属性编辑器。

- 上端面板中的按钮允许全局应用权限。您可以允许任何用户调用方法或禁止一切访问。
- 如果选择 [为所有方法访问设置角色]，将能获得更精确地控制。您可以使用按钮下方的小表，从模块中已声明的角色中进行选择。如果选定某一列，则表示相应的角色将被允许执行所有的企业 bean 方法。在图 7-7 中 `EveryoneElse` 列被选定，这样，使用这一角色的用户可以执行任何的企业 bean 方法。 `Me` 列未被选定，这样，具有 `Me` 角色的用户将无法执行所有的企业 bean 方法。
- 使用下方的表能够获取最精确的控制。单击某行，这样就可以只定义一种方法的权限。为某一方法设置权限与为编辑器中的其他方法设置权限是完全独立的。

  例如，您可以单击第二行的 `getAllItems` 方法，并将 [访问] 字段设置为 [所有用户]。这样任何角色都可以执行 `getAllItems` 方法。然后您可以单击代表另一个方法的行，将其 [访问] 字段设置为 [设置角色]，并为选定的方法单独设置角色。

表下方的按钮为在表中进行编辑提供了多种快捷方式。

EJB 模块的可编程安全性

如果模块中的任何企业 bean 使用可编程安全性，您就需要将方法代码中的安全角色引用映射到 EJB 模块属性表单中声明的安全角色。

使用可编程安全性特性的企业 bean 包含直接访问调用方的凭证的代码，并在应用程序服务器运行声明性安全性机制之前就对其进行验证。编码范例 7-2 显示了方法代码中的少数几行，该代码使用了名为 `everyOne` 的安全角色引用。

编码范例 7-2 使用 `everyOne` 安全角色引用的方法代码

```
...
context.isCallerInRole(roleRefMe);
...
```

安全角色引用是实际引用名称的占位符。在模块级声明角色以前就已完成了代码的编写，然而那时实际角色名称仍是未知的。组装模块时将声明安全角色引用，并将其映射到已声明的安全角色。

要声明安全角色引用并将其映射到安全角色：

1. **右键单击 EJB 逻辑节点，选择 [属性] →[引用] 标签 →[安全角色引用] → 省略号 (...) 按钮。**

   [安全角色引用] 属性编辑器打开。

2. **单击 [增加] 按钮。**

   [增加安全角色] 对话框打开。

3. 声明安全角色引用，并将其链接到安全角色：

   a. 在 [名称] 字段中，输入方法代码中使用的安全角色引用名称。

   b. 在 [安全角色链接] 字段中，输入安全角色名称。也可以选择将该字段保留为空白，这样引用处于已声明但未链接的状态。

   c. 单击 [确定]。

   [增加安全角色] 对话框将关闭，并返回 [安全角色引用] 属性编辑器。图 7-8 中显示的 [安全角色引用] 属性编辑器包含有编码范例 7-2 中使用的名为 `everyOne` 的安全角色引用。这一角色尚未链接。

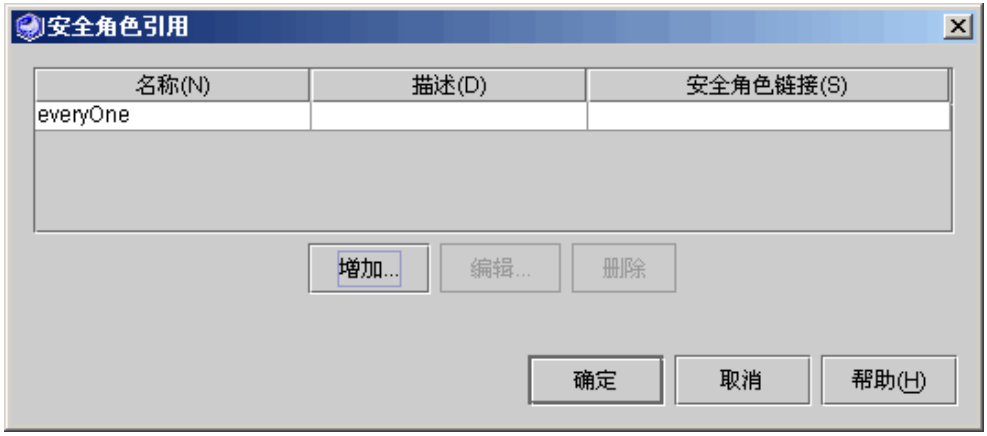

图 7-8 声明的安全角色引用 `everyOne`

在本方案中，在 EJB 模块的 [安全角色] 属性编辑器中链接 `everyOne` 安全角色引用。当企业 bean 组装到 EJB 模块中时就要执行该映射。

要在 EJB 模块的 [安全角色引用] 属性编辑器中链接安全角色引用：

1. **右键单击 EJB 模块节点并选择 [属性] →[安全角色引用] →省略号 (...) 按钮。**

   [安全角色引用] 属性编辑器打开。

2. **评估模块中的引用。**

   EJB 模块的 [安全角色引用] 属性编辑器显示了模块中的所有安全角色引用以及它们的链接状态。在 图 7-9 中， `everyOne` 引用处于未链接的状态。

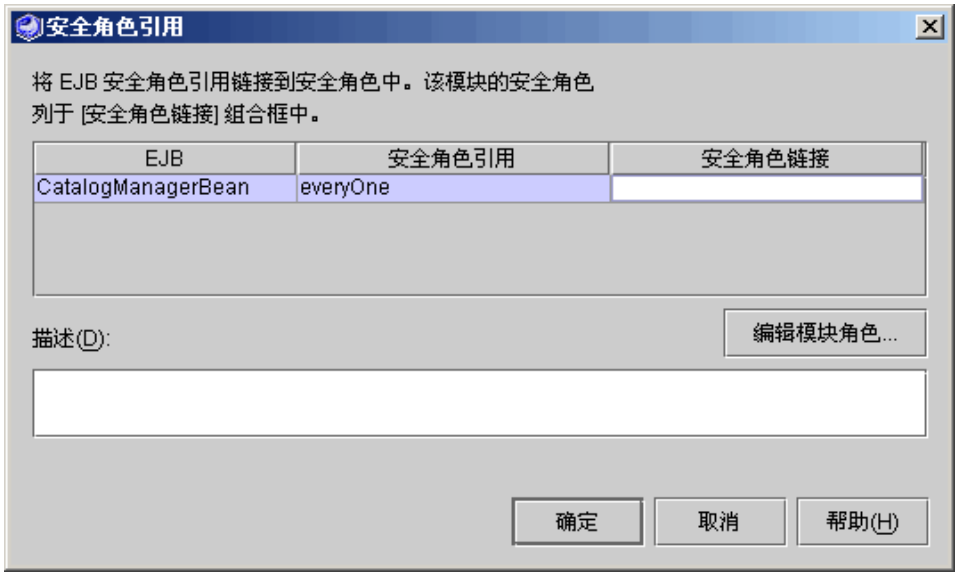

图 7-9 EJB 模块属性编辑器中的 `everyOne` 安全角色引用

3. **链接未链接的角色。**

   a. 单击引用以选定它。

   b. 在 **[** 安全角色链接 **]** 字段中，选择合适的安全角色。

   c. 单击 **[** 确定 **]**。

图 7-10 显示了 EJB 模块的 [安全角色引用] 属性编辑器，名为 `everyOne` 的安全角色引用被映射到名为 EveryoneElse 的安全角色。

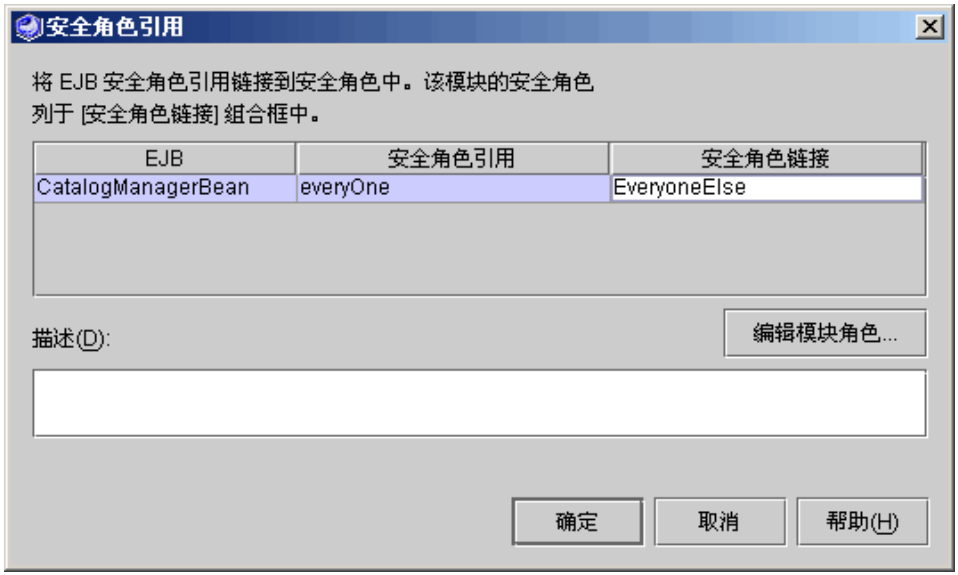

图 7-10 EJB 模块的 [安全角色引用] 属性编辑器

J2EE 应用程序的安全性

在组装 J2EE 应用程序时，需要决定该程序是否有安全性。请注意下列情况：

- 如果应用程序中有一个或多个模块没有定义安全角色，就需要在模块级别定义安全角色。请参阅第 100 页的 “Web 模块的安全性”和第 107 页的 “EJB 模块的安全性”。
- 如果模块有安全性，在不同的模块中可能含有类似的角色，但其名称不同。如果有这样的情况存在，需要在 J2EE 应用程序的属性表单中将所有等价的角色映射到同一个角色。

要在 J2EE 应用程序级别映射安全角色：

- **右键单击应用程序节点并选择 [属性] →[安全角色] →省略号 (...) 按钮。**

  这时将打开 J2EE 应用程序的 [安全角色] 属性编辑器。图 7-11 中显示了此编辑器以及模块中定义的安全角色。

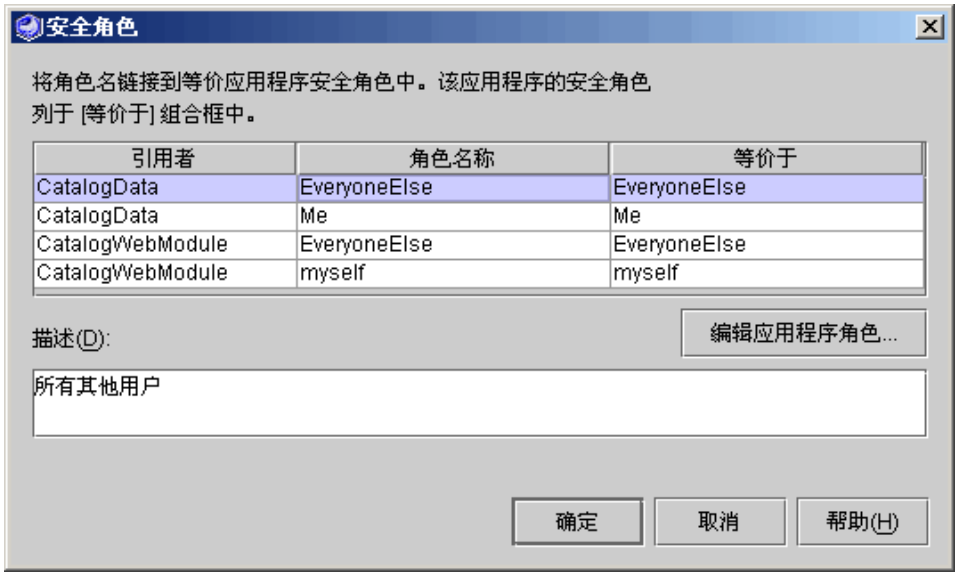

图 7-11 J2EE 应用程序 [安全角色] 属性编辑器中的安全角色

其中前两行显示的是模块中声明的安全角色。每一个角色都通过其所在模块以及角色名称来标识。每一个模块级安全角色都被映射到缺省的应用程序级角色。应用程序级角色具有与模块级角色相同的名称。应用程序级角色显示在 [等价于] 列中。

例如，编辑器中显示的第一个安全角色是 `CatalogData` 模块中的 `EveryoneElse`。该角色被映射到名为 `EveryoneElse` 的应用程序级角色。

图 7-11 中显示了安全角色映射的差异。`CatalogWebModule` 有一个名为 `myself` 的角色，而 `CatalogData` 模块有一个名为 `Me` 的角色。这些角色是等价的，因而最好只使用一个应用程序级角色。

在 图 7-12 中，这种差异可以通过将角色 myself 重新映射到角色 Me 来解决。

图 7-12 名为 myself 的角色被映射到名为 Me 的角色

模块级角色 Me 和 myself 现在被映射到同一个名为 Me 的应用程序级角色。

您也可以在应用程序级创建一个全新的角色，然后将多个模块级角色映射到它。假设在应用程序的某模块中有一个名为 sa 的角色以及另一个名为 sadmin 的角色。您想要通过建立一个名为 sysadmin 的应用程序级角色来解决这一命名上的差别。要实现这一想法，可以单击 [编辑应用程序角色] 并声明一个名为 sysadmin 的新的应用程序级角色。

在完成 sysadmin 角色的声明以后，返回 [安全角色] 属性编辑器。通过单击两个模块级角色的 [等价于] 列，可以对其执行重新映射操作。这样将会显示应用程序级角色，选择 sysadmin 角色。

第 8 章

部署并执行 J2EE 模块和应用程序

IDE 部署和执行特性支持企业应用程序的重复开发。您可以开发、组装、部署以及执行应用程序而不必离开 IDE。

执行应用程序之后，您可以修改源代码或属性，重新部署并且再次执行该应用程序。除非测试暴露出组装中存在的问题，否则在重新部署前不必重新组装该应用程序。

本章描述了从 IDE 内部中部署并且执行已组装的应用程序的基本知识。

服务器的可视化表示

要将应用程序部署到应用程序服务器，您需要与应用程序服务器进行交互。为了简化与应用程序服务器的交互， IDE 将应用程序服务器表示为资源管理器窗口中的节点。

和其它资源管理器窗口节点一样，应用程序服务器节点也有属性表单和菜单命令。您可以使用这些属性表单和菜单命令从 IDE 内部来部署并且执行应用程序。此外取决于使用的应用程序服务器产品，您还可以管理应用程序服务器。

本节标识并且描述了服务器节点，此外也描述了使用这些节点所执行的基本任务。

服务器注册节点

图 8-1 显示资源管理器的 [运行环境] 标签以及用来对服务器进行配置、部署及执行的节点。顶层节点是 [服务器注册] 节点。该节点将其它与服务器相关的节点分组，而自身没有任何命令或属性。

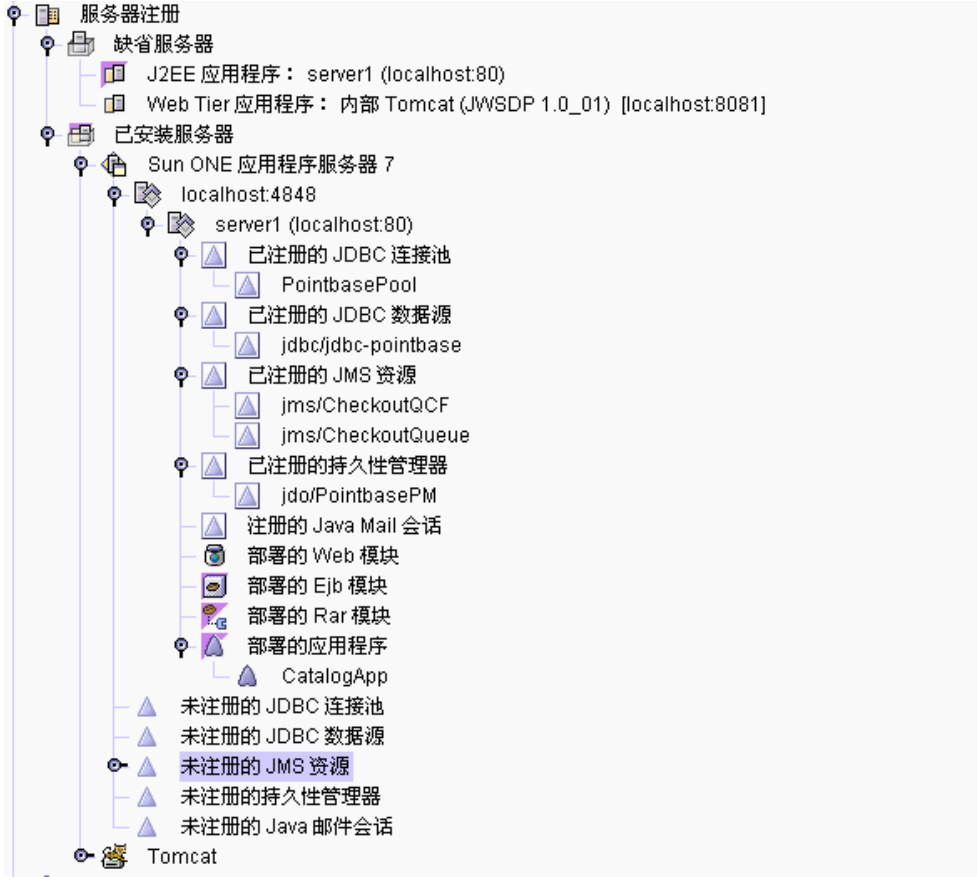

图 8-1 服务器注册节点

图 8-1 中的 [服务器注册] 来自于 Microsoft Windows 平台上的 IDE 安装。该特定安装是用户使用管理或超级用户特权进行的独立安装。系统管理员为没有管理特权的用户安装 IDE 时，节点显示的主机名和端口号是不一样的。主机名和端口号在多用户安装中也不一样。

想要了解不同安装选项的完整描述，请参阅 *Sun ONE Studio 5，Standard Edition Getting Started Guide*。

已安装的服务器节点

已安装的服务器节点将服务器产品节点分组，它自身没有任何命令或属性。在图 8-1 中已安装的服务器节点具有表示两个服务器产品的子节点，这两个产品包括在 IDE、 Sun ONE 应用程序服务器及 Tomcat web 服务器的大多数安装中。

服务器产品节点

已安装服务器节点下面是表示特定 web 及应用程序服务器产品的节点。每个服务器产品节点表示一个已安装的 IDE 服务器插件。

服务器产品节点具有上下文菜单和属性表单，每个服务器产品节点的能力由服务器产品和插件模块决定。

不同服务器产品的过程是不同的，但是通常情况下，要使用应用程序服务器，您需要配置合适的服务器节点来识别服务器产品特定的安装。此外也可以使用服务器产品节点来创建服务器实例。

在图 8-1 中已安装的服务器节点具有表示两个插件的子节点，这两个产品包括在 IDE、Sun ONE 应用程序服务器 及 Tomcat web 服务器的大多数安装中。下一节描述了 Sun ONE 应用程序服务器 7 节点及其子节点。

关于设置其它应用程序服务器产品和 IDE 一起工作的信息，请参阅 *Sun ONE Studio 4，Enterprise Edition for Java Getting Started Guide*。

Sun ONE 应用程序服务器节点

本节标识并描述在资源管理器中表示 Sun ONE 应用程序服务器的节点。

Sun ONE 应用程序服务器 7 节点

Sun ONE 应用程序 7 服务器节点用于管理应用程序服务器。

安装 IDE 以创建应用程序服务器域，同时创建管理应用程序服务器域的管理服务器实例。在大多数情况下，您可以在已安装的应用程序服务器域及管理服务器中完成所有工作。

- 您可以在安装 IDE 所创建的应用程序服务器域中部署并执行应用程序。
- 您可以使用安装 IDE 所创建的管理服务器实例来管理该域。

在图 8-1 中显示的 [服务器注册] 包含了安装 IDE 所创建的应用程序服务器域及管理服务器。管理服务器实例由标为 localhost:4848 的 Sun ONE 应用程序服务器 7 节点下的节点表示。您可以打开管理服务器的属性表单来查看应用程序服务器域名。

如果需要并且有超级用户或管理特权，您可以使用 Sun ONE 应用程序服务器 7 节点来创建附加的应用程序服务器域和管理服务器。

管理服务器节点

Sun ONE 应用程序服务器 7 节点的下面是管理服务器节点。在图 8-1 中，管理服务器节点标为 localhost:4848。管理服务器节点表示 Sun ONE 应用程序服务器管理服务器的实例。每个管理服务器实例都管理应用程序服务器域。

使用 Sun ONE 应用程序服务器安装 IDE 并且管理 Sun ONE 应用程序服务器来创建应用程序服务器域和管理服务器实例。所使用的安装类型决定了如何创建应用程序服务器域和服务器实例。各种可能性将在下表中进行描述：

- 如果使用超级用户或管理员特权来安装 IDE，此安装将会创建初始服务器域及管理服务器实例。图 8-1 显示了安装进程所创建的服务器域及管理服务器实例。
- 如果您有标准用户特权，那么系统管理员将会创建服务器域及管理服务器实例供您使用。您可以使用管理服务器节点来管理域。

管理服务器节点显示的主机名是应用程序服务器运行所在的机器的名称。图 8-1 显示了独立的、单用户安装并且应用程序服务器在本地主机上运行。在多用户安装中，应用程序服务器可在其它机器上运行。

管理服务器显示的端口号是与管理服务器通讯的端口号。创建应用程序服务器域及管理服务器实例时才设置此端口号。图 8-1 显示了在 Microsoft Windows 平台上单用户安装的缺省端口号。

使用管理服务器节点执行的某些任务是在管理服务器控制的服务器域中启动及停止服务器实例。

要启动管理服务器和服务器实例：

1. **右键单击 [管理服务器] 节点然后选择 [启动]。**

   进度监视器窗口打开。管理服务器启动时，进度管理器关闭。 IDE 在管理服务器节点下显示服务器实例节点。

2. **右键单击服务器实例节点然后选择 [状态]。**

   [状态] 对话框打开，其中的 [状态] 字段显示服务器实例的状态已停止。

3. **单击 [启动服务器]。**

   [状态] 对话框显示消息以告诉您它正在启动应用程序服务器实例。服务器实例启动时， [状态] 字段显示为正在运行。

4. **单击 [关闭]。**

   服务器实例已可使用。

上下文菜单列出使用管理服务器节点执行的其它任务。

服务器实例节点

在管理服务器节点以下是应用程序服务器实例节点。在图 8-1 中，应用程序服务器实例节点标为 server1(localhost:81)。应用程序服务器实例节点表示服务器实例。

部署模块或应用程序时，您将其部署到特定的服务器实例。在部署并且执行之前必须有服务器实例节点，并且该节点所表示的实例必须正在运行。

图 8-1 显示了安装 IDE 所创建的服务器实例。您也可以通过管理应用程序服务器域来创建服务器实例。

注册的资源节点

服务器实例节点以下是表示命名资源的节点集，命名资源可以被服务器实例中运行的应用程序使用。图 8-1 显示了由本书中方案使用并且创建的资源的节点：

- 在标为 [已注册的 JDBC 连接池]、[已注册的 JDBC DataSource] 及 [已注册的持久性管理器] 的节点以下是表示已安装的 PointBase 数据库 （命名为 `sample`）的节点。这些资源在安装 IDE 时预配置。在其中一个方案中， `CatalogData` EJB 模块配置为通过输入资源的名称来使用 PointBase `sample` 数据库。关于配置 `CatalogData` EJB 模块的过程，请参阅第 60 页的 “指定实体企业 bean 的数据源”。
- 在标为 [已注册 JMS 资源] 的节点以下是表示本手册方案中所创建的队列及队列连接工厂资源的节点。关于创建及注册这些资源的过程，请参阅第 81 页的 “设置应用程序服务器”。关于使用这些资源来配置应用程序的过程，请参阅第 83 页的 “Web 模块编程”及第 90 页的 “EJB 模块编程”。

已部署的应用程序节点

在已注册资源节点下面是表示部署到服务器实例的模块及应用程序的节点集。图 8-1 显示了命名为 `CatalogApp` 的应用程序的节点。本手册中的方案包含了 `CatalogApp` 的编程及部署。关于部署 `CatalogApp` 的过程，请参阅第 69 页的 “创建 J2EE 应用程序”。

未注册的资源节点

在已部署的应用程序节点下是表示未注册的服务器资源的节点集。这些节点是管理服务器节点的子节点并且表示了服务器实例还没有注册的资源。

这些节点具有用来创建并且注册新资源的菜单命令。本书中的方案使用 [未注册的 JMS 资源] 节点来创建队列及队列连接工厂。关于创建资源的过程，请参阅第 81 页的 “设置应用程序服务器”。

缺省服务器节点

这些节点表示目前被指定为缺省服务器实例的服务器实例。部署应用程序时，除非在应用程序属性表单中另外指定，它将被部署到缺省的服务器实例。

在图 8-1 中，缺省节点显示 J2EE 应用程序的缺省服务器是 Sun ONE 应用程序服务器。

要使服务器实例成为缺省的服务器：

- **右键单击服务器实例节点然后选择 [成为缺省设置]。**

特定服务器属性

您所使用的模块及应用程序具有属性表单，可以使用属性表单来标识模块及应用程序需要应用程序服务器提供的那些服务。

很多属性表单具有特定服务器标签。特定服务器标签列出了为特定服务器产品定义的属性。

例如，图 8-2 显示了 `CatalogData` EJB 模块的属性表单。[Sun ONE AS] 标签被选定。此标签具有 CMP 资源属性，您可以使用该属性来为 `CatalogData` 模块中的 CMP 实体标识数据源。注意，在图 8-2（`jdo/PointbasePM`）中命名的数据源在图 8-1 中作为 Sun ONE 应用程序服务器实例的已注册持久性管理器出现。

图 8-2 EJB 模块的 Sun ONE AS 标签

使用服务器实例节点来部署并执行

本节将简单介绍从 IDE 内部部署并且执行 J2EE 应用程序的过程。

要部署并且执行应用程序：

1. **从组装的 J2EE 应用程序开始，查看组装完整性的应用程序。**
2. **选择应用程序服务器实例。**

   该应用程序节点具有 [应用程序服务器] 属性。 [应用程序服务器] 属性的初始化设置是缺省应用程序服务器。如果继续该设置， IDE 就将应用程序部署到目前指定为 J2EE 应用程序的缺省服务器的服务器实例。

   您也可以为本属性打开属性编辑器并且按名称来选择服务器实例。属性编辑器是一个浏览器对话框，使您能够查看服务器注册中的所有服务器实例并且选择其中之一。
3. **右键单击应用程序节点然后选择 [执行] 命令来部署并且执行应用程序。**

   这将启动部署进程，请在输出窗口中监视进程。部署完成时， IDE 将在应用程序服务器环境中执行应用程序。您所看到的内容取决于应用程序，例如，如果应用程序包含 web 模块，那么应用程序服务器将启动 web 浏览器并且打开应用程序的欢迎页。
4. **您也可以使用独立步骤来部署并且执行应用程序。右键单击应用程序然后选择 [部署] 命令。部署完成时，您自己就可以执行应用程序。**

   例如，如果应用程序包含 web 模块，那么您就可以在 IDE 以外启动 web 浏览器并且打开应用程序的欢迎页。

附录 A

IDE 如何支持 J2EE 模块及应用程序的部署

本附录将简要描述服务器插件。服务器插件是 IDE 用来部署并且执行 J2EE 模块和应用程序的机制，该插件提供以下 IDE 功能：

- 服务器产品节点以及它们的子节点
- 特定服务器属性表单
- 部署及执行菜单命令

本附录主要说明部署及执行菜单命令，它描述了部署应用程序所需的处理并且解释插件如何执行该处理。

如果您了解部署工具的工作原理，那么您就可以有效地使用它。使用部署工具的过程包含在本书的许多方案中。

部署进程

部署是将模块或应用程序的可执行窗体发送到 J2EE 应用程序服务器的过程。发送到服务器的可执行窗体作为包含构成模块或应用程序的源文件编译版本的归档，描述归档内容和组织的部署描述符与编译文件在一起。归档文件安装在应用程序服务器管理的目录下。

执行模块或应用程序时，应用程序服务器在其控制的进程中执行应用程序已安装的副本，该进程提供所需的运行环境。

要想部署成功，应用程序源文件必须以与特定的应用程序服务器产品相兼容的方式进行编译。部署描述符必须包括特定的应用程序服务器产品所需的所有信息。

这些要求通过为产品标识目标应用程序服务器、编译源文件以及为目标应用程序服务器生成特定的部署描述符来满足。

服务器插件概念

由于要让 IDE 能够部署多种 web 及应用程序服务器，所以才有了服务器插件的概念。插件是管理 IDE 与特定的服务器产品之间交互操作的 IDE 模块。部署应用程序时，您需要选择它将要部署到的服务器。 IDE 使用合适的插件来处理 [部署] 命令，这使得插件能够为服务器部署工具生成合适的命令并且在其传送到服务器的文件中包括合适的特定服务器部署描述符。这一部分如图 A-1 所示。

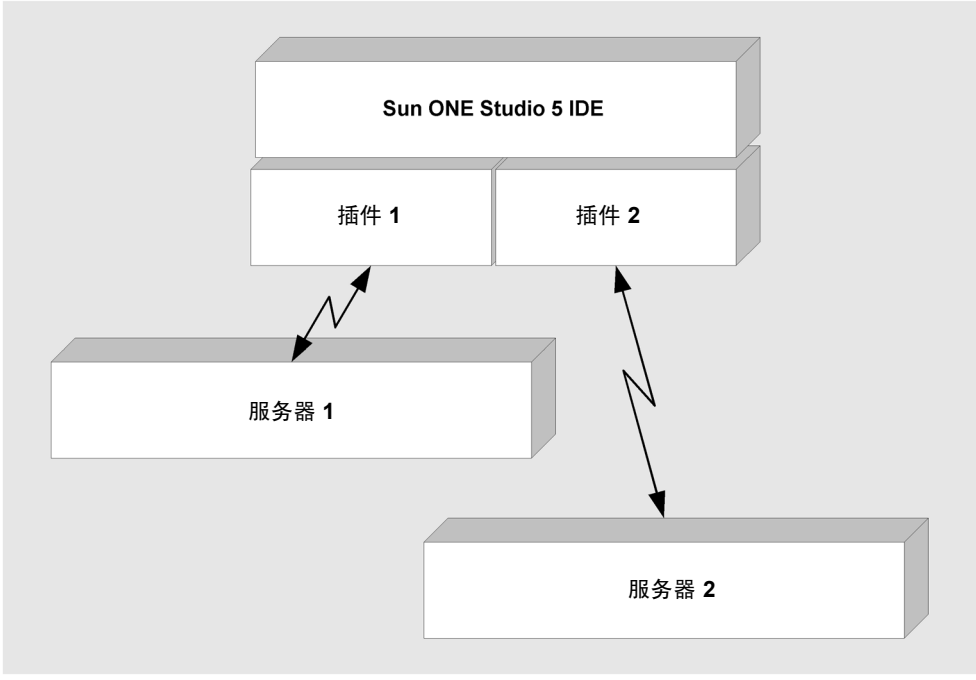

图 A-1 服务器插件使得 IDE 能够与 J2EE 运行环境通讯

插件为部署应用程序的应用程序开发人员提供：

- 插件在 IDE 资源管理器窗口中的可视化表示。每一个插件都由一个服务器产品节点表示。关于服务器产品节点显示及使用的更多信息，请参阅第 119 页的 “服务器产品节点”。
- 运行服务器实例作为服务器产品节点的子节点的可视化表示。您可以选择任何在资源管理器窗口中表示的服务器实例作为部署目标。关于服务器实例节点显示及使用的更多信息，请参阅第 121 页的 “服务器实例节点”。
- 组件、模块及属性表单上的特定服务器标签这些标签显示了服务器产品所需的非标准属性并且提示开发人员服务器产品所需的值。
- 处理特定于选定服务器 [部署] 命令的机制。该处理将在下一节中详细描述。

使用插件的部署过程

本节总结了部署并执行应用程序时插件所执行的处理。

1. 组装应用程序。使用属性表单来提供服务器所需的 J2EE 标准部署描述符元素及非标准元素。
2. 组装完应用程序之后，指定目标服务器实例。
3. 选择 IDE 的 [部署] 命令开始部署过程。
4. IDE 标识为应用程序创建 WAR 或 EAR 文件所需的所有文件。这其中包括在部署描述符中标识的 J2EE 组件以及那些文件所使用的 Java 类或静态资源。IDE 标识组件中所有的文件相关性。
5. IDE 标识应用程序正被部署到的服务器产品。
6. 插件为 WAR 或 EAR 文件验证文件。
7. IDE 为应用程序生成 WAR 或 EAR 文件。这其中包括 J2EE 部署描述符、具有特定服务器部署标记的独立文件以及任何远程方法调用所需的桩模块或骨架类。
8. 插件将 WAR 或 EAR 文件传送到服务器。

   插件可以自动清除同一应用程序的早期部署，或者试图解决与已经部署到服务器实例的应用程序发生的冲突，这些操作取决于服务器产品。
9. 服务器根据自己的标准来接管、读取部署描述符及特定服务器部署文件，并且部署 WAR 或 EAR 文件。

该过程完成时， IDE 将自动启动 web 浏览器并且打开应用程序欢迎页。如果选择分别进行部署与执行，那么您可以启动 web 浏览器以及某一应用程序 web 页。

部署除 web 模块及 J2EE 应用程序外的组件

Web 模块和 J2EE 应用程序是实际上唯一能够部署到服务器并且执行的项。但是，您可能想要测试所开发的商业逻辑的较小单元。Sun ONE Studio 5 IDE 可以部署并执行商业逻辑的较小单元，做法是为想要测试的组件自动生成模块及应用程序。它也可以为组件的某些类型生成测试客户端。关于这些功能的更多信息，请参阅《构建 Web 组件》和《构建 Enterprise JavaBeans 组件》。

索引

M

Q

R

S

T

U

W

X

Y

Z