IBM® WebSphere Commerce

商店开发者指南

版本 5.4

IBM® WebSphere Commerce

商店开发者指南

版本 5.4

注:

在使用本资料及其支持的产品之前，请确保阅读『声明』部分中的信息。

第二版（2002 年 5 月）

此版本适用于以下产品:

- IBM® WebSphere® Commerce 商务版 Windows NT® 和 Windows® 2000 版，版本 5.4
- IBM WebSphere Commerce 商务版 AIX® 版，版本 5.4
- IBM WebSphere Commerce 商务版 Solaris 版，版本 5.4
- IBM WebSphere Commerce Studio 商务开发版 Windows NT 和 Windows 2000 版，版本 5.4
- IBM WebSphere Commerce 专业版 Windows NT 和 Windows 2000 版，版本 5.4
- IBM WebSphere Commerce 专业版 AIX 版，版本 5.4
- IBM WebSphere Commerce 专业版 Solaris 版，版本 5.4
- IBM WebSphere Commerce Studio 专业开发版 Windows NT 和 Windows 2000 版，版本 5.4

以及以上所列产品的所有后续发行版和修订版，直到在新版本中另有声明为止。确认您正在使用本产品级别的正确版本。

通过您当地的 IBM 代表或 IBM 分部可订购出版物。以下地址不备有出版物。

IBM 欢迎您提出宝贵意见。您可以用以下任一方法发送您的意见:

1. 通过电子的方式发送到下列网络标识。如果您需要回复，请确保包含您的完整网络地址。

   因特网: torrcf@ca.ibm.com
2. 邮至以下地址:

   IBM Canada Ltd. Laboratory
   B3/KB7/8200/MKM
   8200 Warden Avenue
   Markham, Ontario, Canada L6G 1C7

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

目录

开始之前

《*IBM WebSphere Commerce 商店开发者指南*》提供了关于 WebSphere Commerce 商店体系结构和商店开发过程的信息。同时还特别提供了关于以下主题的详细信息:

- 商店开发选项
- 商店归档文件
- 商店开发工具
- 开发商店前台
- 开发商店数据
- 商店数据体系结构
- 商店数据信息模型
- 向商店添加访问控制
- 封装商店
- 发布商店
- 向商店添加 WebSphere Commerce 功能部件

本书中使用的约定

本书使用以下突出显示约定:

粗体字表示命令或者诸如字段名、按钮或菜单选项之类的图形用户界面（GUI）控件。

等宽字表示完全按显示原样输入的文本示例和目录路径。

*斜体字*用于表示强调和用自己的值替换的变量。

此图标标记技巧 — 可帮助完成任务的附加信息。

NT 表示专用于 Windows NT 的信息。

2000 表示专用于 Windows 2000 的信息。

AIX 表示专用于 AIX 的信息。

Solaris 表示专用于 Solaris Operating Environment 的信息。

400 表示专用于 IBM eserver iSeries™ 400®（以前称作 AS/400®）的信息。

Linux 表示专用于 Linux 的信息。

Business 表示专用于 WebSphere Commerce 商务版的信息。

Professional 表示专用于 WebSphere Commerce 专业版的信息。

查找新信息

以后可能会对本书进行更新。请检查以下 WebSphere Commerce Web 站点以获取更新:

Business

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

Professional http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html

更新可能包括新信息。

第 1 部分 概述

第 1 章 商店体系结构概述

本章提供了对 WebSphere Commerce Server 商店体系结构的简介。

什么是网上商店?

网上商店是使用因特网技术销售或交换商品或者服务的商店。由显示产品并允许顾客购买这些产品的一组 Web 页面组成。主页将顾客带入商店并将其引向产品和服务。在线产品目录将产品组合在一起，并引导顾客进入产品页面，顾客可以在其中找到关于产品的详细信息。在“商家到消费者”商店中，“购物车”页面与实际购物车的角色相同: 您可以向其中添加希望购买的产品，然后在“结帐”页面中进行支付。在“商家到商家”站点中，某些页面让您能够提交订单和报价请求（RFQ）。

商店页面是使用 JavaServer Pages（JSP）技术创建的。每个页面都包含用于静态内容的 HTML、用于处理输入数据和复杂数据显示的客户机端 JavaScript™、用于调用 WebSphere Commerce Server 命令和其它视图的 URL，以及用于生成动态内容的 JSP 标记和 Java™ 代码。WebSphere Commerce Studio 和 WebSphere Commerce 包含的一组贸易数据 bean 可以由 JavaServer Page 文件使用，允许您访问数据库中的信息（例如产品价格或产品属性）。

商店还由创建正常运作的商店所必需的数据库有用资源组成。例如，正常运作的商店必须包括关于产品目录、税款、装运和货币的数据。

商店的组成

网上商店由以下有用资源组成:

- 商店前台

  商店的外部部分或显示给顾客的部分称为商店前台。商店前台由 HTML 页面、JSP 文件、样式表、图像、图形和其它多媒体文件类型等 Web 有用资源组成。

  本指南讨论了创建用于构建商店页面的 JSP 文件时涉及到的概念和任务。关于更多信息，请参阅第 15 页的第 3 章，『开发商店前台』。

- 商店后端

  商店中顾客看不到的部分（允许顾客在商店前台购买产品的命令、定制代码和业务逻辑实现）称为商店后端。

  关于创建业务逻辑或定制代码的更多详细信息，请参阅《*IBM WebSphere Commerce 程序员指南*》。

- 商店数据

  组成商店的数据有用资源。为了正常运作，商店必须准备好数据以支持所有顾客活动。例如，为让顾客能够进行购买，商店必须包含待售商品的产品目录、处理订单的过程、实现请求的库存以及准备好装运过程。您还必须有处理和收取支付的方法。

  创建商店数据中涉及到的概念和任务在第 33 页的第 4 部分，『开发商店数据』中进行讨论。

商店体系结构

WebSphere Commerce 商店体系结构由以下组件组成:

- WebSphere Commerce Server
- WebSphere Commerce Server 实例
- 商店配置

WebSphere Commerce Server

WebSphere Commerce Server 是处理电子交易解决方案的商店及贸易相关功能的服务器。商店前台有用资源和商店后端业务逻辑驻留在 WebSphere Commerce Server 内的 Web 应用程序中。WebSphere Commerce 提供一个缺省的 Web 应用程序（`WCS Stores`）供您使用，或者您可以创建自己的应用程序。

Web 应用程序可以包含一个商店的有用资源，或者多个商店的有用资源。当 Web 应用程序包含多个商店前台和商店后端时，每个商店的有用资源按商店目录（storedir）分开。

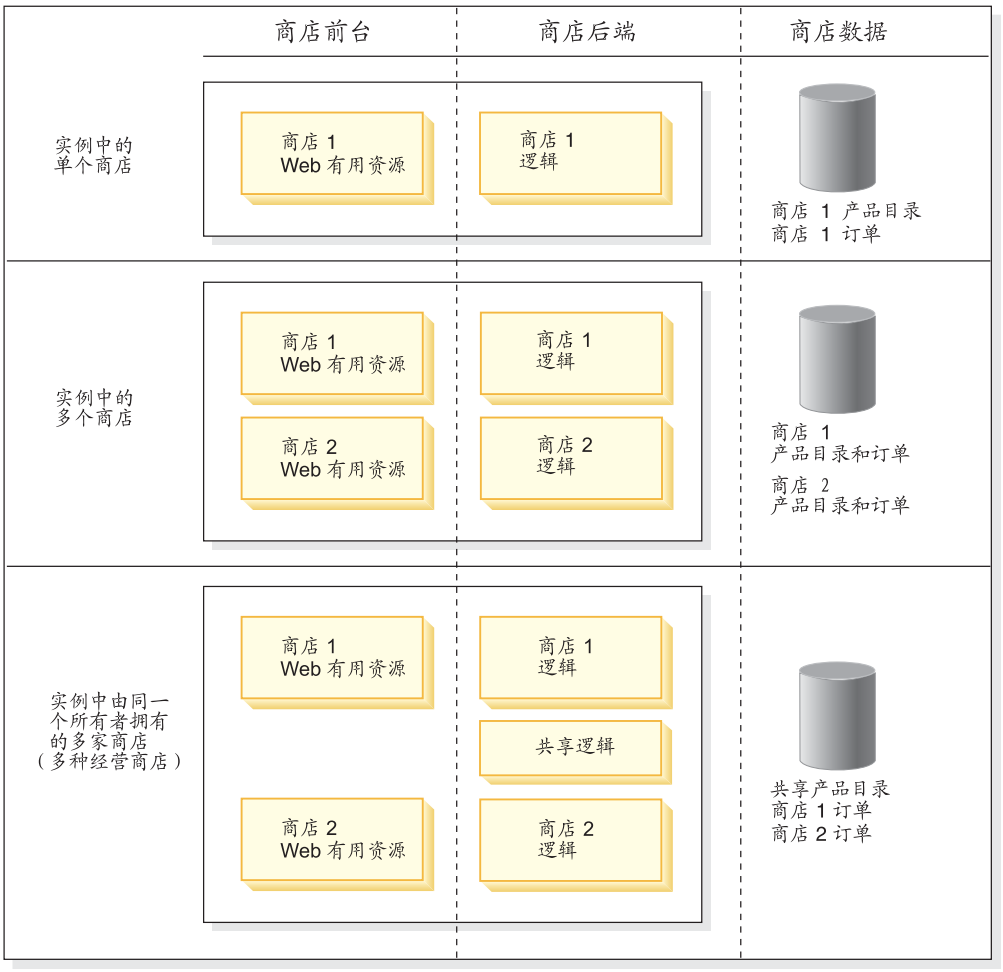
WebSphere Commerce Server 实例

WebSphere Commerce Server 实例是带有关联数据库的 WebSphere Application Server 应用程序。一个实例可以支持多个商店。实例中的所有商店共享同一个数据库，并且可以共享某些类型的数据，例如产品目录、实现或接收。实例中的所有商店还共享同一个 EJB 容器。

商店配置

WebSphere Commerce 支持多种商店配置。即，使用 WebSphere Commerce 可以在实例中创建一个商店，也可以在实例中用单独的商店前台、商店后端和商店数据创建多个商店。或者，可以在实例中用单独的商店前台、共享的商店后端和共享的产品目录

创建多个商店。下图说明了几种可能的商店配置。



注: 每一商店都有其自己的标识。WebSphere Commerce 的许可证对您可以创建的商店数目设置了特定限制。您可以购买附加的权利。关于详细信息，请参阅许可证协议。

第 2 章 商店开发

本章提供了 WebSphere Commerce 中商店开发过程的概述。

商店开发选项

WebSphere Commerce 提供了多种开发商店的选项:

- 基于样本创建商店
- 通过开发新的商店有用资源创建商店
- 使用样本商店和新的商店有用资源的组合创建商店

基于样本创建商店

在 WebSphere Commerce 中，创建网上商店最迅捷最容易的方法是复制随 WebSphere Commerce 提供的样本商店之一，然后对其进行定制以满足需要。

样本商店

WebSphere Commerce 包括多个完全可正常运作的网上样本商店，您可以将其用作创建您自己商店的基础。这些样本（包括“商家到消费者”商店和“商家到商家”商店）实现了当前顶级电子交易站点的很多最常使用的功能，并提供了所有必需的商店有用资源。关于随 WebSphere Commerce 提供的网上商店的更多信息，请参阅 WebSphere Commerce 联机帮助。

为什么从样本商店开始?： 尽管可以用 WebSphere Commerce 创建一个全新的网上商店，但是使用样本商店之一的副本作为您自己商店的基础使您可以更快地创建正常运作的商店。

WebSphere Commerce 需要将特定数据装入 WebSphere Commerce Server 数据库以创建正常运作的商店，而且要求此数据必须以模式确定的顺序装入。由于样本商店在顺序和结构中包含了 WebSphere Commerce Server 数据库所需的所有必填数据，因此使用样本商店作为您自己商店的基础在初始创建期间会节省大量时间。

创建样本商店的副本之后，可以或多或少地对其进行编辑，这取决于您商店的需要。例如，您可能仅需要使用 WebSphere Commerce 提供的工具编辑数据并使用 WebSphere Commerce Studio 更改商店页面的外观。或者可能需要直接编辑 XML 文件或数据库以对数据进行更全面的更改并重写商店页面以更改商店的流程和功能。关于编辑商店的更多信息，请参阅 WebSphere Commerce 联机帮助主题 *“更改商店数据库有用资源”*。

WebSphere Commerce 还提供了多个参考商店，设计为用作突出显示的功能的代码样本。参考商店是一个网上商店，包含网上商店选定功能（例如赠券）的完全可正常运作的代码。以下地址提供了参考商店:

Business `http://www.ibm.com/software/webservers/commerce/wc_be/downloads.html`

Professional `http://www.ibm.com/software/webservers/commerce/wc_pe/downloads.html`

关于基于样本创建商店的更多信息，请参阅 WebSphere Commerce 联机帮助。

通过开发新的商店有用资源创建商店

并非每个人都希望基于样本商店创建他们的商店。例如，如果您商店页面的流程明显不同于提供的所有样本，或者如果您计划对 WebSphere Commerce Server 数据库模式进行大量定制，则您可能希望通过开发自己的商店前台、商店后端和商店数据有用资源创建商店。请参阅第 9 页的『商店开发工具』以获取随 WebSphere Commerce 提供的工具的列表。

使用样本商店和新的商店有用资源的组合创建商店

使用样本商店和开发新的商店有用资源两者的组合可能是最适合您的商店开发方法。例如，如果某个样本商店中的某些数据库有用资源与您的商店需要非常匹配，但是该商店页面的流程与您的不符，则开发全新的 Web 有用资源时可以从该商店复制数据库有用资源并进行定制。

商店归档文件

WebSphere Commerce 包含的样本商店以商店归档文件格式提供。商店归档文件（.sar）是已压缩的归档文件（例如 ZIP 文件），其中包含创建商店所需的所有有用资源。它主要用作以可方便复制的格式封装和传递商店的媒介，然后用作创建新商店的基础。商店归档文件只需发布到 WebSphere Commerce Server 上即可创建可以查看、浏览和购物的正常运作的商店。

通常，商店归档文件由以下文件组成:

- Web 有用资源: 创建商店页面的文件，如 HTML 文件、JSP 文件、图像、图形和包含文件。Web 有用资源在商店归档文件中作为压缩文件组合在一起。
- 属性资源绑定（可选）: 包含商店页面的文本。如果商店支持多种语言，则资源绑定将包含多个绑定；也就是说，一种语言一个绑定。
- 商店数据有用资源: 要装入到数据库中的数据。商店数据有用资源包括竞销、产品目录条目、货币、实现信息、定价、装运、存储和税务信息等数据。关于商店数据有用资源的更详细列表，请参阅第 33 页的第 4 部分，『开发商店数据』。

  随 WebSphere Commerce 提供的样本商店归档文件中的商店数据库有用资源是精心编成、对装入程序软件包有效的 XML 文件。商店归档 XML 文件计划成为可移植文件并且不应包含特定于数据库特定实例的已生成主键。相反，它们使用商店发布时由标识解析器解析的内部别名。这些约定的使用允许多次复制和发布样本商店归档文件。关于更多信息，请参阅第 289 页的附录 B，『创建数据』。

  关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。
- 支付有用资源: IBM Payment Manager 的配置信息。
- 描述符: XML 文件 sarinfo.xml，它描述商店归档文件，包含 Web 有用资源压缩归档文件、资源绑定和商店数据库有用资源 XML 文件三者的名称。sarinfo.xml 文件还包括包含文件和一致性检查文件的名称，以及关于发布过程中所需归档文件的信息。sarinfo.xml 是商店归档文件中唯一的强制文件。

注: Business “多乐五金店”和“新时尚”样本商店归档文件还包含以下文件:

- tools_properties.zip
- tools_xml.zip
- runtime_xml.zip

“商店服务”使用这些文件配置商店。这些文件不得更改、除去或者复制到其它商店。

样本商店归档

样本商店归档文件（.sar）是要作为创建新商店的基础复制和使用的商店归档文件。样本商店归档包括一些允许多次复制和发布这些归档的约定。这些约定包括以下内容:

- 不引用生成的主键或外键: 样本商店归档不包含特定于数据库特定实例的已生成主键。相反，它们使用商店发布时由标识解析器解析的内部别名。关于更多信息，请参阅第 289 页的附录 B, 『创建数据』。

随 WebSphere Commerce 提供的样本商店是样本商店归档文件。这些商店可以从“商店服务”中的“创建商店归档文件”页面中的**样本**列表获取。

确定何时使用商店归档文件

商店归档文件设计为封装和传递商店的媒介。如果希望使用您的商店作为要传递给其他人的样本，将其部署在另一个服务器或平台上，或用其作为创建其它商店的基础，那么请考虑以商店归档文件形式绑定它。

如果您的商店很大部分以其中一个样本商店为基础并且无需对归档进行很多更改，则也可能希望使用商店归档文件。

如果选择使用商店归档文件，且已经使用样本商店创建商店，则您的商店将已经使用商店归档文件格式。然后可以通过确保商店开发期间所作的所有更改都反映在商店归档文件中，维护使用商店归档文件格式的新商店。

也可以将使用其它方法创建的商店封装成商店归档文件。关于创建商店归档文件的更多信息，请参阅第 185 页的第 6 部分, 『封装商店』。

何时不希望使用商店归档文件?

如果您正创建商店的单一实例或对现有 WebSphere Commerce 模式进行大量更改，您可以选择不使用商店归档文件。框外的商店归档文件和 WebSphere Commerce 商店开发工具不立即支持对 WebSphere Commerce 模式进行的更改。如果想在进行大量模式更改后维护商店归档文件，请与 IBM 代表联络以获取更多信息。

商店开发工具

WebSphere Commerce 提供了多种工具帮助您开发商店。使用哪些工具取决于您选择如何开发和封装商店。

开发商店前台的工具

开发商店前台有用资源可能包括定制样本商店页面、用您自己的现有页面替换它们、创建新页面或执行上述三者的组合操作。

WebSphere Commerce 提供了以下工具用以创建或编辑商店前台有用资源:

- WebSphere Commerce Studio: Commerce Studio 包括创建和编辑商店前台有用资源（包括 HTML、图形、多媒体和 JavaServer Pages（JSP）文件）所需的工具。Commerce Studio 中包含的 Page Designer 允许您创建 HTML 或 JSP 以及动画图

像。您还可以配置 WebSphere Commerce Studio 使其使用您选择的另一种 Web 开发工具。关于向 WebSphere Commerce Studio 注册您自己的工具的更多信息，请参阅 WebSphere Studio 联机帮助。

如果计划以商店归档文件格式处理商店，WebSphere Commerce Studio 允许您在保持商店归档文件结构不变的同时将 Web 有用资源从商店归档文件导入 Studio 项目。使用 Studio 工具对 JSP 文件、HTML 文件和图像进行更改后，可以将文件导回 WebSphere Commerce Server 上的商店归档文件并重新发布 Web 有用资源。

如果不在维护商店归档文件，您可以使用 WebSphere Commerce Studio 将这些文件直接发布到正常运作的商店。

- 商店服务: 商店服务中的 “Web 有用资源” 对话框允许您用另一组 Web 有用资源替换商店归档文件中的 Web 有用资源已压缩归档文件，或者将现有的 Web 有用资源下载到选择的位置，您可以在此位置使用自己偏好的 Web 开发工具对其进行编辑。如果正在处理使用商店归档文件格式的商店，则您可以使用 “Web 有用资源” 对话框将已更改的有用资源放回到商店归档文件中。 “配置商店” 页面允许您启用或禁用已发布商店中的 JSP 文件的不同功能。 “商店服务” 当前仅支持配置协作功能: 协作工作空间和客户看管。这些功能仅可用于基于 Business “多乐五金店” 样本商店和 “新时尚” 样本商店的商店中的配置。

关于使用 WebSphere Commerce Studio 和商店服务中的工具创建和编辑商店前台有用资源的更多信息，请参阅 WebSphere Commerce 联机帮助。关于在 WebSphere Commerce 中创建商店前台的更多信息，请参阅第 13 页的第 2 部分，『开发商店前台』。

开发商店数据的工具

您有多种选项可用于在商店中开发和编辑数据库有用资源。

- 商店服务

  商店服务是处理商店归档文件的一组基于浏览器的工具。使用商店服务，您可以基于随 WebSphere Commerce 提供的样本快速创建商店归档文件。一旦创建了商店归档文件， “商店服务” 就允许您执行以下任务:

  – 发布商店归档文件以创建正常运作的商店。
  – 使用 “税款” 笔记本更改税款设置。
  – 使用 “装运” 笔记本更改装运设置。
  – 使用 “商店简要表” 笔记本更改一般商店设置。

  商店服务不允许您编辑商店归档文件中的所有商店数据有用资源。关于您可以使用商店服务编辑的有用资源列表，请参阅 WebSphere Commerce 联机帮助主题 *“更改商店数据库有用资源”*。要编辑商店归档文件中的其它有用资源，请直接编辑 XML 有用资源。

  关于使用商店服务的更多信息，请参阅 WebSphere Commerce 联机帮助。

  何时使用商店服务: 使用商店服务复制样本商店归档文件并以商店归档文件格式编辑数据库有用资源。

- WebSphere Commerce 装入程序软件包

  WebSphere Commerce 装入程序软件包主要由准备数据并将数据装入 WebSphere Commerce 数据库的实用程序组成。使用装入程序软件包装入大量数据并更新

WebSphere Commerce 数据库中的数据。此软件包中的装入实用程序使用有效、精心编成的 XML 作为输入将数据装入数据库。XML 文档的元素映射到数据库中的表名，元素属性映射到列。

关于使用装入程序软件包开发和装入数据有用资源的信息，请参阅第 193 页的第 7 部分，『发布商店』。

何时使用 WebSphere Commerce 装入程序软件包：使用 WebSphere Commerce 装入程序软件包将数据库有用资源初始装入 WebSphere Commerce 数据库并对其进行更新。

重要信息：如果已更改数据库模式，则装入程序软件包是将数据装入数据库的唯一选项。

- WebSphere 贸易加速器

  WebSphere 贸易加速器是主要用于通过各种商店操作维护网上商店的联机工具的工作台。但是，由于 WebSphere 贸易加速器允许您编辑已在数据库中的数据，所以一旦您初始填充了数据库（不管是用样本商店数据还是用您创建的数据），您就可以使用其作为商店开发工具。关于您可以使用 WebSphere 贸易加速器编辑的数据库有用资源列表，请参阅 WebSphere Commerce 联机帮助主题 *"更改商店数据库有用资源"*。

  何时使用 WebSphere 贸易加速器：在您已经填充 WebSphere Commerce 数据库之后使用 WebSphere 贸易加速器。

- 直接编辑数据库

  您始终可以选择使用 SQL 插入直接编辑数据库。

  注：SQL 是特定于数据库的。Oracle 可能需要不同的 SQL 语法。请注意 SQL 语句将需要特定于数据库的值并且另一 WebSphere Commerce Server 实例中可能无法重用 SQL 语句。

开发商店后端的工具

开发商店后端的工具（包括创建和扩展命令、创建定制的代码和实现业务逻辑）在《*IBM WebSphere Commerce 程序员指南*》中进行讨论。

商店开发者角色

商店开发者开发所有三种类型的商店有用资源。他们设计并实现商店前台有用资源（包括 JavaServer Pages 文件）和商店后端（包括创建新命令和任何必要的定制代码）。他们还创建商店数据，并且可以修改 WebSphere Commerce 包含的所有标准功能。

正在创建商店前台和商店数据的商店开发者必须具有 Java、JavaScript、HTML 和 JSP 技术的编程技能，并熟悉 WebSphere Commerce 商店体系结构、商店数据和商店归档文件。

正在创建商店后端的商店开发者必须具有 Java、JavaBeans™、VisualAge® for Java 和 J2EE 编程的编程技能，并熟悉 WebSphere Commerce 编程模型和对象模型。《*IBM WebSphere Commerce 程序员指南*》提供了关于定制商店后端的更多信息。

商店开发者可与数据库开发者和 Web 设计者协同工作。数据库开发者为实现定制的商店功能或与现有的数据库信息集成而修改和扩展 WebSphere Commerce 数据库模式。此成员通常具有 DB2® 或 Oracle 的数据库管理员技能。

Web 设计者创建站点的外观，并与商店开发者一起创建商店页面。Web 设计者应具有使用多媒体工具、HTML 和 JavaScript 技能的经验并熟悉 JSP 技术。

注: 数据库开发者和 Web 设计者角色不是在 WebSphere Commerce Server 中定义的。如果需要，应该对数据库开发者和 Web 设计者分配商店开发者访问权。

一旦创建了商店归档文件，商店开发者就具有手工进行更改或使用“商店简要表”笔记本和“税款”以及“装运”笔记本对其进行更改的权限，但他们不具有将商店归档文件发布到 WebSphere Commerce Server 的权限。

第 2 部分 开发商店前台

第 3 章 开发商店前台

本章提供了 WebSphere Commerce 商店前台体系结构的概述，包括商店的外部部分、Web 有用资源（如 HTML 页面、JSP 文件、样式表、图像、图形或其它多媒体文件类型）如何显示给顾客。

商店前台体系结构

WebSphere Commerce 使用一个*命令*和*视图*系统将商店前台中的 Web 有用资源显示给顾客。

- *命令*执行特定的业务过程，如将产品添加到购物车中、处理订单、更新顾客的通讯录或显示特定的产品页面。操作完成时，命令返回一个视图。
- *视图*显示命令和用户操作的结果，也就是将商店页面（JSP 文件）呈现给顾客的视图。为使视图调用 JSP 文件，必须向视图注册表（VIEWREG）中的视图注册 JSP 文件名。相应的 JSP 文件使用 JSP 文件名存储在 WCS 商店 webapp 文档根路径下的商店子目录（stordir）中。

命令和视图都是使用 URL 调用的。例如，当顾客在样本商店中单击**购物车**时，顾客调用 URL `https://hostname/path/OrderItemDisplay?`，此 URL 传递到 WebSphere Commerce Server 中。WebSphere Commerce Server 调用 OrderItemDisplay 命令，购物车页面显示给顾客。

当顾客在样本商店中单击**帮助**时，顾客调用 URL `https://hostname/path/HelpView?`，此 URL 传递到 WebSphere Commerce Server 中。WebSphere Commerce Server 调用 HelpView，返回“帮助”页面。

WebSphere Commerce Server 也可以将多条命令映射到一个 URL，此操作允许每个商店具有各自的命令实现（可选）。

同样，WebSphere Commerce Server 也允许将多个 JSP 文件映射到单个视图，这样每个商店都可以为不同的设备类型注册不同的 JSP 文件名（可选）

注: 产品显示和产品类别显示命令返回视图和 JSP 文件名。这些显示产品和类别的 JSP 文件名存储在产品目录数据中。关于更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。可以指定不同的 JSP 文件名以显示商店支持的每个成员组或每种语言的产品和类别（可选）。

缺省命令和视图

WebSphere Commerce 提供了可以在商店中使用的缺省命令和视图。这些缺省命令和视图列在 `wcs.bootstrap.xml` 文件中。引导程序文件位于以下目录中:

- NT `drive:\WebSphere\CommerceServer\schema\xml`
- 2000 `drive:\Program Files\WebSphere\CommerceServer\schema\xml`
- AIX `/usr/WebSphere/CommerceServer/schema/xml`

- Solaris /opt/WebSphere/CommerceServer/schema/xml
- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

如果未提供所需的命令或视图，可以创建自己的命令或视图。关于创建命令和视图的信息，请参阅《*IBM WebSphere Commerce 程序员指南*》。

创建商店页面

创建商店前台中最大的任务是创建实际的商店页面。开始商店页面的开发工作之前，您应当完成以下规划活动:

- 列出所需商店页面的列表
- 列出命令和视图 URL 的列表
- 使 JSP 文件名与视图关联

列出商店页面的列表

为了列出创建商店所需页面的列表，您需要了解商店的业务和功能需求以及任何已定义的业务过程。

从使用案例开始

很多人以使用案例的形式收集需求。使用案例以顾客和建议的系统之间交互的形式定义商店中的业务过程。对网上商店而言，使用案例可以定义顾客如何在商店注册、如何浏览产品目录或如何订购商品。

联机帮助中提供了一组详细描述样本商店业务过程的使用案例。这些使用案例可以帮助您更全面地了解样本商店的流程，而且可以用作指南（如果希望创建自己商店的使用案例）。

以下是“注册”使用案例的示例:

“注册”使用案例: 注册过程允许顾客在数据库中输入个人信息。

操作者:

- 顾客

主要流程: 顾客从辅助栏选择**注册**。系统然后显示包含以下字段的页面:

- 电子邮件
- 密码
- 验证密码
- 名字
- 姓氏
- 年龄（可选）
- 性别（可选）

顾客在以上字段中输入相应的信息，并选择**提交**。系统在系统中创建一个新顾客并保存该顾客的信息（E1、E2、E3）。系统提示顾客按照“管理个人帐户”使用案例中的过程管理他们的帐户。

备用流程: 无。

异常流程: E1: 电子邮件地址已经存在:

- 如果电子邮件地址在系统中已经存在，则系统显示一条错误消息，要求用户输入另一个电子邮件地址。使用案例从头开始恢复。

E2: 遗漏必填字段:

- 如果以下任何字段（电子邮件、密码、验证密码、名字或姓氏）中有未填写的，则系统发出一条错误消息。使用案例从头开始恢复。

E3: 无效密码:

- 如果密码无效或与验证密码不匹配，则系统发出警告。

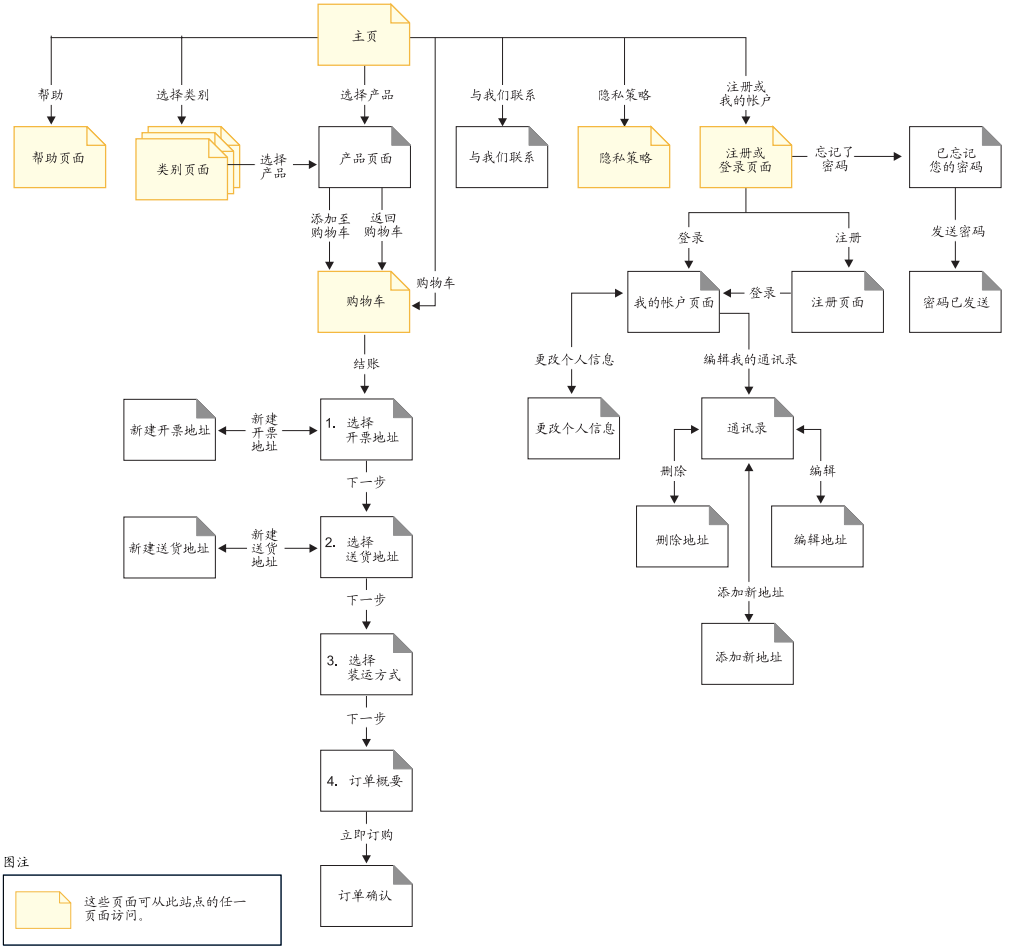
确定商店购物流程: 不管是开发使用案例来说明商店的业务过程，还是使用另一种方法，一旦业务过程可用，就可以创建商店的购物流程。

注: 由于使用案例通常包含流程信息，如“如果顾客选择**提交**，则显示‘订单’页面”，因此使用案例可以为创建购物流程图提供有用信息。

购物流程反映为商店定义的需求和业务过程，说明顾客将如何在商店中移动。例如，顾客可以通过主页进入站点并在浏览产品目录之前被要求进行注册，或者您可以选择允许顾客作为临时顾客查看产品目录而不必注册。一些购物流程允许顾客完成“快速结帐”，而其它流程则需要顾客在每次进行购买时完成所有结帐步骤。或者，您的购物流程可以允许顾客在两种结帐中进行选择。

要验证商店流程图是否完整，请确保商店使用案例中的所有步骤在商店流程图中都已说明。

绘制购物流程（如以下“流行时尚”样本商店购物流程图中所示）允许您查看顾客如何在商店中移动。



“流行时尚”购物流程图十分简单。尽管其中包括顾客在商店中移动的主要流程，但不包括任何错误方案。例如，顾客使用错误密码登录或输入无效信用卡号时会发生什么情况？但是，即使类似此图的简单图也能够让您列出商店所需的页面的列表。要开始此操作，您需要为购物流程图中列出的每个页面创建一个视图。

例如，如果要使用与“流行时尚”图中相同的购物流程创建商店，您就必须创建以下页面:

注: 下表列出了“流行时尚”商店中使用的视图名称

“流行时尚”购物流程图页面（如顾客所见）	相应视图
主页	StoreCatalogDisplayView
帮助页面	HelpView
与我们联系	ContactView
隐私策略	PrivacyView
注册或登录页面	LogonForm
忘记密码	LogoffView
密码已发送	ResetPasswordForm
我的帐户页面	LogonForm

“流行时尚”购物流程图页面（如顾客所见）	相应视图
更改个人信息	UserRegistrationForm
通讯录	AddressBookForm
添加新地址	AddressForm
删除地址	AddressBookForm
编辑地址	AddressForm
注册页面	UserRegistrationForm
购物车	OrderItemDisplayViewShiptoAssoc
选择开票地址	OrderItemDisplayViewShiptoAssoc
新建开票地址	OrderItemDisplayViewShiptoAssoc
选择送货地址	OrderItemDisplayViewShiptoAssoc
新建送货地址	AddressForm
选择装运方式	OrderItemDisplayViewShiptoDsp
订单摘要	OrderDisplayPendingView
订单确认	OrderOKView

注: “流行时尚”中使用的很多视图都是专为“流行时尚”创建的。这些视图在“流行时尚”商店归档文件中的 `command.xml` 文件中列出。关于更多信息，请参阅第 45 页的『在 WebSphere Commerce 中注册命令、视图和 URL』。

上表仅指出需要创建的基本页面集。要确定需要创建的其它页面，您可以更仔细地查看用于定义业务过程的使用案例或其它方法。

错误页面: 使用案例中的异常流程也可以帮助您确定需要为商店创建的错误页面。“流行时尚”的注册使用案例指定了以下异常流程:

- 电子邮件地址已经存在: 如果电子邮件地址在系统中已经存在，则系统显示一条错误消息，要求用户输入另一个电子邮件地址。使用案例从头开始恢复。
- 遗漏必填字段: 如果以下任何字段（电子邮件、密码、验证密码、名字或姓氏）中有未填写的，则系统发出一条错误消息。使用案例从头开始恢复。
- 无效密码: 如果密码与验证密码不匹配，则系统发出一条警告。

结果，您将需要为每个异常流程创建一个错误页面或一条错误消息。

列出命令和视图 URL 的列表

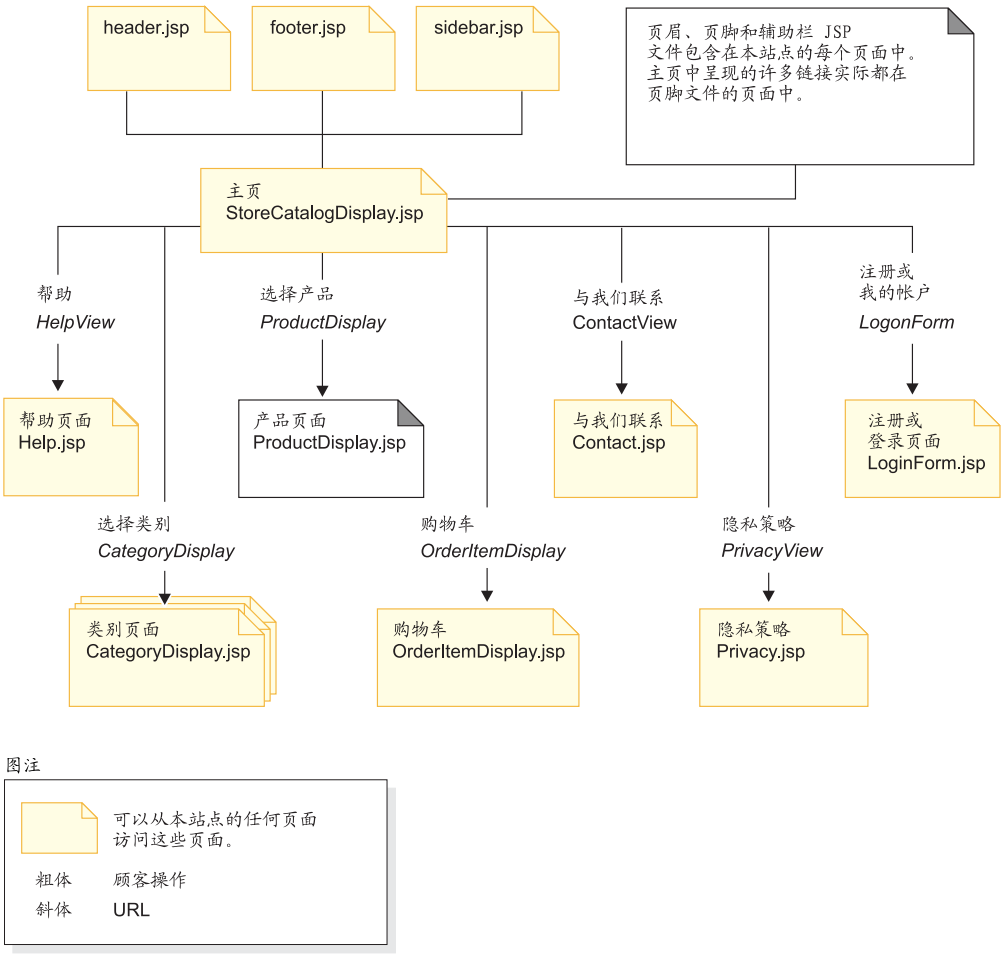
如“流行时尚”购物流程图中所示，业务过程（如结帐和注册）可能需要多个页面。要将这些页面结合到一个正常工作的业务过程或流程中（而不是仅仅一个页面集），您必须在页面中包含命令和视图。

列出所需 URL 的列表

就如同列出创建商店所需页面的列表，您还需要列出实现商店业务过程所需的命令和视图 URL 的列表。使用商店的购物流程图以及缺省命令和视图的列表确定完成每个操作所需的 URL。

了解样本商店中使用的命令和视图 URL 也可以帮助您确定商店中所需的 URL。下图标识了“流行时尚”购物流程图中某些操作的 URL。关于更多详细信息，请参阅

WebSphere Commerce 联机帮助中关于样本商店的信息。



使 JSP 文件名与视图关联

WebSphere Commerce Server 使用视图命令编辑作为请求响应的视图。WebSphere Commerce Server 提供了以下视图命令:

- `HttpForwardViewCommandImpl`: 此视图命令将视图请求转发到 JSP 文件。
- `HttpRedirectViewCommandImpl`: 此视图命令将视图请求重定向到另一个 URL。
- `HttpDirectViewCommandImpl`: 此类型的视图命令将响应视图直接发送到客户机。它不调用 JSP 文件。直接视图允许控制器命令产生输出响应（而非视图命令）。

使用 `HttpForwardViewCommandImpl` 视图命令直接提供 JSP 文件。例如，在说明“流行时尚”中所用 URL 的图中，为显示“帮助”页面（`Help.jsp`），HelpView 在视图注册表中注册并与 `Help.jsp` 和 `HttpForwardViewCommandImpl` 命令关联。这在以下示例中演示:

```
<viewreg
viewname="HelpView"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
```

```
properties="docname=help.jsp"
internal="0"
https="0"
/>
```

请注意使用接口和实现类的全限定类名。

注: 在此示例中，不限制调用视图的 URL。即，任何人都可以直接访问 URL。如果使用此技术，请确保 JSP 文件仅提供公共数据。

使用 HttpForwardViewCommandImpl 视图命令提供从显示命令返回的视图。显示命令从数据库读取数据，但不对其进行更改。例如，在说明“流行时尚”中所用 URL 的图中，OrderItemDisplay 命令返回 OrderItemDisplayViewShiptoAssoc 视图。在视图注册表中注册此视图时，OrderItemDisplay.jsp 和 HttpForwardViewCommandImpl 与其相关联。这在以下示例中演示:

```
<viewreg
viewname="OrderItemDisplayViewShiptoAssoc"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=OrderItemDisplay.jsp"
internal="0"
https="0"
/>
```

注: 在此示例中，限制调用视图的 URL。即，只有那些具有指定访问权限的用户可以直接调用 URL。对此视图的访问由任何对命令 URL 具有访问权的用户控制。

您必须使每个关联视图的 JSP 文件名与所使用的每条显示命令（例如 OrderItemDisplay）关联。关于使 JSP 文件名与视图关联的更多信息，请参阅第 45 页的『在 WebSphere Commerce 中注册命令、视图和 URL』。

注: ProductDisplay 和 CategoryDisplay 列出产品目录数据而非视图注册表中的关联 JSP 文件名。

使用 HttpRedirectViewCommandImpl 视图命令提供从更改数据库的命令返回的视图。要使用重定向视图，请在 URL 上使用 &URL= 参数指定视图名称。例如，在“流行时尚”样本商店 AddressForm 中添加地址信息并单击**提交**时，它将调用 AddressAdd 命令。用于调用 AddressAdd 命令的 URL 使用 &URL= 参数将 AddressBookForm 指定为视图。这会导致重定向到 AddressBookForm 视图。在视图注册表中注册 AddressBookForm 视图时，AddressBookForm.jsp 和 HttpForwardViewCommandImpl 与其关联。

必须对所有非显示命令使用 URL=parameter 技术。非显示命令是导致数据库中的数据发生更改的命令。

第 3 部分 商店数据概述

第 4 章 商店数据

本章提供了 WebSphere Commerce Server 商店数据体系结构和创建商店的数据有用资源两者的概述。本章中还介绍了 WebSphere Commerce Server 信息模型。

什么是商店数据?

商店数据是装入 WebSphere Commerce Server 数据库的信息，使您的商店可以正常运作。为了正常运作，商店必须准备好数据以支持所有顾客活动。例如，要使顾客可以进行购买，商店必须包含待售商品的产品目录（产品目录数据）、与处理订单关联的数据（税款和装运数据）和用于实现请求的库存（库存和实现数据）。

商店数据信息模型

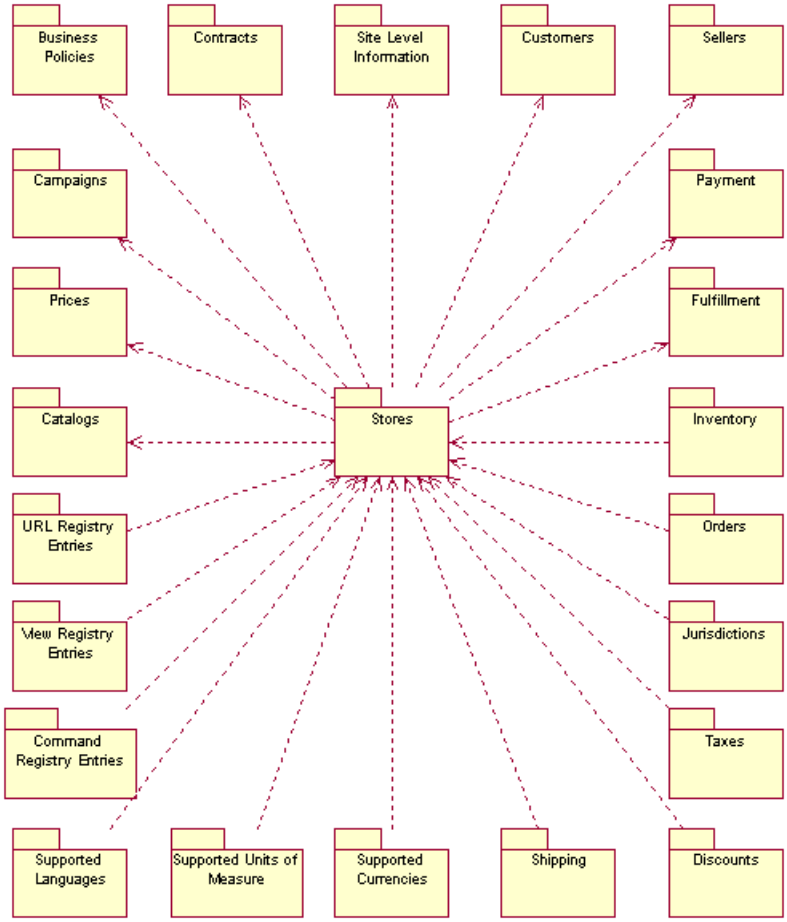
本指南使用信息模型说明商店数据在 WebSphere Commerce Server 中是如何组织的。WebSphere Commerce Server 信息模型是对 WebSphere Commerce Server 数据和对象模型中包含的信息的高级抽象。信息模型突出显示数据和对象模型的最重要功能，但不包括特定于模式和对象实现的较低级详细信息。

例如，包含实体关系数据（如外键对）的数据或对象模型中的某些表和对象不作为实体显示在信息模型中。这些实体关系由信息模型中实体之间的关系线指出。信息模型也不说明*详细信息*扩展（作为实现的结果存储在单独表中的实体附加数据属性：例如，产品描述是产品实体单独存储的扩展）。关于关系对象和详细信息扩展的更多信息，请参阅 WebSphere Commerce 联机帮助中的对象模型。

关于 WebSphere Commerce 对象和数据模型的更多信息，请参阅 WebSphere Commerce 联机帮助。

商店数据有用资源

下图说明 WebSphere Commerce 商店的数据有用资源。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

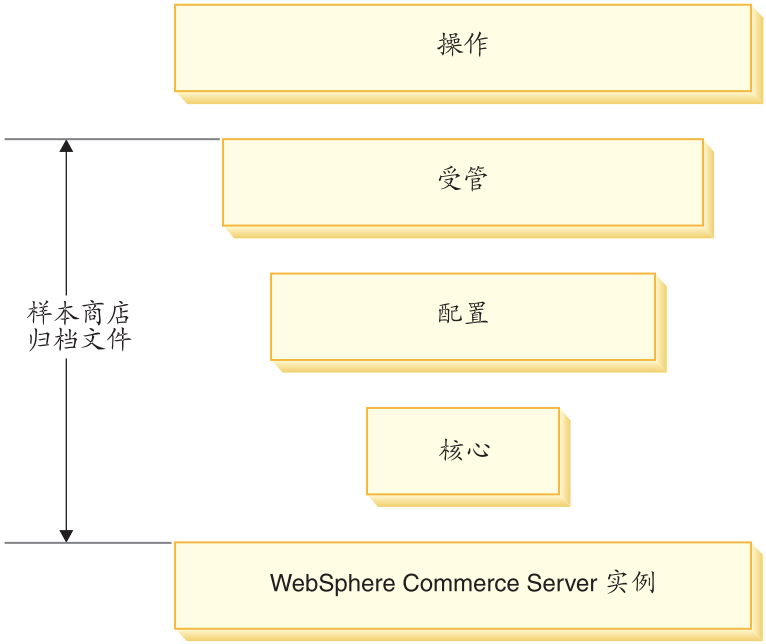
请注意图中箭头的两个方向。在某些情况下，箭头指向商店，例如，从“货币”到“商店”。在此情况下，货币有用资源专属于此特定商店并且是该商店的一部分。这些是此商店支持的货币。如果删除商店，则商店支持的货币列表也将被删除。

当箭头从“商店”指向一个有用资源（例如“产品目录”）时，此有用资源可以由其它商店共享。一个产品目录可以由若干商店共享。但是，如果删除商店，产品目录仍将存在。

上图中说明的每个数据有用资源在第 33 页的第 4 部分，『开发商店数据』中的章节进行了更详细的讨论。

商店数据体系结构

WebSphere Commerce 商店中的数据符合下图中描写的体系结构。第 26 页的『商店数据有用资源』的图中说明的每个商店数据有用资源可以分类为属于以下说明的一种或多种类型的商店数据。



WebSphere Commerce Server 实例

数据的基本级别包含在 WebSphere Commerce Server 实例中。创建实例时，以 XML 格式装入的引导程序文件将用信息填充数据库。引导程序文件创建以下类型的数据:

- 计算用法类型、设备类型（浏览器、电子邮件、I 方式等等）、消息类型和角色
- 缺省管理标识 WCSADMIN
- 缺省命令、视图和 URL
- 缺省业务策略
- 实例支持的语言和货币
- 可以用作商店所有者的缺省组织
- 缺省站点组织
- 缺省商店组

此信息可用于该实例中存在的所有商店，并且在第 26 页的『商店数据有用资源』的图中标识为站点级别信息。

关于引导程序文件及其填充的数据库表的更多详细信息，请参阅 WebSphere Commerce 联机帮助。

核心数据

下一级别的商店数据是核心数据。核心数据创建商店的最小数据，包括:

- STOREENT 表中的商店标识。这在数据库中创建商店。
- 缺省合同。

- 合同数据库表中的存储标识。
- 拥有合同数据库表中商店的组织的成员标识。
- STORE 表中的商店目录。商店目录是商店的 Web 有用资源所在的目录。
- STADDRESS 表中商店地址的别名或标识。别名对于每个商店是唯一的。

如果使用“商店服务”创建商店，则为您创建此信息与新的商店归档文件。“商店服务”使您能够选择可以作为商店所有者的缺省组织，或者您可以使用管理控制台创建另一个组织作为所有者。如果未使用“商店服务”创建商店，则您必须使用装入程序软件包将此信息装入数据库，或者直接编辑数据库。

第 26 页的『商店数据有用资源』的图中的商店数据是核心数据。

配置数据

配置数据控制贸易服务器运行时。公共服务器运行时提供一种在其中部署与执行商业应用程序的框架。该框架由编程模型、过程模型、异常处理、事务控制、数据访问和持久性模型组成。公共服务器运行时利用 WebSphere Application Server 提供的运行时服务支持 WebSphere Commerce Server 应用程序。配置数据确定商店将用于显示商店页面的命令、视图和 JSP 文件。

第 26 页的『商店数据有用资源』的图中标识的以下数据有用资源分类为配置数据:

- 命令注册表项
- 视图注册表项
- URL 注册表项

受管数据

受管数据是卖方创建的数据，对于卖方站点的顾客是只读的。由于卖方完全控制此数据的状态，所以可以通过内容管理系统管理受管数据。

第 26 页的『商店数据有用资源』的图中标识的以下数据有用资源分类为受管数据:

- 竞销
- 业务策略
- 合同
- 实现中心
- 地区
- 税款
- 折扣
- 装运
- 货币
- 计量单位
- 语言
- 产品目录
- 价格
- 顾客
- 卖方
- 支付

操作数据

操作数据是站点顾客通过与站点的交互创建或更改（直接或间接）的数据。例如，顾客订单被视为库存级别（随商店操作上升或下降）的操作数据。顾客也被视为操作数据。卖方创建的数据也是可操作的。

由于对操作数据的更改不完全处于卖方的控制之中，所以使用内容管理系统管理此数据没有意义。

第 26 页的『商店数据有用资源』的图中标识的以下数据有用资源分类为操作数据:

- 订单
- 库存
- 实现
- 顾客

注: 在某些情况下，操作数据和受管数据之间的界线可能很难确定。例如，在一个商店中，顾客和合同数据可能被视为受管数据，而在另一个商店中，同一类型的数据可能就被视为操作数据。第一个商店可以管理其顾客数据与相关合同，因为它们拥有一组特定的顾客（即顾客无法在线注册）。但是，第二个商店允许顾客在线注册并在线创建合同信息。

另一个示例涉及产品目录数据。在单个卖方站点中，产品目录被视为受管数据。在电子市场站点中，产品目录数据则可能被视为操作数据。

在一些站点中，相同数据类型的某些记录可能被视为受管数据，而另一些记录则被视为操作数据。例如，缺省合同可能是受管数据，而在线协商的特定合同则是操作数据。

商店数据体系结构和样本商店

随 WebSphere Commerce 提供的样本商店包含了商店数据体系结构中大多数类型的商店数据。例如，WebSphere Commerce Server 实例必须存在，才能使用样本商店创建商店或发布样本商店。这样，当您使用“商店服务”中的工具基于样本商店创建商店时，就创建了核心数据。样本商店包含所有必需的配置以及正常运作的商店所需的大多数受管数据。基于特定样本商店创建商店时，可能会指示您使用 WebSphere 贸易加速器中的工具完成数据的一些设置。

创建数据的工具

WebSphere Commerce 提供了创建和操作商店数据的多种工具。下面列出了这些工具:

WebSphere Commerce 装入程序软件包

装入程序软件包主要由准备数据并将数据装入 WebSphere Commerce 数据库的实用程序组成。关于更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

商店服务

“商店服务”以商店归档文件的形式编辑预发布数据，而不是在数据库中编辑实际数据。“商店服务”还使您能够将所有商店数据有用资源发布到数据库。关于更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

管理控制台

管理控制台使您可以通过完成管理操作和配置任务来控制站点和商店。您还可以使用管理控制台创建新的组织和用户，以及为用户指定角色（商店开发者、商店管理员、站点管理员等等）。管理控制台还使您可以确定将在商店中可用的通知和消息传递类型。

WebSphere 贸易加速器

WebSphere 贸易加速器是使您可以创建和维护各种商店有用资源的在线工具的工作台。大部分商店数据可以使用 WebSphere 贸易加速器中的工具创建和管理。但是某些情况下，在可以使用 WebSphere 贸易加速器创建数据之前必须已使用“商店服务”的发布工具或装入程序软件包将特定数据装入数据库。关于更多信息，请参阅『工具和商店数据摘要图表』。

组织管理控制台

组织管理控制台使您可以管理访问您站点或商店的组织。组织管理控制台还使买方的管理员可以管理其组织内的买方。

工具和商店数据摘要图表

以下图表列出了可用于创建每种类型的数据的工具。

创建数据的工具	核心数据	配置数据	受管数据	操作数据
WebSphere Commerce 装入程序软件包	使用装入程序软件包以 XML 文件形式装入核心数据。关于更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。	使用装入程序软件包以 XML 文件形式装入配置数据。关于更多信息，请参阅第 45 页的『创建 XML 文件以注册命令、视图和 URL』。	使用装入程序软件包以 XML 文件形式装入受管数据。关于更多信息，请参阅关于受管数据有用资源的相应章节。	通常，操作数据不能使用装入程序软件包装入。

创建数据的工具	核心数据	配置数据	受管数据	操作数据
商店服务（仅用于以商店归档文件形式预发布的数据）	使用“商店服务”创建新的商店归档文件时，就为您创建了核心数据。关于使用“商店服务”的更多信息，请参阅 WebSphere Commerce 联机帮助。	未提供。	“商店服务”使您可以创建和编辑以下受管有用资源部分: • 地区 • 税款 • 装运 • 货币 • 语言 关于“商店服务”允许您编辑或创建的数据库有用资源部分的更多信息，请参阅 WebSphere Commerce 联机帮助中的“更改商店数据库有用资源”。	未提供。
管理控制台	使用管理控制台创建作为商店所有者的组织。	未提供。	未提供。	未提供。
WebSphere 贸易加速器	未提供。	未提供。	使用 WebSphere 贸易加速器创建或编辑以下数据: • 竞销 • 合同（缺省合同必须存在于数据库中，您才可以使用 WebSphere 贸易加速器中的“业务关系管理”工具创建附加合同或更改现有合同。使用装入程序软件包或“商店服务”在数据库中创建缺省合同）。	顾客在向商店注册或从商店进行购买时创建操作数据。但是，在某些情况下，可以使用 WebSphere 贸易加速器为顾客下订单或者创建退货。 WebSphere 贸易加速器还使您可以管理库存。

创建数据的工具	核心数据	配置数据	受管数据	操作数据
WebSphere 贸易加速器（续）	未提供。	未提供。	• 实现 • 折扣 • 产品目录（主产品目录必须存在于数据库中，您才可以使用 WebSphere 贸易加速器中的“产品管理”工具创建可浏览的产品目录以及添加或更改产品信息。使用装入程序软件包或“商店服务”在数据库中创建主产品目录。） • 价格	未提供。
组织管理控制台	未提供。	未提供。	未提供。	顾客和买方是在他们进入商店时创建的。但是，通过使用组织管理控制台，您可以核准买方或创建新买方。

第 4 部分 开发商店数据

本部分中的章节更详细地解释了每个商店数据有用资源。本部分中的商店数据有用资源按照 WebSphere Commerce 商店数据体系结构进行组织:

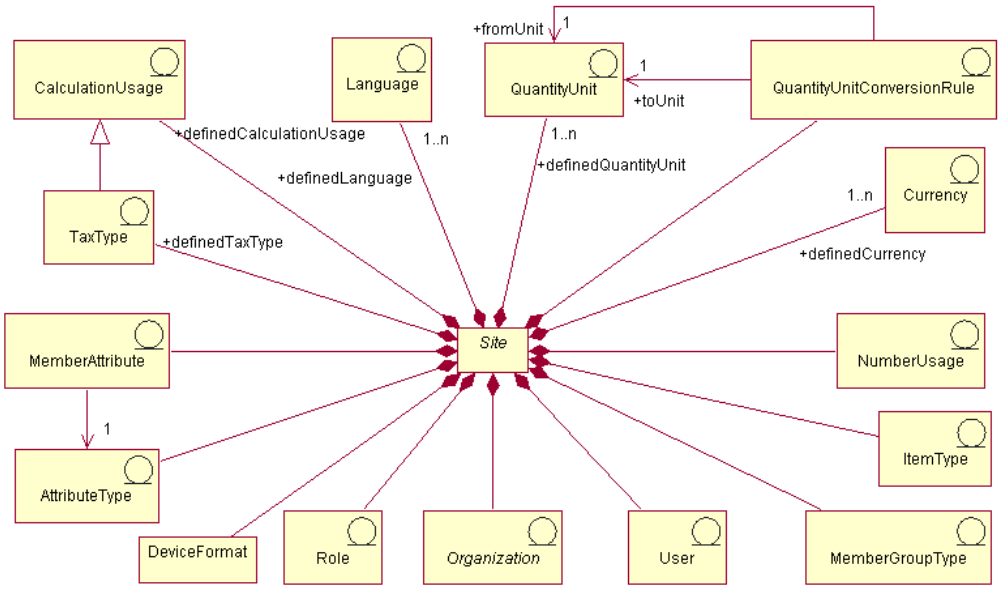
- WebSphere Commerce Server 实例
  - 站点
- 核心数据
  - 商店
- 配置数据
  - 命令注册表
  - 视图注册表
  - URL 注册表
- 受管数据
  - 共享的有用资源
    - 产品目录
    - 价格
    - 合同（包括业务策略）
    - 实现
    - 竞销
    - 支付
  - 专属于商店的有用资源
    - 支持的语言
    - 支持的货币
    - 支持的计量单位
    - 地区
    - 装运
    - 税务
    - 折扣
- 操作数据
  - 库存
  - 订单
  - 顾客

第 5 章 站点有用资源

每个 WebSphere Commerce Server 实例都具有其自己的关系信息数据库。实例由引导程序文件创建，这些文件在创建模式之后用信息填充数据库表。一旦装入数据，您就可以在相应的数据库表中查看预装入的信息。许多数据库表包含特定于商店或商店组的商店或商店组级别信息。此信息通常由商店管理员管理。但是，一些表包含的信息表示可由实例中所有商店使用、并且由 WebSphere Commerce 站点管理员管理的 WebSphere Commerce 站点级别功能。这些功能在本章中进行讨论。关于引导程序文件的更多信息，请参阅 WebSphere Commerce 联机帮助。关于特定于商店的有用资源信息的更多信息，请参阅第 39 页的第 6 章，『商店有用资源』。

了解 WebSphere Commerce 中的站点有用资源

下图说明了站点包含的数据类型及其与站点的关系。



关于本图中使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。本图和商店数据部分中的其它所有图都是 WebSphere Commerce 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。

语言

站点可以在 LANGUAGE 表中定义许多语言，在 LANGUAGEDS 表中对其进行描述。每个商店通常通过向 STORELANG 表添加行支持这些语言的子集。预定义的十种语言是：德语、繁体中文和简体中文、日语、韩国语、意大利语、法语、西班牙语、巴西葡萄牙语和英语。

成员属性

*成员属性*存储在 MBRATTR 表中，代表一组已定义的属性名称，其值可以为组织或用户存储。此类属性名称的示例包括 JobFunction、ProcurementCard、SpendingLimit、ReferredBy 和 CountryOfOperation。特定组织或用户的属性值可以存储在 MBRATTRVAL 表中，且这些值对不同的商店或商店组可以不同。

属性类型

*属性类型*存储在 ATTRTYPE 表中，代表可以用于代表属性值的已定义数据类型。数据类型的示例包括: INTEGER、STRING 和 FLOAT。

成员组类型

*成员组类型*存储在 MBRGRPTYPE 表中，代表一组已定义的成员组用法。通过向 MBRGRPUSG 表添加行为成员组指定用法。成员组用法的示例包括 AccessGroup（供访问控制策略使用）和 UserGroup（供一般用途使用，如顾客组）。

用户

*用户*代表已认证的用户身份。用户通常表示以买方组织名义下订单或核准订单的顾客、为销售组织处理订单或维护商店级别有用资源的销售代理，或者维护 WebSphere Commerce Server 实例的站点管理员。每个用户与一个站点关联，并且在 USERS 表中定义。

组织

*组织*代表组织和组织内的组织单位。组织通常代表负责购买或销售的商业实体。顾客在“商家到商家”购买组织中下的订单被记录为以买方组织名义下订单。商店、产品目录和实现中心的所有者是负责销售的特定方面的组织。组织在 ORGENTITY 表中定义。

角色

*角色*代表可在组织内为用户指定的一组已定义角色。例如，可以在销售组织内为用户指定客户服务代表角色，或者可以在买方组织内为用户指定买方核准员角色。缺省角色的名称和描述填充在 ROLE 表中。关于特定角色的更多信息，请参阅 WebSphere Commerce 联机帮助。

数量单位转换

每个站点都有*数量转换*。这些转换代表用于在不同计量单位之间进行转换的乘法或除法运算。这些规则填充在 QTYCONVERT 表中。

数量单位

*数量单位*代表站点的一组计量单位。这些单位在 QTYUNIT 表中定义，QTYUNITDSC 表中对其进行了描述。每个商店可以通过向 QTYFORMAT 表添加行指定如何舍入和格式化以每种计量单位表示的数量以供显示（根据其预期用法）。

税款类型

*税款类型*代表计算税款的计算用法。销售税和装运税是计算税款的两种不同计算用法。税款类型在 TAXTYPE 表中定义。

计算用法

*计算用法*代表可由 OrderPrepare 命令执行的不同类型的计算。为折扣、装运费用、销售税、装运税和电子赠券定义了计算用法。计算用法在 CALUSAGE 表中定义。

货币

每个站点都在 SETCURR 表中定义许*多货币*并在 SETCURRDSC 表中对这些货币进行描述。每个站点都通过向 CURLIST 表添加行（每行对应一种支持的货币）支持这些货币的子集。

数值用法

*数值用法*代表数值的预期用法。商店可以根据使用数值的方式来为显示的数值指定不同的舍入和格式化规则。例如商店可以通过指定“单价”用法将单价舍入到四位小数，而通过指定“缺省”用法将其它货币金额舍入到两位小数。数值用法在 NUMBRUSG 表中定义，NUMBRUSGDS 表中对其进行了描述。

商品类型

*商品类型*代表不同类型的基本商品。WebSphere Commerce 中两种类型的基本商品是可改装的成套商品和一般商品。商品类型在 ITEMTYPE 表中预定义。关于基本商品的更多信息，请参阅第 153 页的第 21 章，『库存有用资源』。

设备格式

*设备格式*存储在 DEVICEFMT 表中，代表站点使用的诸多设备格式，如浏览器、I_MODE、电子邮件、MQXML 和 MQNC。所有这些设备类型都允许用户通过各种介质与站点交互。

注: 对于一些站点有用资源（如语言、货币、数量单位和数量单位转换规则），站点管理员可以通过向适当的表添加行来扩展站点级别功能。对于其它站点有用资源，扩展其代表的站点级别功能可能还需要相关定制。例如，如果站点管理员添加了一种新的数值用法以定制的货币符号显示小计，则必须定制显示小计的程序，以便指定在格式化小计金额以供显示时使用新的小计数值用法。

在 WebSphere Commerce 中创建站点有用资源

站点有用资源是当您在 WebSphere Commerce Server 中创建实例时创建的。关于在 WebSphere Commerce Server 中创建实例的更多信息，请参阅《*IBM WebSphere Commerce 安装指南*》中的第 5 章『创建或修改实例』。

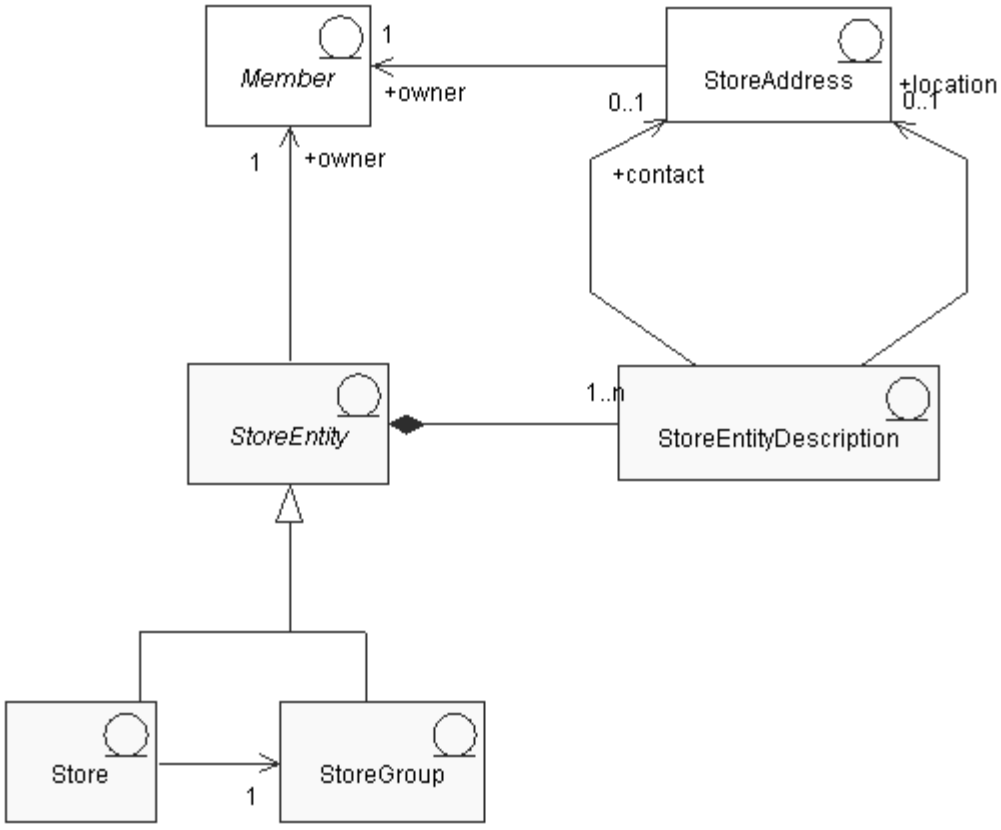
第 6 章 商店有用资源

要在 WebSphere Commerce 中创建商店，必须首先在数据库中创建以下内容:

- 商店
- 商店所属的组
- 双重代表商店或商店组的抽象商店实体对象

了解 WebSphere Commerce 中的商店有用资源

下图说明了 WebSphere Commerce Server 中的商店有用资源。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

商店实体

*商店实体*是可以代表商店或商店组的抽象超类。

商店实体有一个所有者（成员）。关于成员的更多信息，请参阅第 161 页的第 23 章，『顾客和卖方有用资源』。

商店实体描述

*商店实体描述*描述了商店实体。商店实体可能包含描述。如果您的商店支持多种语言，则商店实体描述可能使用多种语言。描述可以包括商店实体的联系地址以及商店实体的位置地址。

商店

*商店*是商店实体。商店必须属于某个商店组。

商店组

*商店组*是商店的集合。商店组是商店实体。商店组作为公共信息的容器，可以商店组级别存储并由商店组中的所有商店共享。例如，同一商店组中的商店可以共享诸如税类别、支持的语言、支持的货币、计算代码和装运地区等信息。

当前，WebSphere Commerce Server 中在站点管理级别上仅可存在一个商店组并对其进行维护。

关于 WebSphere Commerce Server 中商店有用资源结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的商店对象和数据模型。

在 WebSphere Commerce 中创建商店有用资源

WebSphere Commerce 中的“商店服务”工具使您可以创建或编辑以下商店有用资源:

- 合同有用资源中的商店标识和成员标识
- STOREENT 表中的商店标识
- STORE 表中的商店目录
- STADDRESS 表中的地址昵称
- 商店描述
- 商店地址

注: “商店服务”工具以商店归档文件的格式处理预填充的 XML 文件。

结果，您有两个选项可用于创建商店有用资源:

- 编辑随 WebSphere Commerce 提供的样本商店之一的现有商店有用资源，或编辑现有的商店归档文件。
- 以 XML 文件形式创建商店有用资源，这些资源可以作为商店归档文件的一部分发布，或者可以使用装入程序软件包装入。

关于编辑现有商店归档文件中的商店有用资源的信息，请参阅 WebSphere Commerce 联机帮助。关于以 XML 文件形式创建商店有用资源的信息，请参阅『在 XML 文件中创建商店数据有用资源』。

在 XML 文件中创建商店数据有用资源

以 XML 文件（可以使用装入程序软件包装入数据库）格式创建商店有用资源。如果正在创建多文化商店，则您可能希望为商店支持的每种语言环境创建各自的 XML 文件。

特定于语言环境的文件应当指定所有描述信息，以便比较容易进行翻译。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

样本商店（这些任务中的很多示例是从其中获取的）对不需要翻译的所有信息使用一个 store.xml 文件，对需要翻译的信息为商店支持的每种语言环境使用另一个 store.xml 文件。特定于语言环境的文件包含所有描述信息。

要创建商店有用资源，请执行以下操作:

1. 复查用于为样本商店创建商店有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

     注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包括两个 store.xml 文件，其中包括商店信息。要查看商店归档文件中的 store.xml 文件，请使用 ZIP 程序将商店归档文件解压缩。store.xml 文件位于数据目录中。特定于语言的 store.xml 位于数据目录的特定于语言环境的子目录中。
2. 请复查第 289 页的附录 B，『创建数据』中的信息。
3. 通过复制样本商店归档文件中的 store.xml 文件之一或通过创建新文件创建 store.xml 文件。关于更多信息，请参阅相应于 store.xml 的 DTD 文件。DTD 文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. 创建商店实体。

   a. 使用以下示例作为指南，在您的 XML 文件中为 STOREENT 表定义商店实体。

```
<storeent
 storeent_id="@storeent_id_1"
 member_id="&MEMBER_ID"
 type="S"
 identifier="ToolTech"
 setccurr="USD"
/>
```

其中

- storeent_id 是生成的唯一键。
- member_id 是商店实体的所有者。
- type 是商店实体的类型: G = 商店组, S = 商店。
- identifier 是与所有者一起唯一标识商店实体的字符串。
- setccurr 是商店实体的缺省货币，换句话说，是将由尚无首选货币的顾客使用的货币。如果其对某商店为 NULL，则缺省货币从其商店组中获取。

5. 创建商店地址。
   a. 使用以下示例作为指南，在您的 XML 文件中为 STADDRESS 表创建商店地址。如果正在创建多文化的产品目录，则应将此信息包含在特定于语言环境的 XML 文件中。

```
<staddress
staddress_id="@staddress_id_en_US_1"
member_id="&MEMBER_ID"
nickname="storeaddress_English"
address1="12xx Martindale Avenue"
address2="Suite 9xx"
businesstitle="ToolTech"
city="Toolsville"
state="Ontario"
zipcode="Lxx 1xx"
country="Canada"
phone1="1-800-555-1234"
fax1="1-800-555-4321"
email1="info@tooltech.xxx"
/>
```

其中

- staddress_id 是生成的唯一键。
- member_id 是商店实体的所有者。

6. 创建商店实体的描述。
   a. 使用以下示例作为指南，在您的 XML 文件中为 STOREENTDS 表创建商店实体的描述。如果正在创建多文化的产品目录，则应将此信息包含在特定于语言环境的 XML 文件中。

```
<storeentds
description="Commerce Models Store entity"
language_id="&en_US"
displayname="ToolTech"
storeent_id="@storeent_id_1"
staddress_id_cont="@staddress_id_en_US_1"
staddress_id_loc="@staddress_id_en_US_1"
```

其中

- description 是商店实体的详细描述，适合于对顾客显示。
- language_id 是对在商店中购物的顾客显示信息时使用的缺省语言。
- displayname 是商店实体的简短描述，适合于对顾客显示。
- storeent_id 是商店实体。
- staddress_id_cont 是商店实体的联系地址。
- staddress_id_loc 是商店实体的物理位置。

7. 在数据库中创建商店。

a. 使用以下示例作为指南，在您的 XML 文件中为 STORE 表定义商店。

```
<store
store_id="@storeent_id_1"
directory="ToolTech"
ffmcenter_id="@ffmcenter_id_1"
language_id="&en_US"
storegrp_id="-1"
allocationgoodfor="43200"
bopmpadfactor="0"
defaultbooffset="2592000"
ffmcselectionflags="0"
maxbooffset="7776000"
rejectedordexpiry="259200"
rtnffmctr_id="@ffmcenter_id_1"
pricerefflags="0"
storetype="B2B"
/>
```

其中

- store_id 是生成的唯一键。
- directory 是特定于商店的 Web 有用资源所在的目录。这些有用资源在文件系统中的实际位置基于此列的值，以及 WebSphere Commerce 配置文件中的若干配置参数: StoresDocRoot、StoresWebPath 和 StoresPropertiesPath。例如，如果 StoresDocRoot 是 D:\WebSphere\wcs\stores，StoresWebPath 是 web，StorePropertiesPath 是 properties 且此列的值为 mystore，则 JSP 文件将位于目录 D:\WebSphere\wcs\stores\web\mystore 中，而属性文件将位于 D:\WebSphere\wcs\stores\properties\mystore 中。
- ffmcenter_id 是商店的缺省实现中心。
- language_id 是对在商店中购物的顾客显示信息时使用的缺省语言。
- storegrp_id 是商店与之关联的商店组。此数字在 STOREGRP 表中生成。
- allocationgoodfor 意味着 ReleaseExpiredAllocations 调度程序作业可以在进行分配后很多秒用于逆转 ATP 库存分配。
- bopmpadfactor 意味着如果该商店在计算订单金额（如税款或装运费用）时对不同实现中心使用不同方法，则先前所提交订单的订单金额可能在最终将实现中心分配到延迟交货的订购商品时发生更改。此填空因子代表提供给 Payment Manager 的订单金额可以增加的百分比（如有必要）。例如，指定 5 允许最多增加百分之五。
- defaultbooffset 是无法为延迟交货的 OrderItem 确定估计可提供时间之后的时间，以后将设置为这一秒数。
- maxbooffset 意味着如果延迟交货的 OrderItem 的估计可提供时间将来通常超出这一秒数，则将来它会被设置为这一秒数。
- rejectedordexpiry 是支付处于“拒绝”状态的时间长于此秒数的订单，是可以考虑取消的对象。
- rtnffmctr_id 是将商品退回商店的缺省实现中心。
- pricerefflags 包含位标志，这些位标志控制在 GetContractUnitPrices 任务命令的缺省实现刷新价格时，搜索的是哪些 TradingAgreements 和 Offers:
  - 1 = usePreviousOnly — 使用 OrderItems 引用的内容。如果其再也无法使用则失败。

– 2 = usePreviousOrSearchAgain — 与 usePreviousOnly 相同，但是再也无法使用时不会失败，而是搜索 ORDIOFFER 和 ORDITRD 表中保存的内容
– 4 = alwaysSearchAgain — 总是搜索 ORDIOFFER 和 ORDITRD 表中保存的内容。

- storetype 指示以下商店类型之一，以供基于以下 StoreType 提供适当功能的用户界面使用: B2B = 商家到商家。B2C = 商家到消费者。

8. 定义商店的支持语言。
   a. 使用以下示例作为指南，在 XML 文件中定义商店的支持语言以向 STORELANG 表添加信息。如果您的商店支持多种语言，则应将此信息包含在特定于语言环境的 XML 文件中（商店支持的每种语言各一个）。

```
<storelang
language_id="&en_US"
storeent_id="@storeent_id_1"
/>
```

   其中

   - language_id 是商店实体支持的语言。
   - storeent_id 是商店实体。

   b. 使用以下示例作为指南，将关于语言的信息添加到 STORELANGDS 表中。如果您的商店支持多种语言，则应将此信息包含在特定于语言环境的 XML 文件中（商店支持的每种语言各一个）。

```
<storlangds
 description="United States"
 language_id="&en_US"
  storeent_id="@storeent_id_1"
  language_id_desc="& en_US"
/>
```

   其中

   - description 是语言的简短描述，适合于对选择列表中的顾客显示。
   - language_id 是描述的语言。
   - storeent_id 是支持语言的商店实体。
   - language_id_desc 是被描述的语言。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 7 章 命令、视图和 URL 注册表数据

创建 WebSphere Commerce Server 实例时，随 WebSphere Commerce 提供的缺省命令、视图和 URL 注册在 WebSphere Commerce Server 数据库相应的表（CMDREG、VIEWREG 和 URLREG）中。这些命令、视图和 URL 可以在驻留于实例中的所有商店内使用。

WebSphere Commerce 还提供显示缺省视图的缺省 JSP 文件。这些 JSP 文件与 VIEWREG 表中的视图关联。

如果创建新的命令、视图或 URL，或者定制现有的命令、视图或 URL，您必须在相应数据库表（CMDREG、VIEWREG 和 URLREG）中对其进行注册，才可在商店中使用。如果创建新的 JSP 文件以供在商店中使用，则必须使其与 VIEWREG 表中的相应视图关联。

注: 如果创建新的 JSP 文件，但赋予其与关联视图的缺省 JSP 文件相同的名称，则您不需要在 VIEWREG 表中注册新的 JSP 文件。

关于创建或定制命令、视图或 URL 的更多信息，请参阅《*IBM WebSphere Commerce 程序员指南*》。《*程序员指南*》还包含关于注册命令、视图、URL 以及 JSP 文件的方法和时间的信息。

关于 WebSphere Commerce Server 中命令和视图有用资源结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的命令和视图数据模型。

在 WebSphere Commerce 中注册命令、视图和 URL

如果为商店创建或定制多个新的命令、视图、URL 或 JSP 文件，您可能希望使用 XML 文件对其进行注册，然后可以使用装入程序软件包将此文件装入数据库，或者作为可使用“商店服务”发布的商店归档文件的一部分装入数据库。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

注: 创建 XML 文件装入新的或定制的命令之前，请参阅《*程序员指南*》以获取关于命令如何工作的更多详细信息。

创建 XML 文件以注册命令、视图和 URL

要创建 XML 文件以注册商店新的命令、视图和 JSP 文件，请执行以下操作:

1. 复查用于注册样本商店的命令、视图、JSP 文件的 XML 文件。每个样本商店包括一个 `command.xml` 文件，其中包含注册信息。商店归档文件位于以下目录中:
   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`

- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

  注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

  要查看商店归档文件的内容，请使用解压缩程序。command.xml 文件位于 data 目录中。

2. 请复查第 289 页的附录 B, 『创建数据』中的信息。
3. 通过复制样本商店归档文件中的 command.xml 文件之一或通过创建新文件创建 command.xml 文件。关于更多信息，请参阅相应于 command.xml 的 DTD 文件。DTD 文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer \xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. 控制器命令必须在 URLREG 表和 CMDREG 表中注册。要在 URLREG 表中注册新的或定制的控制器命令，请使用以下示例作为指南在 XML 文件中为每个新的定制控制器命令创建一个条目:

   ```
   <urlreg
   url="MyProductDisplay"
   storeent_id="@storeent_id_1″
   interfacename="com.mystore.commerce.catalog.commands.ProductDisplayCmd"
   https="0"
   description="Product display command for my store"
   authenticated="0"
   internal="0" />
   ```

   其中

   - urlreg 是此信息将填充的数据库表（URLREG）的名称。
   - url 是 URI 名称
   - storeent_id 是商店实体标识，使用 @ 符号称为内部别名解析。使用内部别名解析时，以别名替换 XML 文档中的主键（标识）。然后在 XML 文件中的其它位置使用此别名来引用该元素。这样就无需了解构建 XML 文件所需的唯一索引。发布期间，标识解析器会以唯一值替换 @ 符号。关于更多信息，请参阅第 289 页的附录 B, 『创建数据』。
   - interfacename 是控制器命令接口名称
   - https 是此 URL 请求所需的安全 HTTP。需要安全 HTTP 时请使用 1，不需要时请使用 0。

- authenticated 是此 URL 请求是否需要用户登录。需要认证时请使用 1，不需要时请使用 0。
- internal 指示命令对于 WebSphere Commerce 是否是内部的。内部的 URL 由 WebSphere Commerce 工具使用。是内部时请使用 1，是外部时请使用 0。您创建的 URL 必须是外部的。

5. 要在 CMDREG 表中注册新的控制器命令或新的任务命令，请使用任务命令的以下示例（来自“多乐五金店”样本商店的 command.xml 文件）作为指南在 XML 文件中为每个新的或定制的控制器或任务命令创建一个条目:

```
< cmdreg
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.payment.commands.DoPaymentCmd"
classname="com.ibm.commerce.payment.commands.DoPaymentMPFCmdImpl" />
```

其中

- cmdreg 是此信息将填充的数据库表（CMDREG）的名称。
- storeent_id 是商店实体标识，使用 @ 符号称为内部别名解析。使用内部别名解析时，以别名替换 XML 文档中的主键（标识）。然后在 XML 文件中的其它位置使用此别名来引用该元素。这样就无需了解构建 XML 文件所需的唯一索引。发布期间，标识解析器会以唯一值替换 @ 符号。关于更多信息，请参阅第 289 页的附录 B，『创建数据』。
- interfacename 是命令接口名称
- classname 是命令实现类名称。通常，此名称是在末尾附加 Impl 的接口名称。

6. 要注册新视图，或使新的 JSP 文件与视图关联，请使用以下示例（来自“多乐五金店”样本商店的 command.xml 文件）作为指南在 VIEWREG 表中创建一个条目:

```
< viewreg
viewname="OrderOptionsView"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=Shipping.jsp"
internal="0"
https="0" />
```

其中

- viewreg 是此信息将填充的数据库表（VIEWREG）的名称。
- viewname 是视图的名称。
- devicefmt_id 是将使用此视图的设备类型（例如浏览器）。
- storeent_id 是商店实体标识，使用 @ 符号称为内部别名解析。使用内部别名解析时，以别名替换 XML 文档中的主键（标识）。然后在 XML 文件中的其它位置使用此别名来引用该元素。这样就无需了解构建 XML 文件所需的唯一索引。发布期间，标识解析器会以唯一值替换 @ 符号。关于更多信息，请参阅第 289 页的附录 B，『创建数据』。

- `interfacename` 是视图命令接口名称。缺省选项为 ForwardView、DirectView 和 RedirectView。
- `classname` 是视图实现类名称。通常，此名称是在末尾附加 Impl 的接口名称。
- `properties` 是设置为命令的输入属性的缺省名称-值对。如果总是显示同一页面，请设置此属性中的 JSP 文件名，例如，docname=Shipping.jsp。
- `internal` 指示视图对于 WebSphere Commerce 是否是内部的。内部视图由 WebSphere Commerce 工具使用。是内部时请使用 1，是外部时请使用 0。您创建的视图必须是外部的。
- `https` 是此 URL 请求所需的安全 HTTP。需要安全 HTTP 时请使用 1，不需要时请使用 0。

关于 **@** 和 **&** 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

第 8 章 产品目录有用资源

和传统的产品目录一样，在线产品目录也是由提供待售的商品和服务组成的。尽管在线产品目录的大小和结构在商店与商店之间可能极不相同（取决于可购买商品的类型和数量），但产品目录需要以下内容:

- 所销售的商品，包括:
  - 价格，它几乎总是包含在在线产品目录中。
  - 产品数据，如商品的描述和图像。
  - 类别，大部分（但是不全部）产品目录都将商品分类以便顾客浏览。
- 所销售商品的显示方法。产品目录显示页面概述顾客对页面外观的评价，并且在各产品目录页面之间提供一致的外观。如何构建产品目录取决于商品。

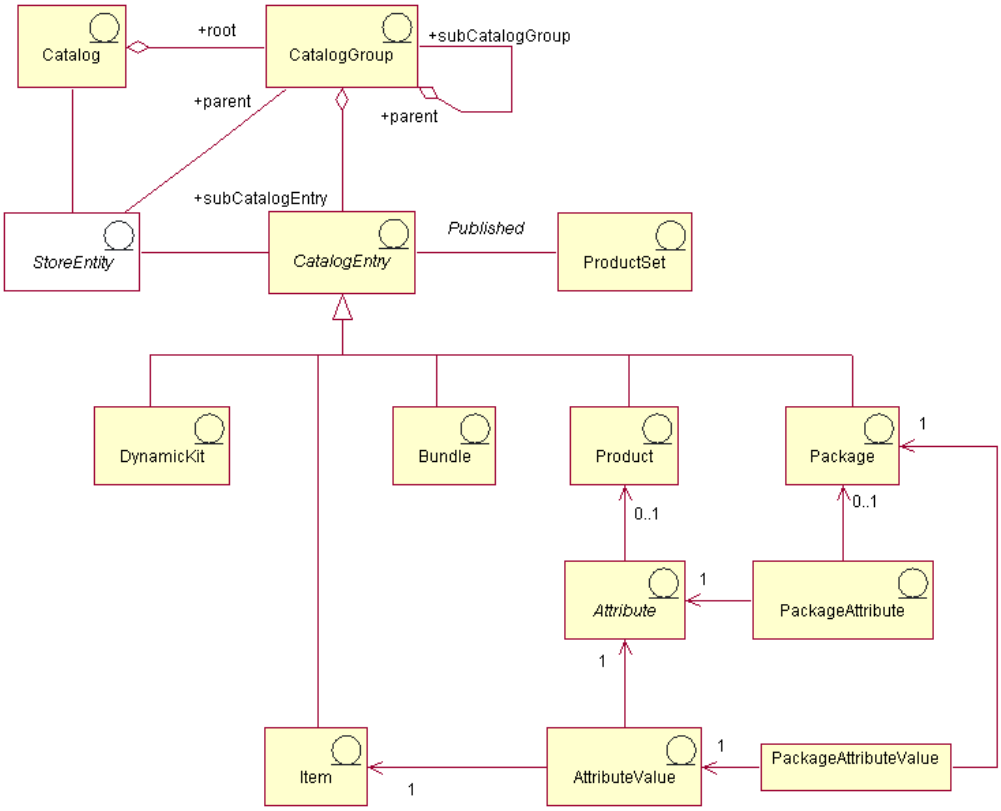
了解 WebSphere Commerce 中的产品目录

WebSphere Commerce 对商店的在线产品目录有若干要求。WebSphere Commerce 系统中的每个商店都必须具有主产品目录。主产品目录是管理商店商品的中心位置。它是单个目录，其中包含所有产品、商品、关系和商店中所有待售商品的标准定价。

您可以在商店之间共享主产品目录，也可以根据需要定义任意数量的商店。除创建用于产品目录管理的主产品目录以外，您还可以选择创建用于显示的一个或多个导航产品目录。导航产品目录可以包含与主产品目录相同的条目，但将具有用于顾客显示的更灵活结构。您可以具有任意数量的导航产品目录。但是，由于它是用于管理在线商品的主产品目录，因此建议您也使用主产品目录作为导航产品目录，从而最小化维护开销。

如果正创建与 WebSphere Commerce 商店一同使用的新主产品目录，或正使用 WebSphere Commerce 样本商店（例如“多乐五金店”）提供的现有主产品目录，您将必须修改产品目录以满足这些需求。下图概述了 WebSphere Commerce 中产品目录的基

本结构。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

产品目录

*产品目录*是信息模型的起点。产品目录包含在线产品目录的所有层次结构和导航信息，是网上商店中可用于显示和购买的产品目录组和产品目录条目的集合。

在 WebSphere Commerce 中，产品目录在数据库中以*产品目录实体*表示。产品目录实体由唯一的产品目录标识和产品目录的描述（例如产品目录名称）组成。由于每个产品目录都是独立的唯一实体，因此可以很容易与一个或多个商店关联。WebSphere Commerce 系统中的每个商店必须至少与一个产品目录实体相关。主产品目录是特殊的产品目录，包含网上商店中的所有待售商品。

产品目录组

*产品目录组*是产品目录条目的一般分组，是为导航和产品目录划分而创建的。产品目录组属于产品目录，并可以包含多个产品目录组或产品目录条目。可以使产品目录组与多个产品目录关联。产品目录组也称为*类别*。

平面产品目录是不将其产品分组为类别的产品目录，而是显示产品列表。尽管可以在 WebSphere Commerce 中创建平面产品目录，但是还是建议您出于结构化和导航的目的创建产品目录组。

创建产品目录组时，必须首先以层次结构或倒排树排列产品目录。该树从一般产品目录组（称为根产品目录）开始，分支扩展出越来越特定的子类别，直到无法继续细分。每个仅包含产品的最低级产品目录组为叶。产品目录组是其下类别的父级，是其上类别的子级。例如，男式时装是男式时装类别的分组，而类别组长裤和衬衫是产品的分组。

产品目录条目

每个产品目录组都包含*产品目录条目*。在线产品目录中的产品目录条目代表可订购的商品。条目通常具有名称或部件号、描述、一个或多个价格、图像和其它详细信息。产品目录条目可以是产品、商品、打包销售商品、捆绑销售商品或可改装的成套商品。如果需要，您可以创建不适合以下任一现有型号（共五种）的新产品目录条目类型。以下提供了每种类型产品目录条目的更多信息。

产品

*产品*是一种产品目录条目类型。产品作为显示相同属性的商品组或库存标识组的模板。例如，衬衫是产品目录中的一种产品。将属性和属性值添加到衬衫之后，每个变体都成为商品，如小号黑色衬衫。

商品

*商品*是商品的有形单位，具有特定名称、部件号和价格。例如，小号黑色衬衫是商品，而衬衫是产品。与特定产品相关的所有商品都显示同一组属性，并通过其属性值区分。

注意：对于 WebSphere 贸易加速器用户，商品和*库存标识*被视为同义词。使用 WebSphere 贸易加速器中的“产品管理”工具时，可订购商品称为库存标识。在 WebSphere Commerce 数据库模式中，此特定类型的产品目录条目称为商品。

打包销售商品

*打包销售商品*是产品目录条目的基本集合。例如，计算机打包销售商品可能包含不能单独销售的特定中央处理单元、显示器和硬盘驱动器。与产品类似，打包销售商品具有定义属性，是用于完全解析的打包销售商品的容器。完全解析的打包销售商品可以和库存标识相比。打包销售商品具有自己的价格而且是可以添加到购物车的实际可订购库存标识。浏览期间或将打包销售商品放入购物车之后不能分解或修改打包销售商品。

捆绑销售商品

*捆绑销售商品*是产品目录条目的集合，使顾客可以一次购买多种商品。例如，计算机的捆绑销售商品可能由中央处理单元、显示器、硬盘驱动器和 CD-ROM 驱动器组成。捆绑销售商品是一组商品，或产品、商品和完全解析的打包销售商品的组合。如果您选择仅包含商品的捆绑销售商品，则此捆绑销售商品分解为独立的可订购库存标识，这些库存标识是单独添加到购物车的。但是，如果您选择包含产品的捆绑销售商品，则需要通过库存标识解析将这些产品解析为商品才能将其添加到购物车。任一情况下一旦分解了捆绑销售商品，且将其组件商品添加到购物车，您就可以修改或除去每个商品。

可改装的成套商品

*可改装的成套商品*是一类可由顾客动态配置的产品目录条目。此产品配置（或分组）以顾客需求为基础并作为单个单元销售。可改装的成套商品的组件由外部产品配置器通过一组预定义的规则和用户交互编写。将可改装的成套商品添加到订单与添加打包销售商品类似。如同打包销售商品，不能修改可改装的成套商品的单独组件，且整个配置必须作为整体实现。但是，您可以使用外部产品配置器进行重新配置，来更改可改装的成套商品的组件。

产品集

*产品集*与已发布的 CatalogEntry 对象关联。产品集提供将产品目录划分为逻辑子集的机制。此划分操作使您可以向不同用户显示产品目录的不同部分。您可以创建合同并指定合同参与方仅有权购买属于预定义产品集的产品。WebSphere Commerce 提供从主产品目录创建产品集的工具，并在合同中使用这些工具进行权利过滤。

属性

*属性*是网上商店中产品的属性，如颜色或大小。属性属于产品。属性和属性值的每种可能组合都定义一个商品。

属性值

*属性值*是属性的特性，如特定颜色（蓝色或黄色）或大小（小号、中号或大号）。在将属性值指定给商品之前必须对其进行预定义。属性和属性值的每种可能组合都定义一个商品。

打包销售商品属性

*打包销售商品属性*是从打包销售商品中包含的产品的属性继承的。打包销售商品可以具有零个或多个打包销售商品属性。一个打包销售商品属性引用一个属性。

打包销售商品属性值

*打包销售商品属性值*是对打包销售商品属性指定的值。打包销售商品属性值是从打包销售商品中包含的产品的属性值继承的。一个打包销售商品属性值引用一个属性值。

关于 WebSphere Commerce 中产品目录有用资源结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的产品目录数据模型。

在 WebSphere Commerce 中创建产品目录有用资源

要为商店创建产品目录有用资源，您需要通过向若干 WebSphere Commerce 数据库表添加信息创建主产品目录。可以使用由装入程序软件包装入数据库的 XML 文件创建产品目录。如果正在创建多文化产品目录，则将需要用于商店所支持的每种语言环境的各自的 XML 文件。每个特定于语言环境的 XML 文件添加产品目录、产品目录组和产品目录条目的可翻译信息（例如描述）。

以下是产品目录创建过程的概述:

1. 在 WebSphere Commerce 中，产品目录是使用 XML 文件来创建的。创建产品目录的过程以一个产品目录实体开始，您的数据库等同于一个记录在纸上的产品目录。

2. 通过添加产品目录组来创建产品目录结构和导航，以此确定您的商品的产品目录和布局。
3. 创建库存信息作为产品目录条目的基础。
4. 以产品目录条目的形式添加您的商品，产品目录条目可以代表产品、库存标识、打包销售商品、捆绑销售商品和可改装的成套商品。
5. 向产品目录中的产品添加属性和属性值以将不同的库存标识彼此区分开。
6. 您可以为了促销的目的创建打包销售商品和捆绑销售商品以将某些产品目录条目组合在一起。
7. 接下来创建产品目录组和产品目录条目之间的关系。这将确定哪些条目属于产品目录组。
8. 您可以创建产品目录条目的销售策略关联以作为产品推荐措施。
9. 将您的产品目录、产品目录组和产品目录条目关联到您的 WebSphere Commerce 商店。
10. 在最后的步骤中，您需要创建:
    a. 商品的税款,
    b. 装运方式,
    c. 充当库存仓库以及装运和接收中心的实现中心。一个商店可以定义多个实现中心,
    d. 商品的价格。

创建主产品目录

要创建包含多个类别级别的主产品目录，请完成以下任务:

第 1 部分: 准备产品目录创建

1. 请复查产品目录信息及其在 WebSphere Commerce 内对应的对象和数据模型。产品目录信息是 WebSphere Commerce Server 的组件，为可订购商品提供在线产品目录导航、划分、分类和关联。
2. 请复查 WebSphere Commerce 装入程序软件包信息。装入程序软件包主要由准备数据并将数据装入 WebSphere Commerce 数据库的实用程序组成。可以使用装入程序软件包装入大量数据并更新数据库中的数据。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。
3. 请复查第 289 页的附录 B，『创建数据』中的信息。
4. 通过管理控制台创建组织作为产品目录所有者。关于更多信息，请参阅 WebSphere Commerce 联机帮助主题 “创建组织”。
5. 使用 “多乐五金店” 样本商店中现有的 XML 条目和 `catalog.xml` 文件作为指南为主产品目录创建新的 XML 文件。如果正在创建多文化的产品目录，则为商店支持的每种语言环境创建各自的 `catalog.xml` 文件。特定于语言环境的文件应当指定所有描述信息，以便比较容易进行翻译。在此示例中，一个 `catalog.xml` 文件将用于无需翻译的所有信息，另一个 `catalog.xml` 将用于商店支持的每种语言环境并将包括需要翻译的信息。或者，如果愿意，可以按需要使用 “多乐五金店” 样本商店现有的 XML 文件并更改信息。 “多乐五金店” 样本商店中的 `catalog.xml` 文件位于其商店归档文件中。要查看 `catalog.xml` 文件，请使用 ZIP 程序将该商店归档文件解压缩。`catalog.xml` 文件位于以下数据目录中:
    - NT *drive*:\WebSphere\CommerceServer\samplestores

- 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

catalog.dtd 文件位于以下目录中:

- NT *drive*:\WebSphere\CommerceServer\xml\sar
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

第 2 部分: 创建产品目录实体

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向 CATALOG 和 CATALOGDSC 表添加信息创建产品目录条目。一个产品目录实体代表数据库中的一个产品目录。

```
<catalog
catalog_id="@catalog_id_1"
member_id="&MEMBER_ID;"
identifier="ToolTech"
description="ToolTech Catalog"
tpclevel="0"
/>
```

其中

- catalog_id 是内部引用号。
- member_id 是标识产品目录所有者的内部引用号。
- identifer 是产品目录的外部名称。
- description 是产品目录的描述。

2. 使用“多乐五金店”样本商店的以下示例作为指南，在特定于语言环境的 XML 文件中添加产品目录的描述以便翻译:

```
<catalogdsc
catalog_id="@catalog_id_1"
language_id="&en_US;"
name="Store master catalog"
/>
```

其中

- catalog_id 是使此语言特定信息与产品目录相关的内部引用号。
- language_id 是语言的标识。
- name 是产品目录的依赖于语言的名称。

第 3 部分: 创建产品目录组

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向 CATGROUP 和 CATGRPDESC 表添加信息创建产品目录组。产品目录组（也称为类别）是其它产品目录组或产品的分组。对产品目录中的每个产品目录组完成此任务:

```
<catgroup
catgroup_id="@catgroup_id_1"
member_id="&MEMBER_ID;"
identifier="Woodworking"
markfordelete="0"
/>
```

   其中

   - catgroup_id 是产品目录组的内部引用号
   - member_id 是标识产品目录所有者的内部引用号。
   - identifer 是产品目录的外部名称。
   - markfordelete 指示产品目录组是否已标记为删除。
     - 0 = 否。
     - 1 = 是。

2. 使用“多乐五金店”样本商店的以下示例作为指南，在特定于语言环境的 XML 文件中添加产品目录组的描述以便翻译。对产品目录中的每个产品目录组完成此任务:

```
<catgrpdesc
language_id="&en_US;"
catgroup_id="@catgroup_id_1"
name="Woodworking"
shortdescription="Woodworking"
longdescription="Woodworking"
published="1"
/>
```

   其中

   - language_id 是语言的标识。
   - catgroup_id 是产品目录组的内部引用号。
   - name 是产品目录的依赖于语言的名称。
   - shortdescription 是产品目录组的简短描述。
   - longdescription 是产品目录组的详细描述。
   - published 指示是否应对 language_id 指示的语言显示此产品目录组:
     - 0 = 否。
     - 1 = 是。

   注: 每次创建产品目录组及其描述时，catgroup_id 都更改为代表新的产品目录组。例如，catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3"、catgroup_id="@catgroup_id_4"，等等。

3. 创建产品目录组后，请通过向 CATTOGRP 表添加信息将顶级产品目录组指定给产品目录。此产品目录组是其下紧挨着的产品目录组的父级。对产品目录中的每个顶级产品目录组完成此任务。使用“多乐五金店”样本商店的以下示例作为指南:

```
<cattogrp
catalog_id="@catalog_id_1"
catgroup_id="@catgroup_id_1"
/>
```

其中

- catalog_id 是产品目录的引用号。
- catgroup_id 是产品目录组的引用号。

注: 每次将顶级产品目录组指定给产品目录时，catgroup_id 修改为代表新的产品目录组关联。例如，catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3" 和 catgroup_id="@catgroup_id_4"，等等。

4. 一旦对产品目录组确定了父级和子级结构，请通过向 CATGRPREL 表添加信息创建产品目录组之间的关系。对产品目录中的每个父级和子级产品目录结构完成此任务。使用“多乐五金店”样本商店的以下示例作为指南:

```
<catgrprel
catgroup_id_parent="@catgroup_id_1"
catgroup_id_child="@catgroup_id_11"
catalog_id="@catalog_id_1"
sequence="0"
/>
```

其中

- catgroup_id_parent 是此关系的源产品目录组。
- catgroup_id_child 是此关系的目标产品目录组。
- catalog_id 是产品目录的引用号。
- sequence 是确定产品目录组内容的显示顺序的编号。

注: 对于每种产品目录组关系，catgroup_id_child 和 sequence 修改为代表新关系。例如，后续关系将显示为 catgroup_id_child="@catgroup_id_12" 与 sequence="1"、catgroup_id_child="@catgroup_id_13" 与 sequence="2"，等等。如果不在产品目录中使用浏览结构，则可以除去 CATGRPREL 关系。

第 4 部分: 创建库存信息

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向 BASEITEM、BASEITEMDSC、ITEMSPC、ITEMVERSN、VERSIONSPC、DISTARRANG 和 STOREITEM 表添加信息创建库存信息。从通过向 BASEITEM 表添加信息创建基本商品开始。基本商品表示具有公共名称和描述的一般产品系列。对产品目录中的每组库存商品完成此任务:

```
<baseitem
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID;"
markfordelete="0"
partnumber="tooltech_sku_102"
itemtype_id="ITEM"
quantitymeasure="C62"
quantitymultiple="1.0"
/>
```

其中

- baseitem_id 是生成的唯一键。
- member_id 是基本商品的所有者。

- markfordelete 指示基本商品是否已标记为删除:
  - 0 = 否。
  - 1 = 是。
- partnumber 唯一标识所有者的基本商品。
- itemtype_id 是基本商品的类型:
  - ITEM = 商品、打包销售商品或捆绑销售商品
  - DNKT = 可改装的成套商品。
- quantitymeasure 是数量倍数的计量单位。
- quantitymultiple 是以整数单位计量的基本商品数量。它和 quantitymeasure 一同指示每个整数单位所表示的金额。

注: 必须为您在产品目录中创建的每一产品创建一个基本商品。每次创建基本商品，baseitem_id 和 partnumber 编号都会更改以创建新的基本商品。例如，新的基本商品将包含 baseitem_id="@baseitem_id_147" 和 partnumber="tooltech_sku_147" 作为条目，而另一基本商品将包含 baseitem_id="@baseitem_id_192" 和 partnumber="tooltech_sku_192" 作为条目，等等。

2. 使用“多乐五金店”样本商店的以下示例作为指南，将关于指定商品的信息添加到数据库中。指定商品是所有属性都具有值的商品，代表产品目录中的商品、打包销售商品、捆绑销售商品或可改装的成套商品。对产品目录中的每个指定商品完成此任务:

```
<itemspc
itemspc_id="@itemspc_id_106"
baseitem_id="@baseitem_id_102"
markfordelete="0"
partnumber="T0000106"
member_id="&MEMBER_ID;"
discontinued="N"
/>
```

其中

- itemspc_id 是生成的唯一键。
- baseitem_id 是产品基本商品。
- markfordelete 指示指定商品是否已标记为删除:
  - 0 = 否。
  - 1 = 是。
- partnumber 唯一标识所有者的指定商品。
- member_id 是指定商品的所有者。
- discontinued 指示指定商品是否已停止。
  - Y = 已停止，如果有足够库存就可以订购，但是不能延迟交货。
  - N = 正在运行，如果脱销可以延迟交货。

注: 必须为您在产品目录中创建的每一商品创建一个指定商品。每次定义指定商品时，itemspc_id、baseitem_id、partnumber 编号都会更改以创建新的指定商品。例如，新的指定商品将包含 itemspc_id="@itemspc_id_108"、baseitem_id="@baseitem_id_102" 和 partnumber="T0000108" 作为条目，而另一指定商品将包含 itemspc_id、baseitem_id 和 partnumber 作为条目，等等。

3. 使用“多乐五金店”样本商店的以下示例作为指南，将商品版本和基本商品之间关系的信息添加到数据库中。对产品目录中的每个此类关系完成此任务:

```
<itemversn
itemversn_id="@itemversn_id_102"
baseitem_id="@baseitem_id_102"
expirationdate="2010-01-01 00:00:00.000000"
versionname="version"
/>
```

其中

- itemversn_id 是生成的引用号，标识商品版本。
- baseitem_id 是基本商品。
- expirationdate 是商品版本失效的时间。
- versionname 唯一标识基本商品的商品版本。

注: 每次创建商品版本和基本商品之间的关系时，itemversn_id 和 baseitem_id 编号都会更改以创建新的关系。baseitem_id 与现有的基本商品匹配。例如，新关系将包含 itemversn_id="@itemversn_id_107" 和 baseitem_id="@baseitem_id_107" 作为条目，而另一关系将包含 itemversn_id="@itemversn_id_108" 和 baseitem_id="@baseitem_id_108" 作为条目，等等。

4. 使用“多乐五金店”样本商店的以下示例作为指南，将产品版本和指定商品之间关系的以下信息添加到数据库中。对产品目录中的每个此类关系完成此任务:

```
<versionspc
versionspc_id="@versionspc_id_106"
itemspc_id="@itemspc_id_106"
itemversn_id="@itemversn_id_102"
/>
```

其中

- versionspc_id 是生成的唯一标识。
- itemspc_id 是产品目录条目相关的指定商品。
- itemversn_id 标识商品版本。

注: 每次创建产品版本和指定商品之间的关系时，versionspc_id 和 itemspc_id 编号都会更改以创建新的关系。itemspc_id 与现有的指定商品匹配。例如，新关系将包含 versionspc_id="@versionspc_id_107" 和 itemspc_id="@itemspc_id_107" 作为条目，而另一关系将包含 versionspc_id="@versionspc_id_108" 和 itemspc_id="@itemspc_id_108" 作为条目，等等。

5. 使用“多乐五金店”样本商店的以下示例作为指南，将分发安排添加到数据库中。分发安排使商店可以销售其自己的库存。对产品目录中的每个分发安排完成此任务:

```
<distarrang
distarrang_id="@distarrang_id_102"
wholesalestore_id="@storeent_id_1"
merchantstore_id="@storeent_id_1"
baseitem_id="@baseitem_id_102"
pickingmethod="F"
startdate="2000-12-25 00:00:00.000000"
enddate="2010-01-01 00:00:00.000000"
/>
```

其中

- distarrang_id 是分发安排的引用号。
- wholesalestore_id 是拥有商家商店可销售的库存的批发商店。此批发商店必须与 merchantstore_id 相同。
- merchantstore_id 是可以销售批发商店库存的商家商店。此商家商店必须与 wholesalestore_id 相同。
- baseitem_id 是分发安排所包括的产品。
- pickingmethod 确定依照此安排从 RECEIPT 表中选取库存的顺序。
  - F = FIFO（先进先出）— 最早接收到的库存。
  - L = LIFO（后进先出） — 最近接收到的库存。
- startdate 是分发安排生效的时间。
- enddate 是分发安排失效的时间。

注: 每次创建分发安排时，distarrang_id 和 baseitem_id 编号都会更改以创建新的分发安排。例如，另一分发安排可能包含值 distarrang_id="@distarrang_id_147" 和 baseitem_id="@baseitem_id_147"，而再一个可能包含 distarrang_id="@distarrang_id_192" 和 baseitem_id="@baseitem_id_192"，等等。

6. 使用“多乐五金店”样本商店的以下示例作为指南，将影响特定商店为特定基本商品的指定商品分配库存的属性添加到数据库中。对产品目录中的每个基本商品完成此任务:

```
<storeitem
baseitem_id="@baseitem_id_102"
storeent_id="@storeent_id_1"
trackinventory="Y"
forcebackorder="N"
releaseseparately="N"
returnnotdesired="N"
backorderable="Y"
creditable="Y"
minqtyforsplit="0"
/>
```

其中

- baseitem_id 是基本商品。
- storeent_id 是商店或商店组。
- trackinventory 控制是否在 RECEIPT 表中跟踪库存:
  - N = 不跟踪库存，且 RECEIPT 表中没有条目。
  - Y = 在 RECEIPT 表中跟踪库存。
- forcebackorder 暂挂基本商品的指定商品的分配:
  - N = 可以分配库存（正常行为）。
  - Y = 即使库存足够，也不能分配库存。
- releaseseparately 控制如何发出基本商品的指定订购商品:
  - N = 订购商品可与其它订购商品一起发出。
  - Y = 订购商品必须分别发货（装在各自的箱子中）。

- returnnotdesired 指示不期望退回商品（例如，易腐食品），即使顾客愿意或能够退回:
  - N = 根据顾客退回商品的意向评估返款请求，但并不期望退货。
  - Y = 视同期望退货，评估返款请求。
- backorderable 指示基本商品的指定商品不能延迟交货:
  - N = 商品不可以延迟交货。
  - Y = 商品可以延迟交货。
- creditable 指示商家是否将给予此商品的返款而无需佣金:
  - N = 按现状销售。
  - Y = 可返款。
- minqtyforsplit 表示如果新订购商品中其余未分配的数量小于特定的最低数量，不会在库存分配期间自动分割订购商品。

注: 每次为商店商品定义库存分配规则时，baseitem_id 编号都会更改以代表新的基本商品。例如，一个新的分配可能包含 baseitem_id="@baseitem_id_147"，而另一个可能包含 baseitem_id="@baseitem_id_192"，等等。

7. 使用“多乐五金店”样本商店的以下示例作为指南，在特定于语言环境的 XML 文件中添加基本商品的描述以便翻译。对产品目录中的每个基本商品描述完成此任务:

```
<baseitmdsc
baseitem_id="@baseitem_id_102"
language_id="&en_US;"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
/>
```

其中

- baseitem_id 是生成的唯一键。
- language_id 是此信息的语言。
- shortdescription 是基本商品的简短描述。
- longdescription 是基本商品的详细描述。

第 5 部分: 创建产品目录条目

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向 CATENTRY 和 CATENTDESC 表添加信息创建产品目录条目。每种类型的产品目录条目（产品、商品、打包销售商品、捆绑销售商品和可改装的成套商品）都代表产品目录中待售商品的可订购商品。您需要为每一产品目录条目定义一个基本商品。对产品目录中的每个产品目录条目完成此任务:

```
<catentry
catentry_id="@product_id_102"
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID"
catenttype_id="ProductBean"
partnumber="T0000102"
mfpartnumber="Sprain-Tools-102"
```

```
mfname="Sprain Tools"
markfordelete="0"
buyable="1"
/>
```

其中

- catentry_id 是产品目录条目的内部引用号。
- baseitem_id 是产品目录条目相关的基本商品。
- member_id 是标识产品目录条目的引用号。
- catenttype_id 标识产品目录条目的类型:
  - ItemBean = 标识商品。
  - ProductBean = 标识产品。
  - PackageBean = 标识打包销售商品。
  - BundleBean = 标识捆绑销售商品。
  - DynamicKitBean = 标识可改装的成套商品。
- partnumber 是标识产品目录条目部件号的引用号。
- mfpartnumber 是生产商用于标识产品目录条目的部件号。
- mfname 是产品目录条目生产商的名称。
- markfordelete 指示产品目录条目是否已标记为删除。
  - 0 = 否。
  - 1 = 是。
- buyable 指示是否可以单独购买产品目录条目:
  - 0 = 否。
  - 1 = 是。

  注: 每次将基本商品添加到产品目录条目中时，catentry_id 和 baseitem_id 顺序都会更改以代表新的产品目录条目。catenttype_id 根据产品目录条目的类型更改。
- 使用“多乐五金店”样本商店的以下示例作为指南，为每一产品目录条目定义指定商品。对产品目录中的每个产品目录条目完成此任务:

  ```
  <catentry
  catentry_id="@catentry_id_106"
  itemspc_id="@itemspc_id_106"
  member_id="&MEMBER_ID"
  catenttype_id="ItemBean"
  partnumber="T0000106"
  mfpartnumber="Sprain-Tools-106"
  mfname="Sprain Tools"
  markfordelete="0"
  buyable="1"
  />
  ```

  其中
  - catentry_id 是产品目录条目的内部引用号。
  - itemspc_id 产品目录条目所属的指定商品。
  - member_id 是标识产品目录条目的引用号。
  - cattentype_id 标识产品目录条目的类型:

- ItemBean = 标识商品。
- ProductBean = 标识产品。
- PackageBean = 标识打包销售商品。
- BundleBean = 标识捆绑销售商品。
- DynamicKitBean = 标识可改装的成套商品。

– partnumber 是标识产品目录条目部件号的引用号。
– mfpartnumber 是生产商用于标识产品目录条目的部件号。
– mfname 是产品目录条目生产商的名称。
– markfordelete 指示产品目录条目是否已标记为删除。
  - 0 = 否。
  - 1 = 是。
– buyable 指示是否可以单独购买产品目录条目:
  - 0 = 否。
  - 1 = 是。

注: 每次将指定商品添加到产品目录条目中时，catentry_id 和 itemspc_id 顺序都会更改以代表新的产品目录条目。catenttype_id 根据产品目录条目的类型更改。在主产品目录结构限制下，产品目录条目不能属于多个类别。要将产品目录条目放到多个类别中，您必须使用导航产品目录。

• 使用“多乐五金店”样本商店的以下示例作为指南，将描述添加到特定于语言环境的 XML 文件中。对产品目录中的每个产品目录条目完成此任务:

```
<catentdesc
catentry_id="@product_id_102"
language_id="&en_US"
name="Circular"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
thumbnail="images/circular_saw_sm.gif"
fullimage="images/circular_saw.gif"
available="1"
published="1"
/>
```

其中

– catentry_id 是内部引用号，指示此特定于语言的信息所相关的产品目录条目。
– language_id 是语言的标识。
– name 是产品目录条目的依赖于语言的名称。
– shortdescription 是产品目录条目的简短描述。
– longdescription 是产品目录条目的详细描述。
– thumbnail 是缩略图的路径。
– fullimage 是全图像的路径。
– available 指示产品目录条目可用的时间长度。
– published 指示是否应以 language_id 指示的语言显示此产品目录条目
  - 0 = 显示。

- 1 = 不显示。

第 6 部分: 创建属性和属性值

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向特定于语言环境的 XML 文件中的 ATTRIBUTE 和 ATTRVALUE 表添加信息创建产品的属性和属性值以便翻译。产品目录中的每个产品都具有一组特定属性，如衬衫或长裤的大小和颜色。商品是由属性值定义的。例如，衬衫是产品，中号黑衬衫是商品。对产品目录中的每个属性完成此任务:

```
<attribute
attribute_id="@attribute_id_103"
language_id="&en_US"
attrtype_id="STRING"
name="Amps"
sequence="0"
description="Amps"
catentry_id="@product_id_102"
description2="Amps"
/>
```

其中

- attribute_id 是属性的内部引用号。
- language_id 是属性所属的语言。
- attrtype_id 是相应属性值的类型。
- name 是属性的名称。
- sequence 是确定给定产品的属性显示次序的顺序编号。
- description 是属性的描述。
- catentry_id 是此属性所属产品的引用号。
- description2 是属性的附加描述。

注: 每次将属性添加到由 catentry_id 定义的产品中时，attribute_id 顺序都会更改以代表新属性。

2. 使用“多乐五金店”样本商店的以下示例作为指南，添加属性值。对产品目录中的每个属性值完成此任务:

```
<attrvalue
attrvalue_id="@attrvalue_id_114"
language_id="&en_US"
attribute_id="@attribute_id_103"
name="12.0amps"
attrtype_id="STRING"
stringvalue="12.0amps"
sequence="0"
catentry_id="@catentry_id_106"
/>
```

其中

- attrvalue_id 是属性值的内部引用号
- language_id 是此属性值所属的语言
- attribute_id 是与此值关联的属性的内部引用号
- name 是属性值的名称
- attrtype_id 是属性值的类型

- stringvalue 是属性值
- sequence 是确定给定属性的属性值显示次序的顺序编号
- catentry_id 是此属性值描述的商品标识

注: 每次将属性值添加到属性中时，attrvalue_id 顺序都会更改以代表不同值。attribute_id 顺序会更改以代表不同属性。sequence 随着每个新的属性值增加。例如，后续属性值将是 sequence="1"、sequence="2" 和 sequence="3"，等等。

第 7 部分: 创建产品和商品之间的关系

1. 为产品目录创建产品和商品之后，请通过向 CATENTREL 表添加信息定义产品和商品之间的关系。使用“多乐五金店”样本商店的以下示例作为指南。对产品目录中的每个产品和商品关系值完成此任务:

```
<catentrel
catentry_id_parent="@product_id_147"
catreltype_id="PRODUCT_ITEM"
catentry_id_child="@catentry_id_152"
sequence="2"
quantity="1"
/>
```

其中

- catentry_id_parent 是此关系中源产品目录条目的引用号，即产品。
- catreltype_id 是关系类型: PRODUCT_ITEM
- catentry_id_child 是此关系中目标产品目录条目的引用号，即商品。
- sequence 是用于确定显示顺序的顺序编号。
- quantity 是可与关系关联的数量。

注: 每次添加产品和商品之间的关系时，catentry_id_parent 和 catentry_id_child 编号都会更改以基于 catreltype_id 创建不同的关系。对于每种新关系，sequence 编号都不同。例如，如果您具有 sequence="2"，则下一关系将是 sequence="3"，接着是 sequence="4"，等等。

第 8 部分: 创建打包销售商品和捆绑销售商品

1. 一旦创建了产品和商品，请通过向 CATENTRY、CATENTDESC 和 CATENTREL 表添加信息创建打包销售商品和捆绑销售商品。使用“多乐五金店”样本商店的以下示例作为指南，从通过向 CATENTRY 表添加信息创建打包销售商品或捆绑销售商品开始。对产品目录中的每个打包销售商品和捆绑销售商品完成此任务:

```
<catentry
catentry_id="@package_id_102"
member_id="&MEMBER_ID"
catenttype_id="PackageBean"
partnumber="sku-@package_id_102"
mfpartnumber="sku-@package_id_102"
mfname="ToolTech"
markfordelete="0"
buyable="1"
/>
```

其中

- catentry_id 是产品目录条目的引用号。

- member_id 是标识产品目录条目所有者的引用号。
- catenttype_id 标识产品目录条目的类型:
  - PackageBean = 标识打包销售商品。
  - BundleBean = 标识捆绑销售商品。
- partnumber 是标识产品目录条目部件号的引用号。
- mfpartnumber 是生产商用于标识产品目录条目的部件号。
- mfname 是产品目录条目生产商的名称。
- markfordelete 指示产品目录条目是否已标记为删除:
  - 0 = 否。
  - 1 = 是。
- buyable 指示是否可以单独购买产品目录条目。
  - 0 = 否。
  - 1 = 是。

  注: 每次您创建打包销售商品或捆绑销售商品时，catentry_id、partnumber 和 mfpartnumber 编号都会更改以创建不同的打包销售商品或捆绑销售商品。例如，要创建新的打包销售商品，您可以使用 catentry_id="@package_id_103"、partnumber="sku-@package_id_103" 和 mfpartnumber="sku-@package_id_103"，包括将此条目标识为打包销售商品的 catenttype_id="PackageBean"。要创建新的捆绑销售商品，您可以使用 catentry_id="@package_id_110"、partnumber="sku-@package_id_110" 和 mfpartnumber="sku-@package_id_110"，包括将此条目标识为捆绑销售商品的 catenttype_id="BundleBean"，等等。
- 使用“多乐五金店”样本商店的以下示例作为指南，通过向特定于语言环境的 XML 文件中的 CATENTDESC 表添加信息添加打包销售商品和捆绑销售商品描述以便翻译。对产品目录中的每个打包销售商品和捆绑销售商品的描述完成此任务:

```
<catentdesc
catentry_id="@catentry_id_102"
language_id="-1"
name="computer"
shortdescription="Computer"
longdescription="A combination of a central processing unit, monitor, hard drive,
and color printer. An ideal starter system."
thumbnail="images/package_system_sm.gif"
fullimage="images/package_system.gif"
available="1"
published="1"
/>
```

  其中
  - catentry_id 是内部引用号，指示此语言特定信息相关的产品目录条目。
  - language_id 是语言的标识。
  - name 是产品目录条目的依赖于语言的名称。
  - shortdescription 是产品目录条目的简短描述。
  - longdescription 是产品目录条目的详细描述。
  - thumbnail 是产品目录条目的缩略图路径。

– fullimage 是产品目录条目的全图像路径。
– available 指示产品目录条目可用的时间长度。
– published 指示是否应以 language_id 指示的语言显示产品目录条目:
  - 0 = 不显示产品目录条目。
  - 1 = 显示产品目录条目。

• 使用“多乐五金店”样本商店的以下示例作为指南，通过向 CATENTREL 表添加信息创建打包销售商品或捆绑销售商品及其组件之间的关系。对产品目录中的每个打包销售商品或捆绑销售商品组件关系完成此任务:

```
<catentrel
catentry_id_parent="@catentry_id_102"
catreltype_id="PACKAGE_COMPONENT"
catentry_id_child="@catentry_id_97"
sequence="1.0"
quantity="1.0"
/>
```

其中

– catentry_id_parent 是此关系中源产品目录条目的引用号，即打包销售商品或捆绑销售商品。
– catreltype_id 是此关系的类型:
  - PACKAGE_COMPONENT 表示打包销售商品与其组件之间的关系。
  - BUNDLE_COMPONENT 表示捆绑销售商品与其组件之间的关系。
– catentry_id_child 是此关系中目标产品目录条目的引用号，即组件。
– sequence 是用于确定显示顺序的顺序编号。
– quantity 是可与关系关联的数量。

注: 每次创建打包销售商品和捆绑销售商品之间的关系时，catentry_id_parent 和 catentry_id_child 编号都会更改以匹配现有的产品目录条目。对于每种新关系，sequence 编号都不同。例如，如果以 sequence="1.0" 开始，则下一关系将为 sequence="2.0"，接着是 sequence="3.0"，等等。

第 9 部分: 创建产品目录组和产品目录条目之间的关系

1. 在产品目录中创建产品目录组和产品目录条目之后，请通过向 CATGPENREL 表添加信息定义产品目录组和产品目录条目之间的关系。在主产品目录结构限制下，产品目录条目不能属于多个类别。要将产品目录条目放到多个类别中，您必须使用导航产品目录。使用“多乐五金店”样本商店的以下示例作为指南。对产品目录中的每个产品目录组和产品目录条目关系完成此任务:

```
<catgpenrel
catgroup_id="@catgroup_id_11"
catalog_id="@catalog_id_1"
catentry_id="@product_id_102"
sequence="0"
/>
```

其中

• catgroup_id 是此关系的源产品目录组。
• catalog_id 是此关系所在的产品目录。
• catentry_id 是此关系的目标产品目录条目。

- sequence 是确定产品目录组的内容显示次序的顺序编号。

注: 每次创建产品目录组和产品目录条目之间的关系时，catgroup_id 和 catentry_id 号码都会更改以形成不同产品目录组和产品目录条目的新关系。对于每种新关系，sequence 编号都不同。例如，如果以 sequence="0" 开始，则下一关系将为 sequence="1"，接着是 sequence="2"，等等。

第 10 部分: 创建销售策略关联

1. 使用“多乐五金店”样本商店的以下示例作为指南，通过向 MASSOCECE 表添加信息创建产品目录条目之间的销售策略关联。对产品目录中的每个销售策略关联完成此任务:

```
<massoccece
massoccece_id="@relationship_id_100"
massoctype_id="X-SELL"
catentry_id_from="@product_id_1"
catentry_id_to="@product_id_15"
massoc_id="REQUIRES"
quantity="2.0"
rank="1.00000"
/>
```

其中

- massoccece_id 是此条目的引用号。
- massoctype_id 是关联类型的标识:
  - X-SELL = 交叉销售。
  - UPSELL = 优选销售。
  - ACCESSORY = 辅助销售。
- catentry_id_from 是产品目录条目，该条目是关联的源。
- catentry_id_to 是产品目录条目，该条目是关联的目标。
- massoc_id 是语义说明符的标识:
  - REQUIRED
  - OPTIONAL
  - COMES WITH
- quantity 是与此关联相关的数量。
- rank 是用于显示次序的顺序编号。

注: 每次添加销售策略关联时，massoccece_id 编号都会更改以代表新关系。catentry_id_from 和 catentry_id_to 编号会更改以创建关联的新商品内容。

第 11 部分: 将产品目录与商店关联

1. 使用“多乐五金店”样本商店现有的 storecatalog.xml 作为指南，通过在数据库中对商店指定产品目录、其产品目录组和产品目录条目使产品目录与商店关联。您还应当对产品目录组和产品目录条目指定显示页面。将此信息添加到 STORECAT、STORECENT、STORECGRP、DISPCGPREL 和 DISPENTREL 表。如果正在创建多文化的产品目录，则为商店支持的每种语言环境创建各自的商店 - 产品目录关系 XML 文件。

```
<storecat
catalog_id="@catalog_id_1"
storeent_id="@storeent_id_1"
mastercatalog="1"
/>
```

其中

- catalog_id 是产品目录的引用号。
- storeent_id 是数据库中商店实体的引用号。
- mastercatalog 指定商店的主产品目录。值 1 指示此产品目录指定为主产品目录。

2. 使用“多乐五金店”样本商店的以下示例作为指南，将产品目录条目添加到“商店－产品目录”关系。对产品目录中的每个产品目录条目完成此任务:

```
<storecent
storeent_id="@storeent_id_1"
catentry_id="@product_id_102"
/>
```

其中

- storeent_id 是数据库中商店实体的引用号。
- catentry_id 是产品目录条目的引用号。

注: 每次将 catentry_id 添加到商店实体中时，引用号都会更改以匹配现有产品目录条目。

3. 使用“多乐五金店”样本商店的以下示例作为指南，将产品目录组添加到商店实体中。对产品目录中的每个产品目录组完成此任务:

```
<storecgrp
storeent_id="@storeent_id_1"
catgroup_id="@catgroup_id_1"
/>
```

其中

- storeent_id 是数据库中商店实体的引用号。
- catgroup_id 是产品目录组的引用号。

注: 每次将 catgroup_id 添加到商店实体中时，引用号都会更改以匹配现有产品目录组。

.

第 12 部分: 将税款与产品目录关联

1. 使税款与特定商店的产品目录条目中的产品和服务关联。您必须通过将此信息添加到 CATENCALCD 表使税款计算代码与产品目录条目关联。关于更多信息，请参阅第 137 页的『在 WebSphere Commerce 中创建税款有用资源』。

第 13 部分: 将装运方式与产品目录关联

1. 要使装运方式与产品目录中的产品和服务关联，您必须使装运费用计算代码与产品目录条目关联。将此信息添加到 CATENCALCD 表。关于更多信息，请参阅第 121 页的『在 WebSphere Commerce 中创建装运有用资源』。

第 14 部分: 将实现中心与产品目录关联

1. 使产品目录与将产品装运给顾客的实现中心关联。实现中心管理商店的产品库存和装运。将此信息添加到 FFMCENTER 表。关于更多信息，请参阅第 99 页的『在 WebSphere Commerce 中创建实现有用资源』。

第 15 部分: 为产品目录条目创建价格

1. 为产品目录条目创建定价。定价代表产品目录条目的价格范围和使用此价格而必须满足的所有条件。要创建正常运作的产品目录，您需要将出售物信息添加到数据库。将此信息添加到 TRADEPOSCN、TDPSCNCNTR、MGPTRDPSCN、OFFER 和 OFFERPRICE 表。关于更多信息，请参阅第 82 页的『在 WebSphere Commerce 中创建定价有用资源』。或者可以使用 WebSphere 贸易加速器中的“产品管理”工具创建或更新产品目录条目的定价。

第 16 部分: 装入 XML 文件

1. 创建数据之后，可以使用装入程序软件包或通过“商店服务”中的发布功能将 XML 文件装入数据库。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

   注: 您也可以使用 WebSphere 贸易加速器的“产品管理”工具为主产品目录创建产品目录有用资源。关于“产品管理”工具的更多详细信息，请参阅 WebSphere Commerce 联机帮助。

显示商店产品目录有用资源

使产品目录、产品目录组和产品目录条目与商店关联后，通过在数据库中创建这些关系指定 JSP 模板显示产品目录条目和产品目录组。以 XML 文件（可以使用装入程序软件包装入数据库）格式创建这些关系。

“多乐五金店”商店样本的 `storecatalog.xml` 文件位于其商店归档文件中。要查看 `storecatalog.xml` 文件，请使用 ZIP 程序解压缩该商店归档文件。`storecatalog.xml` 文件位于以下数据目录中:

- NT *drive*`:\WebSphere\CommerceServer\samplestores`
- 2000 *drive*`:\Program Files\WebSphere\CommerceServer\samplestores`
- AIX `/usr/WebSphere/CommerceServer/samplestores`
- Solaris Linux `/opt/WebSphere/CommerceServer/samplestores`
- 400 `/qibm/proddata/WebCommerce/samplestores`

  注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

`store-catalog.dtd` 文件位于以下目录中:

- NT *drive*`:\WebSphere\CommerceServer\xml\sar`
- 2000 *drive*`:\Program Files\WebSphere\CommerceServer\xml\sar`
- AIX `/usr/WebSphere/CommerceServer/xml/sar`
- Solaris Linux `/opt/WebSphere/CommerceServer/xml/sar`

- 400 /qibm/proddata/WebCommerce/xml/sar

在可以创建“商店 – 产品目录”关系之前，请确保已创建了商店数据有用资源。请完成以下任务，其中每项任务都在 storecatalog.xml 文件中创建条目:

1. 要在商店中显示产品目录组（类别），您必须对产品目录组指定 JSP 模板。可以对产品目录组指定特定显示页模板，也可以指定缺省模板显示所有产品目录组。使用“多乐五金店”样本商店的以下示例作为指南，通过向 DISPCGPREL 表添加信息指定产品目录组模板。对希望指定给产品目录组的每个模板完成此任务:

```
<dispcgprel
catgroup_id="@catgroup_id_1"
devicefmt_id="-1"
dispcgprel_id="@dispcgprel_id_1"
mbrgrp_id="0"
pagename="/ToolTech/CategoryDisplay.jsp"
storeent_id="@storeent_id_1"
rank="0"/>
```

   其中

   - catgroup_id 是将显示此页面名称的产品目录组的引用号。值 0 指示此页面名称将用于所有产品目录组。
   - devicefmt_id 是将显示页面的设备类别的引用号。值 –1 指示此模板页面将由 HTTP 浏览器使用。
   - dispcgprel_id 是此条目的引用号。
   - mbrgrp_id 是将显示模板页面的成员组的引用号。值 0 指示此模板页面将用于所有成员组。
   - pagename 是显示模板页面的名称和路径。
   - rank 是当多个页面满足选择条件时用于断开连接的顺序编号。

   注: 每次对产品目录组指定 JSP 模板时，catentry_id 都会更改顺序以匹配现有产品目录条目。

2. 要显示商店中的产品目录条目（产品、商品、打包销售商品、捆绑销售商品和可改装的成套商品），您必须对产品目录条目指定 JSP 模板。可以指定缺省模板显示所有产品目录条目，或者指定缺省模板显示每种类型的产品目录条目，例如，一个模板用于产品，另一模板用于商品，或特定模板用于特定产品目录条目。使用“多乐五金店”样本商店的以下示例作为指南，通过向 DISPENTREL 表添加信息指定模板。对要指定给产品目录条目的每个模板完成此任务:

```
<dispentrel
auctionstate="0"
catentry_id="0"
catenttype_id="ProductBean"
devicefmt_id="-1"
dispentrel_id="@dispentrel_id_1"
mbrgrp="0"
pagename="/ToolTech/ProductDisplay.jsp"
storeent_id="@storeent_id_1"
rank="0"/>
```

   其中

   - auctionstate 指示此模板页面显示正在拍卖的产品目录条目。
     - 0 = 不是拍卖模板。
     - 1 = 拍卖模板。

- catentry_id 是将显示此页面名称的产品目录条目的引用号。值 0 指示此页面名称将用于所有产品目录条目。
- catenttype_id 是此页面将用于显示的产品目录条目类型。
  - ProductBean = 显示产品。
  - ItemBean = 显示商品。
  - PackageBean = 显示打包销售商品。
  - BundleBean = 显示捆绑销售商品。
  - DynamicKitBean = 显示可改装的成套商品。
- devicefmt_id 是将显示页面的设备类别的引用号。值 –1 指示此模板页面将由 HTTP 浏览器使用。
- dispentrel_id 是产品目录条目的引用号。
- mbrgrp 是将显示此模板页面的成员组的引用号。值 0 指示此模板页面将用于所有成员组。
- pagename 是显示模板页面的名称和路径。
- storeent_id 是将显示此页面的商店的引用号。
- rank 是当多个页面满足选择条件时用于断开连接的顺序编号。

注: 每次对产品目录条目指定 JSP 模板时，catentry_id 都会更改顺序以匹配现有产品目录条目。

创建导航产品目录

WebSphere Commerce 商店允许两种类型的产品目录: 主产品目录和导航产品目录。遍历的产品目录不需符合施加在主产品目录上的结构限制。这些产品目录用于提供一个灵活的显示结构以允许您创建适合商店需求的遍历产品目录。

特别的，遍历的产品目录无需满足以下施加在主产品目录上的限制:

- 主产品目录必须是恰当的树，这意味着不存在循环且不能使用以下结构: 父类别 A 拥有子类别 B。则 B 和 B 的所有子类别都不是 A 的父类别，这一点很重要。
- 产品不能属于多个类别。

以下任务通过修改“新时尚”样本商店产品目录来创建一个遍历产品目录。得到的产品目录不再可以分类为主产品目录，因为它所遵循的步骤将类别周期和某些产品的分类都引入多个类别中。典型的导航产品目录是通过向类别关系表添加信息创建的: CATGRPREL（它保存子类别关系）和 CATGPENREL（它保存“类别 – 产品”关系）。尽管这些示例涉及“新时尚”，但是您可对您自己的主产品目录遵循这些基本步骤，进行适当的调整以与产品目录信息、结构和设计相匹配。

创建类别循环

“新时尚”产品目录包含四个顶级类别: **男式时装**、**女式时装**、**新品**和**主页促销**。本示例显示了如何将**新品**类别复制到**男士时装**中，以及将**主页促销**复制到**女式时装**中，然后更名为**新品**。

要使用类别循环将“新时尚”样本商店主产品目录更改为导航产品目录，请执行以下操作:

1. 发布“新时尚”商店归档文件以创建“新时尚”样本商店。随 WebSphere Commerce 提供了美国英语以及九种本地语言之一的“新时尚”。选择一个 NewFashion_en_US_*locale*.sar 文件用于发布。
2. 在编辑器中打开 catalog.xml 文件。此文件位于以下 WebSphere Commerce 目录中:
   - NT *drive*:\WebSphere\CommerceServer\samplestores\NewFashion*locale*\data
   - 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores\NewFashion*locale*\data
   - AIX /usr/WebSphere/CommerceServer/samplestores/NewFashion/*locale*/data
   - Solaris Linux /opt/WebSphere/Commerce/samplestores/NewFashion/*locale*/data
   - 400 /qibm/proddata/WebCommerce/samplestores/NewFashion/*locale*/data
3. 在 catalog.xml 文件中定位 CATGRPREL 数据部分。在**男式时装**和**新品**之间创建新的顶级类别关系。当前，这两个类别都处于顶级。对于导航关系，创建新部分，其中在保留了原始结构的同时，将**新品**置为**男式时装**的子类别。在 CATGRPREL 部分下，添加以下摘要:

   ```
   <catgrprel
   catgroup_id_parent="@catgroup_id_11"
   catgroup_id_child="@catgroup_id_21"
   catalog_id="@catalog_id_1"
   sequence="7"
   />
   ```

   其中
   - catgroup_id_parent 是由“新时尚”样本商店定义的父类别的产品目录组内部引用号。在此示例中，@catgroup_id_11 是**男式时装**类别。
   - catgroup_id_child 是“新时尚”样本商店定义的子类别的产品目录组内部引用号。在此示例中，@catgroup_id_21 是**新品**类别。
   - catalog_id 是“新时尚”样本商店定义的产品目录的内部引用号。
   - sequence 是确定由“新时尚”样本商店定义的产品目录组的内容显示顺序的编号。在此示例中，**新品**类别将在前六个**男式时装**子类别之后最后显示。
4. 重复上述步骤，此次在**女式时装**和**主页促销**之间创建新的顶级类别关系。当前，这两个类别都处于顶级。对于导航关系，创建新部分，其中在保留了原始结构的同时，将**主页促销**置为**女式时装**的子类别。在 CATGRPREL 部分下，添加以下摘要:

   ```
   <catgrprel
   catgroup_id_parent="@catgroup_id_20"
   catgroup_id_child="@catgroup_id_22"
   catalog_id="@catalog_id_1"
   sequence="9"
   />
   ```

   其中
   - catgroup_id_parent 是由“新时尚”样本商店定义的父类别的产品目录组内部引用号。在此示例中，@catgroup_id_20 是**女式时装**类别。

- catgroup_id_child 是“新时尚”样本商店定义的子类别的产品目录组内部引用号。在此示例中，@catgroup_id_22 是**主页促销**类别。
- catalog_id 是“新时尚”样本商店定义的产品目录的内部引用号。
- sequence 是确定由“新时尚”样本商店定义的产品目录组的内容显示顺序的编号。在此示例中，**主页促销**类别将最后显示。

5. 既然**主页促销**已是**女式时装**的子类别，则类别名就不正确了。将类别名重命名为**新品**，以与**男式时装**中的新子类别相匹配。
6. 保存 catalog.xml 文件。
7. 要查看更改，请执行以下操作之一：通过“商店服务”发布修改后的“新时尚”商店归档文件，或者如第 252 页的『装入数据库有用资源组』中所示利用装入程序软件包装入 catalog.xml 文件。

向另一类别添加产品

此示例显示了如何在保留原始结构的同时，将产品从一个类别复制到另一类别。**主页促销**类别包含**夏日女式睡衣**产品，该产品也可归属于**女式时装**顶级类别的**睡衣**子类别之下。这些指导将向您显示如何将**夏日女式睡衣**产品及其库存标识复制到**睡衣**类别。

要通过向另一类别添加产品将“新时尚”样本商店主产品目录更改为导航产品目录，请执行以下操作:

1. 发布“新时尚”商店归档文件以创建“新时尚”样本商店。随 WebSphere Commerce 提供了美国英语以及九种本地语言之一的“新时尚”。选择一个 NewFashion_en_US_*locale*.sar 文件用于发布。
2. 在编辑器中打开 catalog.xml 文件。此文件位于以下 WebSphere Commerce 目录中:
   - NT *drive*:\WebSphere\CommerceServer\samplestores\NewFashion*locale*\data
   - 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores\NewFashion*locale*\data
   - AIX /usr/WebSphere/CommerceServer/samplestores/NewFashion/*locale*/data
   - Solaris Linux /opt/WebSphere/CommerceServer/samplestores/NewFashion/*locale*/data
   - 400 /qibm/proddata/WebCommerce/samplestores/NewFashion/*locale*/data
3. 在 catalog.xml 文件中定位 CATGPENREL 数据部分。对原为**主页促销**类别下产品的**夏日女式睡衣**创建新的产品条目。在 CATGPENREL 部分下，添加以下摘要以包含产品:

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@product_id_2692"
sequence="2"
/>
```

其中

- catgroup_id 是由“新时尚”样本商店定义的产品目录组内部引用号。在此示例中，@catgroup_id_18 是**睡衣**类别。
- catalog_id 是“新时尚”样本商店定义的产品目录的内部引用号。
- catentry_id 是由“新时尚”样本商店定义的产品目录条目内部引用号。在此示例中，@catentry_id_2692 是**夏日女式睡衣**产品。
- sequence 是确定由“新时尚”样本商店定义的产品目录组的内容显示顺序的编号。在此示例中，**夏日女式睡衣**产品将最后显示。

4. 在添加了**夏日女式睡衣**产品条目之后，如“新时尚”样本商店中所定义，在 CATGPENREL 部分下添加产品的库存标识条目。当前，**夏日女士睡衣**产品包含十个已定义的库存标识。在 CATGPENREL 部分下，添加以下摘要以包含库存标识:

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2695"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2696"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2697"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2698"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2699"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2700"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2701"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
```

```
catentry_id="@catentry_id_2702"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2703"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2704"
sequence="2"
/>
```

其中

- catgroup_id 是由“新时尚”样本商店定义的产品目录组内部引用号。在此示例中，@catgroup_id_18 是**睡衣**类别。
- catalog_id 是“新时尚”样本商店定义的产品目录的内部引用号。
- catentry_id 是由“新时尚”样本商店定义的产品目录条目内部引用号。在此示例中，@catentry_id_2695 到 @catentry_id_2704 表示已对**夏日女式睡衣**产品定义的十个库存标识。
- sequence 是确定由“新时尚”样本商店定义的产品目录组的内容显示顺序的编号。在此示例中，**夏日女式睡衣**库存标识将最后显示。

5. 保存 catalog.xml 文件。
6. 要查看更改，请执行以下操作之一：通过“商店服务”发布修改后的“新时尚”商店归档文件，或者如第 252 页的『装入数据库有用资源组』中所示利用装入程序软件包装入 catalog.xml 文件。

在 WebSphere Commerce 中管理产品目录有用资源

随着时间的流逝，您将需要更新主产品目录的数据库有用资源信息。维护产品目录是一个进行中的过程，因为您将需要不断地添加和除去商品、创建和关联类别或产品目录组，以及更新诸如描述和价格之类的产品信息。

您可以使用现有的数据库条目和商店的 catalog.xml 文件编辑 WebSphere Commerce XML 数据，以此更改您的产品目录有用资源。使用 WebSphere Commerce 样本商店 XML 文件作为参考，该文件位于以下数据目录中:

- NT *drive*:\WebSphere\CommerceServer\samplestores
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

注: 这些示例来源于“新时尚”样本商店，它标识了为了更改产品目录有用资源信息必须修改哪些 XML 元素。

产品目录组

使用 CATGROUP 和 CATGRPDESC 数据库表在 WebSphere Commerce 产品目录中创建产品目录组。在 catalog.xml 文件中，典型的产品目录组看上去与以下抽取的内容相似:

```
<catgroup
catgroup_id="@catgroup_id_1"
member_id="&MEMBER_ID"
identifier="Accessories"
markfordelete="0"
/>
```

catgroup_id 是产品目录组的内部引用号。在 WebSphere Commerce 中，每个产品目录组都被指定了一个内部引用号，它在添加产品目录条目时标识该组。identifer 是产品目录组的外部名称。在数据库有用资源中这两个元素都是唯一的，并且不可以复制它们。

名称和描述属于特定于语言环境的 catalog.xml 文件，对于您的商店支持的每种语言环境都需要其中一个文件。包含有可翻译信息的典型产品目录组看上去与以下抽取的内容相似:

```
<catgrpdesc
language_id="&en_US"
catgroup_id="@catgroup_id_1"
name="Accessories"
shortdescription="Accessories"
longdescription="Accessories"
published="1"
/>
```

language_id 标识产品目录信息所用的语言。该标识必须更改以匹配商店支持的每种语言。name 向顾客显示，这与 shortdescription 和 longdescription 元素是相同的，这两个元素可以包含产品目录组的简短描述和详细描述。

当创建新的产品目录组时，请遵循以上信息的结构。

注:

1. 虽然 identifer 和 name 元素在以上示例中是等价的，但是其内容可能不同。例如，您可能选择将您的产品目录组重命名为 **Complementary Additions**。在这种情况下，您无需更改 identifer 中的信息，仅需更改 name 中的信息。
2. 当删除产品目录组时，请确保相应的更新了 catgroup_id 显示。例如，如果您还希望删除产品目录组下的产品目录条目，则应该除去整个 XML 条目。然而，如果您计划保留产品目录条目，则需要将 catgroup_id 更改到正确的组。

产品目录条目

产品目录条目是使用 CATENTRY 和 CATENTDESC 数据库表中信息在 WebSphere Commerce 产品目录中创建的。产品目录条目可以是产品、商品、打包销售商品、捆绑销售商品或可改装的成套商品。在 catalog.xml 文件中，典型的产品目录条目看上去与以下抽取的内容相似:

```
<catentry
catentry_id="@product_id_102"
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID"
catenttype_id="ProductBean"
partnumber="product-sku-nf-102"
```

```
mfpartnumber="product-sku-nf-102"
mfname="NewFashion"
markfordelete="0"
buyable="1"
/>
```

catentry_id 是产品目录条目的内部引用号。baseitem_id 是产品目录条目为了库存目的而相关的基本商品。partnumber 是标识产品目录条目部件号的引用号。mfpartnumber 是生产商用于标识产品目录条目的部件号。在数据库有用资源中这些元素都是唯一的，并且不可以复制它们。

catenttype_id 标识产品目录条目的类型：ItemBean、ProductBean、PackageBean、BundleBean 或 DynamicKitBean。

名称和描述属于特定于语言环境的 catalog.xml 文件，对于您的商店支持的每种语言环境都需要其中一个文件。此文件中还包含了商品图像。包含有可翻译信息的典型产品目录组看上去与以下抽取的内容相似:

```
<catentdesc
catentry_id="@product_id_102"
language_id="&en_US"
name="Belt"
shortdescription="Classic belt"
longdescription="This classic belt looks great with your favorite jeans,
or takes you to work in style. 1 1/2 inches wide in full-grain leather
with a solid nickel buckle."
thumbnail="images/mens_accessories_belt_sm.gif"
fullimage="images/mens_accessories_belt.gif"
available="1"
published="1"
/>
```

language_id 标识产品目录信息所用的语言。该标识必须更改以匹配商店支持的每种语言。name 向顾客显示，这与 shortdescription 和 longdescription 元素是相同的，这两个元素可以包含产品目录条目的简短描述和详细描述。

当创建新的产品目录条目时，请遵循以上信息的结构。

注:

1. 当删除产品目录条目时，请确保相应更新了出现的每个唯一元素。例如，如果您还希望删除产品目录组下的产品目录条目，则应该除去整个 XML 条目。然而，如果您计划保留产品目录条目，则需要将 catgroup_id 更改到正确的组。
2. 在创建其它类型的产品目录条目之前，必须先创建产品。

如果不希望手工更改 XML 文件，请选择以下一种方法：产品管理工具或 Catalog Manager 实用程序。

产品管理工具

WebSphere 贸易加速器中的“产品管理”工具让您能够使用各种向导和笔记本来管理商店主产品目录中的产品。可更新产品目录内容或创建新的产品目录数据:

- 使用向导或笔记本创建、更新和删除产品和产品详细信息。产品充当库存标识（即最终销售给顾客的单个商品）的模板。产品详细信息包括产品代码（唯一地标识了产品）、产品名称和描述、任何销售策略选项（例如向顾客显示产品或指示该产品是否是特价商品促销的一部分）、产品图像、税款和装运规格、指定给产品的折扣以及生产商信息。

- 生成、更新和删除可购买的库存标识（或商品）。库存标识表示待售商品的每个可订购商品。与特定产品相关的所有库存标识都显示同一组属性并按其属性值加以区分。对库存标识所作的添加或更改包含了与产品相同的信息，但基于可订购的除外。
- 创建、更新和删除类别（或产品目录组），它们是具有用于组织商店所提供产品或服务的类似属性的一组对象。可通过创建、更改和删除类别及其相关详细信息（例如类别代码、名称和描述，包括父类别和图像），来管理主产品目录的类别层次结构。
- 通过选择父类别或将产品和库存标识从一个类别移动到另一个类别，将产品和库存标识与类别相关联。
- 创建产品的属性和属性值。属性和属性值的每种可能组合都等于一个新库存标识。在将属性值指定给库存标识之前必须对其进行预定义。创建了属性及其值之后，可以创建或更新诸如名称、描述（文本、整数或小数）以及属性和属性值出现顺序之类的信息。
- 创建、更新、删除产品目录定价，并将之与产品相关联。可以用一种或多种货币以及一组条件（例如设置单件或成批数量的价格，为了使用此价格必须满足这些条件），为产品或库存标识定义价格。

可以参阅联机帮助中的“产品管理”部分以获取关于每个任务的详细指导。

注:

1. 建议仅对“产品管理”工具作微小更改。对于大量产品目录更新（例如添加或除去季节性商品或准备清货销售），请使用装入程序软件包。
2. 对产品目录数据的任何更改只有禁用了高速缓存或除去了当前高速缓存的 JSP 页面时才能显示在商店中。关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CacheDelete 命令。CacheDelete 命令启动对动态页面高速缓存的远程清除，并让您能够在不需要直接访问文件系统的情况下管理高速缓存。在使用此命令之前，确保启用了“页面自动失效”。请注意必须作为站点管理员或商店开发者登录才能使用此命令。

Catalog Manager

也可使用装入程序软件包或 Web 编辑器（它们是 Catalog Manager 提供的两个实用程序），来维护产品目录。装入程序软件包对于将大量现有的产品信息导入数据库是很理想的。在 WebSphere Commerce 中，这是用于创建和管理产品目录信息的主要工具。此软件包主要由用于准备数据和将数据装入 WebSphere Commerce 数据库的命令实用程序组成。装入程序软件包还允许将数据库中的数据抽取为 XML 文档、将 XML 数据转换为备用的 XML 文档，以及在以字符定界的变量格式和 XML 数据格式之间转换数据。

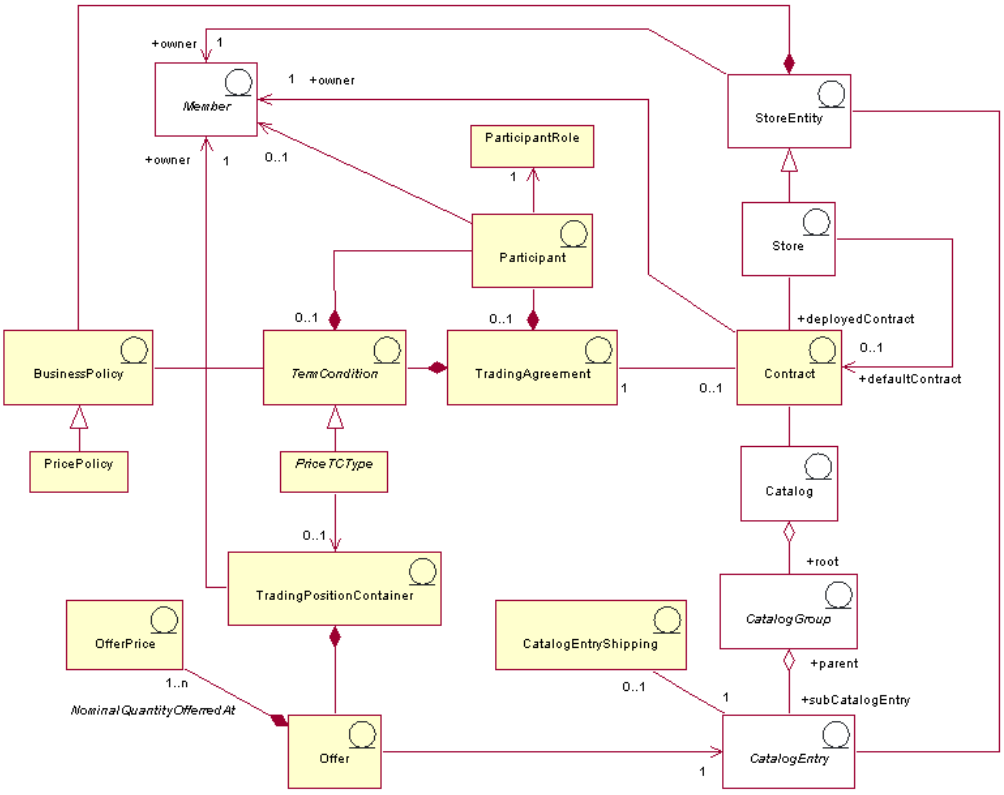
Web 编辑器使用 Web 浏览器界面，您可在其中创建、编辑或删除产品目录数据。用于查看和更新信息的数据输入表单对于 Web 编辑器处于中心位置。在最简单的情况下，表单对应于 WebSphere Commerce 数据库中的表。管理员可以选择是使用所提供的缺省表单，还是定制可用的表单。关于更多信息，请参阅《*IBM WebSphere Commerce 版本 5.4 Catalog Manager 用户指南*》。

第 9 章 对有用资源进行定价

定价代表产品目录条目的价格以及使用此价格而必须满足的所有条件。要创建正常运作的产品目录，您需要将定价信息添加到数据库中。您可以用 XML 文件（可以使用装入程序软件包装入数据库）格式创建定价信息。或者对于少量定价数据，您可以使用 WebSphere 贸易加速器的“产品管理”工具。

了解 WebSphere Commerce 中的定价

下图说明了 WebSphere Commerce Server 中的定价有用资源。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

报价

报价（或定价）是相同产品或商品针对不同顾客或组织的不同价格。报价代表产品目录条目的价格以及顾客为支付此价格而必须满足的条件，例如购买数量。例如，商品或服务对儿童、学生、成人和老人通常有不同的价格。在 WebSphere Commerce 中，报价也称为贸易状态，是贸易状态容器的一部分。

卖价

*卖价*是一种价格，商店通过贸易协议或合同以此价格提供产品目录条目。报价可以具有以多种货币定义的一个或多个卖价。

贸易状态容器

报价是*贸易状态容器*（由成员所拥有）的一部分。贸易状态容器包含贸易状态。通过贸易协议或合同以及合同中的条款和条件，可以使其对所有顾客都可用，或者仅对特定组中的顾客可用。依据合同，贸易状态容器是价格商业对象，可以由多个价格业务策略引用，并且可以由商店组中的某个商店或所有商店共享。贸易状态容器也称为*价格列表*。

条款和条件

*条款和条件*定义了贸易协议的行为和属性。许多条款和条件都引用业务策略，因为商店操作的若干方面都是由业务策略定义的。

定价条款和条件的类型

*定价条款和条件*定义依据合同可用的产品以及顾客为产品支付的价格。一份合同中至少需要以下定价条款之一:

- *定制的价格列表*指定待售产品的列表及其价格都在合同中定制待售并且其价格已定制。商品不限于商店产品目录中的某一部分，可以来自商店产品目录中的任何地方。
- *带有调价条款的整个产品目录*为商店产品目录中的所有可销售产品提供商店产品目录中定义的基本价格基础上的百分比调价（加价或折扣）。如果未指定调价，则商品以基本价格销售。
- *带有调价条款的价格列表*为价格列表中的所有可销售产品提供商店产品目录中定义的基本价格基础上的百分比调价（加价或折扣）。如果未指定调价，则商品以基本价格销售。
- *带有可选调价条款的价格列表*类似于带有调价的价格列表，不同之处在于其调价并不适用于整个价格列表。调价是对价格列表的子集进行的。价格列表的子集可以是产品集业务策略，也可以是定制的产品集。

贸易协议

*贸易协议*可以是合同、RFQ、商业帐户或拍卖。贸易协议是在卖方和买方之间协商的协议，根据该协议，买方可以依据合同中规定的指定条款和条件以及业务策略购买特定商品。例如，它使顾客可以依据定价条款和条件在指定的时间期限内以指定的价格从商店购买产品。在 WebSphere Commerce 中，所有顾客必须依据合同在商店中进行购物，一个商店可以部署一个或多个合同，且其中之一可以指定为缺省合同。缺省合同包含与一组商店缺省策略关联的一组条款和条件。贸易协议可以包含零个或多个不同角色的参与方。

参与方

*参与方*可以是贸易协议的一部分，也可以是条款和条件的一部分。参与方是个成员，可以是成员组、组织等等。如果为合同指定了买方角色的参与方，则买方必须是买方参与者的成员以根据合同进行购物。合同中的条款和条件同样也可以包含零个或多个参与方。

参与方角色

参与方可以具有以下*参与方角色*之一:

- 创建者
- 卖方
- 买方
- 供应商
- 核准员
- 帐户持有者
- 买方联系人
- 卖方联系人
- 代理人
- 管理员。

合同

*合同*包含产品的卖价。在 WebSphere Commerce 中，所有顾客都必须依据合同进行购物。合同使顾客可以依据合同中规定的条款和条件及业务策略，在指定的时间内以指定的价格从商店购买产品。商店拥有零个或多个合同，并拥有至少一个缺省合同。

业务策略

*业务策略*是商店或商店组所遵循的规则集，定义业务过程、行业惯例，以及商店或商店组出售物的范围和特征。业务策略的实施涉及以下各项的组合：实现业务策略规则的一个或多个业务策略命令、对规则作用的商业对象的引用以及配置业务策略命令操作的一组属性。

价格策略

*价格策略*包含对价格列表的引用，可以与多个业务策略命令关联，这些命令定义将如何在价格列表上实现业务策略。可以为商店或商店组定义策略。如果为商店组注册策略，则该组中的所有商店都可以使用此策略。

产品目录条目装运

*产品目录条目装运*信息包含关于如何包装产品进行装运的信息。每个产品目录条目可以定义不同类型的装运信息。例如，产品包装后的高度、重量和产品长度。

其它定价有用资源

以下有用资源与定价关联:

- 拥有贸易状态容器的*成员*。一个贸易状态容器仅有一个所有者。
- *商店实体*表示 WebSphere Commerce Server 数据库中的商店。
- *产品目录*包含将在合同中引用的产品目录条目。产品目录包含在线产品目录的所有层次结构和导航信息，是网上商店中可用于显示和购买的产品目录组和产品目录条目的集合。
- *产品目录组*（或类别）是产品目录条目的一般分组，是为导航和产品目录分割而创建的。产品目录组属于产品目录，并可以包含多个产品目录组或产品目录条目。可以使产品目录组与多个产品目录关联。

- *产品目录条目*代表在线产品目录中的可订购商品。产品目录条目属于产品目录组。一个报价总与一个产品目录条目关联。

关于 WebSphere Commerce Server 中定价有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的定价对象和数据模型。

在 WebSphere Commerce 中创建定价有用资源

您有两个选项可用于创建定价有用资源:

- 使用 WebSphere 贸易加速器中的“产品管理”工具创建价格。使用 WebSphere 贸易加速器中的工具最适合为特小型产品目录创建价格。
- 在 XML 文件中创建价格，此文件可以用 WebSphere Commerce 装入程序软件包装入，或可以作为商店归档文件的一部分，可以通过“商店服务”发布。此方法最适合创建大量数据。

关于使用 WebSphere 贸易加速器中的“产品管理”工具创建价格的更多信息，请参阅 WebSphere Commerce 联机帮助。关于在 XML 文件中创建价格的更多信息，请参阅『在 XML 文件中创建定价有用资源』。

在 XML 文件中创建定价有用资源

请以 XML 文件（这些文件可以使用装入程序软件包装入数据库）格式创建定价有用资源。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

1. 复查用于为样本商店创建定价有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包括两个 `offering.xml` 文件，其中包括定价信息。要查看商店归档文件中的 `offering.xml` 文件，请使用 ZIP 程序将其解压缩。`offering.xml` 文件位于数据目录中。特定于语言的 `offering.xml` 位于数据目录中特定于语言环境的子目录中。
2. 请复查第 289 页的附录 B，『创建数据』中的信息。
3. 通过复制样本商店归档文件中的 `offering.xml` 文件之一或通过创建新文件创建 `offering.xml` 文件。关于更多信息，请参阅对应于 `offering.xml` 的 DTD 文件。DTD 文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\xml\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

4. 创建一个贸易状态容器。要为商店中的商品提供价格，您必须首先创建贸易状态容器。要创建贸易状态容器，请向 TRADEPOSCN 表添加信息。
   a. 使用以下示例作为指南，在您的 XML 文件中为 TRADEPOSCN 表创建贸易状态容器:

```
<tradeposcn
tradeposcn_id="@tradeposcn_id_101"
member_id="&MEMBER_ID"
markfordelete="0"
name="ToolTech"
precedence="0"

/>
```

   其中

   - tradeposcn_id 是生成的唯一键
   - member_id 是贸易状态容器的所有者。
   - markfordelete 如下:
     - 0 = TradingPositionContainer 可以使用
     - 1 = TradingPositionContainer 已经标记为删除（请参阅 DBClean 实用程序）并且不应使用
   - name 是贸易状态容器的助记名称，对于特定所有者是唯一的。
   - precedence 是在特定时间有多个贸易状态容器合格时，使用 PRECEDENCE 最高的贸易状态容器。

5. 通过向 CATGRPTPC 表添加信息，使主产品目录与贸易状态容器关联。使主产品目录与贸易状态容器关联时，主产品目录中的每个产品目录条目必须有标准价格。关于创建主产品目录的更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。
   a. 使用以下示例作为指南，通过向 CATGRPTPC 表添加信息，使主产品目录与贸易状态容器关联:

```
<catgrptpc
catalog_id="@catalog_id_1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

   其中

   - catalog_id 是主产品目录。
   - tradeposcn_id 是贸易状态容器。

6. 通过向 OFFER 和 OFFERPRICE 表添加信息，创建产品目录条目的报价和卖价

a. 使用以下示例作为指南，通过向 OFFER 表添加信息，为产品目录条目创建报价。请注意，在可以创建价格之前必须已创建产品目录条目。关于创建产品目录条目的更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。

```
offer
offer_id="@offer_id_138"
startdate="2000-06-19 00:00:00.000000"
catentry_id="@product_id_102"
precedence="0"
published="1"
identifier="1"
flags="1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

其中

- offer_id 是生成的唯一键。
- startdate 是该报价有效的时间范围的开始时间。
- catentry_id 是提供待售的产品目录条目。
- precedence 是在特定时间有多个报价有效时，使用 PRECEDENCE 最高的报价。
- published 是
  - 0 = 未发布（临时禁用）
  - 1 = 已发布
  - 2 = 标记为删除（且未发布）
- identifier 是与其指定的产品目录条目和贸易状态容器一起唯一标识此报价的数值。
- flags 是
  - 1 = shiptoAddressRequired — 如果为 1，则当 OrderItem 引用此 Offer 但是没有送货地址时，OrderPrepare 将返回错误。
- tradeposcn_id 是贸易状态容器，此报价是该贸易状态容器的组成部分。

b. 使用以下示例作为指南，通过向 OFFERPRICE 表添加信息，为产品目录条目创建卖价。卖价是产品目录条目提供待售的实际价格。请注意，在可以创建价格之前必须已创建产品目录条目。关于创建产品目录条目的更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。

```
<offerprice
offer_id="@offer_id_138"
currency="USD"
price="590.00"
/>
```

其中

- offer_id 是与此价格关联的报价。
- currency 是提供此价格所用的货币。
- price 是报价所引用的产品额定数量（请参阅 CATENTSHIP.NOMINALQUANTITY）的价格。

注: 要在商店中显示多种货币，请在 OFFERPRICE 表中为每种货币创建一个单独的 XML 条目。例如，要以加拿大元显示货币，请在新的 XML 条目中使用 currency="CAD"。price 值将更改为反映使用加拿大元计算的价格。或

者，您可以使用转换，使顾客可以基于所选择的货币显示不同的兑换率。
关于更多信息，请参阅第 111 页的『使用 XML 文件创建货币有用资源』。

c. 对产品目录中所有的产品目录条目重复步骤 a 和 b。

关于 **@** 和 **&** 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 10 章 合同有用资源

在 WebSphere Commerce 中，所有商店顾客都必须依据合同进行购物。合同使顾客可以依据特定条件，在指定的时间内以指定的价格从商店购买产品。当浏览商店的产品目录时，顾客仅可看见商店内对其授权的合同中所涵盖的产品。

如果您希望与商店没有任何合同的顾客（例如临时购物者）可以在商店中购物，或者如果您希望顾客可以购买他们的合同中未涵盖的产品，则您的商店将需要*缺省合同*。

重要信息

WebSphere Commerce 专业版仅支持商店缺省合同。

除商店缺省合同之外的合同仅受 WebSphere Commerce 商务版支持。

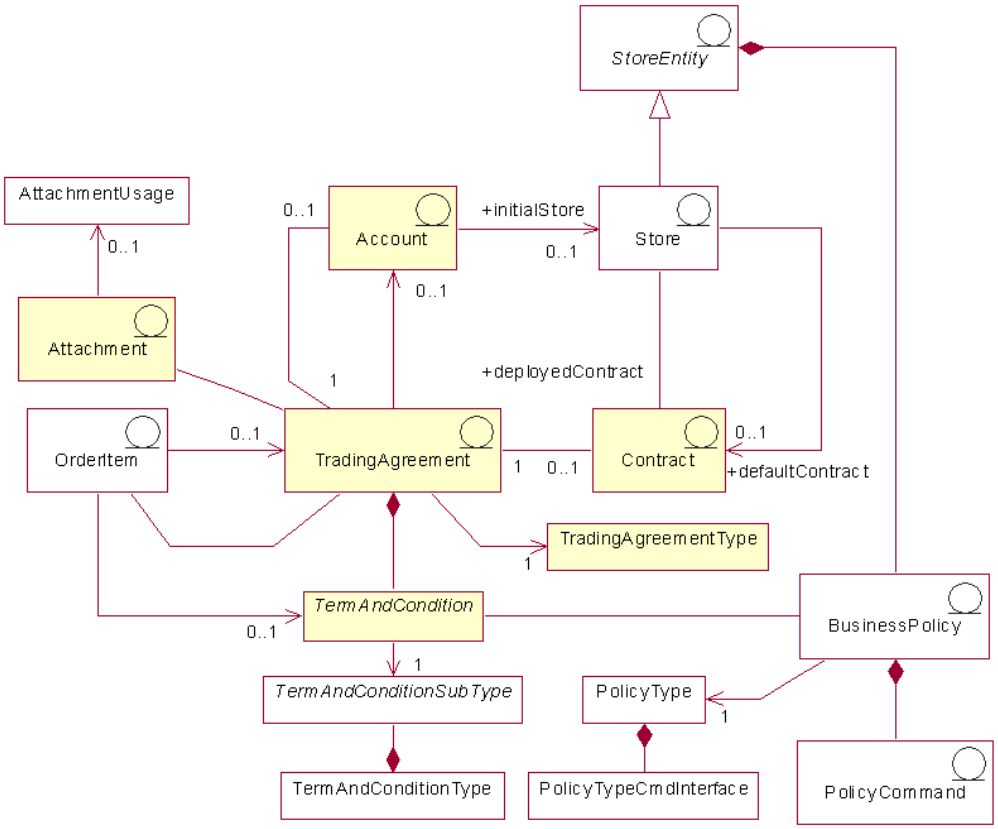
要使所有顾客都可以在商店中购物，用 WebSphere Commerce 创建的商店必须包含以下信息:

- 业务策略
- 缺省合同

业务策略由缺省合同引用，由此使所有顾客都能在商店中购物。

了解 WebSphere Commerce 中的合同

下图说明了 WebSphere Commerce 中的合同结构:



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

帐户（商业帐户）

商业帐户代表买方组织和卖方组织之间的关系。商业帐户可以用于组织各种贸易协议，还可以用于指定与买方和卖方之间的关系相关的条款和条件，例如: 发票定制、购买订单验证或维持买方在卖方下的信用限额。

合同与商业帐户关联，因为其代表买方和卖方之间的协议。此处的特例是不能与商业帐户关联的商店缺省合同。商业帐户可以有很多合同与之关联。

商业帐户是一种贸易协议类型。关于贸易协议的描述，请参阅第 89 页的『贸易协议』。

重要信息: 商业帐户仅受 WebSphere Commerce 商务版支持。

合同

有两种类型的与商店关联的活动合同: 已部署合同和缺省合同。已部署合同对特定买方组织或个人买方作了授权，并可以在创建商店之后使用 WebSphere 贸易加速器进行创

建。一个已部署合同与一个商业帐户相关联。缺省合同定义了商店对与商店没有其它任何合同的买方所采取的缺省行为。仅可使用 XML 文件创建缺省合同，且对一个商店只能定义一个缺省合同。关于合同的更多信息，请参阅联机信息。关于创建缺省合同有用资源的信息，请参阅第 90 页的『在 WebSphere Commerce 中创建缺省合同有用资源』。

合同是一种贸易协议类型。关于贸易协议的描述，请参阅『贸易协议』。

贸易协议

WebSphere Commerce 提供了很多贸易机制管理买方与卖方之间的交互。以下贸易机制受不同版本的 WebSphere Commerce 支持:

- 拍卖（商务版和专业版均支持）
- 商业帐户（仅商务版支持）
- 合同（请参阅本章先前所讨论的限制条件）
- 报价请求（RFQ）（仅商务版支持）

所有这些贸易机制都具有公共属性。例如，所有的贸易机制都有参与方，并且都有用于管理贸易机制行为的规则。在 WebSphere Commerce 中，管理贸易机制行为的规则称为*条款和条件*。

贸易协议代表贸易机制的一个实例，并记录贸易机制的该实例的属性。WebSphere Commerce 中的每个合同、商业帐户和 RFQ 都由一个贸易协议表示。有一个管理 WebSphere Commerce 中所有拍卖的贸易协议。

贸易协议的组成部分包括: 存储在 TRADING 表中的简要表; 存储在 PARTICIPNT 表中的参与方; 存储在 TERMCOND 表中的条款和条件; 以及作为“全局资源标识”（URI）存储在 ATTACHMENT 表中的可选附件。因为一个贸易协议可以有多个附件，所以附件通过 TRDATTACH 表与贸易协议相关。请注意，RFQ 不支持附件。

除了一般贸易协议以外，每种类型的贸易协议还在它自己的表中存储特定于该类型贸易协议的附加信息: CONTRACT 存储特定于合同的信息; RFQ 存储特定于 RFQ 的信息; ACCOUNT 存储特定于商业帐户的信息。

条款和条件

条款和条件定义贸易协议的行为和属性。

对于合同，条款和条件定义了如何对买方组织实现合同。它们定义了正依据合同销售的商品; 正在销售的商品的价格; 如何装运商品; 如何支付订单; 如何处理商品退货; 如何核准订单; 以及从何处装运订单。

一些条款和条件引用业务策略，因为商店操作的很多方面都是由业务策略定义的。条款和条件为其所引用的业务策略提供参数。向业务策略提供参数使您可以对每个合同的业务策略的行为进行修改。

业务策略

业务策略是商店或商店组所遵循的规则集。业务策略定义业务过程、行业惯例，以及商店或商店组出售物的范围和特征。它们是商店或商店组内所有允许的和支持的惯例的中心源和引用模板。

在 WebSphere Commerce 中，业务策略的实施涉及以下各项的组合：实现业务策略规则的一个或多个业务策略命令、对规则作用的商业对象的引用以及配置业务策略命令操作的一组属性。条款和条件可以为其所引用的业务策略提供参数。这使得可以根据引用业务策略的条款和条件来修改业务策略的行为。

附件

附件提供了关于贸易协议的附加信息，该信息未由该贸易协议中的其它元素所涵盖。示例是一个文件，它提供了关于 RFQ 需求和有关 RFQ 的所有一般注解的附加信息。贸易协议可具有多个附件。附件存储在 WebSphere Commerce 之外，且贸易协议存储了附件的全局资源标识（URI）。URI 的示例包含以下项:

- http://www.mycompany.com/information/document1.txt
- file:///home/joeuser/mydocs/document1
- ftp://ftp.mycompany.com/information/attachment.txt

可以为所有附件指定附件用法，该用法指示附件的用途。附件用法是附件的可选属性。

订购商品

订购商品是包含在订单中的产品。一份订单中的不同订购商品可以依据不同的合同贸易协议购买。根据商店设计，买方可以在购物流程开始时或在向其订单添加商品时，选择进行购物所依据的合同贸易协议。依据不同的合同贸易协议购买商品时，应用以下规则:

- 订单中所有商品的合同贸易协议必须至少共享一种支付方式。如果某个商品的合同不共享支付方式，则买方无法将此商品添加到订单。只有订单中所有商品都共享的支付方式才可以用于支付订单。
- 订单中的所有商品必须来自商店缺省合同或属于同一个商业帐户的合同贸易协议。

关于 WebSphere Commerce 中合同有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的合同数据模型。

在 WebSphere Commerce 中创建缺省合同有用资源

缺省合同定义商店的缺省行为。对于所有合同，可设置可供产品、价格、支付方式、装运方式和其它商店行为。

随 WebSphere Commerce 样本商店提供的商店缺省合同所包含的条款和条件指定了以下内容:

- 顾客可按主产品目录中设置的标准价格（没有折扣或加价）购买商店主产品目录中所有可供产品。
- 所有的装运费用都支付给卖方（商店）。
- 顾客可在特定天数之内退回购买的产品，而不受到惩罚性收费。
- 顾客可使用与用于原始购买的支付方式相同的支付方式接收退款。

并且，商店缺省合同的最普通版本省略了限制顾客可使用的支付和装运方式的条款和条件。省略这些条款将让买方能够使用商店支持的任何缺省支付方式对购买进行支付，且能够使用商店中任何可用的装运方式。

缺省合同的属性定义在其条款和条件中。一些条款和条件引用了业务策略。关于业务策略以及条款和条件的更多信息，请参阅联机信息。

要创建缺省合同有用资源，请执行以下操作:

1. 复查关于条款和条件、合同、缺省合同以及业务策略的联机信息。
2. 复查 wcs.bootstrap.xml 文件中定义的业务策略。关于 wcs.bootstrap.xml 文件的信息，请参阅联机信息。
3. 复查用于为样本商店创建缺省合同有用资源的文件。所有的样本商店文件都位于对应的商店归档文件中。每个样本商店都包含 businesspolicy.xml 和 contract.xml，它们包含了附加的业务策略信息和缺省合同信息。商店归档文件位于以下目录中:
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores
   - NT *drive*:\WebSphere\CommerceServer\samplestores

   注:

   a. WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   b. 要查看商店归档文件中的 businesspolicy.xml 和 contract.xml 文件，请使用 ZIP 程序将其解压缩。文件位于 data 目录中。

   c. 随 WebSphere Commerce 商务版提供的“多乐五金店”样本商店的合同有用资源文件包含了除商店缺省合同之外的合同的信息。
4. 请复查第 289 页的附录 B, 『创建数据』中的信息。
5. 通过复制商店归档文件中的一个 businesspolicy.xml 文件，或者通过创建新文件，来创建 businesspolicy.xml 文件。关于创建新文件的指导位于第 92 页的『创建业务策略 XML 文件』中。如果希望创建与所讨论不同的业务策略，请参阅与 businesspolicy.xml 对应的 DTD 文件。DTD 文件位于以下目录中:
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - 2000 *drive*:\Program Files\WebSphere\CommerceServer\xml\sar
   - NT *drive*:\WebSphere\CommerceServer\xml\sar
6. 使用装入程序软件包装入 businesspolicy.xml 文件。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。如果正在创建多文化商店，则可能要为商店支持的每种语言环境创建各自的 XML 文件。特定于语言环境的文件应当指定所有描述信息，以便比较容易进行翻译。
7. 通过复制商店归档文件中的一个 contract.xml 文件，或者通过创建新文件，来创建 contract.xml 文件。创建新文件的指导位于第 93 页的『创建缺省合同 XML 文件』

中。如果希望创建更为复杂的缺省合同，请复查 B2BTrading.dtd 文件，该文件定义了 contract.xml 文件的结构。B2BTrading.dtd 位于以下目录中:

- AIX /usr/WebSphere/CommerceServer/xml/trading
- 400 /qibm/proddata/WebCommerce/xml/trading
- Linux /opt/WebSphere/CommerceServer/xml/trading
- Solaris /opt/WebSphere/CommerceServer/xml/trading
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\xml\trading
- NT *drive*:\WebSphere\CommerceServer\xml\trading

8. 使用 ContractImportApprovedVersion 命令发布合同。关于更多信息，请参阅第 257 页的第 29 章，『发布商业帐户和合同』。联机信息中也提供了关于 ContractImportApprovedVersion 命令的信息。

WebSphere Commerce 商务版用户可使用 WebSphere 贸易加速器为特定顾客定义合同。关于为特定顾客创建合同的更多信息，请参阅联机信息。

创建业务策略 XML 文件

虽然 WebSphere Commerce 提供很多业务策略可供商店缺省合同中的条款和条件引用，但是您仍然必须定义一些业务策略。您必须定义商店缺省合同条款将要引用的所有退货收费、退货核准以及定价业务策略。这些业务策略的命令已提供并且可以在不修改的情况下使用。如果希望创建自己的业务策略，请参阅《*IBM WebSphere Commerce 程序员指南*》。

要创建商店的业务策略，您必须创建业务策略，并使一个或多个命令与此业务策略关联。要创建业务策略，请向 POLICY 表添加信息。要使命令与业务策略关联，请向 POLICYCMD 表添加信息。

要创建业务策略并使命令与此策略关联，请执行以下操作:

1. 通过向 POLICY 表添加信息，在您的业务策略 XML 文件中创建业务策略。使用以下示例作为指南:

```
<policy
policy_id="@policy_id_10"
policyname="MasterCatalogPriceList"
policytype_id="Price"
storeent_id="@storeent_id_1"
properties="name=InFashion&amp;member_id=&MEMBER_ID"
/>
```

其中

- policy_id 是业务策略的唯一的数字标识。
- policyname 是此业务策略的唯一名称。
- policytype_id 是正在定义的策略的类型。有效的 policytype_id 是:
  - InvoiceFormat
  - Payment
  - Price
  - ProductSet

- ReturnApproval
- ReturnCharge
- ReturnPayment
- ShippingCharge
- ShippingPayment

• storeent_id 是商店或商店组。

• properties 是“名称值”对的列表，将该列表发送到业务策略命令。

2. 通过向 POLICYCMD 表添加信息，使命令与业务策略 XML 文件中的业务策略关联。使用以下示例作为指南:

```
<policycmd
policy_id="@policy_id_10"
businesscmdclass=
  "com.ibm.com.commerce.price.commands.CalculateContractPricesCmdImpl"
/>
```

其中

• policy_id 是命令与之关联的业务策略的数字标识。

• businesscmdclass 是实现业务策略的 Java 类的名称。

businesscmdclass 属性中的断行仅出于显示目的。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

创建缺省合同 XML 文件

要创建缺省合同，您必须定义合同、合同所有者、合同描述、合同参与方以及合同的条款和条件。合同信息存储在四个表中: CONTRACT、PARTICIPNT、TRADING 和 TERMCOND。

使用 STOREDEF 数据库表将缺省合同与商店关联。对于 WebSphere Commerce 商务版用户，使用 STORECNTR 数据库表将除缺省合同之外的合同与商店关联。

要创建缺省合同，请执行以下操作:

1. 在您的 XML 文件中定义缺省合同。缺省合同如下定义在 XML 文件的开头:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Trading SYSTEM "B2BTrading.dtd">
<Trading>
<Contract state="Active" origin="Manual"
  name="&STORE_IDENTIFIER; Default Contract" majorVersionNumber="1"
  minorVersionNumber="0" contractUsage="Default">
```

请注意，Contract 元素中的断行仅出于显示目的。

2. 定义合同所有者。使用以下示例作为指南:

```
<ContractOwner>
          <Member>
    <Organization distinguishName="&MEMBER_IDENTIFIER;" />
          </Member>
</ContractOwner>
```

其中 distinguishName 是 LDAP 专有名称格式的拥有合同的用户的名称。例如，uid=erickoeck,ou=People,dc=ibm,dc=com。

3. 在您的合同 XML 文件中定义合同描述。使用以下示例作为指南:

```
<ContractDescription title="This is a store default contract." languageId="-1">
</ContractDescription>
```

其中

- title 是合同的文本描述。
- languageId 是标题使用的语言。对于 languageId 预定义了以下值:
  - -1（英语 — 美国）
  - -2（法语）
  - -3（德语）
  - -4（意大利语）
  - -5（西班牙语）
  - -6（巴西葡萄牙语）
  - -7（简体中文）
  - -8（繁体中文）
  - -9（韩国语）
  - -10（日语）

可以通过更新商店的语言有用资源，对 languageId 定义附加值。关于语言有用资源的更多信息，请参阅第 107 页的第 14 章，『语言有用资源』。

4. 在您的合同 XML 文件中定义合同参与方。使用以下示例作为指南:

```
<Participant role="Buyer">
</Participant>
<Participant role="Seller">
          <Member>
    <Organization distinguishName="&MEMBER_IDENTIFIER;"/>
          </Member>
</Participant>
```

其中 distinguishName 是 LDAP 专有名称格式的作为此合同卖方的用户的名称。例如，uid=erickoeck,ou=People,dc=ibm,dc=com。在很多情况下，这将与合同所有者相同。

注: 没有成员指定为买方参与者角色，因为合同对具有买方角色的所有用户都可用。

5. 在您的合同 XML 文件中定义条款和条件。对于不同类型的条款和条件，XML 元素和属性是不同的。使用 B2BTrading.dtd 文件学习用于每种条款类型的 XML 元素和属性。定义条款和条件时，通常使用以下属性:

policyName

条款和条件引用的业务策略的名称。此名称存储在 POLICY.POLICYNAME 中。

policyType

条款和条件引用的业务策略的类型。有效值为:

- Price
- ProductSet

- InvoiceFormat
- Payment
- ReturnApproval
- ReturnCharge
- ReturnPayment
- ShippingCharge
- ShippingMode

storeIdentity

条款和条件的商店或商店组。

distinguishName

拥有商店或商店组的用户的名称。名称必须是 LDAP 专有名称格式。例如, `uid=wcsadmin,o=Root Organization`。

以下样本条款和条件跟在对其所定义内容的描述之后:

- 所有买方都可以按照主产品目录中设置的价格购买商店主产品目录中的所有商品:

```
<TermCondition>
  <PriceTC>
    <PriceTCMasterCatalogWithOptionalAdjustment>
    </PriceTCMasterCatalogWithOptionalAdjustment>
  </PriceTC>
</TermCondition>
```

- 买方向卖方支付所有的装运费用:

```
<TermCondition>
  <ShippingTC>
    <ShippingTCShippingCharge>
      <PolicyReference policyName="StandardShippingChargeBySeller"
        policyType="ShippingCharge" storeIdentity="&STORE_IDENTIFIER;">
          <Member>
          <User distinguishName="&MEMBER_IDENTIFIER;">
          </Member>
        </PolicyReference>
    </ShippingTCShippingCharge>
  </ShippingTC>
</TermCondition>
```

`PolicyReference` 元素中的断行仅出于显示目的。

- 买方可以退回产品而不支付任何退货收费。必须在 `ApprovalByDays` 业务策略中定义的天数内退回产品:

```
<TermCondition>
  <ReturnTC>
    <ReturnTCReturnCharge>
      <ReturnChargePolicyReference>
        <PolicyReference policyName="NoCharges"
         policyType="ReturnCharge"
         storeIdentity="&STORE_IDENTIFIER;">
          <Member>
            <Organization distinguishName="&MEMBER_IDENTIFIER;">
          </Member>
        </PolicyReference>
      </ReturnChargePolicyReference>
      <ReturnApprovalPolicyReference>
        <PolicyReference policyName="ApprovalByDays"
         policyType="ReturnApproval"
```

```
          storeIdentity="&STORE_IDENTIFIER;">
           <Member>
             <Organization distinguishName="&MEMBER_IDENTIFIER;">
           </Member>
         </PolicyReference>
       </ReturnApprovalPolicyReference>
     </ReturnTCReturnCharge>
   </ReturnTC>
</TermCondition>
```

PolicyReference 元素中的断行仅出于显示目的。

WebSphere Commerce 商务版用户请注意:
从商店缺省合同中省略这些条款和条件指示了在缺省情况下商店不接受退货。但是其它合同可能通过定义退货条款和条件，允许买方退货。

WebSphere Commerce 专业版用户请注意:
从商店缺省合同中省略这些条款和条件指示了商店不接受退货。

- 使用与完成订单时买方所使用的同一支付方式来支付退款:

```
<TermCondition>
  <ReturnTC>
    <ReturnTCRefundPaymentMethod>
      <PolicyReference policyName="UseOriginalPayment"
       policyType="ReturnPayment" storeIdentity="&STORE_IDENTIFIER;">
          <Member>
          <User distinguishName="&MEMBER_IDENTIFIER;">
          </Member>
        </PolicyReference>
    </ReturnTCRefundPaymentMethod>
  </ReturnTC>
</TermCondition>
```

请注意，PolicyReference 元素中的断行仅出于显示目的。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 11 章 实现有用资源

商店将实现中心同时作为库存仓库以及装运和接收中心来使用。一个商店可能具有一个或多个实现中心与之关联。实现中心为商店管理产品的库存和装运。实现包含了提货、装货和装运。提货是指在实现中心的一个或多个发货批次中产品的选择，装货是指将这些产品装入装运容器，而装运是指将它们发送给顾客。

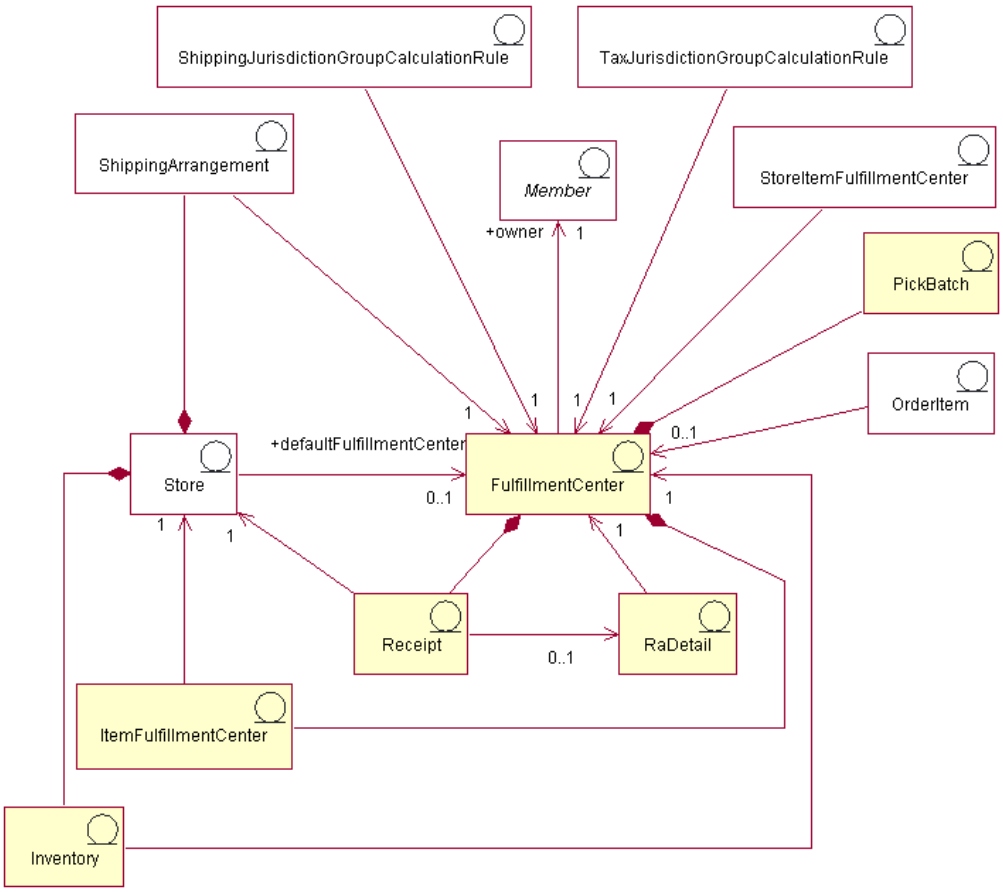
使用“产品”向导和“产品”笔记本配置产品以供实现。产品配置提供了选项以跟踪库存、允许延迟交货、强制延迟交货、分别发货以及指定产品不可以退回。

通常，同一时间在实现中心中有很多人员在工作，每个人都有不同的任务要执行。WebSphere 贸易加速器将最常见的任务分成角色，然后将这些角色指定给用户。在 WebSphere 贸易加速器中，如果对您指定了一个或多个与实现相关的角色，则在登录时必须选择一个实现中心。

注: 关于实现、实现中心和角色的更多信息，请参阅 WebSphere Commerce 联机帮助。

了解 WebSphere Commerce 中的实现有用资源

为了了解实现有用资源，有必要了解实现和商店之间的关系。这可以使用信息模型来解释。以下部分描述了库存所具有的与商店和其它有用资源之间的关系。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

实现中心

在前图中，*实现中心*处于实现过程的中心位置。实现中心具有一个所有者，他在 MEMBER 表中定义。每个商店与一个实现中心关联，并具有一个缺省的实现中心。一个实现中心可以具有多个商店与之关联。如图所示，在商店和实现中心之间存在若干交互。关于商店有用资源的更多信息，请参阅第 39 页的『了解 WebSphere Commerce 中的商店有用资源』。

接收

实现中心按每天、每周或每月来接收商品的库存。当接收到商品的库存时，在 RECEIPT 表中就创建了一个*接收*，它记录了有关接收数量的信息以及拥有此库存的商店。处理订单时，将更新 RECEIPT 表以反映当前可用的库存级别。关于创建接收的信息，请参阅第 156 页的『在 WebSphere Commerce 中创建库存有用资源』。

RaDetail

RaDetail 是关于预期库存记录中的商品的详细信息。此信息可用于估计在实现中心预计何时可以接收到库存，以及用于向顾客提供延迟交货商品的预期装运日期。

库存

商店拥有与实现中心关联的*库存*。库存包含可以在实现中心中实际计算的任何东西。库存与一个商店以及一个实现中心关联。还记录有关商店在实现中心中拥有的库存的信息，例如保留数量、延迟交货的数额，以及分配的延迟交货的数额。此信息存储在 ITEMFFMCTR 表中。关于库存和库存有用资源的更多信息，请参阅第 153 页的第 21 章，『库存有用资源』。

装运安排

商店和实现中心之间的最后一个关系涉及*装运安排*。装运安排指示实现中心可以使用一个装运方式以商店的名义装运产品。每个商店与实现中心之间有一个装运安排，反之亦然。装运安排在 SHPARRANGE 表中设置。关于创建装运安排的信息，请参阅第 130 页的『创建装运实现有用资源』。

其它实现有用资源

其它一些与实现中心的关系并不与商店直接有关。一个提货批量与一个实现中心关联。提货批量将各订单发货批次分组在一起以在实现中心作为一个单位进行处理，并且创建提货单和装货单。一旦提取并包装了商品，就可以装运订单发货批次，并确认装运。提货批量信息存储在 PICKBATCH 表中。一个订购商品也与一个实现中心关联。一个商品是产品的一个特定实例，它由属性定义。关于订单中每个商品的信息都存储在 ORDERITEM 表中。关于订单有用资源的更多信息，请参阅第 157 页的第 22 章，『订单有用资源』。

如同其它实体一样，实现中心具有一系列规则来管理一些它自身的活动。每个实现中心有税款费用和装运费用的规则。这些分别在 TAXJCRULE 和 SHPJCRULE 表中定义。关于税款和装运有用资源的更多信息，请参阅第 119 页的第 18 章，『装运有用资源』和第 135 页的『了解 WebSphere Commerce 中的税款有用资源』。

关于 WebSphere Commerce Server 中实现有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的实现数据模型。

在 WebSphere Commerce 中创建实现有用资源

在您的商店可以向顾客装运商品之前，您必须定义一个或多个供应这些商品的实现中心。以 XML 文件格式创建此信息，可使用装入程序软件包将 XML 文件装入数据库。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

要使用 XML 文件为您的商店创建实现有用资源，请执行以下操作:

1. 复查用于为样本商店创建实现有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包含 `fulfillment.xml` 文件，该文件包含实现信息。要查看商店归档文件中的 `fulfillment.xml` 文件，请使用 ZIP 程序将其解压缩。`fulfillment.xml` 文件位于 data 目录中。
2. 请复查第 289 页的附录 B，『创建数据』中的信息。
3. 通过复制样本商店归档文件中的一个 `fulfillment.xml` 文件或通过创建一个新文件来创建 `fulfillment.xml` 文件。关于更多信息，请参阅相应于 `fulfillment.xml` 的 DTD 文件。DTD 文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\xml\sar`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`

4. 定义商店支持的一个或多个实现中心:
   a. 使用以下示例作为指南，在 XML 文件中定义 FFMCENTER 表中的实现中心:

```
<ffmcenter
   ffmcenter_id="@ffmcenter_id_1"
 member_id="&MEMBER_ID"
   name="ToolTech Home"
  defaultshipoffset="0"
  markfordelete="0"
   />
```

   其中

   - ffmcenter_id 是生成的唯一键
   - member_id 是实现中心的所有者
   - name 是字符串，它与所有者一起唯一标识此实现中心。
   - defaultshipoffset 是商品从该实现中心处装运所花费时间的估计秒数。可以在 STORITMFFC 表中重设此值。
   - markfordelete 指示是否应该删除实现中心，如下: 0 = 不删除。1 = 如果不再使用，就删除。关于更多详细信息，请参阅 WebSphere Commerce 联机帮助中有关数据库清理实用程序的信息。

   b. 使用以下示例作为指南，在 XML 文件中描述 FFMCENTDS 表中的实现中心。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中。

```
<ffmcentds
   ffmcenter_id="@ffmcenter_id_1"
   description="The fulfillment center that supplies products to ToolTech."
   language_id="&en_US"
  displayname="ToolTech Fulfillment"
staddress_id="@staddress_id_en_US_1"
   />
```

   其中

   - ffmcenter_id 是生成的唯一键
   - description 是适合于显示给顾客的对实现中心的描述。
   - language_id 是显示此信息的语言。
   - displayname 是适合于显示给顾客的实现中心的名称。
   - staddress_id 实现中心的物理位置。

   c. 对商店支持的所有实现中心重复步骤 a 和 b。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

创建商店实现有用资源

在定义了将为商店供应商品的一个或多个实现中心之后，必须将实现中心与每个产品关联。即，必须标识哪个实现中心将供应哪些产品。要创建此关系，请向 INVENTORY 表添加信息。以 XML 文件格式创建此信息，可使用装入程序软件包将 XML 文件装入数据库。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

注:

1. 在可以将商店与实现中心关联之前，您必须先创建商店有用资源。关于创建商店有用资源的更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。在可以创建商店实现有用资源之前，您还必须创建产品目录有用资源。关于更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。
2. 仅在实施非 ATP 实现时才创建商店实现有用资源。包含 ATP 功能的商店不使用 INVENTORY 表。

要使用 XML 文件创建商店实现关系，请执行以下操作:

1. 复查用于为样本商店创建商店实现有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包含 `storefulfill.xml` 文件，该文件包含商店实现信息。要查看商店归档文件中的 `storefulfill.xml` 文件，请使用 ZIP 程序将其解压缩。`storefulfill.xml` 文件位于 data 目录中。
2. 请复查第 289 页的附录 B, 『创建数据』中的信息。
3. 通过复制样本商店归档文件中的一个 `storefulfill.xml` 文件或通过创建一个新文件来创建 `storefulfill.xml` 文件。关于更多信息，请参阅相应于 `storefulfill.xml` 的 DTD 文件。DTD 文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\xml\sar`
   - 2000 `drive:\Program Files\WebSphere\CommercServer\xml\sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`
4. 使用以下示例作为指南，通过向 INVENTORY 表添加信息，在 XML 文件中创建“商店 – 实现中心”关系。

```
<inventory
catentry_id="@catentry_id_1470"
quantity="100"
ffmcenter_id="@ffmcenter_id_1"
```

```
store_id="@storeent_id_1"
quantitymeasure="C62"
inventoryflags="0"
/>
```

其中

- `catentry_id` 是此实现中心将供应的产品目录条目。
- `quantity` 是从该实现中心中可以得到的数额（以 QUANTITYMEASURE 指示的单位表示）。
- `ffmcenter_id` 是供应此库存的实现中心。
- `store_id` 是为其供应此库存的商店。
- `quantitymeasure` 是 QUANTITY 的计量单位。
- `inventoryflags` 是指示如何使用 QUANTITY 的位标志:
  - 1 = noUpdate。缺省 UpdateInventory 任务命令不更新 QUANTITY。
  - 2 = noCheck。缺省 CheckInventory 和 UpdateInventory 任务命令不检查 QUANTITY。

5. 对商店中的每个产品目录条目重复步骤 3。

关于 **@** 和 **&** 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 12 章 竞销有用资源

竞销有助于您组织进行市场营销。竞销通常是市场部经理或销售部经理创建的。它们经常与某一组目标相关联。例如，“回到学校”竞销就可能有在竞销期间增加销售儿童服装的目标。

了解 WebSphere Commerce 中的竞销

在 WebSphere Commerce 中，竞销包含任意数量的竞销活动，这些活动将定义条件。当定义的条件评估为真时，竞销活动将为顾客生成目标内容。结果是竞销是组织活动的高级市场营销元素。

竞销活动与包含活动集合的竞销相关联。此关系的一个例子是：如果某一办公用品商店开展“回到学校”竞销，则活动将是负责较低级别的活动，例如向以学生身份注册或列出职业的顾客以折扣价宣传钢笔或推荐格记录纸。

竞销活动能够显示三种类型的动态内容:

- 推荐销售活动
- 基于协作过滤的推荐
- 推广广告

推荐销售内容是设计用来提供基于规则的类别和产品推荐的，该推荐针对特定的顾客观众，以顾客简要表和其它顾客行为为基础。显示此类内容的活动意在用于创建交叉销售和优选销售机会。

基于协作过滤的推荐也意在创建产品推荐，但是它们使用不同的推荐算法，该算法针对基于顾客整体行为的商品，而非预定义的规则。

推广广告是设计用来针对特定顾客受众提供广告内容的，它以与推荐销售的相同条件为基础，但是推广广告意在用来让顾客更加了解网上商店的活动，突出显示特价商品并提高品牌意识。

活动可以并入到站点上的任何页面上。站点设计好后，称为电子广告位的特定占位符将置于站点上。当显示给顾客时，这些占位符由特定的目标内容替换。通过调度活动指定目标位置可以在理想位置上显示电子广告位。关于向商店添加电子广告位的更多信息，请参阅第 279 页的第 32 章，『向商店添加电子广告位』。

竞销活动包含确定显示时间和显示对象的条件。此条件在活动创建时定义，而且在活动的有效期间可以更改条件以调整活动的可视性和显示的内容。

竞销活动生成关于其使用的统计信息。商家、市场部经理和销售部经理可以使用 WebSphere 贸易加速器查看这些统计信息。统计信息说明了被点击的每一电子广告位的活动点击率。这些统计信息提供了关于活动效果的反馈，以及在不同显示位置的相对成功率。

在 WebSphere Commerce 中创建竞销有用资源

竞销和竞销活动通常是由市场部经理或销售部经理使用 WebSphere 贸易加速器中的“竞销和竞销活动”向导创建的。关于更多信息，请参阅 WebSphere Commerce 联机帮助。

关于向商店添加电子广告位的更多信息，请参阅第 279 页的第 32 章，『向商店添加电子广告位』。

第 13 章 支付有用资源

WebSphere Commerce 支持 IBM Payment Manager。要为您的商店创建支付有用资源，请指定商店是否使用 Payment Manager；如果使用了 Payment Manager，请再指定商店接受什么类型的支付卡匣和品牌。

通过执行以下操作指定此信息:

- 以 XML 文件（paymentinfo.xml）格式创建支付数据，该数据在使用“商店服务”进行商店发布时装入。这为正在发布的商店使用指定的商家和品牌类型配置 Payment Manager。关于更多信息，请参阅『使用 XML 文件创建支付有用资源』。

  注: paymentinfo.xml 不填充 WebSphere Commerce Server 数据库中的表。它将配置 Payment Manager。仅当您使用脱机信用卡作为支付方式时，paymentinfo.xml 才适用。要配置其它支付方法，请参阅下一点。

- 使用管理控制台或 Payment Manager 用户界面为您的商店完成 Payment Manager 的设置。如果您使用管理控制台，菜单项将出现在 Payment Manager 菜单上。如果您使用 Payment Manager 用户界面，菜单项将出现在导航框架的“管理”下。关于更多信息，请参阅 WebSphere Commerce 联机帮助主题“*为您的商店设置 Payment Manager*”。

使用 XML 文件创建支付有用资源

要使用 XML 文件为您的商店创建支付有用资源，请执行以下操作:

1. 复查用于为样本商店创建支付有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

     注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包括一个 paymentinfo.xml 文件，该文件包含支付信息。要查看商店归档文件中的 paymentinfo.xml 文件，请使用 ZIP 程序将其解压缩。paymentinfo.xml 文件位于 data 目录中。

2. 通过复制样本商店归档文件中的一个 paymentinfo.xml 文件或通过创建一个新文件来创建 paymentinfo.xml 文件。关于更多信息，请参阅相应于 paymentinfo.xml 的 DTD 文件。DTD 文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\xml\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

3. 启用或禁用 Payment Manager。
   a. 使用以下示例作为指南，在您的 XML 文件中启用或禁用 Payment Manager，指定商店接受的支付卡匣、货币和品牌的类型。

```
<paymentinfo>
<PaymentManager enable="yes"/>
<Cassette type="OfflineCard">
<Account currency="USD">
<Brand type="MasterCard"/>
<Brand type="VISA"/>
<Brand type="American Express"/>
<Account/>
<Account currency="EUR">
<Brand type="MasterCard"/>
<Brand type="VISA"/>
<Brand type="American Express"/>
</Account>
</Cassette>
</paymentinfo>
```

   其中:

   - enable 表示 Payment Manager 是启用的还是禁用的。
   - Cassette type 是受支持的卡匣的类型。
   - Account currency 是商店支持的货币。如果您使用 OfflineCard Cassette 类型，则帐户货币是必需的。货币必须以符合 ISO 4217 标准的三字母代码来标识。例如，“USD” 用以表示美元。
   - Brand type 是帐户和货币支持的信用卡类型。

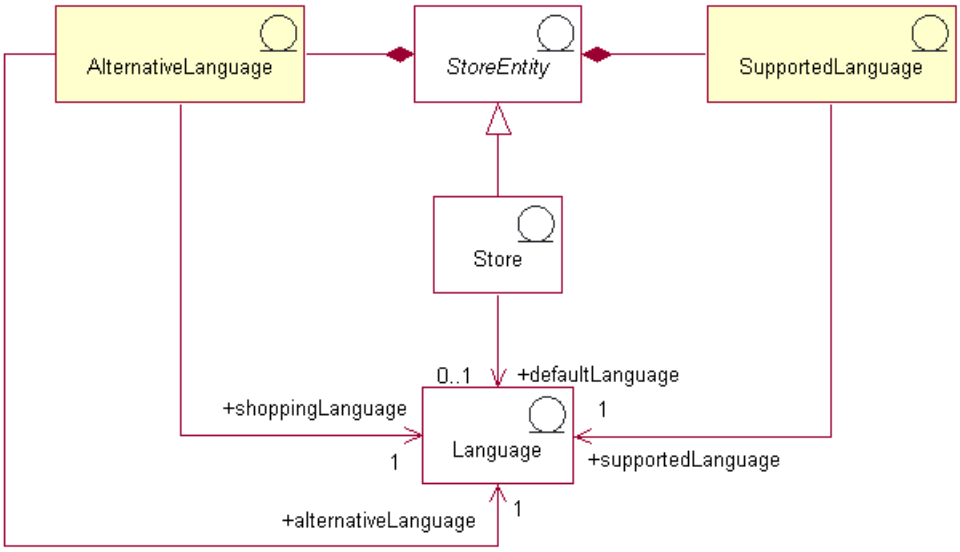
第 14 章 语言有用资源

在 WebSphere Commerce 中，您的站点可以定义许多可在 WebSphere Commerce 中使用的语言。LANGUAGE 表定义了十种支持的语言，其中包括德语、繁体和简体中文、日语、韩国语、意大利语、法语、西班牙语、巴西葡萄牙语和英语。站点可以定义附加的语言，或者现有语言的方言，以便使用适合的方法将信息显示给来自不同文化或地域的顾客。

了解 WebSphere Commerce 中的语言有用资源

为了了解语言有用资源，有必要了解语言和商店之间的关系。这可以使用下面的信息模型来解释。以下部分描述了语言与商店和其它有用资源之间的关系和关联。

下图描绘了语言有用资源信息模型。



在 WebSphere Commerce 中有四个语言分类。它们是缺省语言、支持的语言、备用语言和购物语言。这些分类中的每一个在商店中执行不同的角色。所有的语言都存储在 LANGUAGE 表中。

缺省语言

*缺省语言*与每个商店关联。它是商店选择用作其主要语言的语言，而且缺省语言将是对没有明确选择购物语言的顾客而显示的语言。商店隐式地支持商店的缺省语言；即，商店必须总是能够以缺省语言显示信息，或者以它的备用语言之一显示信息（如果在 LANGPAIR 表中定义了备用语言）。如果不能以商店支持的语言或备用语言之一显示信息，则信息将以缺省语言显示。

支持的语言

STORELANG 表指示每一商店支持的语言。商店必须能够以它的*支持语言*或备用语言之一（如果在 LANGPAIR 表中定义了备用语言）显示信息。商店也支持其商店组所支持的所有语言。

备用语言

如果信息不能以商店的支持语言之一显示，则商店尝试使用*备用语言*（如果有的话）显示信息。商店可以指定尝试其备用语言的顺序。商店的备用语言包括其商店组的备用语言。当某些信息只能以一种语言显示，但又应当向以不同的、相关的语言购物的顾客提供时，备用语言十分有用。例如，并非所有信息都翻译成所有支持的语言时，或者例如支持同一语言的两种十分相近的方言（有时信息完全相同）时，备用语言可能十分有用。

关于 WebSphere Commerce Server 中语言有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的语言对象和数据模型。

在 WebSphere Commerce 中创建语言有用资源

您可以用以下一种方式来定义商店所支持的语言:

- 使用“商店服务”中的工具
- 在 XML 文件中（此文件将由装入程序软件包或“商店服务”中的发布工具装入）

注: “商店服务”工具以商店归档文件的格式处理预填充的 XML 文件。

关于使用“商店服务”定义商店支持语言的更多信息，请参阅 WebSphere Commerce 联机帮助。关于在 XML 文件中定义商店支持语言的更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。

第 15 章 货币有用资源

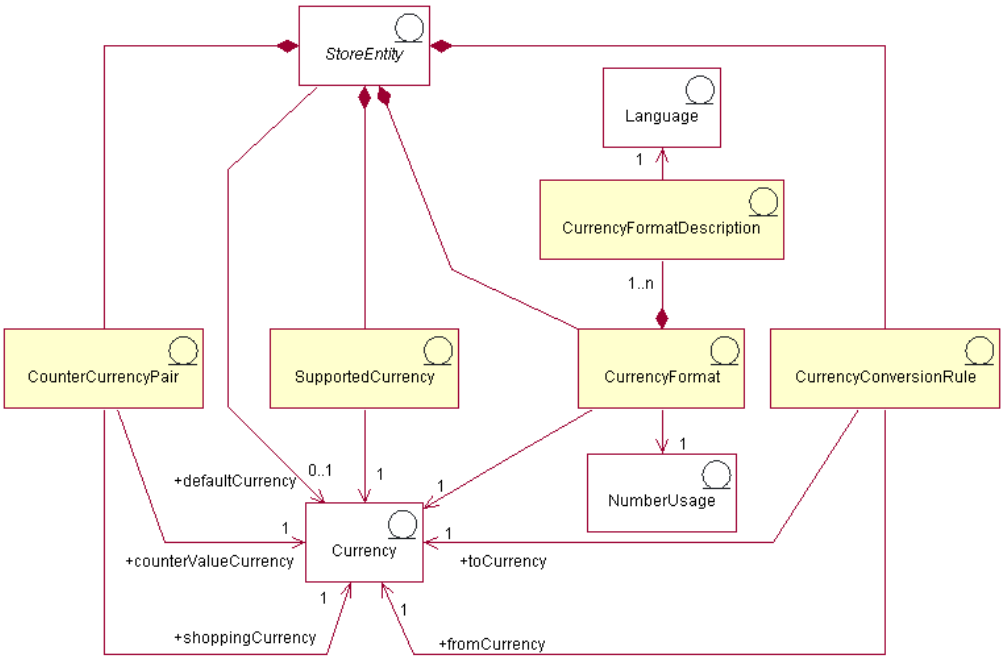
您可以在您的站点中以一种货币显示价格，或者您可以遵循为欧元提供的指导来使用多种货币。对于具有多个商店的站点，您可以为商店使用不同的货币，或者您可以对商店组指定货币。根据您正在创建的站点的特点，您可以指定希望使用的货币以及如何显示它们。

在 WebSphere Commerce 中，你可以允许顾客选择购物货币。购物货币是顾客在特定商店中为产品支付时所用的货币。商店页面上所有的货币金额都以此货币显示。当顾客更改了他们的购物货币时，将自动转换并重新计算已添加到购物车的商品的价格以及订单的总额，并以新的购物货币显示它们。

顾客可以以多种货币（其中包含欧元）进行购物。欧元于 1999 年 1 月 1 日成为欧盟的合法货币，且现已在金融市场上使用。欧元与所有参与国的货币之间的兑换率是固定的。欧盟未来单一货币的纸币™和硬币将于 2002 年 1 月 1 日开始使用。六个月之后，现有的国家标准货币将从流通中永远撤销。在 1999 年到 2001 年的转换阶段，商家必须同时接受国家标准货币和欧元。

了解 WebSphere Commerce 中的货币有用资源

下图说明了 WebSphere Commerce Server 中的货币结构:



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

在上图中，货币处于信息模型的中央。每个商店或商店组都有缺省货币。

货币格式

一个商店实体可以具有许多*货币格式化*规则。如果商店没有特定货币的格式化规则，它将使用其商店组的格式化规则。货币格式在 CURFORMAT 表中设置。

数值用法

每个已格式化的货币规则都与一个*数值用法*关联。根据它们关联用法的不同，数值（例如数量或货币金额）可以用不同方式舍入和格式化。商店可以根据使用数值的方式来为显示的数值指定不同的舍入和格式化规则，例如商店可以通过指定单价用法将单价舍入到四位小数，而通过指定缺省用法将其它货币金额舍入到两位小数。数值用法存储在 NUMBRUSG 表中。

货币格式描述

货币格式规则可以具有许多*货币格式描述*。货币格式描述描述了如何以特定货币和特定语言格式化（为了显示目的）货币金额。每个描述与 LANGUAGE 表中的一个语言相关联。关于语言有用资源的更多信息，请参阅第 107 页的第 14 章，『语言有用资源』。货币格式描述存储在 CURFMTDESC 表中。

支持的货币

一个商店实体可以具有许多*支持的货币*。支持的货币是被承认的可用于支付的货币。

货币转换规则

所有货币都具有规则来管理它们与其它货币之间的相互转换。每个*货币转换规则*可以用于将价格（以特定的货币存储在数据库中）转换为顾客以支持的购物货币支付的金额。

对等货币

*对等货币*是与支持的货币一起显示的货币金额。不可以用它们进行购买，它们只是用于提供信息。如果顾客决定以欧元购物，则他们可以在商店中显示“欧洲货币联盟”货币金额以及其它货币金额。以购物货币表示的金额将转换为该购物货币的所有对等价格币种。对等货币与一种支持的货币结成配对，例如荷兰盾和欧元。对等货币对存储在 CURCVLIST 表中。

关于 WebSphere Commerce Server 中货币有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的货币数据模型。

在 WebSphere Commerce 中创建货币有用资源

WebSphere Commerce 中的“商店服务”工具使您能够将支持的货币添加到您的商店中，并能为您的商店选择缺省的货币。关于您可以使用“商店服务”工具编辑哪些有用资源的更多信息，请参阅 WebSphere Commerce 联机帮助主题*“更改商店数据库有用资源”*。

注: “商店服务”工具以商店归档文件的格式处理预填充的 XML 文件。

您还可以使用 XML 文件将支持的货币和缺省货币添加到您的商店中，XML 文件可以使用装入程序软件包装入到数据库中。此方法还允许您创建其它类型的货币有用资源，包括定义货币兑换率和对等价格货币。

关于在现有商店归档文件中编辑货币有用资源的信息，请参阅 WebSphere Commerce 联机帮助。关于以 XML 文件格式创建新的货币有用资源的信息，请参阅『使用 XML 文件创建货币有用资源』。

使用 XML 文件创建货币有用资源

以 XML 文件（这些文件可以使用装入程序软件包装入到数据库中）格式为您的商店创建货币有用资源。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。

要使用 XML 文件为您的商店创建货币有用资源，请执行以下操作:

1. 复查用于为样本商店创建货币有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包含一个 `currency.xml` 文件，它包含货币信息。要查看商店归档中的 `currency.xml` 文件，请使用 ZIP 程序将其解压缩。`currency.xml` 文件位于 data 目录中。
2. 请复查第 289 页的附录 B，『创建数据』中的信息。
3. 通过复制样本商店归档中的一个 `currency.xml` 文件，或通过创建一个新的文件来创建 `currency.xml` 文件。关于更多信息，请参阅相应于 `currency.xml` 的 DTD 文件。DTD 文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\xml\sar`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer \xml\sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommercServer/xml/sar`
   - Linux `/opt/WebSphere/CommercServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`
4. 定义您的商店支持的货币。

a. 使用以下示例作为指南，在您 XML 文件中为 CURLIST 表定义商店支持的货币:

```
<curlist currstr="USD" storeent_id="@storeent_id_1" />
```

其中:

- currstr 是三个字母的 ISO 4217 货币代码，表示支持的货币。此代码必须出现在 SETCURR 表中的 SETCCURR 列。一个商店必须能够接受以所有其支持的货币进行的支付。
- storeent_id 是商店实体。

b. 为商店支持的每种货币重复此过程。

商店的缺省货币在 STOREENT 表中定义。关于更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。

5. （可选的）商店显示的货币价格，这取决于您如何设置价格。您可以为商店中使用的每种货币定义价格，或者只为缺省货币定义价格。关于设置价格的更多信息，请参阅第 82 页的『在 WebSphere Commerce 中创建定价有用资源』。

   如果您在设置价格时只为缺省货币定义了价格，但又想在商店中以其它支持的货币显示价格，则您必须将兑换率添加到商店中。以此转换率从缺省货币转换为支持的货币。

   a. 确定将进行转换的货币，例如美元（USD），以及将转换成的一种或多种货币，例如日元（JPY）。要确定每种货币的 ISO 货币代码，请参阅国际货币的 ISO 4217 代码。

   b. 使用以下示例作为指南，将转换信息添加到 CURCONVERT 表:

```
<curconvert
storeent_id="@storeent_id_1"
fromcurr="USD"
tocurr="JPY"
factor="105.10"
multiplyordivide="M"
bidirectional="Y"
updatable="Y"
curconvert_id="@curconvert_id_1" />
```

   其中:

   - storeent_id 是商店实体。
   - fromcurr 是您将进行转换的货币。FROMCURR 货币的金额通常是用于确定与待售产品关联的价格、折扣、装运费用或类似金额的规则或者其它信息的组成部分。
   - tocurr 是您将转换成的货币。TOCURR 通常是顾客将支付的货币。此货币的金额通常是订购商品的组成部分，例如单价、装运费用和税款金额。
   - factor 是转换因子。
   - multiplyordivide 如下: 从 FROMCURR 转换为 TOCURR:
     - M = 乘以 FACTOR
     - D = 除以 FACTOR

     对于双向规则，允许使用逆操作进行从 TOCURR 到 FROMCURR 的转换。
   - bidirectional 指示规则是否为双向的。

– Y = 双向的
– N = 非双向的

• updatable 是个标志，用户界面将其用于管理货币转换规则。有效值是:
– N = 转换率是不能取消的 — 永远不能更改
– Y = 转换率是可以更改的
• curconvert_id 是生成的唯一键。

c. 对您想用于显示价格的所有货币重复步骤 a 和 b。

即使您已经在您的定价信息中为所有支持的货币定义了价格，您也可能想为商店中支持的货币定义货币兑换率。

6. （可选的）如果您想同时以购物货币和对等货币显示价格（例如，同时以荷兰盾和欧元显示价格），则您必须将信息添加到 CURCVLIST 表中。

a. 使用以下示例作为指南，将转换信息添加到 CURCVLIST 表:

```
<curcvlist
storeent_id="@storeent_id_1"
currstr="NLG"
countervaluecurr="EUR"
displayseq="1" />
```

其中:

• storeent_id 是商店实体。
• currstr 是符合 ISO 4217 标准的三个字母的货币代码，表示货币。此代码必须出现在 SETCURR 表中的 SETCCURR 列。FROMCURR 货币的金额通常是用于确定与待售产品关联的价格、折扣、装运费用或类似金额的规则或者其它信息的组成部分。
• countervaluecurr 是三个字母的 ISO 4217 货币代码，表示对等价格货币。此代码必须出现在 SETCURR 表中的 SETCCURR 列。
• displayseq 是指示对等价格货币展示顺序的数字。对等价格货币根据 CURCVLIST 表中的 DISPLAYSEQ 列中所指定的对等价格显示顺序以升序显示。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

其它货币任务

关于一般货币和其它货币任务的更多信息，包括:

• 添加 WebSphere Commerce 当前不支持的新货币
• 更改现有的货币格式
• 为新的货币或特定商店添加新的格式化规则

请参阅 WebSphere Commerce 联机帮助。

第 16 章 计量单位有用资源

产品可以按照各种数量单位来出售，库存也可以按照各种数量单位来跟踪，例如，千克、英寸、公升等等。这些单位中，产品可以按最小数量来订购，也可以按特定数量的倍数来订购。

控制器命令使用 UOM（计量单位）来指定数量单位。如果未指定 UOM 参数，则顾客指定的数量将乘以 CATENTSHIP 数据库表中的产品目录条目的额定数量。结果称为请求数量。

请求数量将舍入到此产品目录条目的下一个最高的数量倍数。例如，如果倍数为 2 千克，而请求数量为 4.1 千克，则舍入的结果将是 6 千克。当检查库存时将使用舍入的数量，它具有自己的数量单位。如果库存的数量单位不同于产品目录条目的数量单位，则在这两个单位之间必须有一个转换。

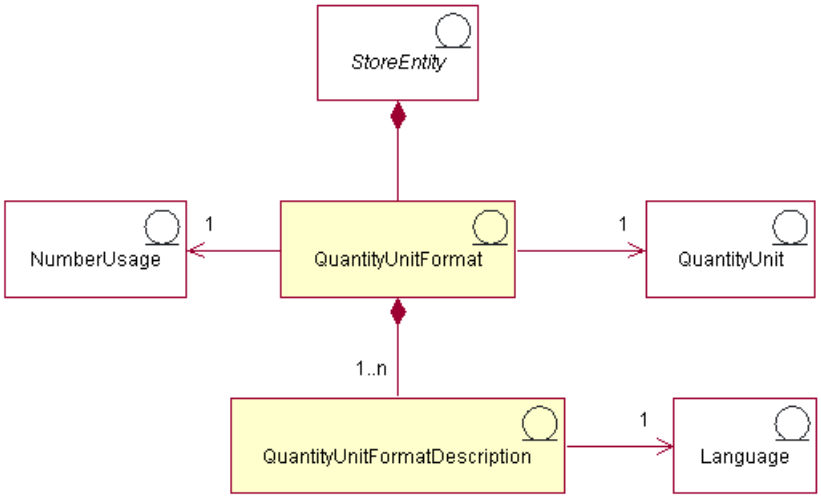
当启用可承诺（ATP）库存时（请参阅 STORE 表的 ALLOCATIONGOODFOR 列），库存的数量单位在 BASEITEM 表的 QUANTITYMEASURE 列中定义。否则，它在 INVENTORY 表的 QUANTITYMEASURE 列中定义。

舍入数量除以产品目录条目的额定数量所得的值称为规格化数量。规格化数量存储在订购商品或兴趣商品中，这取决于正在运行的命令。例如，如果舍入数量为 6 千克，额定数量为 2 千克，则规格化数量为 3。

当找到产品目录条目的报价时，请求数量可以影响哪个报价给出最佳价格，所以请求数量确定了将使用的报价。例如，如果舍入数量为 6 千克，而有两个报价，一个指定 2 千克额定数量和 10 千克最小数量的情况下价格为 ￥32，另一个指定 2 千克额定数量和 2 千克最小数量的情况下价格为 ￥36，则仅使用第二个报价。

了解 WebSphere Commerce 中的计量单位

下图说明了 WebSphere Commerce Server 中的计量单位的结构:



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

数量单位和数量单位格式

*数量单位*是商店中使用的计量单位，例如，千克、磅、米、英寸、公升等等。数量单位格式是在商店中格式化此数量单位的方式，例如当显示此数量单位时将使用几位小数位。

每个*数量单位格式*是一个商店实体的组成部分（且只属于此实体），但是每个商店实体可以具有若干个数量单位格式。

每个数量单位和数值的用法都可以有一个数量单位格式；根据商店支持多少种语言，数量单位格式可以具有一个或多个数量单位格式描述。

数量单位格式描述

*数量单位格式描述*描述了如何（为了显示的目的）格式化以特定数量单位和特定语言表示的数额。

数值用法

*数值的用法*定义了在应用程序中使用数值的方式。例如，通过使用 WebSphere Commerce 代码中的数值用法代码，您可以选择您喜欢的方式来格式化或舍入该数值（货币或数量）。这些代码（在 NUMBRUSG 表中定义）使数值根据在 CURFORMAT、CURFMTDESC、QTYFORMAT 和 QTYFMTDESC 表中为该类型的数值用法指定的规则来进行格式化。这使商店能够以不同的方式来格式化数值以满足不同场合的需要。

关于 WebSphere Commerce Server 中计量单位有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的数量单位数据模型。

在 WebSphere Commerce 中创建计量单位

当创建实例时，在 WebSphere Commerce Server 数据库中预填充了计量单位。关于更多信息，请参阅第 35 页的第 5 章，『站点有用资源』。

您还可以在 WebSphere Commerce 中定义在商店中使用的新的计量单位，或者删除您不想在商店中使用的计量单位。

要定义在商店中使用的新的计量单位，请向以下数据库表添加信息:

- QTYUNIT
- QTUNITDSC
- QTYFORMAT
- QTYFMTDESC
- QTYUNITMAP
- QTYCONVERT

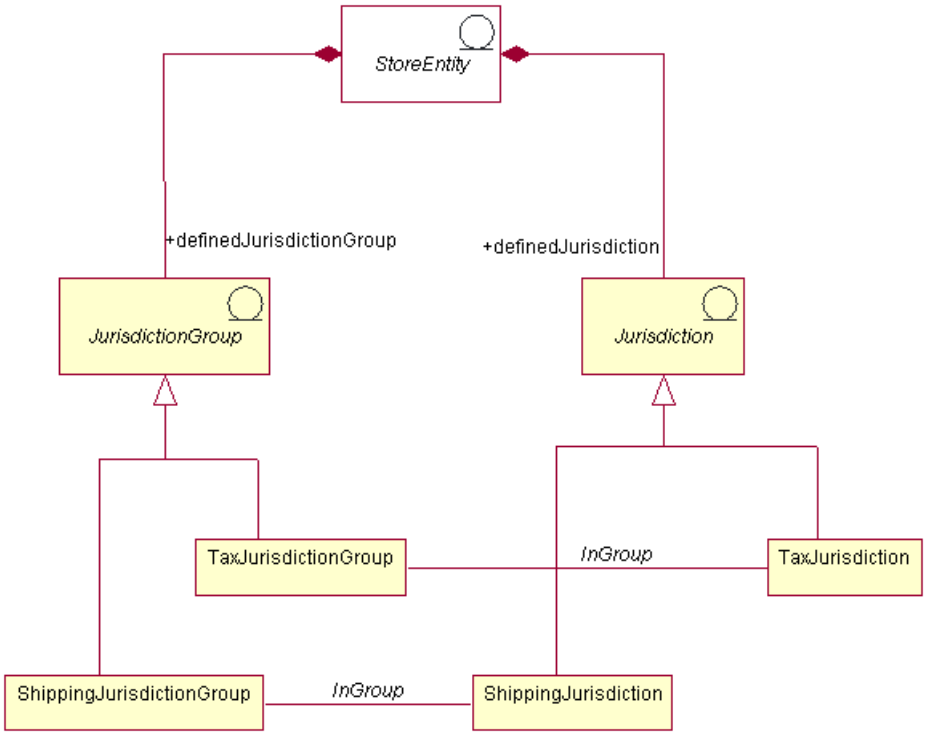
第 17 章 地区有用资源

*地区*是地理意义上的区域，它代表您对其销售商品的一个国家或地区、省／直辖市或领土，或邮政编码范围。可以将地区分组在一起形成*地区组*。

地区组用于订单上的装运费用和税款费用的计算。即，地区组可以用于限定所使用的装运费用和税款计算规则。只有商品将装运到与计算规则关联的地区组中的某个地区内的地址时，这些符合条件的计算规则才适用于订单上的这些商品。所以，根据订单中不同商品的送货地址，装运费用和税款金额的计算会有不同。

了解 WebSphere Commerce 中的地区有用资源

下图说明了地区和地区组如何整合到 WebSphere Commerce Server 中。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

在 WebSphere Commerce 中，地区或地区组是商店的组成部分，而对于为其创建该地区或地区组的商店或商店组而言，该地区或地区组是此商店或商店组专用的。例如，如果为您的商店创建了三个地区，然后删除了该商店，则同时也会删除地区。这些地区对于任何其它现有的商店或可能在将来创建的任何商店而言，都是不可使用的。

不过，如果您为商店组创建了地区，在删除该组中的商店时并不会删除地区。地区将可用于在该商店组中新创建的商店。

WebSphere Commerce 支持两种类型的地区：装运地区和税收地区。可以将装运地区分组在一起形成装运地区组，装运地区组限定装运费用计算规则。类似地，可以将税收地区分组在一起形成税收地区组，税收地区组限定税款计算规则。

关于 WebSphere Commerce Server 中地区有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的地区数据模型。

在 WebSphere Commerce 中创建地区有用资源

为了应用税款和装运费用，您必须为您的商店创建地区有用资源。关于创建地区的更多信息，请参阅第 137 页的『在 WebSphere Commerce 中创建税款有用资源』或第 121 页的『在 WebSphere Commerce 中创建装运有用资源』。

一旦为您的商店创建了地区，您就可以使用“商店服务”中的“税款”笔记本和“装运”笔记本来编辑它们或创建新的地区。

注: “商店服务”自动为创建的每个地区创建一个地区组。“商店服务”为商店而不是商店组创建地区。

注: “商店服务”仅以商店归档文件的格式处理预发布数据。

第 18 章 装运有用资源

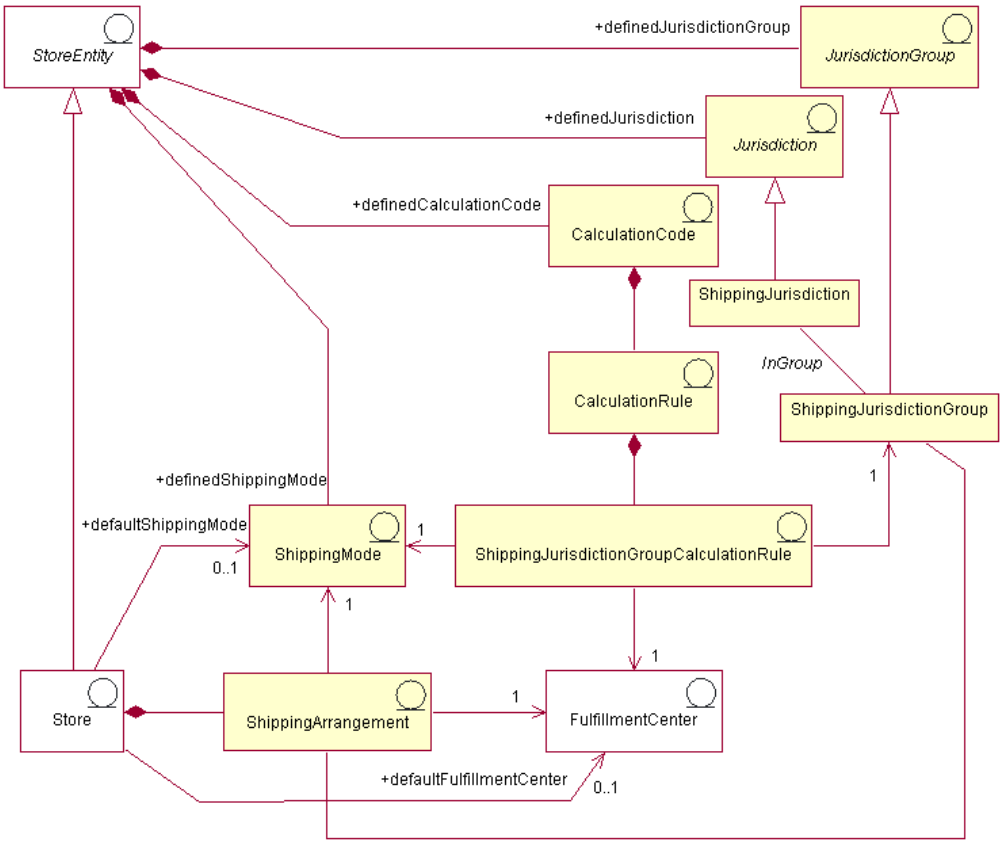
装运指的是商店如何处理物理递送给顾客的商品。大部分情况下，商品从实现中心装运，实现中心是负责储存商店商品的独立中介。

为了提供装运服务，并对这些服务收费，用 WebSphere Commerce 创建的商店应当包含以下内容:

- 至少一种装运方式
- 至少一个装运费用计算代码
- 地区和地区组

了解 WebSphere Commerce 中的装运有用资源

下图说明了 WebSphere Commerce Server 中的装运结构。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A, 『UML 图注』。

装运方式

*装运方式*是装运商品的方式。更具体地说，装运方式是装运递送者（它是提供从实现中心到顾客的装运服务的公司）和该装运递送者所提供的装运服务的组合。例如，“ABC 装运公司，隔夜服务”以及“ABC 装运公司，快递”都是装运方式。

装运方式属于商店实体。如果删除了商店实体，则在该商店实体内定义装运方式也将被删除。商店并不一定要有缺省的装运方式，但建议还是设置一个缺省装运方式。

装运安排

*装运安排*是商店和实现中心之间的一种安排，它指示实现中心将使用指定的装运方式装运特定商店的商品。可以对某个装运安排设置某些限制，包括装运安排的有效期限和装运地区。

如果装运安排与某个装运方式关联，则该装运安排仅适用于该装运方式。否则，装运安排适用于所有可用的装运方式。装运安排是商店的组成部分，如果删除了商店，则也将删除它。

计算代码

*计算代码*用于计算装运费用，即，装运计算代码指示如何计算订购商品的装运费用。要计算订购商品的装运费用，您必须将装运费用计算代码指定给一个产品目录条目或一组产品目录条目。

计算代码是商店实体的组成部分。一个计算代码只能与一个商店实体相关联，但是一个商店实体可以有几个计算代码。如果删除了商店实体，则与该商店实体关联的计算代码也将被删除。

关于使用计算代码的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

计算规则

每一计算代码都有一组*计算规则*。根据装运方式、实现中心以及装运地区的不同，订购商品的装运费用可能会发生变化。ShippingJurisdictionGroupCalculationRules 是将装运费用计算规则与地区、实现中心和装运方式相关联的关系对象，用于确定每一订购商品应当使用的计算规则。

如果删除了计算规则或者 ShippingJurisdictionGroupCalculationRules 引用的任何其它对象，则也将删除 ShippingJurisdictionGroupCalculation 规则。

关于使用计算规则的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

地区和地区组

*地区*是地理意义上的区域，它代表您对其销售商品的一个国家或地区、省／直辖市或领土，或邮政编码范围。将地区分组在一起就形成*地区组*。

WebSphere Commerce 支持两种类型的地区: 装运地区和税收地区。这些地区中每一个都是相应组的组成部分，例如，装运地区在装运地区组中，而税收地区在税收地区组中。

地区组与计算规则关联。计算规则使用地区组作为计算的一部分以确定装运费用金额。

地区和地区组是商店实体的组成部分。如果删除了商店实体，则与该商店实体关联的地区和地区组也将被删除。

一个送货地址可以对应到若干装运地区。例如，美国纽约的送货地址适用于以下装运地区: “美国纽约”、“美国”和“世界”。当送货地址适用于多个装运地区时，将适用若干装运费用计算规则。在此情况下，关联的 ShippingJurisdictionGroupCalculationRules 的优先级用于确定使用什么规则。

关于 WebSphere Commerce Server 中装运有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的税款对象和数据模型。

在 WebSphere Commerce 中创建装运有用资源

WebSphere Commerce 中的“商店服务”工具让您能够在商店归档文件中创建和编辑一些装运有用资源（例如装运方式和地区），但不是所有的装运有用资源。关于您可以使用“商店服务”工具编辑哪些有用资源的更多信息，请参阅 WebSphere Commerce 联机帮助主题 *“更改商店数据库有用资源”*。

注: “商店服务”工具以商店归档文件的格式处理预填充的 XML 文件。
您还能以 XML 文件（这些文件可以使用装入程序软件包装入到数据库中）格式创建装运有用资源。这样，有以下两种选择可用于创建装运有用资源:

- 编辑 WebSphere Commerce 提供的某个样本商店中或现有商店归档文件中现有的装运有用资源。
- 以 XML 文件的格式创建新的装运有用资源。

关于在现有商店归档文件中编辑装运有用资源的信息，请参阅 WebSphere Commerce 联机帮助。关于以 XML 文件格式创建新的装运有用资源的信息，请参阅『使用 XML 文件创建装运有用资源』。

使用 XML 文件创建装运有用资源

请以 XML 文件（这些文件可以使用装入程序软件包装入到数据库中）格式创建装运有用资源。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。如果正在创建多文化商店，则您可能希望为商店支持的每种语言环境创建各自的 XML 文件。特定于语言环境的文件应当指定所有描述信息，以便比较容易进行翻译。

样本商店（这些任务中的很多示例示例从其中获取）对所有不需要翻译的信息使用一个 `shipping.xml` 文件，对需要翻译的信息针对商店支持的每种语言环境使用另一个 `shipping.xml` 文件。特定于语言环境的文件包含所有描述信息，因此可以容易地进行翻译。

要使用 XML 文件为您的商店创建装运有用资源，请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，装运费用）。
2. 复查用于为样本商店创建装运有用资源的 XML 文件。样本商店中的所有文件都位于相应的商店归档文件中。每个样本商店包含两个或更多 shipping.xml 文件，这些文件包含装运信息。商店归档文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   要查看商店归档文件中的 shipping.xml 文件，请使用 ZIP 程序将其解压缩。shipping.xml 文件位于 data 目录中。特定于语言的 shipping.xml 位于 data 目录中特定于语言环境的子目录中。
3. 请复查第 289 页的附录 B, 『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 shipping.xml 文件或通过创建一个新文件来创建 shipping.xml 文件。关于更多信息，请参阅相应于 shipping.xml 的 DTD 文件。DTD 文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
5. 定义地区和地区组，您正在将商品和服务装运到该地区和地区组。所有地区都必须属于一个地区组。
   a. 使用以下示例作为指南，在您的 XML 文件中为 JURSTGROUP 表定义一个地区组:

   ```
   < jurstgroup
    jurstgroup_id="@jurstgroup_id_1"
    description="Jurisdiction Group1 for Shipping"
   subclass="1"
   storeent_id="@storeent_id_1"
   code="World"/>
   ```

   其中
   - jurstgroup_id 是生成的唯一键

- description 是地区组的简短描述，它适合于在管理地区组的用户界面中显示。
- subclass 是如下地区组子类:
  - 1 = ShippingJurisdictionGroup
  - 2 = TaxJurisdictionGroup
- storeent_id 是与此地区组关联的商店实体。
- code 与其商店实体和子类一起唯一标识该地区组。

b. 使用以下示例作为指南，在您的 XML 文件中为 JURST 表定义地区。

```
< jurst
jurst_id="@jurst_id_1"
storeent_id="@storeent_id_1"
code="World"
subclass="1"/>
```

其中

- jurst_id 是生成的唯一键
- storeent_id 是与此地区组关联的商店实体。
- code 与其商店实体和子类一起唯一标识该地区组。
- subclass 是如下地区子类:
  - 1 = ShippingJurisdiction
  - 2 = TaxJurisdiction

c. 使用以下示例作为指南，通过向 JURSTGRPREL 表添加信息，将您在步骤 b 中创建的地区与在步骤 a 中创建的地区组相关联。

```
< jurstgprel
jurst_id="@jurst_id_1"
jurstgroup_id="@jurstgroup_id_1"
subclass="1"/>
```

其中

- jurst_id 是地区
- jurstgroup_id 是地区组
- subclass 是地区和地区组的子类。它们应当匹配:
  - 1 = ShippingJurisdiction[Group]
  - 2 = TaxJurisdiction[Group]

d. 对您商店支持的所有地区和地区组重复步骤 a 到 c。

6. 定义您的商店将使用的装运方式。

a. 使用以下示例作为指南，在您的 XML 文件中为 SHIPMODE 表定义一个装运方式:

```
<shipmode
shipmode_id="@shipmode_id_1"
field1
storeent_id="@storeent_id_1"
code="Ground 1 week"
```

```
carrier="XYZ Carrier" />
```

其中:

- shipmode_id 是生成的唯一键。
- field1 是可供定制的字段。
- storeent_id 是与此装运方式关联的商店实体。
- code 是商家指定的代码，对于商店实体而言它是唯一的。
- carrier 是递送者的名称或标识。

b. 使用以下示例作为指南，将有关装运方式的信息添加到 SHPMODEDSC 表。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中:

```
< shpmodedsc
description="International mail"
field1="USD$5.00 per order plus USD$1.00 for each item"
field2="5 business days"
shipmode_id="@shipmode_id_1"
language_id="&en_US;" />
```

其中:

- description 是 ShippingMode 的简短描述，适合于对顾客显示以供选择。
- field1 和 field2 是可供定制的字段。
- shipmode_id 是生成的唯一键。
- language_id 是使用的语言。

c. 对您的商店中所有的装运方式重复步骤 a 和 b。

7. 定义您的商店要使用的计算代码。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALCODE 表定义计算代码。

```
< calcode
calcode_id="@calcode_id_1"
code="shipping Code 1- per/order"
calusage_id="-2"
storeent_id="@storeent_id_1"
groupby="0″
published="1"
sequence="+0.00E+000"
calmethod_id="-23"
calmethod_id_app="-24"
calmethod_id_qfy="-22"
flags="0" />
```

其中:

- calcode_id 是生成的唯一键。
- code 是在给定特定的 CalculationUsage 和 StoreEntity 时，唯一标识此 CalculationCode 的字符串。

- calusage_id 指示使用此 CalculationCode 的计算的类型。例如，CalculationCode 可能用于计算以下一种货币金额:
  - 折扣（-1）
  - 装运费用（-2）
  - 销售税（-3）
  - 装运税（-4）
  - 赠券（-5）
- storeent_id 是与此计算代码关联的商店实体。
- groupby 是位标志，它向 CalculationCodeCombineMethod 指示在执行计算时应该如何将 OrderItem 分组。0 = 不分组。将所有可应用的 OrderItem 放置在一个组中。关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 *CALCODE 表: 详细信息*。
- published 指定是否发布了计算代码:
  - 0 = 未发布（临时禁用）
  - 1 = 已发布
  - 2 = 标记为删除（且未发布）
- sequence 按照从低到高的顺序计算并应用 CalculationCode。如果两个计算代码具有相同的序号，则将首先计算具有较低 calcode_id 的计算代码。
- calmethod_id 是 CalculationCodeCalculateMethod，它定义了如何计算此 CalculationCode 的货币金额。calmethod_id="-23"（装运的 CalculationCodeCalculateMethod）是随 WebSphere Commerce 提供的唯一装运计算方法。
- calmethod_id_app 是 CalculationCodeApplyMethod，它存储关联的 OrderItem 的已计算金额。calmethod_id_app="-24"（装运的 CalculationCodeApplyMethod）是随 WebSphere Commerce 提供的唯一装运应用方法。
- calmethod_id_qfy 是 CalculationCodeQualifyMethod，它定义了哪些 OrderItem 与此 CalculationCode 关联。calmethod_id_qfy="-22"（装运的 CalculationCodeQualifyMethod）是随 WebSphere Commerce 提供的唯一装运限定方法。
- flags 指定是否应该调用此 CaculationCode 的 CalculationCodeQualifyMethod。
  - 0 = 没有限制。不调用此方法
  - 1 = 限制。将调用此方法

b. 使用以下示例作为指南，在您的 XML 文件中为 CALCODEDSC 表添加计算代码描述信息。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中。

```
<calcodedsc
calcode_id="@calcode_id_3"
description="5.00USD per order"
language_id="&en_US"
longdescription= "This shipping calculation code charges 5.00USD per order."
/>
```

其中

- calcode_id 是此信息应用的计算代码。
- description 是此计算代码的简短描述。

- language_id 是此信息所应用的语言。
- longdescription 是此计算代码的详细描述。

c. 对商店中使用的每个计算代码重复步骤 a 和 b。

8. 定义您的商店的计算规则。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALRULE 表设置计算规则:

```
< calrule
 calrule_id="@calrule_id_1"
calcode_id="@calcode_id_1"
startdate="1900-01-01 00:00:00.000000"
enddate="2100-01-01 00:00:00.000000"
sequence="+1.00000000000000E+000"
combination="2"
calmethod_id="-27"
calmethod_id_qfy="-26"
flags="1"
identifier="1" />其中
```

- calrule_id 是生成的唯一标识。
- calcode_id 是计算代码，此计算规则是其组成部分。
- startdate 是此计算规则生效的时间。
- enddate 是此计算规则失效的时间。
- sequence 是处理此计算规则的顺序。按照从低值到高值的顺序处理相同计算代码的计算规则。
- combination 指定特殊处理的位标志，缺省的 CalculationRuleCombineMethod 实现将执行这些特殊处理。关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALRULE 表。
- calmethod_id 是 CalculationRuleCalculateMethod，它计算一系列 OrderItem 的货币结果。
- calmethod_id_qfy 是 CalculationRuleQualifyMethod，它确定一系列 OrderItem 中的哪些 OrderItem 应该发送到 CalculationRuleCalculateMethod。
- flags 由 CalculationRuleCombineMethod 使用，以确定此计算规则如何与其它计算规则组合。关于更多信息，请参阅 CALRULE 表。
- identifier 与其计算代码结合标识了此计算规则。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALRULE 表。

b. 对您的商店中使用的每个计算规则重复步骤 a。注意，每个计算代码可以具有几个计算规则。例如，calcode_id=″@calcode_id_1″ 可以与若干个 calrule_id 关联。

9. 定义您的商店的计算标度。

计算标度是将应用于计算的一组范围。例如，对于装运费用，您可以具有一组重量范围，每个范围对应一个特定的费用。即，重量在 0 到 5 千克之间的产品装运费用可能为 ￥80.00。而重量在 5 到 10 千克之间的产品装运费用可能为 ￥120.00。这些范围创建了一个标度。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALRULE 表设置计算标度:

```
<calscale
calscale_id="@calscale_id_1"
code="Scale Code 1 per order USD"
storeent_id="@storeent_id_1"
calusage_id="-2"
setccurr="USD"
calmethod_id="-28"/>
```

其中

- calscale_id 是生成的唯一标识。
- code 是在给定特定的计算用法和商店实体时，唯一标识此计算标度的字符串。
- storeent_id 是商店实体，此计算标度是其组成部分。
- calusage_id 指示使用此 CalculationScale 的计算的类型。例如，CalculationScale 可能用于计算以下一种货币金额:
  - 折扣（-1）
  - 装运费用（-2）
  - 销售税（-3）
  - 装运税（-4）
  - 赠券（-5）
- setccurr 如果指定的话，它指示此计算标度的计算范围对象的范围开始值的货币。CalculationScaleLookupMethod 应该返回以此货币表示的 “查找数值”。
- calmethod_id 是 CalculationScaleLookupMethod: 给定的一组订购商品时，该 CalculationScaleLookupMethod 确定查找值、基本货币值、结果乘数以及一组数学权值，它们可以由计算标度用来计算货币金额。要确定使用何种 CalculationScaleLookupMethod，请执行以下操作:
  - 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击 “CALMETHOD 表: 详细信息” 的链接。此表列出了可用的计算方法类型。MonetaryCalculationScaleLookupMethod 方法是 9。
  - 打开引导程序文件 wcs.bootstrap_xx_XX.xml，其中 xx_XX 是语言环境的代码。引导程序文件位于以下目录中:
    - NT drive:\WebSphere\CommerceServer\schema\xml
    - 2000 drive:\Program Files\WebSphere\CommerceServer \schema\xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml
    - Linux /opt/WebSphere/CommerceServer/schema/xml
    - 400 /qibm/proddata/WebCommerce/schema/xml
  - 定位列有可用计算方法（CALMETHOD）的部分。
  - 使用税款的 calusage_ID 值（-3 代表销售税，-4 代表装运税）定位计算方法。

– 定位子类为 7 的计算方法；有若干个。请选择满足您需要的一种。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALSCALE 表。

b. 对您的商店中使用的每个计算标度重复步骤 a。例如，对于装运费用，“流行时尚”创建“按照订单计费”标度和“按照商品计费”标度。

10. 定义计算标度的计算范围。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALRANGE 表设置计算范围。

```
<calrange
calrange_id="@calrange_id_1"
calscale_id="@calscale_id_1"
calmethod_id="-33"
rangestart="0.00000"
cumulative="0"/>
```

其中

- `calrange_id` 是生成的唯一标识。
- `calcode_id` 是计算标度，此计算范围是其组成部分。
- `calmethod_id` 是 CalculationRangeMethod，它确定来自 CalculationRangeLookupResult 的货币金额。例如，FixedAmountCalculationRangeCmd、PerUnitAmountCalculationRangeCmd 或 PercentageCalculationRangeCmd。要确定 CalculationRangeMethod，请执行以下操作:
  - 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击“CALMETHOD 表: 详细信息”的链接。此表列出了可用的计算方法类型。CalculationRangeMethod 是 10。
  - 打开引导程序文件 `wcs.bootstrap_xx_XX.xml`，其中 `xx_XX` 是语言环境的代码。引导程序文件位于以下目录中:
    - NT `drive:\WebSphere\CommerceServer\schema\xml`
    - 2000 `drive:\Program Files\WebSphere\CommerceServer \schema\xml`
    - AIX `/usr/WebSphere/CommerceServer/schema/xml`
    - Solaris `/opt/WebSphere/CommerceServer/schema/xml`
    - Linux `/opt/WebSphere/CommerceServer/schema/xml`
    - 400 `/qibm/proddata/WebCommerce/schema/xml`
  - 定位列有可用计算方法（CALMETHOD）的部分。
  - 使用税款的 calusage_ID 值（-3 代表销售税，-4 代表装运税）定位计算方法。
  - 定位子类为 9 的计算方法；有若干个。请选择满足您需要的一种。
- `cumulative` 是有效值:
  - 0 = 仅使用与最高 RANGESTART 值匹配的 CalculationRange。
  - 1 = 使用所有匹配的 CalculationRange。将已计算的货币金额相加得到最后的结果。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALRANGE 表。

b. 对与商店中使用的计算标度关联的每个计算范围重复步骤 a。

11. 定义计算标度的计算查找值。计算查找值是与计算标度关联的值。例如，计算标度包含以下重量范围以及关联的装运价格:

- 0 - 5 千克的费用是 ￥80.00
- 5 - 10 千克的费用是 ￥120.00

查找值是 ￥80.00 和 ￥120.00。

a. 使用以下示例作为指南，在 XML 文件中为 CALRLOOKUP 表设置计算查找值。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中（商店支持的每种语言环境都有一个特定于该语言环境的 XML 文件）。例如，如果商店为美国和日本的顾客送货，您应该在一个 XML 文件中添加美元查找值，在另一个 XML 文件中添加日元查找值。

```
<calrlookup
calrlookup_id="@calrlookup_id_1"
setccurr="USD"
calrange_id="@calrange_id_1"
value="5.00"/>
```

其中

- `calrlookup_id` 是生成的唯一标识。
- `calrange_id` 是计算范围，此计算范围查找结果是其组成部分。
- `value` 是计算范围查找结果的值，它由计算范围的计算范围方法使用以确定货币结果。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALLOOKUP 表。

b. 对与商店中使用的计算标度关联的每个查找值重复步骤 a。

12. 将计算规则与计算标度关联。

a. 使用以下示例作为指南，在您的 XML 文件中为 CRULESCALE 表将计算标度与计算规则关联。

```
< crulescale
calrule_id="@calrule_id_1"
calscale_id="@calscale_id_1" />
```

其中

- `calrule_id` 是计算规则。
- `calscale_id` 是计算标度。

b. 对每个计算标度和规则的关联重复步骤 a。

关于 **@** 和 **&** 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

创建装运实现有用资源

为使您的装运有用资源在商店中正确工作，您必须将装运地区组与计算规则关联，并将实现中心与商店中使用的装运方式关联。

您必须在可以将装运有用资源关联到实现中心之前创建实现中心有用资源。关于创建实现有用资源的更多信息，请参阅第 99 页的『在 WebSphere Commerce 中创建实现有用资源』。

在创建实现有用资源之后，请通过向 SHPJCRULE 和 SHPARRANGE 表添加信息来将装运有用资源与实现有用资源关联。请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，装运费用）。
2. 复查用于为样本商店创建装运实现有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。每个样本商店包含一个 `shipfulfill.xml` 文件，它包含装运实现信息。要查看商店归档文件中的 `shipfulfill.xml` 文件，请使用 ZIP 程序将其解压缩。`shipfulfill.xml` 文件位于数据目录中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。
3. 请复查第 289 页的附录 B, 『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 `shipfulfill.xml` 文件或通过创建一个新文件来创建 `shipfulfill.xml` 文件。关于更多信息，请参阅相应于 `shipfulfill.xml` 的 DTD 文件。DTD 文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\xml\sar`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`
5. 通过向 SHPJCRULE 表添加信息，将计算规则与装运地区组关联。使用以下示例作为指南。如果您正创建多文化商店，则还需要为商店支持的每种语言环境创建一个 XML 文件。

```
<shpjcrule
calrule_id="@calrule_id_1"
ffmcenter_id="@ffmcenter_id_1"
jurstgroup_id="@jurstgroup_id_1"
precedence="0"
shipmode_id="@shipmode_id_1"
shpjcrule_id="@shpjcrule_id_1"
```

其中

- calrule_id 是使用的计算规则。
- ffmcenter_id 是实现中心。如果这为 NULL，则此关联将应用到所有的实现中心上。
- jurstgroup_id 是装运地区组。如果这为 NULL，则此关联将应用到所有的装运地区组上。
- precedence 是对于同一个实现中心和装运方式，送货地址落在多个指定的装运地区组中的情况。只有具有最高的 SHPJCRULE.PRECEDENCE 值的计算规则才适用。
- shipmode_id 是装运方式。
- shpjcrule_id 是生成的唯一标识。

6. 对商店中每个地区组、实现中心和规则的关联重复步骤 3。
7. 通过向 SHPARRANGE 表添加信息，将装运方式和实现中心与商店关联。使用以下示例作为指南:

```
<shparrange
 shparrange_id="@shparrange_id_2"
   store_id="@storeent_id_1"
   ffmcenter_id="@ffmcenter_id_1"
 shipmode_id= "@shipmode_id_2"
 startdate="1970-06-22 23:00:00.000000"
 enddate= "2008-06-22 23:00:00.000000"
 precedence= "0"
 flags="0"
/>
```

其中

- shparrange_id 是生成的唯一标识。
- store_id 是商店。
- ffmcenter_id 是实现中心。
- shipmode_id 是装运方式。NULL 指示不论何种装运方式都可以使用此装运安排。
- startdate 是此装运安排开始生效的时间。
- enddate 是此装运安排失效的时间。
- precedence 是当特定时间（对于同一个商店和装运方式）有多个装运安排有效时，使用 PRECEDENCE 最高的一个安排。
- flags 包含位标志:
  - 1 = 限制 — 该装运安排仅适用于某些订购商品，这些订购商品的送货地址和（通过 SHPARJURGP 表）与此装运安排关联的装运地区组之一相匹配。

8. 对商店中使用的所有装运方式重复步骤 5。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

创建商店—产品目录—装运有用资源

为了将装运方式与商店关联，您必须对商店包含的每个合同，将计算代码与商店中的产品目录条目关联。

您必须在可以创建“商店—产品目录—装运”有用资源之前，创建您的商店和产品目录有用资源。关于创建商店有用资源的更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。关于创建产品目录有用资源的更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。

要创建“商店—产品目录—装运”有用资源，请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，装运费用）。
2. 复查用于为样本商店创建装运实现有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包含一个 `store-catalog-shipping.xml` 文件，它包含装运实现信息。要查看商店归档文件中的 `store-catalog-shipping.xml` 文件，请使用 ZIP 程序将其解压缩。`store-catalog-shipping.xml` 文件位于 data 目录中。
3. 请复查第 289 页的附录 B，『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 `store-catalog-shipping.xml` 文件或通过创建一个新文件来创建 `store-catalog-shipping.xml` 文件。关于更多信息，请参阅相应于 `store-catalog-shipping.xml` 的 DTD 文件。DTD 文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\xml\sar`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`

- 400 /qibm/proddata/WebCommerce/xml/sar

5. 通过向 CATENCALCD 表添加信息，创建“商店—产品目录—装运”关系。使用以下示例作为指南:

```
<catencalcd
 calcode_id="@calcode_id_1"
 catencalcd_id="@catencalcd_id_1"
   store_id="@storeent_id_1"
/>
```

其中

- calcode_id 是计算代码。
- catencalcd_id 是生成的唯一标识。
- store_id 是商店。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

创建缺省装运方式

为了设置商店的缺省装运方式，您必须向 STOREDEF 表添加信息。要向 STOREDEF 表添加信息，请执行以下操作:

1. 复查用于为样本商店创建商店缺省有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包含一个 store-defaults.xml 文件，它包含缺省装运信息。要查看商店归档文件中的 store-defaults.xml 文件，请使用 ZIP 程序将其解压缩。store-defaults.xml 文件位于 data 目录中。

2. 请复查第 289 页的附录 B, 『创建数据』中的信息。
3. 通过复制样本商店归档文件中的一个 store-defaults.xml 文件或通过创建一个新文件来创建 store-defaults.xml 文件。关于更多信息，请参阅相应于 store-defaults.xml 的 DTD 文件。DTD 文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar

- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

4. 使用以下示例作为指南，在 XML 文件中，通过向 STOREDEF 表添加信息，指定商店的缺省装运方式:

```
<storedef
   store_id="@storeent_id_1"
   shipmode_id="@shipmode_id_1"
/>
```

其中

- store_id 是商店。
- shipmode_id 是商店的缺省装运方式。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 19 章 税款有用资源

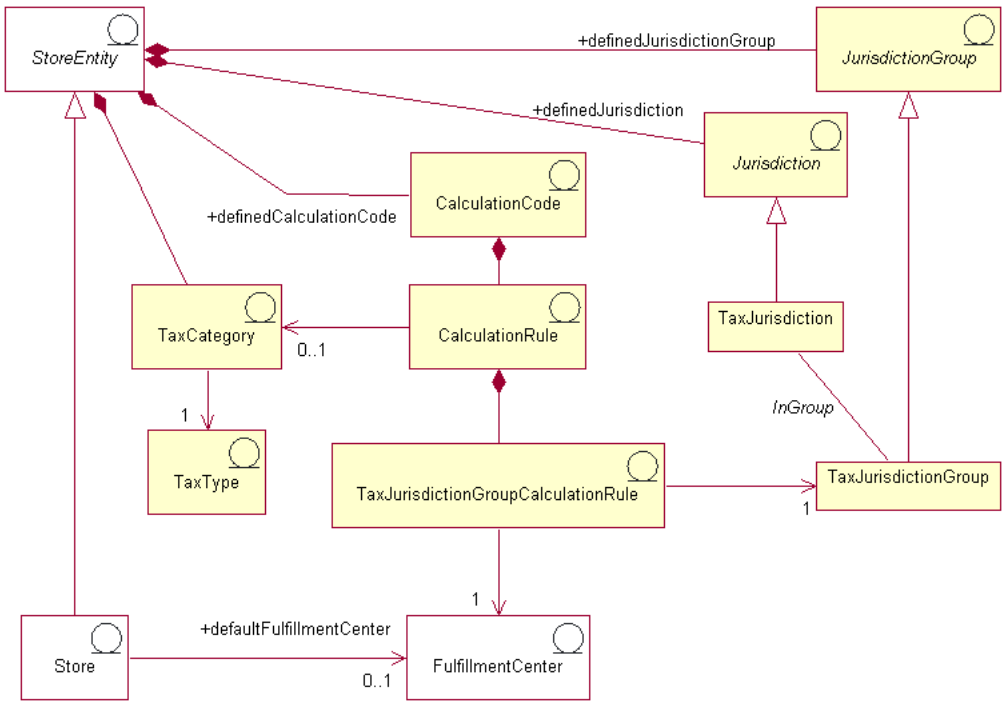
为了对您的商店提供的商品和服务收取税款，用 WebSphere Commerce 创建的商店必须包含以下内容:

- 税类别
- 计算代码
- 地区和地区组

税类别、计算代码以及地区和地区组的组合创建了此商店的税款费用。

了解 WebSphere Commerce 中的税款有用资源

下图说明了 WebSphere Commerce Server 中税务的结构。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

税类别

*税类别*对应于商店可能需要收取的不同类型的税款，例如国税和地方税。

税类别是一个商店实体的组成部分，尽管商店实体可以具有若干个税类别。如果删除了商店实体，则与该商店实体关联的税类别也将被删除。

税款类型

商店通常收取两种类型的税款: 销售税或使用税，以及装运税。每个税类别具有一个*税款类型*。尽管每个税类别可能只属于一种税款类型（例如税类别“国税”是销售税类型），但是可能有若干个不同的税类别属于同一个税款类型（例如，税款类型“销售税”适用于类别“国税”和“地方税”）。

计算代码

*计算代码*用于计算税款费用，即，税款计算代码指示如何计算订购商品的税款。要计算订购商品的税款，您必须将销售税款和装运税款计算代码指定给一个产品目录条目或一组产品目录条目。对于一个特定的产品目录条目或产品目录条目组，仅可以应用每个税款类型的一种税款计算代码。通常情况下，对净价征收销售税或使用税，对装运费用征收装运税。

计算代码是商店实体的组成部分。一个计算代码只能与一个商店实体相关联，但是一个商店实体可以有几个计算代码。如果删除了商店实体，则与该商店实体关联的计算代码也将被删除。

关于使用计算代码的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

计算规则

每个计算代码至少有一个*计算规则*，计算规则定义每个税类别的计算并指定进行计算的条件。每个税款计算规则都与一个税类别、一个地区组和一个实现中心关联，它们共同定义了计算规则的使用条件。例如，根据订单中指定的送货地址和实现中心的不同，可能选择不同的规则来计算特定税类别的金额。

每个计算规则只属于一个计算代码。

一个特定的税款计算代码可以具有若干个计算规则，每个与此商店关联的税类别、税收地区组和实现中心的组合都有一个计算规则。每个销售税款和装运税款计算规则都可以和多个 TaxJurisdictionGroupCalculationRules（TaxRules）关联。如下表所示，计算规则 10001 对地区组 1234 和 1235 都适用。

TAXJCRULE_ID	CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID	PRECEDENCE
10001	10001	NULL	1234	0
10002	10001	NULL	1235	0

每个 TaxRule 定义了应当应用计算代码的条件。例如，您可以为商店装运的每一地区组定义一个计算规则。在以下示例中，计算规则 10001 对地区组 1234 和 1235 都适用。

在以下示例中，当税收地区为阿尔伯特时，税款计算代码对地方销售税类别使用计算规则 A；当税收地区为英属哥伦比亚时，则使用计算规则 C。

税收地区	国家销售税	地方销售税
阿尔伯特，加拿大	计算规则 B，它给出 Y%	计算规则 A，它给出 X%
英属哥伦比亚，加拿大	计算规则 B，它给出 Y%	计算规则 C，它给出 Z%

当一个送货地址与多个税收地区组相匹配时，则将使用其关联的 TAXJCRULE.PRECEDENCE 列值最高的计算规则。

TaxJurisdictionGroupCalculationRules（TaxRule）与计算规则的关联确定了何时适用此计算规则。当满足 TaxRules 给定的任一条件时，将适用销售税或装运税计算规则。在以下示例中，当您装运到地区组 1001 时，或者从实现中心 1001 装运时，适用计算规则 10001。

CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID
10001	NULL	1001
10001	1001	1001

每个 TaxJurisdictionGroupCalculationRule 都至多与一个地区组关联。计算规则自身并不直接与地区组关联。

关于使用计算规则的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

地区和地区组

*地区*是地理意义上的区域，它代表您对其销售商品的一个国家或地区、省／直辖市或领土，或邮政编码范围。将地区分组在一起就形成*地区组*。

WebSphere Commerce 支持两种类型的地区：装运地区和税收地区。这些地区中每一个都是相应组的组成部分，例如，装运地区在装运地区组中，而税收地区在税收地区组中。

地区和地区组确定使用何种计算规则来计算税款费用。

地区和地区组是商店实体的组成部分。每个地区和地区组都是一个商店实体的组成部分，然而一个商店实体可以具有若干个地区或地区组。如果删除了商店实体，则与该商店实体关联的地区和地区组也将被删除。

关于 WebSphere Commerce Server 中税款有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的税款对象和数据模型。

在 WebSphere Commerce 中创建税款有用资源

WebSphere Commerce 中的“商店服务”工具让您能够在商店归档文件中创建和编辑一些税款有用资源（例如税类别和地区），但不是所有的税款有用资源。关于您可以使用“商店服务”工具编辑哪些有用资源的更多信息，请参阅 WebSphere Commerce 联机帮助主题“*更改商店数据库有用资源*”。

注: “商店服务”工具以商店归档文件的格式处理预填充的 XML 文件。
您还能以 XML 文件（这些文件可以使用装入程序软件包装入到数据库中）格式创建税款有用资源。这样，有以下两种选择可用于创建装运有用资源:

- 编辑 WebSphere Commerce 提供的某个样本商店中或现有商店归档文件中现有的税款有用资源。

- 以 XML 文件的格式创建新的税款有用资源，它可以作为商店归档文件的组成部分发布，或者使用装入程序软件包装入。

关于在现有商店归档文件中编辑税款有用资源的信息，请参阅 WebSphere Commerce 联机帮助。关于以 XML 文件格式创建新的税款有用资源的信息，请参阅『使用 XML 文件创建税款有用资源』。

使用 XML 文件创建税款有用资源

请以 XML 文件（这些文件可以使用装入程序软件包装入到数据库中）格式创建税款有用资源。关于装入程序软件包的更多信息，请参阅第 193 页的第 7 部分，『发布商店』。如果正在创建多文化商店，则您可能希望为商店支持的每种语言环境创建各自的 XML 文件。特定于语言环境的文件应当指定所有描述信息，以便比较容易进行翻译。

样本商店（这些任务中的很多示例示例从其中获取）对所有不需要翻译的信息使用一个 tax.xml 文件，对需要翻译的信息针对商店支持的每种语言环境使用另一个 tax.xml 文件。特定于语言环境的文件包含所有描述信息

要使用 XML 文件为您的商店创建税款有用资源，请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，税款）。
2. 复查用于为样本商店创建税款有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包括两个 tax.xml 文件，其中包括税款信息。要查看商店归档文件中的 tax.xml 文件，请使用 ZIP 程序将其解压缩。tax.xml 文件位于 data 目录中。特定于语言的 tax.xml 位于 data 目录中特定于语言环境的子目录中。
3. 请复查第 289 页的附录 B，『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 tax.xml 文件或通过创建一个新文件来创建 tax.xml 文件。关于更多信息，请参阅相应于 tax.xml 的 DTD 文件。DTD 文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar

- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

5. 定义您正将商品和服务装运到的地区和地区组。根据适用的税类别计算规则，将税收地区指定到地区组。
   a. 使用以下示例作为指南，在您的 XML 文件中为 JURSTGROUP 表定义一个地区组:

      ```
      < jurstgroup
      jurstgroup_id="@jurstgroup_id_2"
      description="Tax Jurstiction Group 1"
      subclass="2"
      storeent_id="@storeent_id_1"
      code="World"/>
      ```

      其中
      - jurstgroup_id 是生成的唯一键
      - description 是地区组的简短描述，它适合于在管理地区组的用户界面中显示。
      - subclass 是如下地区组子类:
        - 1 = ShippingJurisdictionGroup
        - 2 = TaxJurisdictionGroup
      - storeent_id 是与此地区组关联的商店实体。
      - code 与其商店实体和子类一起唯一标识该地区组。

   b. 使用以下示例作为指南，在您的 XML 文件中为 JURST 表定义地区。

      ```
      < jurst
      jurst_id="@jurst_id_2"
      storeent_id="@storeent_id_1"
      code="World"
       subclass="2"/>
      ```

      其中
      - jurst_id 是生成的唯一键
      - storeent_id 是与此地区组关联的商店实体。
      - code 与其商店实体和子类一起唯一标识该地区组。
      - subclass 是如下地区子类:
        - 1 = ShippingJurisdiction
        - 2 = TaxJurisdiction

   c. 使用以下示例作为指南，通过向 JURSTGRPREL 表添加信息，将您在步骤 b 中创建的地区与在步骤 a 中创建的地区组相关联。

      ```
      < jurstgprel
       jurst_id="@jurst_id_2"
      jurstgroup_id="@jurstgroup_id_1"
      ```

```
 subclass="2"/>
```

其中

- jurst_id 是地区
- jurstgroup_id 是地区组
- subclass 是地区和地区组的子类。它们应当匹配:
  - 1 = ShippingJurisdiction[Group]
  - 2 = TaxJurisdiction[Group]

d. 对您商店支持的所有地区和地区组重复步骤 a 到 c。

6. 定义您的商店将使用的税类别。

a. 使用以下示例作为指南，在您的 XML 文件中为 TAXCGRY 表定义一个税类别:

```
<taxcgry
taxcgry_id="@taxcgry_id_1"
taxtype_id="-3"
storeent_id="@storeent_id_1"
name="Sales Tax"
displayseq="0"
 displayusage="0"/>
```

其中:

- taxcgry_id 是生成的唯一键。
- taxtype_id="-3" 是此税类别的税款类型。WebSphere Commerce 支持两种税款类型:
  - 销售税或使用税（-3）
  - 装运税（-4）
- storeent_id 是与此税类别关联的商店实体。
- name 是税类别的名称。此名称与商店实体一起唯一标识此税类别。
- displayseq 指定税款金额显示的顺序（从低到高），例如在订单中。
- displayusage 指定此税类别与 PriceDataBean 有关，如下:
  - 0 = 不计算
  - 1 = 已计算

  PriceDataBean 可以用于获取应当与产品价格一起显示的税款金额。

b. 对商店中使用的每个税类别重复步骤 a。

c. 使用以下示例作为指南，在您的 XML 文件中为 TAXCGRYDS 表添加税类别描述信息。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中。

```
<taxcgryds
  taxcgry_id="@taxcgry_id_1"
 description="Sales Tax"
 language_id="&en_US"/>
```

其中

- taxcgry_id 是税类别。
- description 是适合显示给顾客的税类别的简短描述。
- language_id 是显示此信息的语言。

d. 对商店中使用的每个税类别重复步骤 c。

7. 定义您的商店要使用的计算代码。

   a. 使用以下示例作为指南，在您的 XML 文件中为 CALCODE 表定义计算代码。

```
<calcode
calcode_id="@calcode_id_3"
 code="Tax Code 1"
calusage_id="-3"
storeent_id="@storeent_id_1"
groupby="0"
published="1"
sequence="0"
calmethod_id="-43"
calmethod_id_app="-44"
calmethod_id_qfy="-42"
displaylevel="0"
flags="0"
precedence="0"
/>
```

   其中:

   - calcode_id 是生成的唯一键。
   - code 是在给定特定的计算用法和商店实体时，唯一标识此计算代码的字符串。
   - calusage_id 指示使用此计算代码的计算类型。例如，计算代码可能用于计算以下一种货币金额:
     - 折扣（-1）
     - 装运费用（-2）
     - 销售税（-3）
     - 装运税（-4）
     - 赠券（-5）
   - storeent_id 是与此计算代码关联的商店实体。
   - groupby 是位标志，它向计算代码组合方法指示在执行计算时应该如何将 OrderItem 分组。零指定不进行分组（所有适用的订购商品都在单个的组中）。关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 *CALCODE 表: 详细信息*。
   - published 指定是否发布了计算代码:
     - 0 = 未发布（临时禁用）
     - 1 = 已发布
     - 2 = 标记为删除（并且不发布）
   - sequence 是计算此计算代码的顺序。按照从低到高的顺序来计算和应用计算代码。如果两个计算代码具有相同的序号，则将首先计算具有较低 calcode_id 的计算代码。
   - calmethod_id 计算代码计算方法，它定义了如何计算此计算代码的税款金额。为了确定使用何种计算代码计算方法，请执行以下操作:
     - 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击 “CALMETHOD 表: 详细信息” 的链接。此表列出了可用的 CALMETHOD 类型。计算代码计算方法类型是 3。
     - 打开引导程序文件 wcs.bootstrap_xx_XX.xml，其中 xx_XX 是语言环境的代码。引导程序文件位于以下目录中:
       - NT drive:\WebSphere\CommerceServer\schema\xml

- 2000 drive:\Program Files\WebSphere\CommerceServer \schema\xml
- AIX /usr/WebSphere/CommerceServer/schema/xml
- Solaris /opt/WebSphere/CommerceServer/schema/xml
- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 定位列有可用计算方法（CALMETHOD）的部分。
– 使用税款的 calusage_ID 值（-3 代表销售税，—4 代表装运税）定位计算方法。
– 定位子类为 3 的计算方法。此计算方法是 -43。

- calmethod_id_app 是 CalculationCodeApplyMethod，它存储关联的 OrderItem 的已计算金额。使用 calmethod_id 中所述的方法来确定使用何种计算代码应用方法。
  – calmethod_id_app="-44" 是销售税的 CalculationCodeApplyMethod。
- calmethod_id_qfy 是 CalculationCodeQualifyMethod，它定义了哪些订购商品与此计算代码关联。使用 calmethod_id 中所述的方法来确定使用何种计算代码限定方法。
  – calmethod_id_qfy="-42" 是销售税的 CalculationCodeQualifyMethod。
- display level 确定由该计算代码计算的金额是否应该与以下每项一起显示:
  – 0 = 订购商品
  – 1 = 订单
  – 2 = 产品
  – 3 = 商品
  – 4 = 合同
- flags 指定是否应该调用此计算代码的 CalculationCodeQualifyMethod。
  – 0 = 没有限制。不调用此方法
  – 1 = 限制。将调用此方法

b. 使用以下示例作为指南，在您的 XML 文件中为 CALCODEDSC 表添加计算代码描述信息。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中。

```
<calcodedsc
calcode_id="@calcode_id_3"
description="Vitamins
language_id="&en_US"
longdescription= "In Ontario vitamins are taxed federally, but
not provincially."
/>
```

其中

- calcode_id 是此信息应用的计算代码。
- description 是此计算代码的简短描述。
- language_id 是此信息所应用的语言。
- longdescription 是此计算代码的详细描述。

c. 对商店中使用的每个计算代码重复步骤 a 和 b。

8. 定义您的商店的计算规则。

   a. 使用以下示例作为指南，在您的 XML 文件中为 CALRULE 表设置计算规则:

```
<calrule
calrule_id="@calrule_id_10"
calcode_id="@calcode_id_3"
startdate="1900-01-01 00:00:00.000000"
taxcgry_id="@taxcgry_id_1"
enddate="2100-01-01 00:00:00.000000"
 flags="1"
identifier="1"
combination="2"
calmethod_id="-47"
calmethod_id_qfy="-46"
/>
```

   其中

   - calrule_id 是生成的唯一标识。
   - calcode_id 是计算代码，此计算规则是其组成部分。
   - startdate 是此计算规则生效的时间。
   - taxcgry_id 是此计算规则对其生效的税类别。
   - enddate 是此计算规则失效的时间。
   - combination 由 CalculationRuleCombineMethod 使用，以确定此计算规则如何与其它计算规则组合。关于更多信息，请参阅 CALRULE 表。
   - identifier 与其计算代码结合标识了此计算规则。
   - flags 指定位标志以指示将由缺省 CalculationRuleCombineMethod 实现执行的特殊处理。关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALRULE 表。
   - calmethod_id 是 CalculationRuleCalculateMethod，它计算一系列订购商品的货币结果。要确定使用何种计算规则计算方法，请执行以下操作:
     - 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击 “CALMETHOD 表: 详细信息” 的链接。此表列出了可用的 CALMETHOD 类型。计算规则计算方法是 7。
     - 打开引导程序文件 wcs.bootstrap_xx_XX.xml，其中 xx_XX 是语言环境的代码。引导程序文件位于以下目录中:
       - NT drive:\WebSphere\CommerceServer\schema\xml
       - 2000 drive:\Program Files\WebSphere\CommerceServer \schema\xml
       - AIX /usr/WebSphere/CommerceServer/schema/xml
       - Solaris /opt/WebSphere/CommerceServer/schema/xml
       - Linux /opt/WebSphere/CommerceServer/schema/xml
       - 400 /qibm/proddata/WebCommerce/schema/xml
     - 定位列有可用计算方法（CALMETHOD）的部分。
     - 使用税款的 calusage_ID 值（-3 代表销售税，-4 代表装运税）定位计算方法。

– 定位子类为 7 的计算方法。此计算方法是 -47。

- calmethod_id_qfy 是 CalculationRuleQualifyMethod，它确定一系列 OrderItem 中的哪些 OrderItem 应该发送到 CalculationRuleCalculateMethod。使用 calmethod_id 中所述的方法来确定使用何种计算规则限定方法。

b. 对您的商店中使用的每个计算规则重复步骤 a。注意，每个计算代码可以具有几个计算规则，一个规则用于一个适用的税类别。例如，calcode_id="@calcode_id_1" 可以与若干个 calrule_id 关联。

9. 定义您的商店的计算标度。

计算标度是将应用于计算的一组范围。这些范围创建了一个标度。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALRULE 表设置计算标度:

```
<calscale
calscale_id="@calscale_id_19"
code="Sales Tax 1"
 storeent_id="@storeent_id_1"
calusage_id="-3"
setccurr="USD"
calmethod_id="-53"
/>
```

其中

- calscale_id 是生成的唯一标识。
- code 是在给定特定的计算用法和商店实体时，唯一标识此计算标度的字符串。
- storeent_id 是商店实体，此计算标度是其组成部分。
- calusage_id 指示使用此 CalculationScale 的计算的类型。例如，CalculationScale 可能用于计算以下一种货币金额:
  – 折扣（-1）
  – 装运费用（-2）
  – 销售税（-3）
  – 装运税（-4）
  – 赠券（-5）
- setccurr 如果指定的话，它指示此计算标度的计算范围对象的范围开始值的货币。CalculationScaleLookupMethod 将以此货币返回一个 “查找号码”。在此情况下号码是未指定的；CalculationScaleLookupMethod 将以订单中的货币返回一个查找号码。货币是不需要指定的，除非标度范围开始值不是零。
- calmethod_id 是 CalculationScaleLookupMethod，给定一组订购商品，它就确定了一个查找数值、基本货币值、结果乘法器以及一组数学重量，它们可以由计算标度使用来计算货币金额。要确定使用何种 CalculationScaleLookupMethod，请执行以下操作:
  – 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击 “CALMETHOD 表: 详细信息” 的链接。此表列出了可用的 CALMETHOD 类型。MonetaryCalculationScaleLookupMethod 方法是 9。
  – 打开引导程序文件 wcs.bootstrap_xx_XX.xml，其中 xx_XX 是语言环境的代码。引导程序文件位于以下目录中:
    - NT drive:\WebSphere\CommerceServer\schema\xml

- 2000 drive:\Program Files\WebSphere\CommerceServer\schema\xml
- AIX /usr/WebSphere/CommerceServer/schema/xml
- Solaris /opt/WebSphere/CommerceServer/schema/xml
- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 定位列有可用计算方法（CALMETHOD）的部分。
– 使用税款的 calusage_ID 值（-3 代表销售税，-4 代表装运税）定位计算方法。
– 定位子类为 9 的计算方法。有若干个计算方法的子类是 9。从中选取一个适合您需要的方法。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALSCALE 表。

b. 对您的商店中使用的每个计算标度重复步骤 a。
c. 使用以下示例作为指南，在您的 XML 文件中为 CALSCALDS 表添加计算标度描述信息。如果您正创建多文化的商店，则应当将此信息包含在特定于语言环境的 XML 文件中。

```
<calscaleds
calscale_id="@calscale_id_19"
description="Sales Tax 5% "
language_id="&en_US"
/>
```

其中

- calscale_id 是此描述适用的计算标度。
- description 是计算标度的简短描述，适合显示给顾客以解释计算是怎样进行的。例如，“每千克 ￥.80，最少收费 ￥40.00。”或“5 个或 5 个以上 9 折。”
- language_id 是显示此信息的语言。

d. 对您的商店中使用的每个计算标度重复步骤 c。

10. 定义计算标度的计算范围。
    a. 使用以下示例作为指南，在您的 XML 文件中为 CALRANGE 表设置计算范围。

```
<calrange
calrange_id="@calrange_id_37"
calscale_id="@calscale_id_19"
calmethod_id="-59"
rangestart="0.00000"
cumulative="0"
/>
```

其中

- calrange_id 是生成的唯一标识。
- calcode_id 是计算标度，此计算范围是其组成部分。
- calmethod_id 是 CalculationRangeMethod，它确定来自 CalculationRangeLookupResult 的货币金额。例如，

FixedAmountCalculationRangeCmd、PerUnitAmountCalculationRangeCmd 或 PercentageCalculationRangeCmd。要确定 CalculationRangeMethod，请执行以下操作:

– 请参阅 WebSphere Commerce 联机帮助中的 CALMETHOD 表。请参阅 SUBCLASS 列的描述。单击 “CALMETHOD 表: 详细信息” 的链接。此表列出了可用的 CALMETHOD 类型。CalculationRangeMethod 是 10。

– 打开引导程序文件 wcs.bootstrap_xx_XX.xml，其中 xx_XX 是语言环境的代码。引导程序文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\schema\xml
- 2000 drive:\Program Files\WebSphere\CommerceServer \schema\xml
- AIX /usr/WebSphere/CommerceServer/schema/xml
- Solaris /opt/WebSphere/CommerceServer/schema/xml
- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 定位列有可用计算方法（CALMETHOD）的部分。

– 使用税款的 calusage_ID 值（-3 代表销售税，-4 代表装运税）定位计算方法。

– 定位子类为 10 的计算方法。有若干种计算方法的子类为 10。请选择满足您需要的一种。

- rangestart 表示如果查找号码大于或等于 RANGESTART，或者 RANGESTART 是 NULL，则此行与查找号码匹配。
- cumulative 如下:
  – 0 = 仅使用与最高 RANGESTART 值匹配的 CalculationRange。
  – 1 = 使用所有匹配的 CalculationRange。将已计算的货币金额相加得到最后的结果。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALRANGE 表。

b. 对与商店中使用的计算标度关联的每个计算范围重复步骤 a。由于所有金额都以相同的税率征税，所以在以上例子中只有一个范围。

11. 定义计算标度的计算查找值。计算查找值是与计算标度关联的值。例如，计算标度包含以下范围以及关联的税率（对旅馆提供的餐饮征收的安大略地方销售税）:

- $0.00 — $3.99 以 0.00% 税率征税
- $4.00 及以上以 8.00% 税率征税

查找值是 0.00 和 8.00。

a. 使用以下示例作为指南，在您的 XML 文件中为 CALRLOOKUP 表设置计算查找。

```
<calrlookup
 calrlookup_id="@calrlookup_id_37"
 calrange_id="@calrange_id_37"
 value="5.00"
/>
```

其中

- calrlookup_id 是生成的唯一标识。
- calrange_id 是计算范围，此计算范围查找结果是其组成部分。
- value 是计算范围查找结果的值，它由计算范围的计算范围方法使用以确定货币结果。在此例中，税率是 5.00%。

关于更多信息，请参阅 WebSphere Commerce 联机帮助中的 CALLOOKUP 表。

b. 对与商店中使用的计算标度关联的每个查找值重复步骤 a 和 b。在此例中，由于 CALRLOOKUP.SETCCURR 为 NULL，所以只有一个 CALRLOOKUP 值；并且由于所有金额的税率是一样的，所以只有一个 CALRANGE。

12. 将计算规则与计算标度关联。

a. 使用以下示例作为指南，在您的 XML 文件中为 CRULESCALE 表将计算标度与计算规则关联。

```
<crulescale
 calrule_id="@calrule_id_10"
 calscale_id="@calscale_id_19"
/>
```

其中

- calrule_id 是计算规则。
- calscale_id 是计算标度。

b. 对每个计算标度和规则的关联重复步骤 a。在以上示例中，每个计算规则只有一个计算标度。

注: 如果税率依购买金额而发生变化，则您将需要创建范围起始值非零的标度。那样，对于您未建立转换率（请参阅 CURCONVERT 表）的每种支持的货币（将 CALSCALE.SETCCURR 设置为相应的货币）创建计算标度，并且将它们全部与该特定税类别的计算规则关联。例如，对 $4.00 以下的餐饮不征收安大略地方税。如果您的商店支持以美元销售餐饮，则您需要建立从美元到加元的转换，或者以适当的范围起始值（可能是 $6.00 USD）创建单独的税款计算标度，并将其与相同的税款计算规则关联。根据订单的货币，将只使用相应的计算标度。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

创建税款实现有用资源

为使您的税款有用资源在商店中正确工作，您必须将商店中的税款地区组关联到商店使用的实现中心，然后将计算规则同时关联到税款地区组和实现中心。

您必须先创建实现中心有用资源，才能将税款有用资源关联到实现中心。关于创建实现有用资源的更多信息，请参阅第 99 页的『在 WebSphere Commerce 中创建实现有用资源』。

在您创建实现有用资源之后，请通过向 TAXJCRULE 表添加信息来将税款有用资源与其相关联。请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，税款）。

2. 复查用于为样本商店创建税款实现有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

   每个样本商店包括一个 taxfulfill.xml 文件，其中包括税款信息。要查看商店归档文件中的 taxfulfill.xml 文件，请使用 ZIP 程序将其解压缩。taxfulfill.xml 文件位于 data 目录中。
3. 请复查第 289 页的附录 B, 『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 taxfulfill.xml 文件或通过创建一个新文件来创建 taxfulfill.xml 文件。关于更多信息，请参阅相应于 taxfulfill.xml 的 DTD 文件。DTD 文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
5. 使用以下示例作为指南，在您的 XML 文件中为 TAXJCRULE 表添加信息:

```
<taxjcrule
taxjcrule_id="@taxjcrule_id_1"
calrule_id="@calrule_id_10"
ffmcenter_id="@ffmcenter_id_1"
jurstgroup_id="@jurstgroup_id_2"
precedence="0"
/>
```

   其中

   - taxjcrule_id 是生成的唯一标识。
   - calrule_id 是使用的计算规则。
   - ffmcenter_id 是实现中心。如果这为 NULL，则此关联将应用到所有的实现中心上。
   - jurstgroup_id 是税收地区组。如果这为 NULL，则此关联将应用到所有的税收地区组上。

- precedence 当对于同一个实现中心，送货地址落在多个指定的税收地区组中时，则只有 TAXJCRULE.PRECEDENCE 值最高的计算规则适用。

6. 对商店中每个地区组、实现中心和规则的关联重复步骤 3。

关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

创建商店—产品目录—税款有用资源

为了将税款与商店中的商品和服务关联，您必须对商店包含的每个合同将计算代码与商店中的产品目录条目关联。

您必须在可以创建“商店—产品目录—税款”有用资源之前，创建您的商店和产品目录有用资源。关于创建商店有用资源的更多信息，请参阅第 40 页的『在 XML 文件中创建商店数据有用资源』。关于创建产品目录有用资源的更多信息，请参阅第 69 页的『显示商店产品目录有用资源』。

要创建“商店—产品目录—税款”有用资源，请执行以下操作:

1. 复查《*IBM WebSphere Commerce 计算框架指南*》。WebSphere Commerce 计算框架计算与顾客选择购买的产品或服务相关联的货币金额（例如，装运费用）。
2. 复查用于为样本商店创建“商店—产品目录—税款”有用资源的 XML 文件。样本商店的所有文件都位于相应的商店归档文件中。

   商店归档文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   每个样本商店包含一个 store-catalog-tax.xml 文件，它包含装运实现信息。要查看商店归档文件中的 store-catalog-tax.xml 文件，请使用 ZIP 程序将其解压缩。store-catalog-tax.xml 文件位于 data 目录中。

   注: WebSphere Commerce 联机帮助包含关于样本商店中所包含的每个数据有用资源的信息。

3. 请复查第 289 页的附录 B，『创建数据』中的信息。
4. 通过复制样本商店归档文件中的一个 store-catalog-tax.xml 文件或通过创建一个新文件来创建 store-catalog-tax.xml 文件。关于更多信息，请参阅相应于 store-catalog-tax.xml 的 DTD 文件。DTD 文件位于以下目录中:

   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar

- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

5. 通过向 CATENCALCD 表添加信息，创建“商店—产品目录—税款”关系。使用以下示例作为指南:

```
<catencalcd
   calcode_id="@calcode_id_3"
   catencalcd_id="@catencalcd_id_3"
   store_id="@storeent_id_1"
/>
```

其中

- calcode_id 是计算代码。
- catencalcd_id 是生成的唯一标识。
- store_id 是商店。

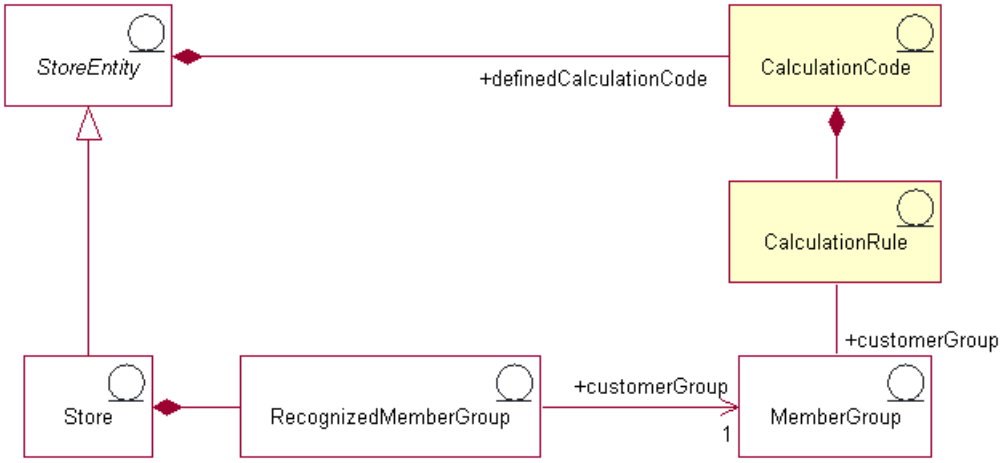
关于 @ 和 & 用法的更多信息，请参阅第 289 页的附录 B, 『创建数据』。

第 20 章 折扣有用资源

折扣让您能够向顾客提供价格刺激以促进其购买。您可以提供百分比折扣（如下调 10%）或固定金额的折扣（如降价 ￥15）。折扣可以适用于特定的产品或适用于整个购买。例如，您可以向老年顾客提供 20% 的折扣；如果库存中有许多红色棒球帽，则可以在一定时期内对这种帽子提供 25% 的折扣。

了解 WebSphere Commerce 中折扣

下图说明了 WebSphere Commerce Server 中折扣的结构。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

计算代码

*折扣*是使用折扣*计算代码*表示和计算的。折扣计算代码指示了如何为订购商品计算折扣。

计算代码属于商店实体。可以在商店实体内定义多个计算代码。如果删除了商店实体，则在该商店实体内定义的计算代码也将被删除。

每个折扣计算代码可以具有一个开始日期和一个结束日期，它们定义了时间周期，在此时间周期内折扣是有效的。折扣计算代码也可以与一个或多个成员组关联，它定义可选择的成员组。

折扣计算代码可以附加在一个或多个产品目录条目以及产品目录组上。将计算代码附加到产品目录组上的效果与将其直接附加到该产品目录组中所有产品目录条目上的效果相同。但是，如果产品目录组 A 包含产品目录组 B，则附加在产品目录组 A 上的计算代码不会附加在产品目录组 B 中的产品和商品上。

产品目录条目或产品目录组可以具有多个与它们关联的折扣。当对一个订单应用多个折扣计算代码时，将按照其计算代码顺序属性的升序执行折扣计算。

注: 定义折扣顺序以在折扣上实现折扣。

按照以下一种方式将订购商品分组以供计算:

- 每个贸易协议
- 每个产品
- 每个报价
- 每个送货地址

关于更多信息，请参阅 WebSphere Commerce 联机帮助。

关于使用计算代码的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

计算规则

每个计算代码具有一组计算规则，这些规则定义了执行计算的条件。每个折扣计算规则与一个或多个成员组关联（折扣对该成员组有效）。成员组一次可以选择多个折扣。

注: 如果可选择的成员组是在计算代码级别定义的，它不必在计算规则级别再次定义。

关于使用计算规则的更多信息，请参阅《*IBM WebSphere Commerce 计算框架指南*》。

在 WebSphere Commerce 中创建折扣有用资源

在用 WebSphere Commerce 创建的商店中创建折扣的主要方法是使用 WebSphere 贸易加速器中的“折扣”向导。关于使用 WebSphere 贸易加速器创建折扣的更多信息，请参阅 WebSphere Commerce 联机帮助。

还可以通过以下方式创建折扣: 先使用一个 XML 文件创建折扣，然后由装入程序软件包装入或由“商店服务”发布。然而，虽然以这种方法创建的折扣以及在从先前版本迁移过程中导入的折扣将具有正常功能，但是在 WebSphere 贸易加速器中可能会无法正常显示。

第 21 章 库存有用资源

库存包含可以在实现中心中实际计算的任何东西。对于可以实现的库存类型有特定的定义，例如商品、产品、库存标识、捆绑销售商品以及打包销售商品；但是我们把这些都当作库存。在“产品”向导和“产品”笔记本中配置产品以供实现。这包含若干选项，有跟踪库存、允许延迟交货、强制延迟交货、分别发货以及指定产品不可以退回。WebSphere 贸易加速器区分可接收的两种主要类型的库存:

- 预期库存（它具有关联的预期库存记录）。
- 特别库存，或者未记录为预期库存的库存。

预期库存是从供应商处接收的，并且通常用购买订单来支付。WebSphere 贸易加速器跟踪带有预期库存记录的预期库存，并允许您记录外部标识，通常是来自外部系统的购买订单号。在这种方式下，您可以轻易的跟踪已订购的库存以及已到货和未到货的库存。预期库存详细信息是有关预期库存记录中产品的具体信息，例如实现中心（它预期此产品）、预期接收日期、期望的数量以及注释。

一旦已根据预期库存记录接收其库存，则不能将其删除；一旦已接收该预期库存中任何库存，则不能更改或删除预期库存记录的详细信息。

当对实现中心可提供的库存下订单时，订单系统将库存分配给这些订单。将库存分配给订单后将使库存不再对订单系统可用。如果取消订单，库存又变为可用。如果为不可用的库存下订单，则可以创建延迟交货订单。如果有可以用于实现延迟交货订单的预期库存，则将预期库存分配给延迟交货订单，并可以向顾客提供预期的装运日期。

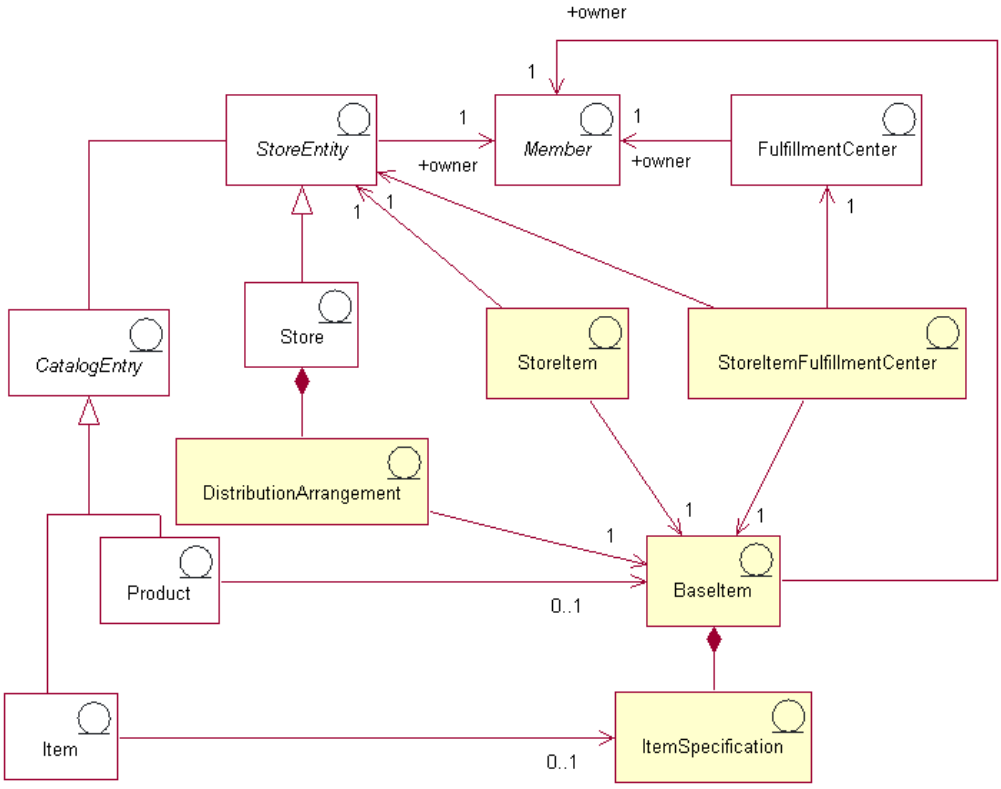
特别库存接收在库存到达实现中心而没有相应的预期库存记录时创建。这可能是因为意外的库存到达，或者可能是因为商家或销售商选择了不使用预期库存记录来记录库存接收。

注: 不论库存接收是预期的还是特别的，产品都必须在 WebSphere Commerce 系统中存在以便接收。

了解 WebSphere Commerce 中的库存有用资源

为了了解库存有用资源，有必要了解库存和商店之间的关系。这可以使用信息模型来解释。以下部分描述了库存与商店和其它有用资源之间的关系和关联。下图描述了 ATP 和非 ATP 库存的关系和关联。非 ATP 库存是产品的先前版本处理库存的方式，如果商店选择不使用 ATP 功能的话，则仍可以使用它。每张图与它的关联描述如下。关于 ATP 的更多信息，请参阅联机帮助。

ATP 库存



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

基本商品

*基本商品*是库存图的中心，它代表具有公共名称和描述的商品的一般系列。基本商品专用于实现，并且不是特定于任何商店的。为了实现的目的，每个产品目录条目（它代表产品目录中的一个产品）都具有一个相应的基本商品。基本商品在 BASEITEM 表中定义。

商品规格

*商品规格*是一个基本商品，它带有为其所有属性定义的值。为了实现的目的，每个产品目录条目（它代表产品目录中的一个商品）都具有一个相应的商品规格。

产品目录条目

产品和商品都是*目录条目*。产品目录条目与商店实体关联，这意味着产品目录条目（例如产品和商品）可以在商店中找到。

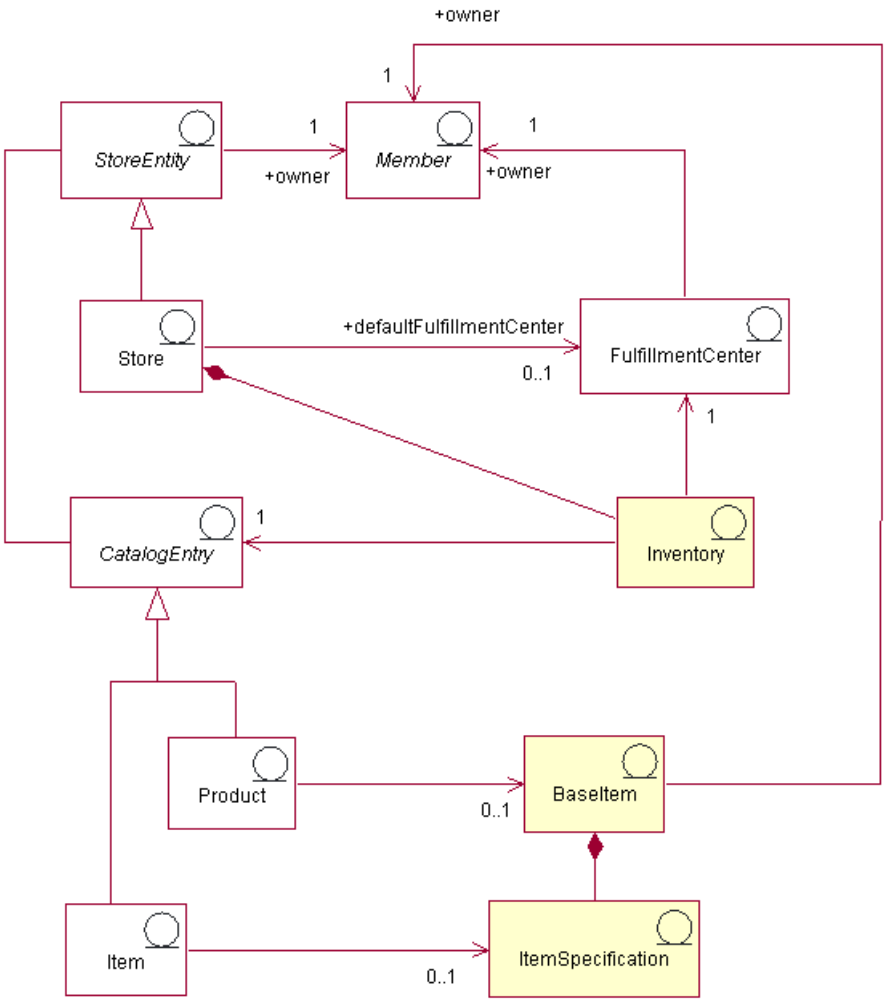
分发安排

*分布安排*与一个基本商品关联，以使商店能够销售它自己的库存。分发安排存储在 DISTARRANG 表中。

商店商品

*商店商品*代表一些属性，这些属性影响了特定商店或商店组为特定基本商品的指定商品分配库存的方式，包括是否允许延迟交货以及是否允许跟踪库存。STORITMFFC 表定义了从发布订购商品以供实现开始算起直至订购商品装运到顾客为止，其间估计经过的时间（以秒为单位）。仅当商店希望定义一个值覆盖商店商品的 FFMCENTER 缺省装运偏移量时，才填充此表。

非 ATP 库存



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

基本商品也是非 ATP 库存图的中心。基本商品与产品、商品和产品目录条目之间的关系与常规库存图中的关系相同。基本商品仍然由一个成员所有，并且一旦该成员定义了基本商品，则该基本商品就可以在商店中销售。然而，在这种情况下，将没有分发安排、商店商品关联或者商店商品实现中心。

实现中心

库存与一个*实现中心*和一个商店关联。商店可以指定一个缺省的实现中心。与基本商品相同，实现中心由成员所拥有，并与此商店共享该所有权。关于实现有用资源的更多信息，请参阅第 97 页的第 11 章，『实现有用资源』。

关于 WebSphere Commerce Server 中库存有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的库存对象和数据模型。

在 WebSphere Commerce 中创建库存有用资源

因为库存是可操作的数据，所以当顾客从您的商店中购买产品或退回产品时，库存每天都会更改。因此，当您销售产品时以及实现中心从供应商处接收新的库存时，您的库存级别会上下浮动。WebSphere 贸易加速器允许您完成以下与库存有关的任务:

- 记录预期库存
- 从供应商处接收预期库存和特别库存
- 调整库存
- 维护退货记录
- 维护退货原因
- 从顾客处接收退回的库存
- 管理退回库存的处理

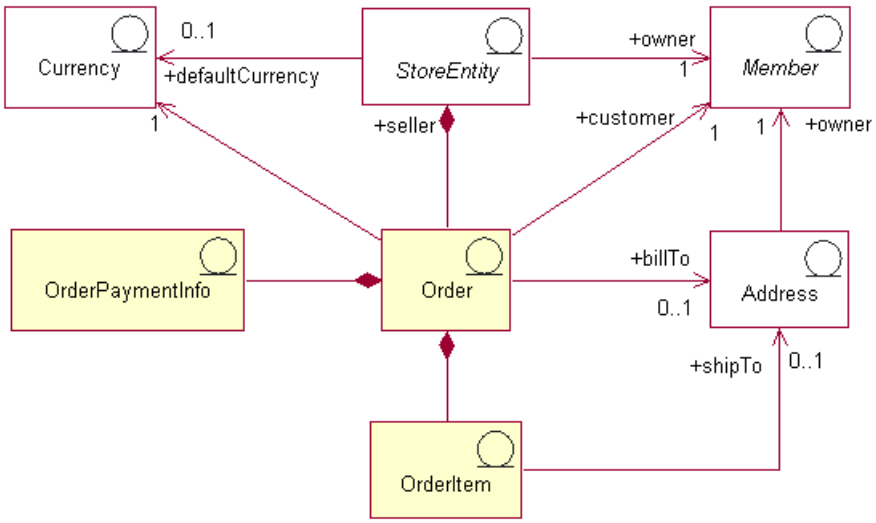
关于使用 WebSphere 贸易加速器管理库存的更多信息，请参阅 WebSphere Commerce 联机帮助。

第 22 章 订单有用资源

WebSphere Commerce 系统中的订单有用资源提供了购物车、订单管理以及订单处理功能。订单处理功能包含快速订购或购买、固定订购、多个未决订单、重新订购以及分割订单和延迟交货订单。相关的服务（例如定价、税务、支付、库存和实现）也是订单有用资源的组成部分。

了解 WebSphere Commerce 中的订单有用资源

下图说明了 WebSphere Commerce Server 中的订单有用资源。每个有用资源的描述在此图之后。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。

订单和订购商品

在 WebSphere Commerce 系统中，对于顾客或购物者而言，*订单*是选定产品的列表（例如订单可以包含两本书和一张 CD），而该列表上的每个产品是一个*订购商品*（例如，每本书和每张 CD 都是同一个订单的订购商品）。当顾客在商店中下了一个订单时，此顾客必须提供开票地址，商店会把发票发送到此地址。每个订单有一个单一货币标识与之关联。从商店角度而言，订单是订购商品的列表。它是商店数据的组成部分。

货币

商店可以用一种*货币*来显示价格，或者也可以使用多种货币。每个商店还必须定义一个*缺省货币*。您还可以允许顾客选择*购物货币*。如果购物货币与商店的缺省货币相同，则它在 STOREENT 表中已受到支持。如果购物货币不是商店的缺省货币，则您必须将此货币添加到 CURLIST 表。顾客使用购物货币在您的商店中下订单。

支付信息

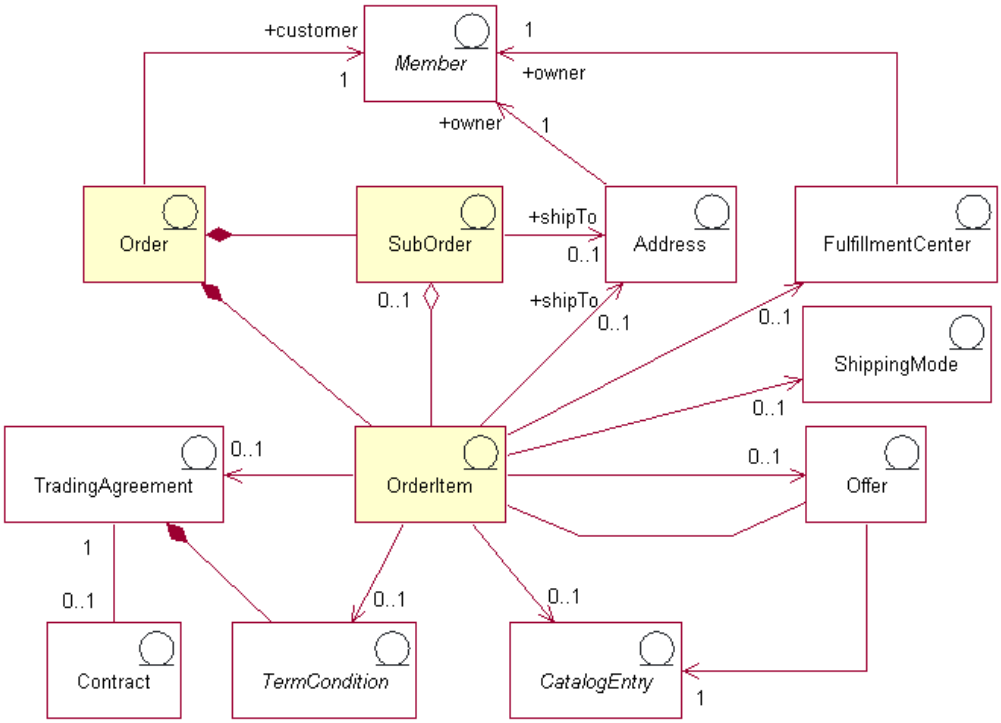
一旦顾客选择了首选购物货币，则所有的支付将使用该货币来处理。根据商店的支付支持和策略，顾客可以使用联机支付（这种情况下，顾客通过因特网在商店的站点上提供支付信息）或脱机支付（这种情况下，顾客不通过因特网通道提供支付信息，例如可通过电话或传真）来支付购买的商品。不论是采用联机支付方式还是脱机支付方式，顾客必须在下订单时提供*支付信息*，包含以下任何信息:

- 支付方式: 顾客支付订单的方式。根据在商店的 Payment Manager 中配置的支付卡匣，您可以设置商店以接受脱机支付、联机支付的电子货币、联机支付的 SET Secure Electronic Transaction™ 和商家发起的授权（MIA）（它们不需要顾客使用联机电子钱包），或者定制的支付方式。
- 对于信用卡支付，有关信用卡的信息: 顾客用于支付订单的信用卡的品牌、号码和失效日期。如果商店支持联机支付，则通常需要信用卡信息。
- 购买订单号码: 购买订单的号码，顾客在商店中订购时可能已经提供了此号码。正如商店和顾客之间的合同中的条款所规定的，购买订单号码将顾客识别成一个已授权可从商店订购商品的人。

订购商品

*订购商品*是构成订单的独立的产品或商品。一个订单必须具有至少一个订购商品。每个订购商品代表顾客选择购买的一件物品。此外，每个订购商品都引用了贸易协议（通常是合同）、装运方式、实现中心以及报价。折扣、装运费用和总税款与每个订购商品存储在一起。

下图说明了 WebSphere Commerce 订购商品有用资源。每个有用资源的描述在此图之后。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

子订单

订购商品被分组形成*子订单*。子订单是订单中被交付到指定地址的那一部分。例如，顾客可以为购物车中的不同产品指定不同的送货地址。每个送货地址以及与其相关的产品即构成一个子订单。子订单中的订购商品具有相同的送货地址，而且可以用于显示其订购商品金额的小计。

OrderItem 对象的数量属性是一个无单位的数值，它可以与 CatalogEntryShippingInformation 对象（此对象与 CatalogEntry 对象关联）的额定数量属性相乘得到由 OrderItem 代表的实际数量。CatalogEntryShippingInformation 对象指定了表示数量的计量单位。

尽管订单通常与单个商店关联，但是订单简要表是一种特殊类型的订单，它可以与一个商店或一个商店组关联。在对象模型中，订单简要表由状态为 “Q” 的订单来代表。订单简要表存放有关顾客的缺省信息，例如支付信息、送货地址、装运方式和开票地址。

其它订购商品有用资源

一个订购商品可以与以下对象中的一个关联，也可以不与以下对象中的任何一个关联。

- 顾客的送货地址（此顾客下的订单中包含订购商品）。顾客必须在订购过程中指定一个送货地址，以便商店的实现中心可以使用此地址来相应地交付订购商品。
- 实现中心，它装运和接收顾客订单所需的订购商品，并且存储订购商品的库存。
- 订购商品的装运方式，它是装运递送者（它是提供从实现中心到顾客的装运服务的公司）和该装运递送者所提供的装运服务的组合。例如，“ABC 装运公司，隔夜服务”以及“ABC 装运公司，快递”都是装运方式。
- 与此订购商品关联的报价。通过将不同的报价包含在不同的价格列表（或“贸易状态容器”）中，商店可以对不同的顾客显示同一个产品或库存标识的不同价格。例如，旅行社可以在四个不同的价格列表中提供机票：成人定价、老人定价、儿童定价和学生定价。
- 订购商品的产品目录条目；即，每个订购商品从产品目录中订购一个商品。
- 贸易协议，它定义了条款和条件，在该条款和条件下才可以订购此商品。这通常是一个合同，但是也可能是一个报价请求（RFQ），它代表一个协商，直至提交此订单以供处理。

关于 WebSphere Commerce Server 中订单有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的订单对象和数据模型。

在 WebSphere Commerce 中创建订单有用资源

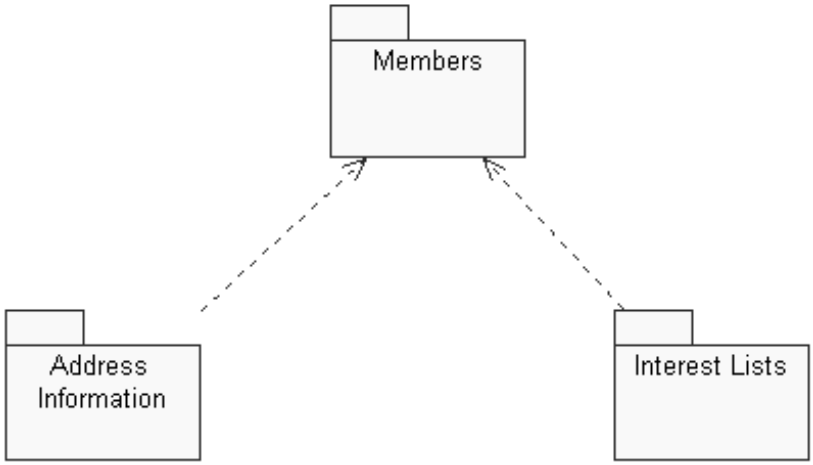
顾客可以从商店中下订单，或（使用 WebSphere 贸易加速器）请求商店的客户服务代表帮助完成此任务。要以 B2C 顾客的名义创建一个订单，请参阅 WebSphere Commerce 联机帮助主题“为注册顾客创建订单”和“为未注册顾客创建订单”。要以 B2B 顾客的名义创建一个订单，请参阅帮助主题“为商业用户创建订单”。

第 23 章 顾客和卖方有用资源

尽管商店可以包含参与商店活动的若干角色，但是在最少情况下，商店应该有一个顾客和一个卖方。

了解 WebSphere Commerce 中的顾客有用资源

*顾客*是这样一类人，他在商店中购物、创建兴趣列表（如果希望的话）、下订单并从商店或卖方购买商品。下图说明了顾客从商店中下订单所需要的有用资源。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

如前图中所示，WebSphere Commerce 系统包含成员。可以对每个用户成员和组织实体成员指定一个角色。

注: 在 WebSphere Commerce 中，*成员*可以是组织实体、用户或成员组。请参阅第 166 页的『成员』以获取更多详细信息。

在这种情况下，用户成员是顾客。顾客必须提供地址信息，并可以拥有兴趣商品列表。该图说明了成员（顾客）和与其关联的顾客有用资源之间的双向关系：顾客必须拥有并提供一个地址且可以拥有一个兴趣列表以在商店购物；而地址和兴趣列表取决于顾客的存在。

地址信息

当顾客从商店购买商品时，他必须提供三种类型的地址信息：联系地址、开票地址和送货地址。下面描述了这三种地址类型；每个地址可以是唯一的或者是相同的：

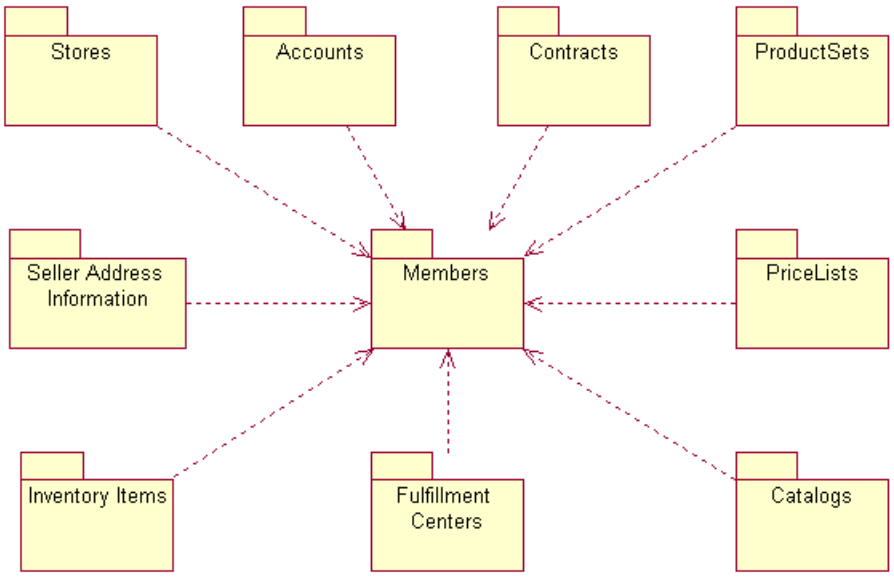
- 联系地址用于为了不同的目的通知顾客，例如关于订单的状态或对订单的更改，以及关于即将进行的商店事件（例如促销或商店维护）的通知。顾客的联系地址包含街道名称和门牌号码、市／县／区、省／直辖市、邮政编码、国家或地区、电子邮件地址、电话号码和传真号码。通常情况下，联系地址是最容易联系上顾客的地址，例如工作地址。
- 开票地址用于发送所购买商品的帐单或发票。开票地址包含街道名称和门牌号码、市／县／区、省／直辖市、邮政编码、国家或地区、电话号码和电子邮件地址。开票地址可以与联系地址或送货地址相同，也可以不同。
- 送货地址用于交付所购买的商品。送货地址包含街道名称和门牌号码、市／县／区、省／直辖市、邮政编码、国家或地区、电话号码和电子邮件地址。送货地址可以与联系地址或开票地址相同，也可以不同。

兴趣列表

商店可以支持*兴趣列表*。即，顾客将未来可能希望订购的产品添加到他们的兴趣列表。兴趣列表不是购物车；兴趣列表可以包含来自多个商店的商品，并且不包含价格、送货地址、装运方式、库存可用性信息或计算的金额（例如折扣、装运费用和税款）。

了解 WebSphere Commerce 中的卖方有用资源

*卖方*除了跟踪商店销售之外，还在总体上监督商店目标和管理。卖方向顾客销售商品和服务。卖方角色等同于商家，他对所有 WebSphere 贸易加速器功能都有访问权。下图说明了卖方维护商店以及向顾客销售时所需要的有用资源。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

如前图中所示，WebSphere Commerce 系统包含成员。每个成员被指定一个角色，例如商店的客户服务代表或在仓库的库存收货员。为了向顾客销售，卖方角色可以维护以下有用资源:

- 商店
- 帐户（可选）
- 合同（或至少是 WebSphere Commerce 缺省合同）
- 产品集
- 价格列表
- 产品目录
- 实现中心
- 库存商品

前图说明了成员（卖方）和卖方有用资源之间的关系: 即，卖方可以具有以上列出的有用资源以维护商店，而这些有用资源需要有一个卖方来部署它们。

商店

WebSphere Commerce 网上*商店*包括一系列 HTML 和 JavaServer Pages 文件，以及税款、装运费用、支付、产品目录和其它数据库有用资源（它们包含在商店归档文件中）。商店还包含商店数据，此数据是填充到 WebSphere Commerce 数据库中以使商店能运作的信息。

关于 WebSphere Commerce 商店的更多信息，请参阅第 39 页的第 6 章，『商店有用资源』和第 33 页的第 4 部分，『开发商店数据』。

帐户

商店可以设置顾客的商业*帐户*以允许他们从商店中购买商品。帐户包含以下信息:

- 帐户名称，它通常是此顾客关联的组织的名称。此组织已与商店之间签订了合同，规定了有关顾客在商店中购物的条款。例如，组织 IBM 可能具有与 “ABC 办公用品公司” 的合同。
- 代表名称，这是在负责此帐户的卖方组织内部的代表组织名称。
- 属于此帐户的合同的数目。

关于 WebSphere Commerce 帐户的更多信息，请参阅第 88 页的『帐户（商业帐户）』和 WebSphere Commerce 联机帮助。

合同

通常情况下，在 WebSphere Commerce 中，所有的顾客都必须在*合同*下进行购物。顾客和卖方之间的每个帐户都必须与一个或多个合同关联（为了使未注册顾客或顾客能在商店购物，或为了顾客能购买其它合同未涵盖的产品，至少要关联一个*缺省合同*）。合

同让顾客能够在合同中规定的条款和条件及业务策略下，在指定的时间内以指定的价格从商店购买产品。卖方部署合同，以便顾客可以从商店购买商品。

关于卖方可以使用的 WebSphere Commerce 合同和缺省合同的更多信息，请参阅第 88 页的『合同』。

产品集

*产品集*为卖方提供了一种机制将在线产品目录分类成逻辑子集，以便卖方可以让不同的顾客使用不同的产品目录视图。此外，卖方还可以为顾客创建合同，并规定顾客仅可以在预定义的产品集下购买产品。

关于 WebSphere Commerce 产品集的更多信息，请参阅第 52 页的『产品集』。

价格列表

*价格列表*与卖方提供的价格关联，或与向顾客显示的价格关联。对于同样的产品，卖方可以对不同的顾客列出不同的价格。在 WebSphere Commerce 中，报价也称为*贸易状态*，它代表一个产品目录条目的价格以及顾客为了有资格使用该价格而必须满足的条件。

在 WebSphere Commerce 中，报价对象是成员所拥有的 TradingPositionContainer 的组成部分。TradingPositionContainer 包含 TradingPosition，并可以通过贸易协议或合同对所有的顾客都可用，或者仅对某组中的顾客可用。有时候，TradingPositionContainer 称为价格列表。有两种价格列表：标准价格列表（它包含商店产品目录中产品的基本价格）或定制价格列表（它指定了产品的列表及其定制的价格）。

关于 WebSphere Commerce 价格列表的更多信息，请参阅第 79 页的第 9 章，『对有用资源进行定价』。

产品目录

WebSphere Commerce 商店至少使用一个在线*产品目录*来陈列卖方为销售而提供的商品和服务。通常情况下，一个在线产品目录包含待售的商品的价格、图像和描述。在线产品目录还可将商品显示为不同的类别以为浏览提供便利。

WebSphere Commerce 系统中的每个商店都必须具有*主产品目录*（它用于产品目录的管理）。主产品目录是管理卖方商品的中心位置；它是一个单一产品目录，包含所有产品、商品和关系以及商店中待售的一切物品的标准价格。如果卖方有多个商店，则主产品目录可以在这些商店之间共享。

关于 WebSphere Commerce 产品集的更多信息，请参阅第 49 页的第 8 章，『产品目录有用资源』。

实现中心

商店将*实现中心*同时作为库存仓库以及装运和接收中心来使用。卖方可能具有一个或多个实现中心。

从 WebSphere Commerce 服务器的角度看，FulfillmentCenter 对象是独立于 Store 对象的。它管理产品库存和装运。要装运一个订单，实现中心依靠由顾客指定的 ShippingMode 对象。ShippingMode 对象为了实现订单而指示装运递送者和装运的方法。

在实现中心中，ShippingArrangement 对象指示 Store 对象已安排好与 FulfillmentCenter 对象一起使用某个 ShippingMode 来装运产品。

关于 WebSphere Commerce 实现中心的更多信息，请参阅第 97 页的第 11 章，『实现有用资源』和第 119 页的第 18 章，『装运有用资源』。

库存商品

*库存商品*包含了可以在卖方实现中心中实际计算的任何东西。WebSphere Commerce 系统定义了可以实现的指定库存类型，例如商品、产品、库存标识、捆绑销售商品以及打包销售商品；但是这些是所有被考虑在内的库存。使用 WebSphere 贸易加速器的“产品管理”工具配置产品以供实现。

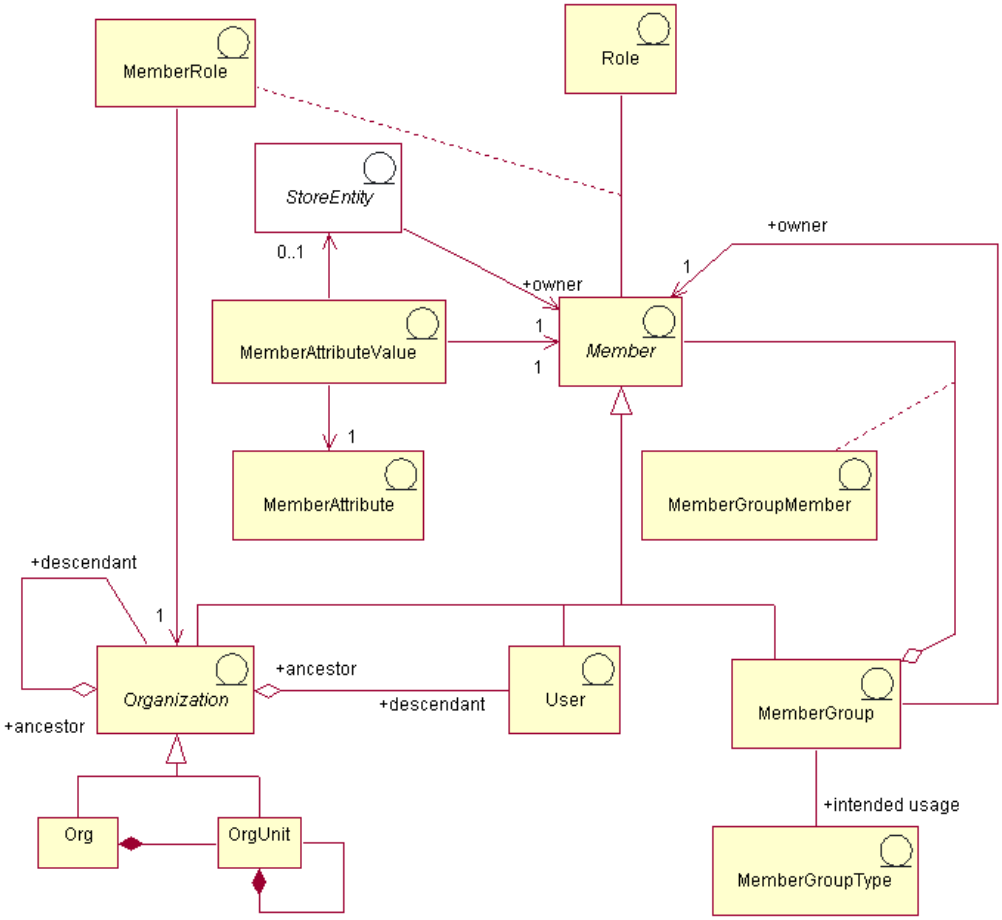
关于 WebSphere Commerce 库存商品的更多信息，请参阅 WebSphere Commerce 联机帮助和第 153 页的第 21 章，『库存有用资源』。

了解 WebSphere Commerce 中的成员有用资源

WebSphere Commerce 成员有用资源包含了 WebSphere Commerce 系统参与方的数据。成员可以是一个用户、一组用户或一个组织实体。管理员（例如站点管理员）对用户和组织实体成员指定角色。一旦给成员指定角色，访问控制组件就授权成员可参与到活动中。例如，组织可以是买方或卖方，或同时是两者。也可以给用户指定多个角色。管理员可以创建成员组，成员组是为了不同商业原因分类而成的用户组。使用 WebSphere Commerce 管理控制台创建和处理组织、用户、角色和成员组。

成员有用资源的业务逻辑提供了成员注册和简要表管理服务。与成员有用资源紧密相关的其它服务包含访问控制、认证和会话管理。关于这些主题的更多详细信息，请参阅 WebSphere Commerce 联机帮助。

下图说明了 WebSphere Commerce 成员有用资源。在图之后是每个有用资源的描述。



本图和商店数据部分中的其它所有图都是 WebSphere Commerce Server 信息模型的一部分。关于信息模型的更多信息，请参阅第 25 页的『商店数据信息模型』。关于本图中所使用的约定的更多信息，请参阅第 287 页的附录 A，『UML 图注』。

成员

WebSphere Commerce 中的*成员*可以是以下任意一个:

- *组织实体*。这可以是组织（例如 IBM）或大型组织中的组织单位（例如 IBM 中的“电子交易分部”）。
- *用户*（已注册的或未注册的）。注册用户具有唯一的标识和密码，且为了注册的目的，需要提供简要表数据。可以根据其简要表类型将注册用户分类: 类型 ‘B’ 表示商业用户（或 B2B 顾客），类型 ‘C’ 表示零售用户（或 B2C 顾客）。关于注册用户和未注册用户的更多信息，请参阅 WebSphere Commerce 联机帮助中的“成员”。
- *成员组*。这是为了不同商业原因分类而成的用户组。分组可以用于访问控制目的、核准目的，以及市场营销目的（例如计算折扣、价格和显示产品）。

每个*商店实体*（即，商店或商店组）由一个成员拥有。

成员属性

WebSphere Commerce 成员具有一系列*属性*，每个属性具有一个与其关联的*值*。成员的基本用户简要表包含了注册信息、情况调查、地址信息、购买历史和其它杂项属性。

商业用户简要表包含与基本用户简要表相同的信息，另外还有职业信息（例如员工编号或职务，或者工作描述）。在注册期间，商业用户应当标识他们所属的商业组织。组织实体的简要表包含诸如组织名称和业务类别之类的附加信息。

访问控制规则强制实施了执行简要表管理的用户权限。成员简要表可以包含各种个人和业务相关的属性（例如角色、支付信息、地址、首选语言和货币以及普及计算设备）。属性可以是区别商店的。对成员组不支持这些属性。

角色

每个用户成员可以在组织中执行一个或多个角色。站点管理员对每个用户成员指定一个或多个角色。例如，作为 IBM 组织的一个成员，John Smith 的角色是客户服务代表，这意味着 John 代表 IBM 顾客来执行任务，并帮助他们查询或关注其注册信息、订单或退货。John 还可能具有客户服务主管的角色，该角色对上述任务负全责，并具有核准和监督其它客户服务代表的权力。

WebSphere Commerce 系统提供了以下一组缺省角色类型:

- 站点和内容开发角色
- 技术操作角色
- 市场营销管理角色
- 产品管理角色
- 业务关系管理角色
- 后勤和操作管理角色
- 组织管理角色

关于这些角色中的每一个的详细信息，请参阅 WebSphere Commerce 联机帮助主题“角色”。站点管理员可以按组织实体指定这些角色以及任何由该站点管理员创建的新角色；即，属于一个组织实体的用户可以担当指定给该组织实体的角色。

为用户指定了角色时，该角色将在组织实体范围内起作用。当对用户指定一个角色时，此用户并不是必须要为他所属的组织实体执行该角色；即，当管理员执行指定时，可以选择该用户为哪个组织实体执行该角色。如果管理员选择根组织，则该用户对所有组织实体扮演该角色。

关于 WebSphere Commerce 中成员有用资源的结构的更多详细信息，请参阅 WebSphere Commerce 联机帮助中的成员对象和数据模型。

在 WebSphere Commerce 中创建成员有用资源

为了创建卖方（充当商店所有者的组织）并维护有关卖方的信息，请使用管理控制台。关于更多信息，请参阅 WebSphere Commerce 联机帮助主题“创建组织”。

顾客不是由商店开发者创建的；当顾客在商店中注册时，注册信息是由 WebSphere Commerce 系统收集和维护的。

第 5 部分 向商店添加访问控制

第 24 章 商店中的访问控制

WebSphere Commerce 允许您通过访问控制来确定特定用户（无论是顾客还是管理员）可以执行的任务。本章着重描述了如何在商店中添加访问控制，以便限制您的顾客可以看到的页面以及他们在商店中可以执行的任务。

关于 WebSphere Commerce 中访问控制模型以及如何在站点级别应用访问控制的更多信息，请参阅《*IBM WebSphere Commerce 访问控制指南*》。关于在定制代码中实现访问控制的更多信息，请参阅《*IBM WebSphere Commerce 程序员指南*》。这些指南可以从以下 URL 获得:

Business

`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

理解 WebSphere Commerce 中的访问控制

关于 WebSphere Commerce 访问控制模型的详细信息包含在《*IBM WebSphere Commerce 访问控制指南*》中。但是为了理解访问控制是如何影响商店开发的，这里也提供了简单的摘要。

WebSphere Commerce 中的访问控制由以下元素组成: 用户、操作、资源和关系。

- 用户是使用系统的人。出于访问控制目的，用户必须分组到相关的访问组中。例如，在您的商店中可能将用户分组到以下访问组中: 注册顾客、临时顾客或诸如客户服务代表的管理组。
- 操作是用户可对资源执行的活动。出于访问控制目的，操作也必须分组到相关的操作组中。例如，商店中使用的常用操作是视图。视图供调用向顾客显示商店页面。商店中使用的视图必须指定到操作组中。
- 资源是受到保护的实体。例如，如果操作是视图，则该操作的资源是类 `com.ibm.commerce.command.ViewCommand`。出于访问控制目的，资源被分组到资源组中。
- 关系是指用户和资源之间的关系。访问控制策略可能要求满足用户和资源之间的一种关系。

访问控制策略

访问控制策略授权访问组对 WebSphere Commerce 的资源执行特定操作，只要访问组中的用户满足对于资源的特定关系。

WebSphere Commerce 提供了两百个以上的缺省访问控制策略，这些策略已在实例创建期间装入。它们覆盖了广泛的常见业务活动，包括订单创建和处理以及贸易，

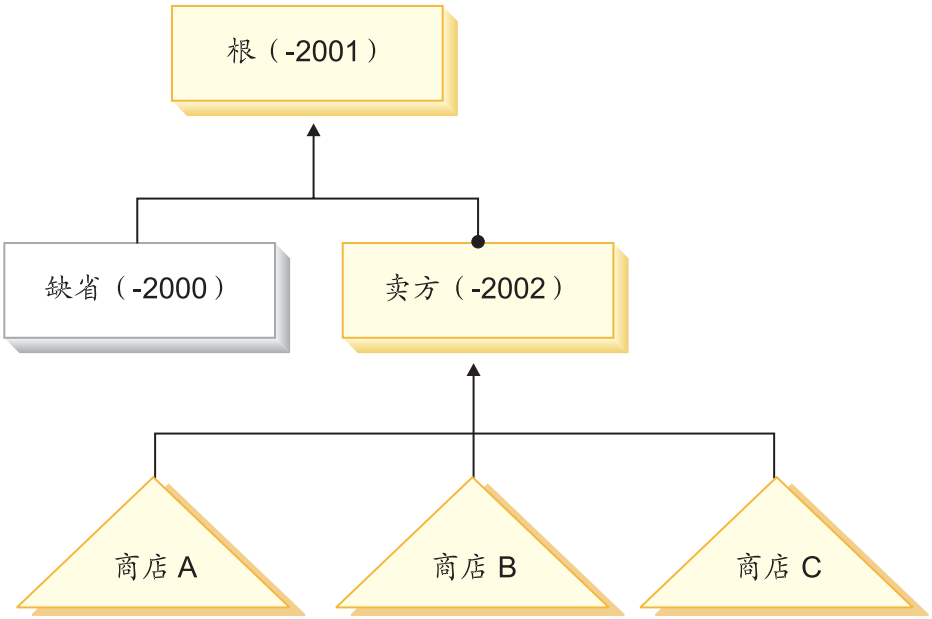
例如 Business 报价请求和 Business 合同。《*IBM WebSphere Commerce 访问控制指南*》中记述了缺省策略。

每个访问控制策略都属于一个组织实体。访问控制策略只能适用于由访问控制策略所有者所拥有的资源。

注: 从访问控制的角度来说，资源的所有权具有特殊的意义。所有资源都必须实现 `com.ibm.commerce.security.Protectable` 接口。在此接口上有一个方法是 `getOwner()`，它返回资源所有者的成员标识。例如，Order 实体 bean 是通过让其远程接口扩展 Protectable 接口而受到保护的资源。Order 的 `getOwner()` 实现是: 特定 Order 资源返回在其中下订单的商店的所有者。对于资源是命令的策略，例如，`com.ibm.commerce.command.ViewCommand`，`getOwner()` 的缺省实现是返回当前处于命令上下文中的商店的所有者。如果在命令上下文中没有任何商店，则使用根组织作为所有者。关于更多信息，请参阅《*IBM WebSphere Commerce 程序员指南*》。

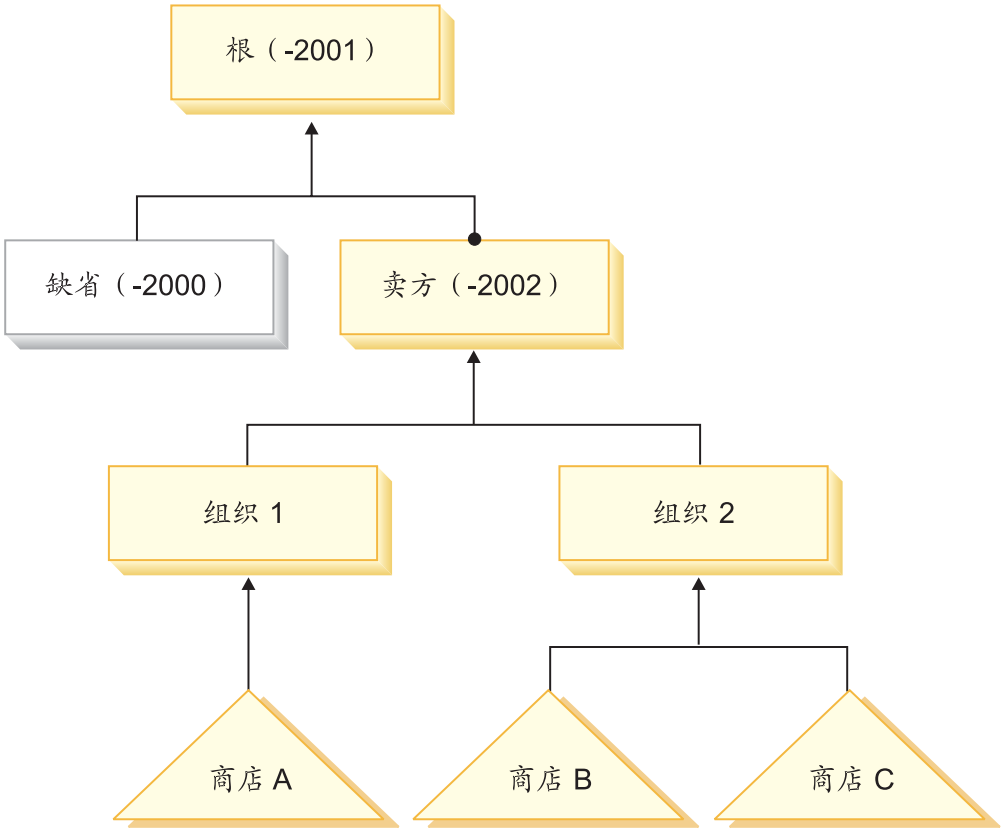
请看下图:

Professional



在 WebSphere Commerce 专业版中，根组织是最高级别的组织。它拥有所有其它组织实体。因此，根组织所拥有的访问控制策略便适用于站点中的所有资源。如果访问控制策略由卖方组织所有，则访问控制策略仅适用于由卖方所拥有的资源。

Business



在 WebSphere Commerce 商务版中，根组织是最高级别的组织。它拥有所有其它组织实体。因此，根组织所拥有的访问控制策略便适用于站点中的所有资源。如果访问控制策略由卖方组织所有，则访问控制策略仅适用于由卖方所拥有的资源。即，它将适用于组织 1 和 2（包括商店 A、B 和 C）所拥有的资源。如果访问控制策略由组织 1 所有，则它将仅适用于包括商店 A 的资源。如果访问控制策略由组织 2 所有，则它将适用于商店 B 和 C 中的资源。

Business 由于访问控制策略为组织实体所有，因此如果您正在站点中创建多个商店，并希望对单个商店应用不同的访问控制，则必须创建各自的组织拥有每个商店。

注: WebSphere Commerce 所提供的多数缺省访问控制策略都由根组织所有，且适用于站点中的所有资源。根组织所拥有的访问控制策略是指站点级别的策略。其它组织实体所拥有的策略称为组织级别的策略。

商店中的访问控制

WebSphere Commerce 中创建的所有商店都受到根组织所拥有的缺省访问控制策略的控制。商店中的资源也受到拥有商店的组织所拥有的任何访问控制策略的控制，且受到该组织上级组织的任何访问控制策略的控制。

缺省情况下，根组织拥有大部分访问控制策略。但是，如果您以 WebSphere Commerce 中提供的样本商店之一为基础创建商店，则将创建拥有该商店的组织所拥有的新访问控制策略。关于样本商店中访问控制策略的更多信息，请参阅第 174 页的『样本商店中的访问控制』。

当您创建自己的商店时，无论它是否基于样本，您都可能希望创建新的访问控制策略或修改现有策略，这些策略将只适用于该组织所有的商店。例如，如果您创建新的视图用来显示商店页面，则必须对这些视图指定访问控制策略。

组织级别的访问控制数据是在高级访问控制策略文件中定义的。这些文件定义了可由任何策略使用的可能的操作、操作组、资源、资源组和关系。它们也定义了特定于某特定组织的策略。与 WebSphere Commerce 一起提供的样本商店包含这些高级访问控制策略文件。以下部分说明了样本商店如何使用这些访问控制策略文件来定义组织信息。

样本商店中的访问控制

所有样本商店都包含高级访问控制策略文件，这些文件定义为商店特别创建的访问控制策略。拥有商店的组织拥有这些策略。下面讨论了为“新时尚”和 Business “多乐五金店”样本商店定义的访问控制策略。

为 Business “多乐五金店”创建的访问控制策略如下:

- AllUsersForToolTechExecuteToolTechAllUsersViews
- RegisteredApprovedUsersForToolTechExecuteToolTech RegisteredApprovedUsersViews

为“新时尚”创建的访问控制策略如下:

- AllUsersExecuteNewFashionAllUsersViews

这些访问控制策略确定哪些用户将看到样本商店中或基于样本的商店中的哪些视图。

这些策略的访问控制信息定义在两个高级访问控制策略文件中，这两个文件定义了样本商店中使用的可能的操作、操作组、资源、资源组和策略定义: *samplestorename*AccessPolicies.xml 和 *samplestorename*AccessPolicies_*locale*.xml。这些文件位于以下目录中:

- NT `drive:\WebSphere\CommerceServer\samples\stores\`*`samplestorename`*
- 2000 `drive:\ProgramFiles\WebSphere\CommerceServer\samples\stores\`*`samplestorename`*
- AIX `/usr/WebSphere/CommerceServer/samples/stores/`*`samplestorename`*
- Solaris `/opt/WebSphere/CommerceServer/samples/stores/`*`samplestorename`*
- Linux `/opt/WebSphere/CommerceServer/samples/stores/`*`samplestorename`*
- 400 `/QIBM/ProdData/WebCommerce/samples/stores/`*`samplestorename`*

  其中 *samplestorename* 是您的商店所基于的样本商店归档文件的名称，例如“新时尚”。

将样本商店作为商店归档文件封装之前，这两个高级访问控制策略文件会进行转换，产生两个 XML 文件，它们适于与装入程序软件包一起使用。然后这些转换过的 XML 文件封装在商店归档文件中，并用剩余的商店归档文件发布。

理解样本商店访问控制策略文件: 要理解访问控制是如何在商店级别添加的，请熟悉高级样本商店访问控制策略文件。以下示例取自 Business ToolTechAccessPolicies.xml 文件。

定义操作: Business ToolTechAccessPolicies.xml 文件的第一部分定义了商店中的新操作，现有访问控制策略中没有覆盖这些操作。在此情况下，操作即为商店中使用的所有视图。为了在商店中使用可直接从 URL 调用的视图来显示页面，或使用可以由其它命令通过重定向来启动的视图（相对于通过转发到视图来启动）来显示页面，必须将其定义为一项操作。请看以下示例:

```
<!-- [Start of Action definitions] -->
<!-- [this is the dictionary of possible actions -->
<Action Name="GenericApplicationError"
CommandName="GenericApplicationError">
</Action>

<Action Name="GenericSystemError"
CommandName="GenericSystemError">
</Action>

<Action Name="OrderOptionsView"
CommandName="OrderOptionsView">
</Action>

 <!--[End of Action definitions] -->
```

其中

- Action Name 是 XML 文件中用来引用此项操作的标号。在这些示例中，标号与视图名称相同。
- CommandName 是存储在 VIEWREG 表 VIEWNAME 列中的视图的名称。CommandName 将存储在 ACACTION 表的 Action 列中。

注: 如果某操作已经定义在 WebSphere Commerce 的缺省策略或任何其它策略中，则无须对每个使用该操作的策略重新定义。

定义资源类别: 此文件中的第二部分定义了资源类别。资源类别是指一类资源。为“多乐五金店”标识的资源类别是类 com.ibm.commerce.command.ViewCommand 和 com.ibm.commerce.tools.command.ToolsForwardViewCommand。

```
<!-- [Start of Resource Category definitions] -->
<!-- the dictionary of Protectable resources -->
<ResourceCategory Name="com.ibm.commerce.command.ViewCommandResourceCategory"
ResourceBeanClass="com.ibm.commerce.command.ViewCommand">
</ResourceCategory>
<ResourceCategory Name="com.ibm.commerce.tools.command.
ToolsForwardViewCommandResourceCategory"
ResourceBeanClass="com.ibm.commerce.tools.command.ToolsForwardViewCommand">
</ResourceCategory>
<!-- [End of Resource Category definitions] -->
```

其中

- ResourceCategory Name 标号用于引用 XML 文件中此资源类别。
- ResourceBeanClass 是类的名称。

注: 如果某资源类别已经定义在 WebSphere Commerce 的缺省策略或任何其它策略中，则无须对每个使用该资源类别的策略重新定义。上面示例中定义的资源类别已经在缺省策略中定义，但为了说明目的，此处我们再次定义。

定义操作组: 第三部分定义操作组。操作组是对文件第一部分中定义的操作的分组。在“多乐五金店”示例中，所有新用户视图都分组到 ToolTechAllUserViews 组中（该组将用在允许所有用户访问那些视图的策略中）或分组到 ToolTechRegisteredApprovedUserviews 中（它将用在仅允许注册用户访问那些视图的策略中）。

注: 您也可以在操作组中添加在 WebSphere Commerce 中其它地方定义的操作。如果在 WebSphere Commerce 中的其它地方定义，则这些操作不必在第 175 页的『定义操作』中讨论的“操作”列表中定义。

```
<!-- [Start of Action Group definitions] -->
<!-- Dictionary of grouped actions usable in policies -->
<!-- cross-component view-related action groups -->
<ActionGroup Name="ToolTechAllUsersViews"
OwnerID="RootOrganization">

<ActionGroupAction Name="UserRegistrationForm"/>
<ActionGroupAction Name="UserRegistrationErrorView"/>
<ActionGroupAction Name="GenericApplicationError"/>
<ActionGroupAction Name="GenericSystemError"/>
<ActionGroupAction Name="LogonForm"/>
</ActionGroup>

<!-- [End of Action Group definitions] -->
```

其中

- `ActionGroup Name` 是操作组的名称。
- `OwnerID` 是操作组的所有者。根组织必须是操作组的所有者。
- `ActionGroupAction Name` 是属于此组的操作的名称。ActionGroupAction Name 必须与第 175 页的『定义操作』的“Action Name”元素中定义的名称匹配。

定义资源组: 下一部分定义了资源组。资源组标识一组相关资源。

```
<!-- [Start of Resource Group definitions] -->
<!-- Dictionary of grouped resources usable in policies -->

<!-- Grouped resources permitting view execution,
 by any command extending com.ibm.commerce.command.ViewCommand
(with or without the Tools Framework) -->
<ResourceGroup Name="ViewCommandResourceGroup"  OwnerID="RootOrganization">

<ResourceGroupResource Name="com.ibm.commerce.command.ViewCommandResourceCategory"/>
</ResourceGroup>

</ResourceGroup> !-- [End of Resource Group definitions] -->
```

其中

- ResourceGroup Name 是资源组的名称。
- OwnerID 是资源组的所有者。根组织必须拥有资源组。
- ResourceGroupResource Name 是包含在该组中的资源类别名称。

定义策略: 最后一部分定义了商店中使用的新策略。

```
!-- [Start of Policy definitions] -->
!-- AllUsers for ToolTech can execute ToolTechAllUsersViews -->


<Policy Name="AllUsersForToolTechExecuteToolTechAllUsersViews"
OwnerID="MEMBER_ID"
```

```
UserGroup="AllUsers"
UserGroupOwner="RootOrganization"
ActionGroupName="ToolTechAllUsersViews"
ResourceGroupName="ViewCommandResourceGroup">
    </Policy>
!-- RegisteredApprovedUsers for ToolTech can execute
ToolTechRegisteredApprovedUsersViews -->

<Policy Name="RegisteredApprovedUsersFor
ToolTechExecuteToolTechRegisteredApprovedUsersViews"
OwnerID="MEMBER_ID"
UserGroup="RegisteredApprovedUsers"
UserGroupOwner="RootOrganization"
ActionGroupName="ToolTechRegisteredApprovedUsersViews"
ResourceGroupName="ViewCommandResourceGroup">
    </Policy>

!-- [End of of Policy definitions] -->
```

其中

- Policy Name 是定义的策略的名称。
- OwnerId 是策略的所有者。在此情况下，策略的所有者是拥有商店的组织。
- UserGroup 是策略所适用的用户组（访问组）。
- UserGroupOwner 是访问组的所有者。在此示例中，访问组的所有者与策略所有者不同。如果策略所有者和 UserGroupOwner 相同，则可以省略此元素。
- ActionGroupName 是策略所适用的操作组。
- ResourceGroupName 是策略所适用的资源组。

向商店添加访问控制

从商店开发角度来看，需要的最常见类型的访问控制是用于为商店创建的新视图和命令的访问控制。但是，您可能希望对商店添加其它类型的访问控制。关于对视图、命令和其它功能的访问控制的更多信息，请参阅《*IBM WebSphere Commerce 访问控制指南*》。在继续本指南中概述的以后步骤之前，请确保复查了《*IBM WebSphere Commerce 访问控制指南*》。

如果您在对基于样本商店的商店添加新的访问控制功能，请编辑现有高级访问控制策略 XML 文件。关于详细指导，请参阅第 180 页的『在商店归档文件中编辑访问控制文件』。如果您在对没有基于样本商店的商店添加访问控制，则需要创建新的高级访问控制策略 XML 文件，然后转换它。关于详细指导，请参阅『在商店中创建访问控制』。

在商店中创建访问控制

访问控制有用资源与商店中的其它有用资源不同，因为对于访问控制，您要创建两个高级访问控制 XML 文件，然后再转换它们。之后，转换好的 XML 文件可以使用装入程序软件包装入数据库，或者用在商店归档文件中。

要创建访问控制有用资源，请执行以下操作:

1. 复查用于为样本商店创建商店有用资源的高级 XML 文件: *samplestorename*AccessPolicies.xml 和 *samplestorename*AccessPolicies_*locale*.xml。这些文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\samples\stores*samplestorename*
- 2000 drive:\ProgramFiles\WebSphere\CommerceServer\samples\stores*samplestorename*
- AIX /usr/WebSphere/CommerceServer/samples/stores/*samplestorename*
- Solaris /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
- Linux /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
- 400 /QIBM/ProdData/WebCommerce/samples/stores/*samplestorename*

其中 *samplestorename* 是您的商店所基于的样本商店归档文件的名称，例如“新时尚”。

2. 请复查第 289 页的附录 B，『创建数据』中的信息。
3. 通过复制 *samplestorename*AccessPolicies.xml 文件之一或通过创建一个新文件来创建 *storename*AccessPolicies.xml 文件。关于更多信息，请参阅相应于 *samplestorename*AccessPolicies.xml 的 DTD 文件。DTD 文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\xml\policies\dtd
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\policies\dtd
   - AIX /usr/WebSphere/CommerceServer/xml/policies/dtd
   - Solaris /opt/WebSphere/CommerceServer/xml/policies/dtd
   - Linux /opt/WebSphere/CommerceServer/xml/policies/dtd
   - 400 /QIBM/ProdData/WebCommerce/xml/policies/dtd
4. 为商店支持的每种语言环境创建各自的高级 XML 文件。特定于语言环境的文件应当指定所有描述和显示名称信息，以便比较容易进行翻译。通过复制 *samplestorename*AccessPolicies_locale.xml 文件之一或者通过创建新的文件，对商店中的每种语言创建此文件，*storename*AccessPolicies_*locale*.xml。

   注: 即便您的商店仅支持一种语言，也将需要创建特定于语言环境的文件。
5. 在文件中添加适当的访问控制信息。关于更多信息，请参阅第 174 页的『样本商店中的访问控制』和《*IBM WebSphere Commerce 访问控制指南*》。
6. 将 *storename*AccessPolicies.xml 和 *storename*AccessPolicies_locale.xml 复制到以下目录:
   - NT drive:\WebSphere\CommerceServer\xml\policies\xml
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\policies\xml
   - AIX /usr/WebSphere/CommerceServer/xml/policies/xml
   - Solaris /opt/WebSphere/CommerceServer/xml/policies/xml

- Linux /opt/WebSphere/CommerceServer/xml/policies/xml
- 400 复制到任何用户数据目录。在现有 XML 文件中指定到 DTD 的全路径。访问控制 DTD 文件位于以下目录中: /QIBM/ProdData/WebCommerce/xml/policies/dtd

7. 运行 xmltransform 命令转换 *storename*AccessPolicies.xml。
   a. 在命令提示符下，转至以下目录:
      - NT drive:\WebSphere\CommerceServer\bin
      - 2000 drive:\ProgramFiles\WebSphere\CommerceServer\bin
      - AIX /usr/WebSphere/CommerceServer\bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin
   b. 然后输入: xmltransform -infile ..\xml\policies\xml\samplestorenameAccessPolicies.xml -transform ..\xml\policies\xsl\accesscontrol.xsl -outfile ..\xml\policies\xml\samplestorenameAccessPoliciesOut.xml

      400 TRNWCSXML INFILE (input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl /accesscontrol.xsl') INSTROOT(instance_root) OUTFILE(output_file)
   c. 检查以下日志文件以确保转换已成功完成:
      - NT drive:\WebSphere\CommerceServer\bin\xmltransform.db2.log
      - 2000 drive:\Program Files\WebSphere\CommerceServer\bin \xmltransform.db2.log
      - AIX /usr/WebSphere/CommerceServer/bin/xmltransform.db2.log
      - Solaris /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
      - Linux /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
      - 400 /QIBM/UserData/WebCommerce/instances/instancename /logs/TRNWCSXML.tx

        如果转换成功，则显示以下消息: "<DATE> <TIME> java.lang.Class main XMLTransformer Transform Successful"
8. 运行 xmltransform 命令转换 *storename*AccessPolicies_locale.xml。
   a. 在命令提示符下，转至以下目录:
      - NT drive:\WebSphere\CommerceServer\bin
      - 2000 drive:\ProgramFiles\WebSphere\CommerceServer\bin
      - AIX /usr/WebSphere/CommerceServer\bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin

b. 然后输入: `xmltransform -infile ..\xml\policies\xml\storenameAccessPolicies_locale.xml -transform ..\xml\policies\xsl\accesscontrolnls.xsl -outfile ..\xml\policies\xml\storenameAccessPoliciesOut_locale.xml`

c. 400 `TRNWCSXML INFILE(input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrolnls.xsl') INSTROOT(instance_root) OUTFILE(output_file)`

9. 对转换后的 XML 文件进行以下更改:

a. 在 *storename*AccessPolicesOut.xml 中，将开始和结束标记替换为:

```
<?xml version="1.0"?>
<!DOCTYPE accesscontrol-asset SYSTEM "accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

b. 在 *storename*AccessPolicesOut_locale.xml 中，将开始和结束标记替换为:

```
<?xml version="1.0" encoding="correct language code for the file"?>
<!DOCTYPE accesscontrol-asset SYSTEM "../accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

c. 在 *storename*AccessPolicesOut_locale.xml 中，将 @locale 替换为 &locale;，例如将 LANGUAGE_ID=″@zh_CN″ 更改为 LANGUAGE_ID=″&zh_CN;″

d. 在 *storename*AccessPolicesOut_locale.xml 中，将引用定位到 “acpoldesc” 表。除去 ACPOLICY_ID 值末尾的 @。例如，将 `"@AllUsersExecuteInFashionAllUsersViews@"` 更改为 `"@AllUsersExecuteInFashionAllUsersViews"`。

e. 在 *storename*AccessPolicesOut.xml 中，将 MEMBER_ID=″MEMBER_ID″ 替换为 MEMBER_ID=″&MEMBER_ID;″

f. 在 *storename*AccessPolicesOut.xml 中，将引用定位到 “acpolicy” 表。除去 ACPOLICY_ID 值末尾的 “@MEMBER_ID”。例如，将 `"@AllUsersExecuteInFashionAllUsersViews@MEMBER_ID"` 更改为 `"@AllUsersExecuteInFashionAllUsersViews"`

10. 此时您有两种选项:

- 使用装入程序软件包装入访问控制数据。关于更多信息，请参阅第 193 页的第 7 部分，『发布商店』。
- 在商店归档文件中添加访问控制数据。关于创建商店归档文件的更多信息，请参阅第 185 页的第 6 部分，『封装商店』。

关于 **@** 和 **&** 用法的更多信息，请参阅第 289 页的附录 B，『创建数据』。

在商店归档文件中编辑访问控制文件

WebSphere Commerce 为每个样本商店提供了两个预先转换好的访问控制 XML 文件，一个适用于所有语言（*samplestorename*AccessPolicies.xml），一个包含特定于语言环境的信息（*samplestorename*AccessPolicies_locale.xml）。您必须转换这些文件中的每个文件，这样就会产生两个 XML 文件，一个适用于所有语言（*samplestorename*AccessPoliciesOut.xml），一个包含特定于语言环境的信息（*samplestorename*AccessPoliciesOut_locale.xml）。

要编辑商店归档文件中的访问控制数据库有用资源，请执行以下操作:

1. 查找您用作商店基础的样本商店的预转换访问控制 XML 文件。这些文件的名称是 *samplestorename*AccessPolicies.xml 和 *samplestorename*AccessPolicies_locale.xml。缺省情况下这些文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\samples\stores*samplestorename*
   - 2000 drive:\ProgramFiles\WebSphere\CommerceServer\samples\stores*samplestorename*
   - AIX /usr/WebSphere/CommerceServer/samples/stores/*samplestorename*
   - Solaris /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
   - Linux /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
   - 400 /QIBM/ProdData/WebCommerce/samples/stores/*samplestorename*

   其中 *samplestorename* 是您的商店所基于的样本商店归档文件的名称，例如“新时尚”。

   注: 更改相应的 DTD 文件可能导致策略不可使用。
2. 对文件进行必要的更改。关于现有文件的更多信息，请参阅第 175 页的『理解样本商店访问控制策略文件』。关于 WebSphere Commerce 中可用的访问控制方法的更多信息，请参阅《*IBMM WebSphere Commerce 访问控制指南*》。
3. 将 *samplestorename*AccessPolicies.xml 和 *samplestorename*AccessPolicies_locale.xml 复制到以下目录中:
   - NT drive:\WebSphere\CommerceServer\xml\policies\xml
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\policies\xml
   - AIX /usr/WebSphere/CommerceServer/xml/policies/xml
   - Solaris /opt/WebSphere/CommerceServer/xml/policies/xml
   - Linux /opt/WebSphere/CommerceServer/xml/policies/xml
   - 400 复制到任何用户数据目录。在现有 XML 文件中指定到 DTD 的全路径。访问控制 DTD 文件位于以下目录中: /QIBM/ProdData/WebCommerce/xml/policies/dtd
4. 运行 xmltransform 命令转换 *samplestorename*AccessPolicies.xml。
   a. 在命令提示符下，转至以下目录:
      - NT drive:\WebSphere\CommerceServer\bin
      - 2000 drive:\ProgramFiles\WebSphere\CommerceServer\bin
      - AIX /usr/WebSphere/CommerceServer\bin
      - Solaris /opt/WebSphere/CommerceServer/bin

- Linux /opt/WebSphere/CommerceServer/bin

b. 然后输入: xmltransform -infile ..\xml\policies\xml\samplestorenameAccessPolicies -transform ..\xml\policies\xsl\accesscontrol.xsl -outfile ..\xml\policies\xml\samplestorenameAccessPoliciesOut.xml

400 TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrol.xsl') INSTROOT(instance_root) OUTFILE(output_file)

c. 检查以下日志文件以确保转换已成功完成:

- NT drive:\WebSphere\CommerceServer\bin\xmltransform.db2.log
- 2000 drive:\Program Files\WebSphere\CommerceServer\bin\xmltransform.db2.log
- AIX /usr/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Solaris /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Linux /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- 400 /QIBM/UserData/WebCommerce/instances/instancename/logs/TRNWCSXML.tx

  如果转换成功，则显示以下消息: "<DATE> <TIME> java.lang.Class main XMLTransformer Transform Successful"

5. 运行 xmltransform 命令转换 *samplestorename*AccessPolicies_locale.xml。

a. 在命令提示符下，转至以下目录:

- NT drive:\WebSphere\CommerceServer\bin
- 2000 drive:\ProgramFiles\WebSphere\CommerceServer\bin
- AIX /usr/WebSphere/CommerceServer\bin
- Solaris /opt/WebSphere/CommerceServer/bin
- Linux /opt/WebSphere/CommerceServer/bin

b. 然后输入: xmltransform -infile ..\xml\policies\xml\samplestorenameAccessPolicies_locale.xml -transform ..\xml\policies\xsl\accesscontrolnls.xsl -outfile ..\xml\policies\xml\samplestorenameAccessPoliciesOut_locale.xml

c. 400 TRNWCSXML INFILE(input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrolnls.xsl') INSTROOT(instance_root) OUTFILE(output_file)

6. 对转换后的 XML 文件进行以下更改:

a. 在 *samplestorename*AccessPolicesOut.xml 中，将开始和结束标记替换为:

```
<?xml version="1.0"?>
<!DOCTYPE accesscontrol-asset SYSTEM "accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

b. 在 *samplestorename*AccessPolicesOut_locale.xml 中，将开始和结束标记替换为:

```
<?xml version="1.0" encoding="correct language code for the file"?>
<!DOCTYPE accesscontrol-asset SYSTEM "../accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

c. 在 *samplestorename*AccessPolicesOut_locale.xml 中，将 @locale 替换为 &locale;，例如将 LANGUAGE_ID="@zh_CN" 更改为 LANGUAGE_ID="&zh_CN;"

d. 在 *samplestorename*AccessPolicesOut_locale.xml 中，将引用定位到 “acpoldesc” 表。除去 ACPOLICY_ID 值末尾的 @。例如，将 "@AllUsersExecuteInFashionAllUsersViews@" 更改为 "@AllUsersExecuteInFashionAllUsersViews"。

e. 在 *samplestorename*AccessPolicesOut.xml 中，将 MEMBER_ID="MEMBER_ID" 替换为 MEMBER_ID="&MEMBER_ID;"

f. 在 *samplestorename*AccessPolicesOut.xml 中，将引用定位到 “acpolicy” 表。除去 ACPOLICY_ID 值末尾的 “@MEMBER_ID”。例如，将 "@AllUsersExecuteInFashionAllUsersViews@MEMBER_ID" 更改为 "@AllUsersExecuteInFashionAllUsersViews"

7. 查找商店的商店归档文件（例如，mystore.sar），缺省情况下，商店归档文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\instances\instancename\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\instances\instancename\sar
- AIX /usr/WebSphere/CommerceServer/instances/instancename/sar
- Solaris /opt/WebSphere/CommerceServer/instances/instancename/sar
- Linux /opt/WebSphere/CommerceServer/instances/instancename/sar
- 400 /QIBM/UserData/WebCommerce/instances/instancename/sar

8. 将 *samplestorename*AccessPolicesOut.xml 和 *samplestorename*AccessPoliciesOut_locale.xml 重命名为: accesscontrol.xml

注: 缺省情况下，特定于语言环境的 accesscontrol.xml 文件位于 data/locale 目录中，例如 data/zh_CN。

9. 使用 ZIP 程序打开商店归档文件。
10. 将商店归档文件中现有的 accesscontrol.xml 和 locale specific accesscontrol.xml 替换为步骤 8 中重命名的文件。
11. 保存商店归档文件。

第 6 部分 封裝商店

第 25 章 封装商店

如果希望使用您的商店作为样本交付给其他人，以在另一个服务器或平台上部署它，或者使用它作为创建其它商店的基础，那么您可能希望以商店归档文件的形式封装它。

与 WebSphere Commerce 一同提供的样本商店封装为商店归档文件。样本商店归档文件中的文件按以下分组:

- Web 有用资源: 用于创建商店页面的文件，如 HTML 文件、JSP 文件、图像、图形和包含文件。Web 有用资源在商店归档文件中以压缩文件（webapp.zip）分组在一起。
- 属性资源绑定（可选）: 包含商店页面的文本。如果商店支持多种语言，资源绑定将包含多个绑定；也就是说，一种语言一个绑定。属性资源绑定在商店归档文件中以压缩归档文件文件（properties.zip）分组在一起。
- 商店数据有用资源: 要装入到数据库中的数据。商店数据有用资源包括诸如竞销、产品目录、货币、实现信息、定价、装运、存储和税务信息这样的数据。
- 支付有用资源: IBM Payment Manager 的配置信息。
- 描述符: XML 文件 sarinfo.xml，它描述商店归档文件，包含 Web 有用资源压缩归档文件、资源绑定和商店数据库有用资源 XML 文件三者的名称。sarinfo.xml 文件还包括包含文件和一致性检查文件的名称，以及关于发布过程中所需归档文件的信息。sarinfo.xml 是商店归档文件中唯一的强制文件。

创建商店归档文件

要将商店封装为商店归档文件，请执行以下操作:

1. 复查与 WebSphere Commerce 一同提供的样本商店归档文件的结构和内容。

   商店归档文件位于以下目录中:

   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WeSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   要查看商店归档文件，请使用解压缩程序。
2. 在 WebSphere Commerce Server 上为商店创建一个临时目录。例如，*mystore*。
3. 在临时目录中，将 Web 有用资源（JSP 文件，HTML，图像）一同分组在名为 webapp 的目录中。使用 ZIP 程序，创建 webapp 文件夹的压缩文件。
4. （可选的）在临时目录中，将属性资源绑定一同分组在名为 properties 的目录中。如果商店支持多种语言，资源绑定将包含多个绑定；也就是说，一种语言一个绑定。使用 ZIP 程序，创建 properties 文件夹的压缩文件。

5. 在临时目录中，将商店数据有用资源一同分组在名为 data 的目录中。如果商店支持多种语言，使用语言环境名称为特定语言的信息创建子目录。例如，en_US。
6. （可选）将 sarrule.xml 文件从现有的商店归档文件复制到 data 目录中。sarrule.xml 文件位于样本商店归档文件的 data 目录中。在您使用“商店服务”发布时，sarrule.xml 担当一致性检查程序。关于 sarrule 文件的更多信息，请参阅第 297 页的附录 D, 『sarrule.xml』。
7. 在临时目录中，创建一个名为 SAR-INF 的目录。
8. 为商店归档文件创建 sarinfo.xml 文件。关于 XML 规范的更多信息，请参阅以下目录中的归档文件描述符 sarinfo.dtd:
   - NT drive:\WebSphere\CommerceServer\xml\sar
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
   - AIX /usr/WeSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar

   a. 使用第 291 页的附录 C, 『sarinfo.xml』 中的示例和信息作为指南，创建 sarinfo.xml 文件。数据有用资源发布的顺序很重要，因为一些数据有用资源必须在其它的数据有用资源发布之前发布。因此，sarinfo.xml 文件中指定的有用资源顺序应当与样本商店的 sarinfo.xml 文件中指定的有用资源顺序一致。

      注: 如果您选择在商店归档文件中包含 sarrule.xml 文件（请参阅步骤 6），则应当在 sarinfo.xml 文件中包含关于 sarrule.xml 的信息。如果选择不包含 sarrule.xml 文件，请确保 sarinfo.xml 文件中没有提及此文件。

   b. 将 sarinfo.xml 保存到步骤 6 中创建的 SAR-INF 目录中。
9. 创建由 Web 有用资源 ZIP 文件、属性资源绑定、商店数据有用资源和 sar-inf 目录组成的 ZIP 文件。将此 ZIP 文件命名为 *storearchivename.sar*。
10. 如果您希望使用“商店服务”编辑或发布商店归档文件，请将 *storearchivename.sar* 保存到以下目录:
    - NT drive:\WebSphere\CommerceServer\instances*instancename*\sar
    - 2000 drive:\Program Files\WebSphere\CommerceServer\instances*instancename*\sar
    - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/sar
    - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/sar
    - Linux /opt/WebSphere/CommerceServer/instances/*instancename*/sar
    - 400 /QIBM/UserData/WebCommerce/instances/*instancename*/sar
11. 商店归档文件现在将显示在“商店服务”中的商店归档文件列表中。

创建样本商店归档文件

在将商店封装为商店归档文件后，您可以在“商店服务”中选择使用它作为样本商店。样本商店归档文件是一个商店归档文件，它将作为创建新商店的基础被复制和使用。要将商店归档文件用作样本商店归档文件，请执行以下操作:

1. 将商店归档文件保存到以下目录:
   - NT drive:\WebSphere\CommerceServer\samplestores*storeachivename*
   - 2000 drive:\Program Files\WebSphere\CommerceServer \samplestores*storeachivename*
   - AIX /usr/WeSphere/CommerceServer/samplestores/*storeachivename*
   - Solaris /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
   - Linux /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
   - 400 /qibm/proddata/WebCommerce/samplestores/*storeachivename*
2. （可选的）创建预览页面。要在“商店服务”中显示商店页面的预览，您必须创建预览页面。请执行以下操作:
   a. （可选的）在“商店服务”中，选择**新建**。将显示“创建商店归档文件”页面。从**样本**列表中选择一个样本商店，然后单击**预览**。显示的页面称为预览页面。这些页面是 HTML 文件，它们显示预定义的样本购物流程，并作为样本商店的预览。
   b. 确定希望在预览页面中显示的购物流程。
   c. （可选的）在发布了的商店中创建一些样本数据。例如，向购物车中添加商品，并创建少许送货地址和开票地址。将正在从此商店创建预览页面，数据使页面看起来更加真实。
   d. 使用 Internet Explorer 浏览商店。通过选择“文件” -> “另存为”保存每个页面的 HTML。您还应当保存样式表（.css）和图像。将文件保存到以下目录:
      - *stylesheet*.css
        - NT drive:\WebSphere\AppServer\installedApps\ WC_Enterprise_App_*instancename*.ear\wcstools.war\ tools\devtools*storearchivename*\preview
        - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\ WC_Enterprise_App_*instancename*.ear\wcstools.war \tools\devtools*storearchivename*\preview
        - AIX /usr/WebSphere/AppServer/installedApps WC_Enterprise_App_*instancename*.ear/wcstools.war /tools/devtools/*storearchivename*/preview
        - Solaris /opt/WebSphere/AppServer/installedApps/ WC_Enterprise_App_*instancename.ear*/wcstools.war /tools/devtools/*storearchivename*/preview
        - Linux /opt/WebSphere/AppServer/installedApps/ WC_Enterprise_App_*instancename.ear*/wcstools.war /tools/devtools/*storearchivename*/preview

- 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview

- HTML
  - NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview*locale*
  - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview*locale*
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
  - 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
- 与语言环境无关的图像
  - NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview\images
  - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview\images
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/images

- 与语言环境有关的图像
  - NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview*locale*\images
  - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstools.war\tools\devtools*storearchivename*\preview*locale*\images
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images
  - 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools*storearchivename*/preview/*locale*/images

e. 由于图像和 css 文件的位置已经更改，您必须更改 HTML 页面中至图象和 css 文件的引用。更改完引用之后，请确保在浏览器中打开 HTML 页面时能够查看图像。

f. 将 HTML 页面中的链接从命令更改为引用 HTML 文件的链接。

3. 创建总结商店归档文件的 HTML 文件。此信息将显示在“商店服务”的“创建商店归档文件”页面中的**样本描述**中。

   a. 使用以下示例作为指南，创建新文件。

```
<doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
在此描述商店
</body>
</html>
```

   b. 将此文件保存为 Feature_*locale*.html，其中 locale 是您正在使用的语言的缩写。例如，en_US。将此文件保存到以下目录:

   - NT drive:\WebSphere\CommerceServer\samplestores*storeachivename*
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores*storeachivename*
   - AIX /usr/WeSphere/CommerceServer/samplestores/*storeachivename*

- Solaris /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
- Linux /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
- 400 /qibm/proddata/WebCommerce/samplestores/*storeachivename*

4. 将商店归档文件添加到 sarregistry.xml 文件。sarregistry.xml 文件确定哪些商店归档文件在**样本**列表（此列表位于“商店服务”的“创建商店归档文件”页面中）中显示。sarregistry.xml 还确定哪些预览页面和功能文件与每个商店归档文件关联。 sarregistry.xml位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\xml\tools\devtools
   - 2000 drive:\Program Files\WebSphere\CommerceServer\xml\tools\devtools
   - AIX /usr/WebSphere/CommerceServer/xml/tools/devtools
   - Solaris /opt/WebSphere/CommerceServer/xml/tools/devtools
   - Linux /opt/WebSphere/CommerceServer/xml/tools/devtools
   - 400 /QIBM/ProdData/WebCommerce/xml/tools/devtools

   a. 使用以下示例作为指南，将新的商店归档文件添加到 sarregistry.xml 中。

```
<SampleSAR fileName="infashion_en_US_es_ES.sar" relativePath="InFashion">
        <html locale="es_ES" featureFile="InFashion/Feature_es_ES.html"
        sampleSite="RetailModel/preview/es_ES/index.html"/>
        <html locale="en_US" featureFile="InFashion/Feature_en_US.html"
        sampleSite="RetailModel/preview/en_US/index.html"/>
</SampleSAR>
```

      此示例定义英语和西班牙语“流行时尚”商店归档文件的预览页面。粗体字的行定义英语和西班牙语预览页面的主页。

5. 您现在应当能够在**样本**列表（位于“商店服务”的“创建商店归档文件”页面中）中查看商店归档文件。

第 7 部分 发布商店

为了创建正常工作的商店，必须将商店前台 Web 有用资源发布到 WebSphere Commerce Server，且必须将商店数据发布到 WebSphere Commerce 数据库。

本部分中的各章讨论 WebSphere Commerce 提供的发布选项:

- 第 195 页的第 26 章，『发布完整的商店』 — 该章讨论在商店处于商店归档文件格式的情况下，使用“商店服务”或命令行发布来发布整个商店（商店前台和商店数据有用资源）。
- 第 207 页的第 27 章，『装入商店数据的概述』 — 该章讨论使用装入程序软件包和 WebSphere Catalog Manager 的其它组件将商店数据有用资源发布到数据库。
- 第 245 页的第 28 章，『装入 WebSphere Commerce 数据库有用资源组』 — 该章讨论使用装入程序软件包和 WebSphere Catalog Manager 的其它组件将商店数据有用资源组或所有商店数据发布到数据库。
- 第 257 页的第 29 章，『发布商业帐户和合同』 — 该章讨论发布帐户、合同和产品集有用资源。
- 第 261 页的第 30 章，『发布商店前台有用资源和商店配置文件』 — 该章讨论发布商店前台有用资源和商店配置文件。

第 26 章 发布完整的商店

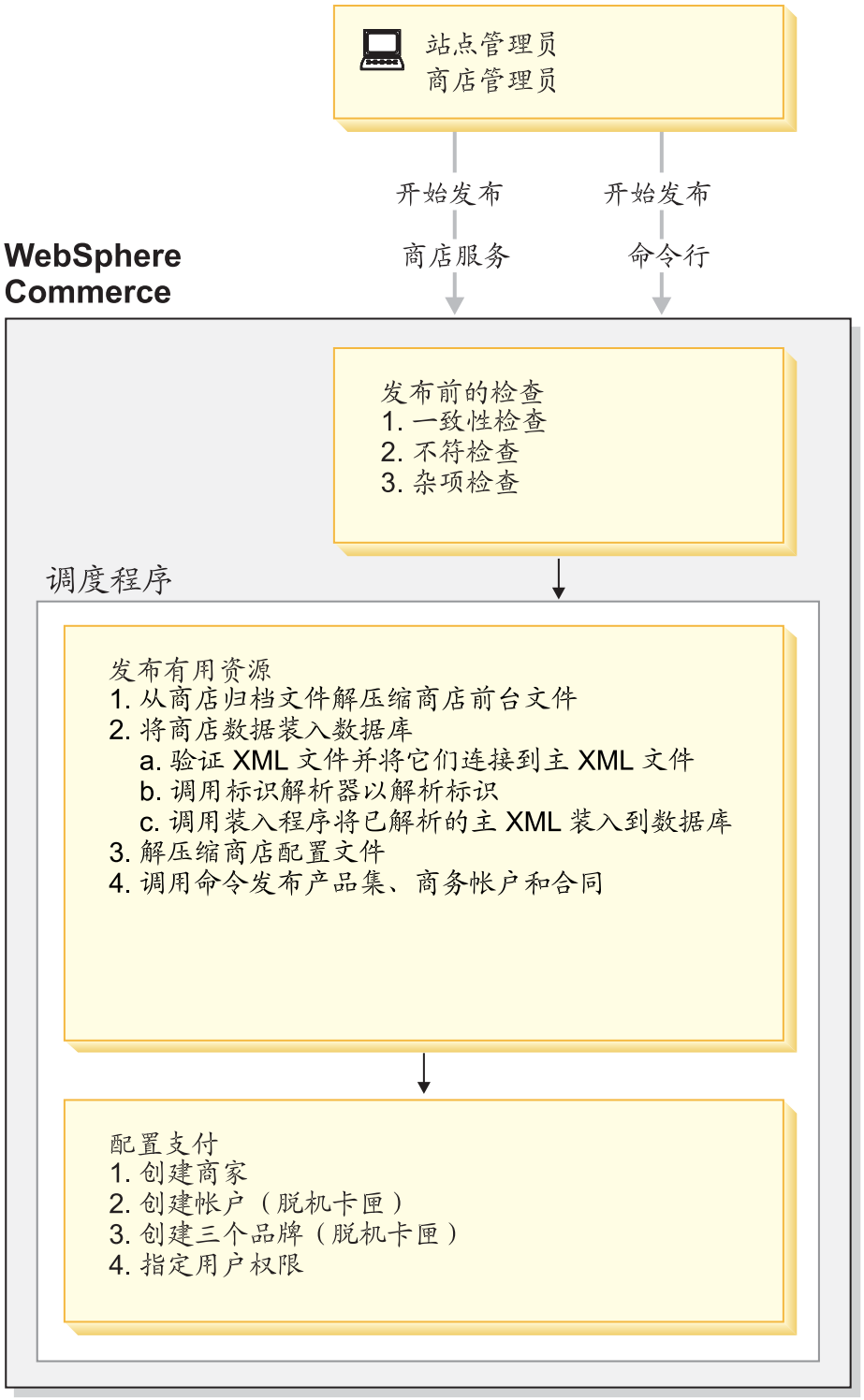
为了创建正常工作的商店，必须将商店前台 Web 有用资源发布到 WebSphere Commerce Server，且必须将商店数据发布到 WebSphere Commerce 数据库。本章讨论在商店处于商店归档文件格式的情况下，使用“商店服务”或命令行发布来发布整个商店（商店前台和商店数据有用资源）。

注: 如果不希望将商店封装为商店归档文件，可单个地发布有用资源。关于更多信息，请参阅第 207 页的第 27 章，『装入商店数据的概述』、第 245 页的第 28 章，『装入 WebSphere Commerce 数据库有用资源组』、第 257 页的第 29 章，『发布商业帐户和合同』和第 261 页的第 30 章，『发布商店前台有用资源和商店配置文件』。

了解 WebSphere Commerce 中的发布

“商店服务”或命令行提供的发布选项让您能够同时发布完整的商店（商店前台和商店数据有用资源）。为使用此选项，必须以商店归档文件的格式封装商店有用资源。关于将商店封装为商店归档文件的更多信息，请参阅第 185 页的第 6 部分，『封装商店』。

下图概述了发布过程中的步骤。



开始发布

为了发布商店，您必须具有站点管理员或商店管理员（对于所有商店）权限。站点管理员或商店管理员可使用以下方法之一启动发布过程:

- 商店服务

- 命令行

两种发布方法都要求您选择希望发布的商店归档文件。使用任一方法，都可选择希望发布的商店归档文件的量。例如，可选择以下项的任意组合用于发布:

- 商店数据库有用资源，带或不带在线产品目录数据
- Web 有用资源，例如 JSP 文件、HTML 文件和图像
- 属性文件（商店的文本）

首次发布时，建议您发布整个商店归档文件（上述所有内容），这样您可查看正常工作的商店。然而在后续的发布中，您可能希望仅更新以下一项: 数据库有用资源、Web 有用资源或属性资源绑定，而不是重新发布整个商店归档文件。

如果在首次发布商店归档文件时选择不发布在线产品目录数据，则一些数据有用资源（合同和帐户）可能未正确发布，因为它们依赖于产品目录中的数据。并且，如果在首次发布时未同时发布产品目录数据和 Web 有用资源，则将不创建商店标识、产品目录标识和语言标识，且将不能从“商店服务”启动商店。关于更多信息，请参阅第 204 页的『创建 parameters.jsp 文件』。

在线产品目录数据由商店归档文件中的以下 XML 文件所组成:

- `catalog.xml`
- `offering.xml`
- `store-catalog.xml`
- `store-catalog-shipping.xml`
- `store-catalog-tax.xml`
- `storefulfill.xml`

如果选择以后发布此信息，则可使用“商店服务”或通过装入程序软件包发布它。关于使用数据库有用资源组装入数据子集（例如上面列出的产品目录组数据）的更多信息，请参阅第 252 页的『装入数据库有用资源组』。

关于如何使用“商店服务”或命令行发布商店归档文件的更详细信息，请参阅 WebSphere Commerce 联机帮助。

注: 在使用“商店服务”或命令行启动了发布过程后，无需再执行任何操作。列在前图和本章中的所有其它步骤都是由 WebSphere Commerce 系统完成的。

发布前的检查

一旦站点管理员或商店管理员启动了发布，则 WebSphere Commerce 在开始实际的发布过程之前执行若干检查。这些检查包含以下内容:

- 一致性检查
- 不符检查
- 杂项检查

注: 命令行发布检查传递给它的参数，并完成一致性检查和不一致性检查。然而，如果您使用命令行发布来发布，则您不会看见如以下段落中所述的消息。相反的，如果您使用更新模式发布，则命令行将覆盖现有的商店而不会提示您。如果您使用插入模式发布，则命令行将装入新的商店。如果发布失败，则显示一个发布失败错误消息。

一致性检查

发布前的检查使用 sarrule.xml 文件中的规则来确保 XML 文件中的信息与商店归档文件中的 Web 有用资源一致。例如，如果 command.xml 文件引用特定的 JSP 文件，则检查将确保 JSP 文件位于商店归档文件的 webapp.zip 中。如果一致性检查发现错误，则将错误写入日志，但发布照常进行。关于日志文件的更多信息，请参阅第 205 页的『发布日志文件』。

关于 sarrule.xml 文件的示例，请参阅第 297 页的附录 D，『sarrule.xml』。

不符检查

发布前的检查期间，WebSphere Commerce 检查商店和产品目录之间的不符。特别地，它检查数据库是否存在以下项:

- 与 STOREENT 表具有相同标识的现有商店：如果商店已经存在，则将显示消息询问您是要覆盖商店还是取消发布。
- 具有相同的产品目录标识的现有产品目录：如果产品目录已经存在，则将显示消息询问您是要覆盖产品目录还是取消发布。如果产品目录属于其它商店，则显示消息声明产品目录所属的商店，并询问您是要继续发布并覆盖产品目录，还是取消发布。

杂项检查

在发布前的检查期间，WebSphere Commerce 还检查是否启用了调度程序、是否禁用了高速缓存触发器和高速缓存，以及是否禁用了摘要表。如果否，则显示警告消息，告知未正确地启用或禁用哪个功能。

注: 命令行发布不会完成这些检查。

调度程序: 调度程序运行发布作业（将在以下部分中更详细讨论）。如果禁用了调度程序，则发布过程将不能运行。

高速缓存和高速缓存触发器: 在发布期间保持高速缓存打开，会导致在发布期间更新数据库时调用高速缓存触发器。高速缓存触发器可能生成不必要的数据库活动，这些活动可导致数据库事务日志溢出并影响发布性能。要禁用高速缓存触发器，请参阅 WebSphere Commerce 联机帮助。

摘要表: DB2 启用摘要表可导致在发布期间更新摘要表，这可导致数据库事务日志溢出并影响发布性能。要禁用摘要表，请参阅 WebSphere Commerce 联机帮助。

发布有用资源

发布过程的发布有用资源阶段是由调度程序运行的已调度作业。当调度程序运行发布作业时，WebSphere Commerce 完成以下操作:

- 对商店归档文件中的商店前台文件执行解压缩
- 将商店数据从商店归档文件的 XML 文件装入到数据库中
- 创建 parameters.jsp 文件
- 对商店配置文件执行解压缩
- 调用命令发布商业帐户和合同
- 更新注册表组件

对商店归档文件中的商店前台文件执行解压缩

将 Web 有用资源从商店归档文件解压缩到 WebSphere Commerce Server 是发布有用资源阶段的第一个操作。当对商店归档文件中的文件执行解压缩时，WebSphere Commerce 执行以下操作:

对 Web 有用资源执行解压缩并将它们复制到 WebSphere Commerce Server 上的以下位置:

- 将 JSP 文件、HTML、包含文件、图像和图形发布到商店 Web 应用程序文档根路径下的商店目录（*storedir*）:
  - NT `drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\wcstores.war\`*storedir*
  - 2000 `drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\wcstores.war\`*storedir*
  - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
- 将资源绑定和属性文件发布到应用程序属性路径:
  - NT `drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\wcstores.war\WEB-INF\classes\`*storedir*
  - 2000 `drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\wcstores.war\WEB-INF\classes\`*storedir*
  - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/WEB-INF/classes/`*storedir*
  - Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/WEB-INF/classes/`*storedir*
  - Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/WEB-INF/classes/`*storedir*
  - 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/WEB-INF/classes/`*storedir*

将商店数据从商店归档文件的 XML 文件装入到数据库中

将商店数据从商店归档文件中的 XML 文件装入到数据库中时，WebSphere Commerce 执行以下操作:

注: 仅将类型为 db-load 的 XML 文件装入数据库。文件类型在 `sarinfo.xml` 文件中指定。关于 sarinfo.xml 文件的更多信息，请参阅第 291 页的附录 C, 『sarinfo.xml』

验证商店归档文件中的 XML 文件并将它们连接至主 XML 文件: WebSphere Commerce 使用对应的 DTD 文件验证 XML 文件。DTD 文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\xml\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

如果 XML 文件无效，则 WebSphere Commerce 向错误日志写入错误。如果错误日志中的错误超出了在 WebSphere Commerce 配置文件 *instance_name*.xml 的 DevTools 部分中指定的最大错误数（缺省情况下 MaxErrorsInSarXML=1），则发布失败。WebSphere Commerce 配置文件 *instance_name*.xml 位于以下目录中:

- NT drive:\WebSphere\CommerceServer\instances\ *instancename*\xml*instance_name*.xml
- 2000 drive:\Program Files\WebSphere\CommerceServer\instances\ *instancename*\xml*instance_name*.xml
- AIX /usr/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- Solaris /opt/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- Linux /opt/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- 400 /QIBM/UserData/WebCommerce/instances/*instancename* /xml/*instance_name*.xml

验证了 XML 文件之后，会将这些文件连接成一个文件: *storename*master.xml。这些文件根据 sarinfo.xml 文件中指定的优先级连接在一起。关于更多信息，请参阅第 291 页的附录 C, 『sarinfo.xml』。*storename*master.xml 文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\temp*instancename* \tools\devtools
- 2000 drive:\Program Files\WebSphere\CommerceServer\temp*instancename*\tools\devtools
- AIX /usr/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Solaris /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Linux /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- 400 /QIBM/UserData/WebCommerce/instances/*instancename* /temp/tools/devtools

调用标识解析器解析标识: 标识解析器（一个装入程序软件包实用程序）在商店归档文件 XML 文件中生成 XML 元素的唯一标识。例如，标识解析器以唯一值替换样本商店 XML 文件中使用的 @ 别名。关于样本商店中使用的内部别名解析的示例，请参阅第 289 页的附录 B, 『创建数据』。

注: 当您重新发布时，标识解析器还可以为已发布的商店解析标识。例如，如果您已发布了商店归档文件一次，而且您需要重新发布商店归档文件或其一部分，则标识解析器从数据库中检索唯一标识并在重新发布的过程中使用它们。

关于标识解析器和装入程序软件包其它组件的更多信息，请参阅第 207 页的第 27 章, 『装入商店数据的概述』。

当“商店服务”或命令行发布调用标识解析器时，它必须指定要使用的标识解析器方法。标识解析器具有若干方法，这些方法可用于处理标识解析器输入；明确地讲，是如同标识存在于原始数据中一样地处理数据（更新方法），还是如同标识不存在于原始数据中一样地处理数据（装入方法）。当一些标识存在而另一些不存在时，则使用混合方法。

注: 混合方法是对“商店服务”建议的方法。如果您使用命令行来发布，则可能还希望指定混合模式，特别是当不知道正在发布的数据是否已存在于数据库中的时候。

您可以指定“商店服务”或命令行发布将在 WebSphere Commerce 配置文件 *instance_name*.xml 中使用何种方法。缺省情况下，“商店服务”使用混合方法。关于标识解析器方法的更多信息，请参阅第 211 页的『标识解析命令』。

“商店服务”和命令行发布还必须指定要用于标识解析器的定制程序文件。如果您未在 WebSphere Commerce 配置文件 *instance_name*.xml 中指定定制程序文件，则发布代码将使用缺省定制程序文件: DBConnectionCustomizer 或 OracleConnectionCustomizer 之一。

Oracle

OracleConnectionCustomizer 定制程序文件位于以下目录中:

- NT `drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\properties`
- 2000 `drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*instancename*`.ear\properties`
- AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/properties`
- Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/properties`
- Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/properties`
- 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_`*instancename*`.ear/properties`

DB2

DBConnectionCustomizer 文件位于以下 ZIP 文件中:

- NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\lib\loader\idresgen.zip
- 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\lib\loader\idresgen.zip
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip

注: 如果希望指定自己的定制程序文件，则必须向 *instance_name*.xml 文件的 DevTools 部分添加以下内容:

- IDResolverCustomizerFile="*myIDResolverCustomizerFile*"

缺省情况下，WebSphere Commerce 配置文件 *instance_name*.xml 不为此属性指定值。

标识解析器使用 *storename*master.xml 和对应的 DTD 文件 wcs.dtd。解析了标识之后，标识解析器创建以下文件 *storenametime_stamp*master.xml 文件，该文件包含唯一标识。如果在标识解析过程中发生错误，则装入程序软件包创建 error.xml 文件 *storename*master.error.xml。

注: 当发布尝试失败时，发布过程将自动保存这些临时文件。然而，如果发布成功，缺省情况下将删除这些文件。您可能希望保留这些文件以供疑难解答，或发布和处理 WebSphere Test Environment 中的商店。要保存临时文件，请参阅 WebSphere Commerce 联机帮助中的“商店服务参数”。

*storenametime_stamp*master.xml 文件和 *storename*master.error.xml 位于以下目录中:

- NT drive:\WebSphere\CommerceServer\temp*instancename*\tools\devtools
- 2000 drive:\Program Files\WebSphere\CommerceServer\temp*instancename*\tools\devtools
- AIX /usr/WebSphere/CommerceServer/temp/*instancename*/tools/devtools

- Solaris /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Linux /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/temp/tools/devtools

调用装入程序软件包将经解析的主 XML 文件装入数据库: 装入程序软件包将经解析的 *storenametime_stamp*master.xml 装入数据库。如果在装入过程中发生错误，则装入程序软件包创建 error.xml 文件 *storenametime_stamp* master.error.xml。

关于装入程序软件包的更多信息，请参阅第 207 页的第 27 章，『装入商店数据的概述』。

当“商店服务”或命令行发布调用装入程序软件包时，它必须指定要使用的装入程序方法。“商店服务”可以使用以下任何一种装入程序方法:

- SQL 导入
- 导入
- 装入

注: 缺省情况下，“商店服务”使用 SQL 导入方法。
您可以使用 DevTools 元素中 LoaderMode 属性来指定“商店服务”或命令行将在 WebSphere Commerce 配置文件 *instance_name*.xml 中调用何种方法。

- SQL 导入: 此方法使用 Java 数据库连接性（JDBC）插入和更新数据，提供了最灵活的操作方法，但是也是将大量数据导入少量表的最慢方法。它允许列级别的更新。建议您使用 SQL 导入。

  注: SQL 导入方法是使用最安全的方法，因为当数据无效时它不会使数据库毁坏。在您可以使用 SQL 导入装入之前，记录必须符合数据库模式约束。其它的装入程序方法更快，因为它们成批装入数据库而不进行过多的检查。因此在使用其它方法之前，您必须确保数据的正确性。
- 导入: 此方法使用 DB2 本机导入功能并允许单元格级别的更新，具有中等的速度和灵活性。此方法不可用于 Oracle。
- 装入: 此方法使用 RDBMS（DB2 Load 或 SQLLoad）的本机工具，且是将大量数据装入到少量表中的最快方法。

关于装入命令中的方法的更多信息，请参阅第 218 页的『装入命令』。

“商店服务”和命令行发布还必须指定要用于装入程序的定制程序文件。如果未在 WebSphere Commerce 配置文件 *instance_name*.xml 中指定定制程序文件，则发布代码将使用缺省定制程序文件: MassLoadCustomizer。

注: 如果希望指定自己的定制程序文件，则必须向 *instance_name*.xml 文件的 DevTools 部分添加以下内容:

- LoaderCustomizerFile="*myLoaderCustomizerFile*"

  缺省情况下，WebSphere Commerce 配置文件 *instance_name*.xml 不为此属性指定值。

创建 parameters.jsp 文件

发布过程创建文件 parameters.jsp。此文件包含三个参数: storeId、catalogId 和 langId。样本商店中的 index.jsp 文件使用这些参数启动商店。

为了让发布过程创建 parameters.jsp 文件，必须并行发布产品目录数据和 Web 有用资源至少一次。如果这两项没有一起发布，则“商店服务”将不能启动商店。

parameters.jsp 位于以下目录中:

- NT `drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_instancename.ear\wcstores.war\storedir\include`
- 2000 `drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_instancename.ear\wcstores.war\storedir\include`
- AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/wcstores.war/storedir/include`
- Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/wcstores.war/storedir/include`
- Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/wcstores.war/storedir/include`
- 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_instancename.ear/wcstores.war/storedir/include`

对商店配置文件执行解压缩

Business “多乐五金店”和“新时尚”样本商店归档文件还包含以下文件:

- tools_properties.zip
- tools_xml.zip
- runtime_xml.zip

这些文件注册在 sarinfo.xml 文件中且由“商店服务”用于配置商店。在发布过程中，这些文件将被解压缩并复制到 WebSphere Commerce 配置文件 *instance_name*.xml 中指定的目录中。关于更多信息，请参阅第 261 页的『通过复制到 WebSphere Commerce Server 发布商店前台有用资源和商店配置文件』。

注: 这些文件不得更改、除去、复制到其它商店或保存到不同的目录中。

调用命令发布商业帐户和合同

一些商店数据库有用资源（合同和商业帐户）不能由装入程序软件包装入，因此发布还调用对应的命令将这些有用资源发布到 WebSphere Commerce Server。这些命令如下:

- AccountImport — 从商店归档文件中的 businessaccount.xml 文件中创建商业帐户。
- ContractImportApprovedVersion — 从商店归档文件中的 contract.xml 文件导入合同。
- ProductSetPublish — 在创建商业帐户和合同之前，使数据库中的产品集数据与产品目录同步。“商店服务”和命令行发布调用 ProductSetPublish 命令，该命令接着调用 AccountImport 和 ContractImportApprovedVersion 命令。

关于发布商业帐户和合同的更多信息，请参阅第 257 页的第 29 章，『发布商业帐户和合同』。

更新注册表组件

发布过程中的最后一个操作是更新注册表组件。发布更新 WebSphere Commerce 中所有的注册表。关于注册表的更多信息，请参阅 WebSphere Commerce 联机帮助。

错误处理

如果在发布过程的发布有用资源阶段发生错误，则可在发布日志（请参阅『发布日志文件』）或通过“商店服务”的“发布摘要”页面来查看错误消息。

配置支付

发布过程的最后一步是配置支付。WebSphere Commerce 支持 IBM Payment Manager。如果计划使用 Payment Manager 作为处理支付的方法，则应当如第 105 页的第 13 章，『支付有用资源』所述创建支付 XML 文件。如果支付 XML 文件包含在正在发布的商店归档文件中，则 WebSphere Commerce 将在发布期间完成以下支付配置:

- 创建商家。
- 创建帐户（仅用于脱机卡匣）。
- 创建 paymentinfo.xml 中指定的品牌（仅用于脱机卡匣）。
- 指定用户权限。

错误处理

如果在发布过程的配置支付阶段发生错误，则您可以在发布日志中查看错误消息（请参阅『发布日志文件』）。

发布日志文件

发布过程的发布有用资源阶段遇到的所有错误写入以下日志和跟踪文件中:

- messages.txt: 包含发布过程的装入程序软件包部分的错误消息。在发布失败时首先检查此日志。在这些错误消息中提及的行或列号指的是临时 master.xml 文件: *storename*master.xml 或 *storenametime_stamp*master.xml。
- trace.txt: 包含发布过程的装入程序软件包和标识解析器部分的跟踪信息。缺省情况下 trace.txt 是关闭的。
- ecmsg_*instancename_timestamp*.log: 记录 WebSphere Commerce Server 中所有正在运行的错误消息。
- wcs.log 文件: 包含从 WebSphere Application Server 中运行的所有应用程序（包括发布）到标准控制台的输出。

日志文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\instances*instancename*\logs
- 2000 drive:\Program Files\WebSphere\CommerceServer*instancename*\logs
- AIX /usr/WebSphere/CommerceServer/instances/*instancename*/logs
- Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/logs
- Linux /opt/WebSphere/CommerceServer/instances/*instancename*/logs
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/logs

要配置 trace.txt 和 messages.txt 日志文件（即，调整日志级别或其它选项），请编辑以下文件:

- NT drive:\WebSphere\CommerceServer\xml\loader\WCALoggerConfig.xml
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\loader\WCALoggerConfig.xml
- AIX /usr/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- Solaris /opt/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- Linux /opt/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- 400 /QIBM/UserData/WebCommerce/instances/instancename/xml/WCALoggerConfig.xml

注: 关于配置 WCALoggerConfig.xml 文件的更多信息，请参阅《*WebSphere Commerce Catalog Manager 用户指南*》。

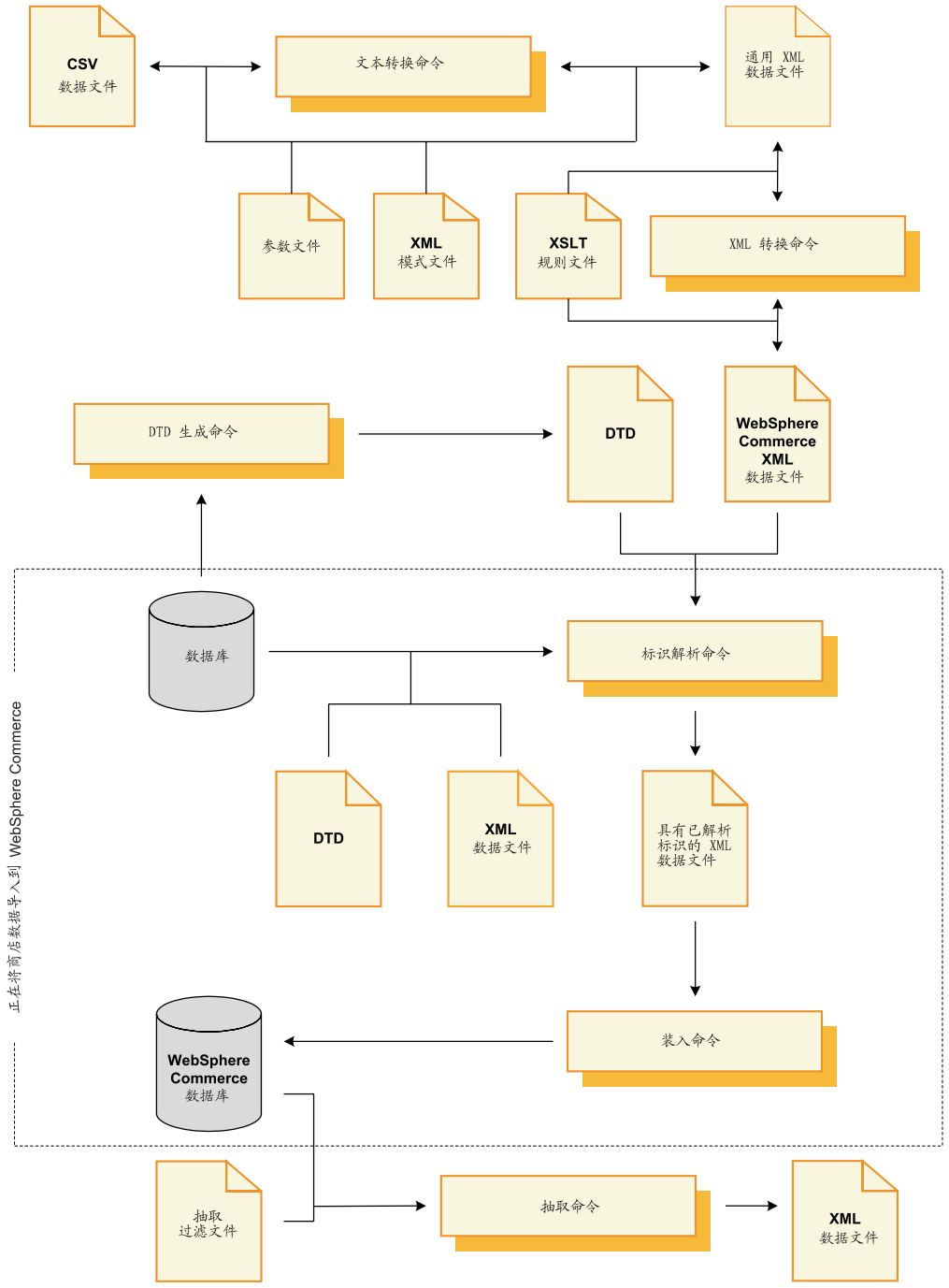
第 27 章 装入商店数据的概述

创建商店数据之后，可选择使用“商店服务”将它封装为商店归档文件并发布它，或者可使用 WebSphere Commerce Catalog Manager 装入程序软件包将它直接装入 WebSphere Commerce Server 数据库。请参阅第 245 页的第 28 章，『装入 WebSphere Commerce 数据库有用资源组』和第 252 页的『装入数据库有用资源组』以获取关于 WebSphere Commerce 数据库有用资源组的装入过程的信息。

Catalog Manager 提供了六个命令行实用程序（这里统称为“装入程序软件包”）和三个相关的管理工具，可用于为装入准备数据以及将数据装入商店。这些命令和工具使用“可扩展标记语言（XML）”数据文件管理信息。

理解 WebSphere Commerce 中的数据装入

下图显示了可使用装入程序软件包命令执行的数据准备、装入和抽取过程。



请注意虚线指示最常用于将商店数据装入 WebSphere Commerce Server 数据库的两个过程：解析标识和装入数据。这些过程是本章的讨论重点。

关于准备数据以供装入 WebSphere Commerce Server 数据库的更多信息，请参阅第 33 页的第 4 部分，『开发商店数据』。

以下两个装入程序软件包命令行实用程序常用于将数据装入 WebSphere Commerce Server 数据库:

- **标识解析命令**

  要使用装入程序软件包将 XML 数据装入 WebSphere Commerce Server 数据库，XML 元素必须直接映射到目标 WebSphere Commerce Server 数据库的模式。其具有的属性相应于数据库模式中的唯一键或主键的所有 XML 元素都必须具有唯一标识；数据库模式所有的非空列都必须用非空值定义相应的属性。标识解析器可以为获得授权的 XML 元素的唯一键或主键属性生成唯一标识。

  注: 正如此文档中所示，标识是数据库表（它为每行给予一个唯一标识）的单个列中的一个值。如果您使用标识解析器来生成标识，则它从 KEYS 或 SUBKEYS 表中获得一个基础值，并顺序递增该值以解析数据库表中每行的标识。

  关于此命令的信息，请参阅第 211 页的『标识解析命令』、第 235 页的『使用装入程序软件包命令和脚本』和第 236 页的『解析标识的示例』。

- **装入命令**

  装入程序使用有效而组织良好的 XML 文件作为输入，将数据装入数据库。XML 文档的元素映射到数据库中的表名；元素属性映射到列。

  注: 关于有效性和格式良好性约束的描述，请参阅万维网联盟（W3C）XML 原则。

  关于此命令的信息，请参阅第 218 页的『装入命令』、第 235 页的『使用装入程序软件包命令和脚本』和第 243 页的『装入数据的示例』。

这些命令是本章主要的重点。

也可使用以下装入程序软件包命令行实用程序管理数据:

- **DTD 生成命令**

  DTD 生成程序生成文档类型定义（DTD），后者描述要向其装入 XML 数据的目标数据库的表和列。DTD 生成程序还可为数据库生成 XML 模式。

  DTD 生成程序可基于 WebSphere Commerce 数据库模式创建 DTD。如果使用随样本商店归档文件提供的 DTD 且未修改数据库模式，则通常无需使用 DTD 生成程序生成 DTD。

  关于更多信息，请参阅第 225 页的『DTD 生成命令』。

- **抽取命令**

  抽取程序对照数据库使用查询，以将选定的数据子集从数据库抽取到 XML 文档。

  您可以使用此命令从数据库中抽取数据并转换为 XML 格式。

  关于更多信息，请参阅第 228 页的『抽取命令』。

- **文本转换命令**

  文本转换程序在字符定界的变量格式与 XML 数据格式之间转换数据。

  如果您的数据无法直接从数据库中抽取为 XML 格式，则（比如说）您可以将数据保存为字符限定变量格式，然后使用此命令将其转换成 XML 格式。

  关于更多信息，请参阅第 230 页的『文本转换命令』。

- **XML 转换命令**

  XML 转换程序将 XML 文档中的数据转换为备用的 XML 格式。它使用“可扩展样式表语言（XSL）”定义转换的映射规则。

  您可以使用此命令将您的 XML 数据转换成一个格式，该格式直接映射到目标 WebSphere Commerce 数据库的模式（您希望将数据装入到此数据库）。

  关于更多信息，请参阅第 231 页的『XML 转换命令』。

这些命令不是本章主要的重点。关于这些命令的详细信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

WebSphere Commerce Catalog Manager 还包含以下工具来帮助管理其数据管理功能:

- **文本转换工具**

  文本转换工具有助于使用“文本转换”命令来处理字符定界的变量格式和 XML 数据格式之间的数据转换。

- **XSL 编辑器**

  XSL 编辑器提供了用于编辑 XSL 文件（可由 XML 转换程序使用）的可视界面。使用 XSL 编辑器，在对 XML 格式之间转换数据定义映射规则时，建立从源 DTD 中的元素至目标 DTD 中的元素的关联。

- **Web 编辑器**

  Web 编辑器让您能够通过 Web 浏览器创建、修改和删除数据库中的数据。

这些工具不是本章主要的重点。关于这些工具的详细信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

用于装入商店数据的装入程序软件包命令

标识解析命令

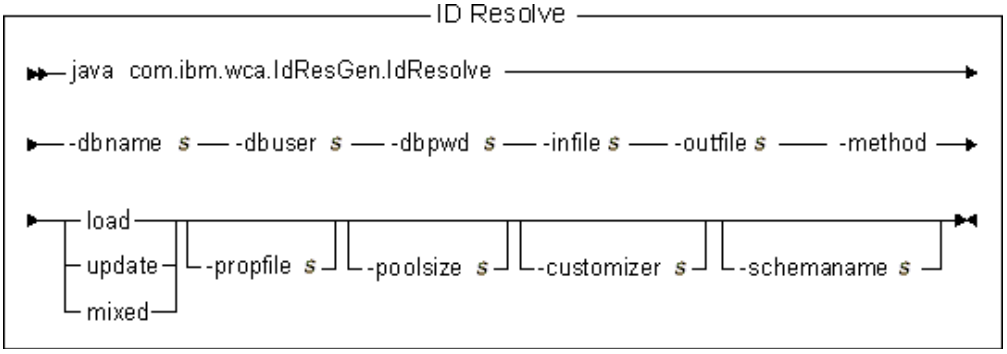
此命令为在可将 XML 数据元素装入数据库之前需要标识的 XML 数据元素生成唯一标识。如果源 XML 数据已提供了必要的唯一标识，则无需运行标识解析器。

WebSphere Commerce 数据库模式定义其表中的主键和外键，这些键用于表示表之间的各种关系。对于此原因，WebSphere Commerce XML 元素必须包含带有唯一标识的相应属性。在 WebSphere Commerce Server 数据库中，必须解析其标识的表是定义在 KEYS 和 SUBKEYS 表中的那些表。这些表在 WebSphere Commerce 中被称为*主表*。有关 KEYS 和 SUBKEYS 表的更多信息，请参阅 WebSphere Commerce 联机帮助。

注: 如果必须解析在 KEYS 或 SUBKEYS 表中未定义的表的标识，在运行标识解析器前将该表添加至 SUBKEYS 表。

因为 WebSphere Commerce XML 元素和属性意在可跨数据库和跨数据库实例进行移植，因此通常使用内部别名表示其标识。在可将数据装入任何 WebSphere Commerce Server 数据库之前，必须将这些别名解析为有效的数字标识。关于更多信息，请参阅第 289 页的附录 B，『创建数据』。

Windows AIX Solaris Linux



注:

1. 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数；因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。
2. 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

-dbname

目标数据库的名称

-dbuser

连接至数据库的用户的名称

-dbpwd

连接至数据库的用户的密码

-infile 包含表记录的输入 XML 文档的名称。

-outfile

要产生的输出 XML 文件的名称; 此文件可用作对装入程序的输入

-method

要用于处理输入文件的方法

- 如果输入文件中的***所有***记录都***不存在***于数据库中，则使用装入方法处理输入文件。
- 如果输入文件中的***所有***记录都***存在***于数据库中，则使用更新方法处理输入文件。
- 如果输入文件中***仅有部分***记录***存在***于数据库中，则使用混合方法处理输入文件。

缺省方法是 load。

-propfile

包含 name=value 对格式的 Java 属性的文本文件。此属性文件设置标识解析器解析标识的方式。它用于描述应当将主条目的哪些列用作对需要主行标识的表的查找。此文件定义用于外键标识查找的列名称和用于主表（例如 CATEGORY 和 PRODUCT）查询的选择谓词。可以省略此文件中具有已定义唯一索引（不包含标识）的表的条目。此参数是可选的。`IdResolveKeys.properties` 是缺省文件。此属性文件可以如以下示例之一所示来指定:

```
-propfile d:\WebSphere\CommerceServer\prop\idresprop.properties
```

```
-propfile d:\WebSphere\CommerceServer\prop\idresprop
```

如果在当前目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-propfile idresprop.properties
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-propfile idresprop
```

关于创建和指定新属性文件以供与标识解析器一起使用的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

-poolsize

要保留的标识数目。此参数是可选的。缺省值是 50。

-customizer

要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置标识解析器起作用的方式。`DB2ConnectionCustomizer.properties` 是缺省文件。定制程序属性文件可以如以下示例之一所示来指定:

```
-customizer d:\WebSphere\CommerceServer\prop\idres.properties
-customizer d:\WebSphere\CommerceServer\prop\idres
```

如果在当前目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-customizer idres.properties
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-customizer idres
```

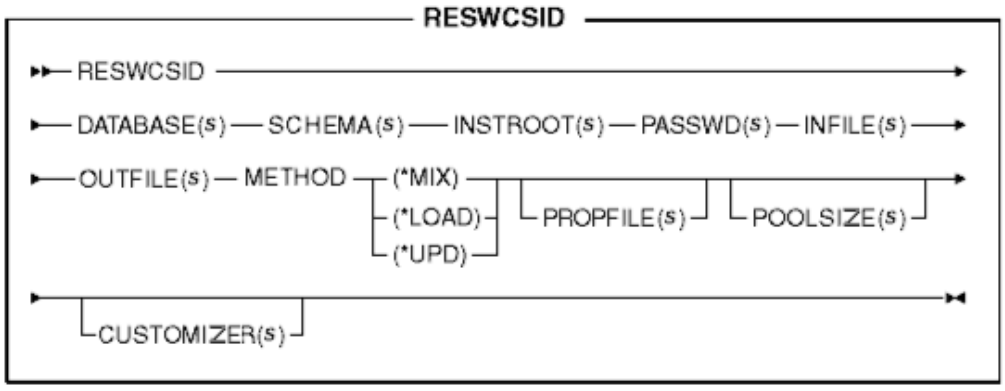
关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

-schemaname

目标数据库模式的名称。此参数是可选的。

如果在运行命令时未指定此参数，则此命令在指定了 SchemaName 的值的定制程序属性文件中查找 name=value 对。如果此对出现在属性文件中，则命令使用指定的值。如果此参数的命令行或属性文件规范都不存在，则命令缺省为数据库中表的模式名。

400



注: 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

DATABASE

目标数据库名称，如在关系数据库目录中所显示

SCHEMA

目标数据库模式名称；此名称与实例名称相同

INSTROOT

WebSphere Commerce 实例根路径的全名，例如 /QIBM/UserData/WebCommerce/instances/*instance_name*

PASSWD

WebSphere Commerce 实例的密码

INFILE

包含表记录的输入 XML 文档的名称。

OUTFILE

要产生的输出 XML 文件的名称；此文件可用作对装入程序的输入

METHOD

要用于处理输入文件的方法

- 如果输入文件中的***所有***记录都***不存在***于数据库中，则使用装入方法（*LOAD）处理输入文件。
- 如果输入文件中的***所有***记录都***存在***于数据库中，则使用更新方法（*UPD）处理输入文件。
- 如果输入文件中***仅有部分***记录***存在***于数据库中，则使用混合方法（*MIX）处理输入文件。

PROPFILE

包含 name=value 对格式的 Java 属性的文本文件。此属性文件设置标识解析器解析标识的方式。它用于描述应当将主条目的哪些列用作对需要主行标识的表的查找。此文件定义用于外键标识查找的列名称和用于主表（例如 CATEGORY 和 PRODUCT）查询的选择谓词。可以省略此文件中具有已定义唯一索引（不包含标识）的表的条目。此参数是可选的。IdResolveKeys.properties 是缺省文件。此属性文件可以如以下示例之一所示来指定:

```
PROPFILE(/wc/prop/idresprop.properties)
```

```
PROPFILE(/wc/prop/idresprop)
```

如果在当前目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
PROPFILE(idresprop.properties)
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
PROPFILE(idresprop)
```

关于创建和指定新属性文件以供与标识解析器一起使用的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

POOLSIZE

要保留的标识数目。此参数是可选的。缺省值是 50。

CUSTOMIZER

要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置标识解析器起作用的方式。缺省文件是 ISeries_RESWCSID_Customizer.properties。定制程序属性文件可以如以下示例之一所示来指定:

```
CUSTOMIZER(/wc/prop/idres.properties)
```

```
CUSTOMIZER(/wc/prop/idres)
```

如果在当前目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
CUSTOMIZER(idres.properties)
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
CUSTOMIZER(idres)
```

关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

解析技术:

根据是否使用属性文件，标识解析器使用两个或三个以下技术的组合解析标识。

- **内部别名解析**

  使用内部别名标识解析时，在源 XML 文档中以别名替换唯一键（标识）。然后在 XML 文件的其它位置使用此别名来引用该元素。

  必须在整个 XML 文件中一致地使用内部别名。例如，如果通讯录标识 ADDRBOOK_ID 的别名是 @addrbook_1，则文件中对该标识的所有外键引用都必须使用 @addrbook_1。

  请注意别名对于特定 XML 文件是瞬时的。它们不会被保存；并且如果不再引进该别名的话它无法在独立的 XML 文件中使用。而在“商店服务”发布期间，发布连接 XML 文件以使解析能够跨所有数据发生。

- **唯一索引解析**

  标识解析器还可分析数据库模式以确定是否存在满足其需求的唯一索引。标识解析器仅当属性文件中不存在被分析的表的条目时或者在没有属性文件时查找唯一索引。如果这些条件为真，则执行唯一索引检查。唯一索引在其存在且未包含表的主键时视为有效。

- **属性文件规范**

  标识解析器让您能够使用备用的 Java 属性文件来描述应当将主条目的哪些列用作对需要主行标识的表的查找。

随 WebSphere Commerce 提供的样本商店归档文件在它们的 XML 文件中使用内部别名。这使商店归档文件可以跨数据库移植。尽管唯一索引和属性文件规范技术也允许跨数据库的移植，但是用户可以随时更改唯一列，这在稍后这些技术用于标识解析时会造成问题。例如，如果用户更改了唯一列，则必须在属性文件定义中更改列名。然而，利用内部别名技术，数据库中的更改不会必然导致 XML 或属性文件中的更改。在“商店服务”中发布或使用装入程序软件包发布期间，标识解析器会以唯一值替换别名。一旦装入数据，别名对用户是透明的。关于更多信息，请参阅第 289 页的附录 B, 『创建数据』。

标识解析器使用以下过程:

- 如果输入 XML 数据具有来自主表的已具有硬编码标识（例如 “12345” ）的元素，则标识解析器不对该元素创建新标识。
- 如果输入 XML 数据具有来自主表的不具有标识的元素，则标识解析器查看数据库以确定是否已存在此元素的行。

  在数据库中查找元素要求使用元素中的其它列构成唯一键。可以在属性文件中指定这些其它列，或者，可让标识解析器确定要使用哪些列。

  – 如果正在使用属性文件且在属性文件中存在正被分析的表的条目，则标识解析器使用在属性文件中指定的列构成唯一键。
  – 如果未在使用属性文件或属性文件中不存在正被分析的表的条目，则标识解析器使用唯一索引解析。

    唯一索引解析使用表中任何已指定的唯一索引作为定位标识的方法。例如，MEMBER_ID 加上 IDENTIFIER 是 CATALOG 表的唯一索引，因此可用作 CATALOGDSC 表的外键 CATALOG_ID 的解析点。

  如果存在着具有相同唯一键的行，则元素被认为已存在于数据库中；否则，它被视为一项新数据。
- 如果元素作为数据库中的行已存在，则将检索并保存其标识以便以后可以使用。否则，将由标识解析器使用 KEYS 或 SUBKEYS 表中提供的值生成新标识。
- 如果在 XML 文档中指定了元素的内部别名（例如 “store_id_1” ），则将该别名与标识关联，以便以后可以使用同一内部别名查找标识。
- 后面的 XML 文档元素（它们需要引用主表中的元素）使用内部别名（如果主表元素具有一个内部别名，例如 “store_id_1” ），或使用查找列的值（如果主表元素没有内部别名，例如 “WC2001@100” ）。对于上述两种情况的任一种，将使用指定的值查找实际标识，并用该标识替换值。
- 当产生了输出 XML 文档时，所有主表元素自身具有实际标识，且引用这些主表元素的所有元素使用实际标识引用它们（而不是使用上面提及的内部别名或查找列值）。这是完全解析的 XML 文档。

标识解析命令的方法:
标识解析命令让您选择 load、update 或 mixed 方法以处理输入文件。

装入方法:
标识解析器的装入方法用于为装入到数据库中的所有新记录生成新标识。

注: 如果为标识解析器指定装入方法，则输入文件中的记录不应已存在于数据库中。如果对标识解析器使用装入方法而源 XML 文件中的记录已存在于目标数据库中，则装入程序在您装入数据时将生成错误。标识解析器在标识解析期间将对 XML 文件中的记录指定新的主键；但是当您将数据装入数据库时，将生成错误。装入程序在处理重复记录时将不会停止；但是将报告错误且不会将重复记录装入数据库。

以下示例用于为对于数据库来说是新元素的数据元素生成标识:

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname wcsadmin
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname wcsadmin
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(WCSADMIN)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) INFILE(input.xml) OUTFILE(output.xml)
METHOD(*LOAD)
```

注: 有关相应的标识解析命令或脚本的位置，请参阅第 235 页的『使用装入程序软件包命令和脚本』。

更新方法:
如果为标识解析器指定更新方法，则输入文件中的记录应当已存在于数据库中。标识解析器如页面215中所述在数据库中定位标识。如果记录不存在于数据库中，则标识解析器无法解析此记录的标识，并指示已发生错误。以下示例用于定位已存在于数据库中的数据元素的标识:

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname wcsadmin
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname wcsadmin
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(WCSADMIN)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*UPD)
```

注: 有关相应的标识解析命令或脚本的位置，请参阅第 235 页的『使用装入程序软件包命令和脚本』。

混合方法:
如果输入数据文件包含已存在于数据库中的记录以及一些新的记录，则必须使用混合方法运行标识解析器。使用此方法，标识解析器仅当记录未存在于数据库中时为记录创建新标识。否则，从数据库获取现有标识。以下示例用于为新数据生成标识以及定位已存在于数据库中的数据元素的标识:

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname wcsadmin
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname wcsadmin
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(WCSADMIN)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*MIX)
```

注:

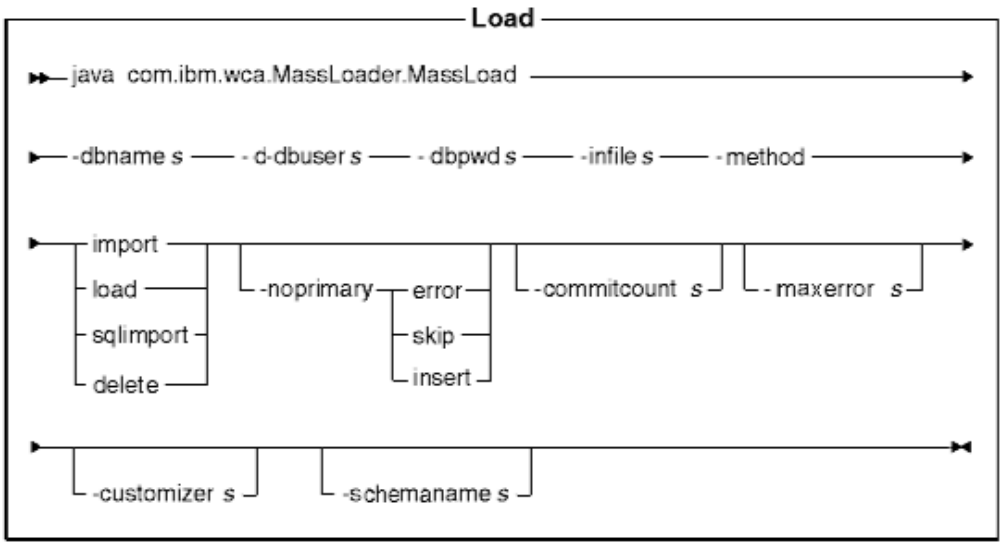
1. 有关相应的标识解析命令或脚本的位置，请参阅第 235 页的『使用装入程序软件包命令和脚本』。
2. 混合方法是对“商店服务”建议的方法。

关于设置和定制用于运行此命令的文件的详细信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

装入命令

此命令将 XML 输入文件装入目标数据库。

Windows AIX Solaris Linux



注:

1. 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数；因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。
2. 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

-dbname

目标数据库的名称

-dbuser

连接至数据库的用户的名称

-dbpwd

连接至数据库的用户的密码

-infile 输入 XML 文件的名称

-method

装入程序在将数据插入到数据库时要使用的操作方式

- load 方法使用来自数据库供应商的本机装入程序。您可以对本地和远程 Oracle 数据库都使用装入方法；但装入方法只能用于本地 DB2 数据库。
- 尽管导入方法可以用于将数据装入本地或远程数据库，但是它通常用于将数据装入到远程 DB2 数据库。此方法使用导入或更新选项（如果可以从数据库供应商处获得它的话）。如果您对不可用导入或更新选项的数据库指定此方法（例如 Oracle），则使用 JDBC 的 SQL 语句用于更新此数据库。
- SQL 导入（sqlimport）方法可以与本地和远程数据库一同使用。
- delete 方法从数据库删除数据。

-noprimary

当输入文件中记录的主键丢失时，装入程序必须执行的操作

- error 选项指示它应当将丢失主键报告为错误并终止。
- skip 选项跳过输入文件中没有主键的任何记录。
- 插入选项尝试插入或删除数据。

此参数是可选的。缺省操作是 error。

-commitcount

在使用操作的 SQL import 方法时，数据库提交发生前处理的记录数目。此参数是可选的。缺省数目是 1。

-maxerror

操作的 SQL import 方法中的错误数目，在这些错误发生后装入程序将终止。此参数是可选的。

-customizer

要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置装入程序起作用的方式。`MassLoadCustomizer.properties` 是缺省文件。定制程序属性文件可以如以下示例所示来指定:

```
-customizer d:\WebSphere\CommerceServer\prop\ml.properties
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-customizer ml
```

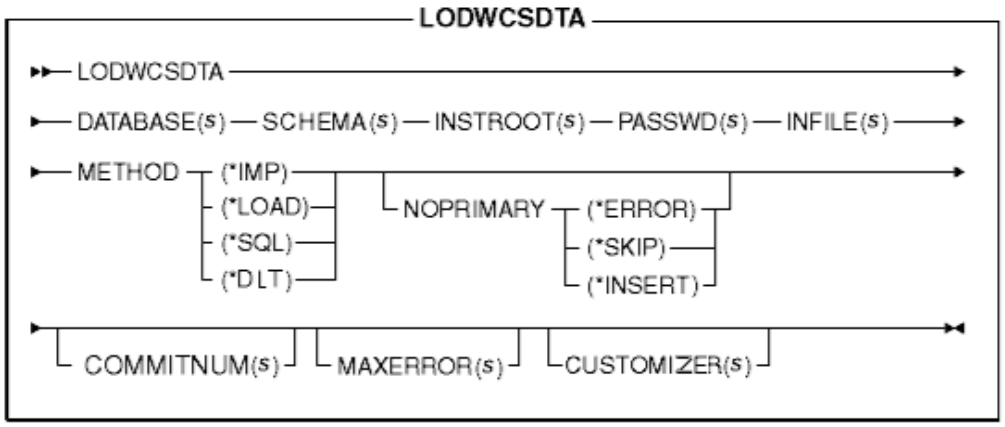
关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

-schemaname

目标数据库模式的名称。此参数是可选的。

如果在运行命令时未指定此参数，则此命令在指定了 SchemaName 的值的定制程序属性文件中查找 name=value 对。如果此对出现在属性文件中，则命令使用指定的值。如果此参数的命令行或属性文件规范都不存在，则命令缺省为数据库中表的模式名。

400



注: 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

DATABASE

目标数据库名称，如在关系数据库目录中所显示

SCHEMA

目标数据库模式名称；此名称与实例名称相同

INSTROOT

WebSphere Commerce 实例根路径的全名，例如 `/QIBM/UserData/WebCommerce/instances/`*instance_name*

PASSWD

WebSphere Commerce 实例的密码

INFILE

输入 XML 文件的名称

METHOD

装入程序在将数据插入到数据库时要使用的操作方式

- load 方法（*LOAD）使用来自数据库供应商的本机装入程序。您可以对本地和远程 Oracle 数据库都使用 load 方法（*LOAD）；但 load 方法（*LOAD）只能用于本地 DB2 数据库。
- 尽管 import (*IMP) 方法可以用于将数据装入本地或远程数据库，但是它通常用于将数据装入到远程 DB2 数据库。此方法使用导入或更新选项（如果可以从数据库供应商处获得它的话）。如果未提供 import 或 update 选项，将用使用 JDBC 的 SQL 语句更新数据库。
- SQL import（*SQL）方法可以与本地和远程数据库一同使用。
- delete（*DLT）方法从数据库删除数据。

NOPRIMARY

当输入文件中记录的主键丢失时，装入程序必须执行的操作

- error 选项（*ERROR）指示它应当将丢失主键报告为错误并终止。
- skip 选项（*SKIP）跳过输入文件中没有主键的任何记录。
- 插入选项（*INSERT）尝试处理（插入或删除）数据。

此参数是可选的。缺省操作是 error。

COMMITNUM

在使用操作的 SQL import 方法时，数据库提交发生前处理的记录数目。此参数是可选的。缺省数目是 1。

MAXERROR

操作的 SQL import 方法中的错误数目，在这些错误发生后装入程序将终止。此参数是可选的。

CUSTOMIZER

要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置标识解析器起作用的方式。缺省文件是 `ISeries_LODWCSDTA_Customizer.properties`。定制程序属性文件可以如以下示例所示来指定:

```
CUSTOMIZER(/wc/prop/ml.properties)
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
CUSTOMIZER(ml)
```

关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

装入命令的方法:

在装入数据之前，应当确定这三种处理方法中的哪一种将产生最佳结果。

装入方法:

在任何以下情况中，请考虑装入方法:

- 源数据是干净的，数据库不包含任何数据

  注: 干净的数据是不违反正装入到的表的任何约束的数据。
- 源数据是干净的，数据库不包含正被装入的数据
- 源数据是干净的，一个或多个目标表不包含主键，数据库不包含正被装入的数据

- 数据库是本地的 DB2 数据库
- 数据库是本地或远程 Oracle 数据库
- 当发生装入时，数据库未由其它用户或应用程序访问

400 使用装入方法时，数据被装入到数据库中。如果数据已经存在，则命令由于重复的键错误而失败，且显示“重复错误”消息。

对于使用装入方法存在以下限制:

- 装入方法无法插入或更新位数据字段中的数据。
- DB2 利用装入方法，只有新的记录才会插入到数据库中；不会更新现有的记录。
- DB2 装入方法可以仅用于本地（而不是远程）DB2 数据库。

导入方法:

Windows AIX Solaris Linux 利用 DB2 的导入方法，数据也可以装入到数据库中。如果数据已经存在，则不删除它，而是用新值更新它。在任何以下情况中，请考虑此方法:

- 数据库管理是 DB2
- 您不知道数据是否干净
- 您必须在列级别更新许多组同类数据
- 正将数据导入其中的所有表都具有主键

400 使用导入方法时，数据也被装入到数据库中。如果数据已经存在，则不删除它，而是用新值更新它。在任何以下情况中，请考虑此方法:

- 您不知道数据是否干净
- 数据已经存在于数据库中
- 正将数据导入其中的所有表都具有主键

对于使用导入方法存在以下限制:

- 数据库管理系统必须是 DB2 以便使用导入方法。
- 导入方法无法插入或更新位数据字段中的数据。
- 使用导入方法时，装入程序仅插入或更新已对其定义了主键的表；导入方法无法在不具有主键的表中插入或更新数据。如果输入记录仅对是主键的列具有值，则拒绝该记录。

SQL 导入方法:
使用 SQL 导入方法时，JDBC 或 SQL 用于向数据库更新或插入数据。如果数据尚未存在则插入数据，并更新现有的数据。在任何以下情况中，请考虑此方法:

- 您正在更新现有的数据并需要列级别的更新
- 一些数据不干净
- 数据库不是本地的

注: 如果正在使用产品顾问搜索空间同步，则对于装入数据必须使用 SQL 导入方法。

删除方法:
删除方法用于从数据库中删除在输入 XML 文档中的数据。元素必须包含主键的值或表的唯一索引。如果正被删除的数据具有另一表中的数据，而该表又以启用了“级联删除”的方式依赖于这些数据，则还将删除相关的数据。

比较方法:

- **SQL 导入和装入方法的比较**

  SQL 导入方法检查数据一致性（包括外键引用），并让您能够更新现有的数据。而装入方法则不。

- **导入和 SQL 导入方法的比较**

  导入和 SQL 导入方法执行类似的功能。导入方法通常更快，但是它需要磁盘空间来存放临时文件。

  导入方法仅可插入或更新对其定义了主键的表；而 SQL 导入方法则不要求对表定义了主键。

- **基于所使用数据库产品的方法比较**

  导入和装入方法使用对 DB2 优化的本机实用程序，而 SQL 导入方法使用 JDBC 调用（通用于许多数据库产品）。

性能注意事项:
当使用装入程序将大文档装入数据库时，请考虑以下事项:

- **Java 虚拟机（JVM）堆大小**

  缺省情况下，分配给 JVM 堆的最大内存是 64 MB。如果未增加此数值，则在装入过程中 JVM 最终可发生内存不足。可以通过在 Java 命令中使用 JVM -mx 选项来更改分配给 Java 堆的最大内存。

- **跟踪记录**

  装入大的 XML 文档时，跟踪记录程序可耗尽 JVM 堆。跟踪信息最常用于在运行失败的情况下对运行进行调试。如果对装入过程的跟踪是不必要的，则应当关闭跟踪。当关闭跟踪时将获得显著的性能提高。通过修改记录配置 XML 文档来关闭跟踪。关于修改记录配置 XML 文档的信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

- **提交计数**

  装入程序在以 SQL 导入方式运作时的缺省提交计数是 1。因此，缺省情况下每次对数据库进行更新或插入时将提交事务。为了改善装入程序在装入大文档时的性能，应增加提交计数。建议值为 “100”，但是根据服务器上物理内存量、DBMS 事务日志大小等，可以取更高的值。

  使用装入命令的 -commitcount *count* 选项来更改装入程序的提交计数（其中 *count* 是提交事务前执行的语句数）。

- **记录配置**

  通常装入数据时进展缓慢可产生于以下情况之一:

  – 调用装入程序的用户不具有许可权以写入在记录配置文档中所指定的目录或更新在记录配置文档中所指定的文件。
  – 在记录配置文档中指定为文件位置的目录不存在。
  – 在记录配置文档中指定为文件位置的驱动器不具有足够的空间。

  当更正所有这些问题时，可能需要通过修改记录配置文档（缺省情况下是 `WCALoggerConfig.xml`）来更改文件的指定位置。关于修改记录配置 XML 文档的信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

关于设置和定制用于运行此命令的文件的详细信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

用于转换和抽取数据的装入程序软件包命令

DTD 生成命令

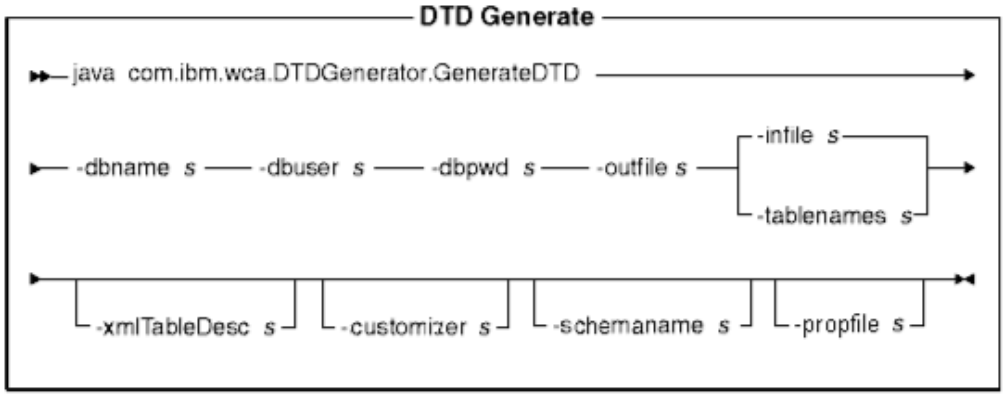
此命令创建用于装入程序软件包的 DTD。在整个数据装入过程中将使用此 DTD。根据调用命令的方式，DTD 生成程序可单独生成一个 DTD 或随 XML 模式一起生成一个 DTD。

DTD 生成程序可基于 WebSphere Commerce 数据库模式创建 DTD。如果使用随样本商店归档文件提供的 DTD 且未修改数据库模式，则无需使用 DTD 生成程序生成 DTD。所提供的 DTD 位于以下目录中:

- NT *drive*:\WebSphere\CommerceServer\xml\sar
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /QIBM/ProdData/WebCommerce/xml/sar

建议您使用所提供的 DTD。但是，如果定制数据库模式，则必须编辑所提供的 DTD 以与所作更改相匹配，或者创建新的 DTD。

Windows AIX Solaris Linux



注:

1. 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数;因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。
2. 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

-dbname
目标数据库的名称

-dbuser
连接至数据库的用户的名称

-dbpwd
连接至数据库的用户的密码

-outfile
输出 DTD 文件的名称（首选使用 .dtd 扩展名）

-infile 每行包含了数据库表名称的输入文件的名称

-tablenames
以逗号分隔且以引号（""）括起的表的名称

-xmltabledesc
要创建的 XML 模式文件的文件路径。此参数是可选的。

-customizer
要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置 DTD 生成程序起作用的方式。DB2ConnectionCustomizer.properties 是缺省文件。定制程序属性文件可以如以下示例所示来指定:

```
-customizer d:\WebSphere\CommerceServer\prop\dtdgen.properties
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-customizer dtdgen
```

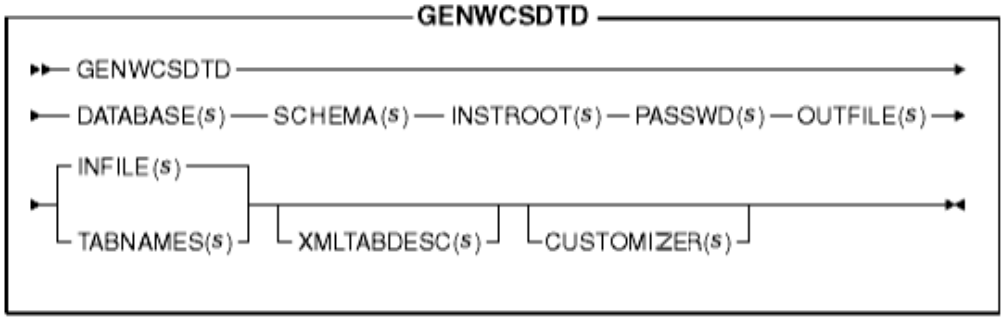
关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

-schemaname
目标数据库模式的名称。此参数是可选的。

如果在运行命令时未指定此参数，则此命令在指定了 SchemaName 的值的定制程序属性文件中查找 name=value 对。如果此对出现在属性文件中，则命令使用指定的值。如果此参数的命令行或属性文件规范都不存在，则命令缺省为数据库用户的名称。

-propfile
属性文件，可存储 Web 编辑器表单描述的帮助文本、缺省值和字段描述信息。此参数是可选的。

400



注: 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

DATABASE
目标数据库名称，如在关系数据库目录中所显示

SCHEMA
目标数据库模式名称；此名称与实例名称相同

INSTROOT
WebSphere Commerce 实例根路径的全名，例如 `/QIBM/UserData/WebCommerce/instances/`*instance_name*

PASSWD
WebSphere Commerce 实例的密码

OUTFILE
输出 DTD 文件的名称（首选使用 `.dtd` 扩展名）

INFILE
每行包含了数据库表名称的输入文件的名称

TABNAMES
以逗号分隔且以引号（`""`）括起的表的名称

XMLTABDESC
要创建的 XML 模式文件的文件路径。此参数是可选的。

CUSTOMIZER
要使用的定制程序属性文件名。此参数是可选的。定制程序属性文件设置 DTD 生成程序起作用的方式。缺省文件是 `ISeries_GENWCSDTD_Customizer.properties`。定制程序属性文件可以如以下示例所示来指定:

```
CUSTOMIZER(/wc/prop/dtdgen.properties)
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
CUSTOMIZER(dtdgen)
```

关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

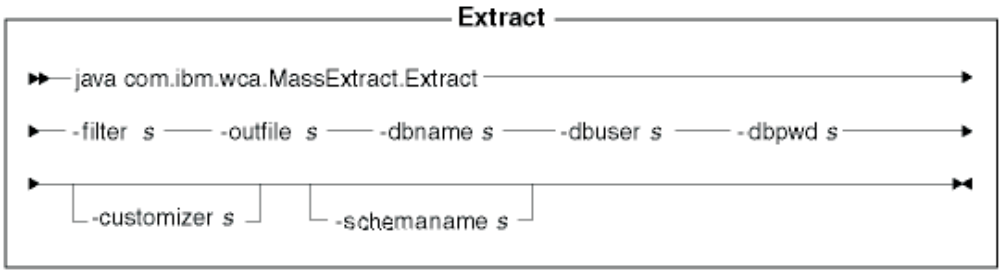
关于设置和定制用于运行此命令的文件的详细信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

抽取命令

此命令以 XML 文件格式从数据库中抽取选定的数据子集。

要使用抽取程序从数据库抽取数据，必须使用抽取过滤文件指定要抽取的数据。使用的抽取过滤取决于希望抽取的数据的类型。

Windows AIX Solaris Linux



注:

1. 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数;因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。
2. 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

-filter 抽取过滤文件的名称

-outfile
将存储抽取后的数据的输出 XML 文件的名称

-dbname
从其中抽取数据的数据库的名称

-dbuser
从其中抽取数据的数据库的数据库用户名

-dbpwd
从其中抽取数据的数据库的用户名关联密码

-customizer
要使用的定制程序属性文件名。定制程序属性文件设置抽取程序起作用的方式。`DB2ConnectionCustomizer.properties` 是缺省文件。定制程序属性文件可以如以下示例所示来指定:

```
-customizer d:\WebSphere\CommerceServer\prop\extract.properties
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
-customizer extract
```

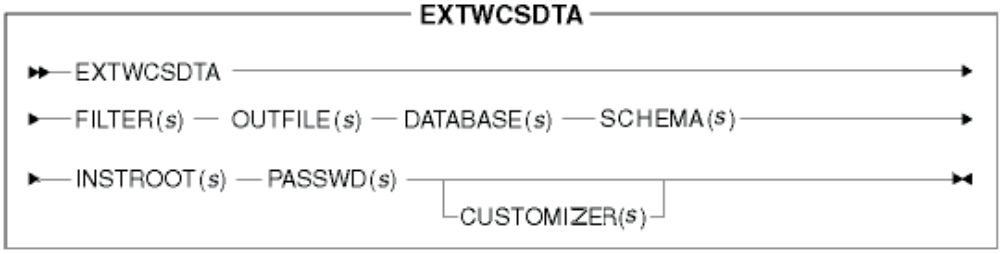
关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

-schemaname

从中抽取数据的数据库模式的名称。此参数是可选的。

如果在运行命令时未指定此参数，则此命令在指定了 SchemaName 的值的定制程序属性文件中查找 name=value 对。如果此对出现在属性文件中，则命令使用指定的值。如果此参数的命令行或属性文件规范都不存在，则命令缺省为数据库中表的模式名。

400



注: 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

FILTER

抽取过滤文件的名称

OUTFILE

将存储抽取后的数据的输出 XML 文件的名称

DATABASE

显示在关系数据库目录中的从其中抽取数据的数据库的名称

SCHEMA

从其中抽取数据的数据库模式的名称; 与实例名称相同

INSTROOT

WebSphere Commerce 实例根路径的全名，例如 `/QIBM/UserData/WebCommerce/instances/`*instance_name*

PASSWD

WebSphere Commerce 实例的密码

CUSTOMIZER

要使用的定制程序属性文件名。定制程序属性文件设置抽取程序起作用的方式。缺省文件是 `ISeries_EXTWCSDTA_Customizer.properties`。定制程序属性文件可以如以下示例所示来指定:

```
CUSTOMIZER(/wc/prop/extract.properties)
```

如果在类路径系统环境变量中指定的目录中存在此文件，则可以如以下示例所示指定相同的文件:

```
CUSTOMIZER(extract)
```

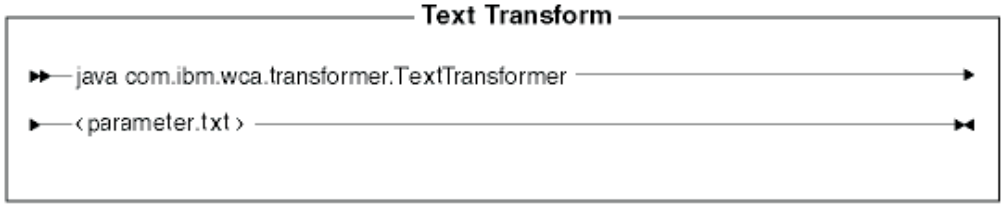
关于创建和指定新的定制程序属性文件的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

关于此命令的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

文本转换命令

此命令在字符定界的变量格式与 XML 格式之间转换数据。

Windows AIX Solaris Linux



注: 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数；因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。

参数值:

在参数文件（*parameter.txt*）中指定了以下值且以逗号作了分隔:

输入文件

要转换的文件的名称

模式文件

要用于转换的 XML 模式文件的名称

输出文件

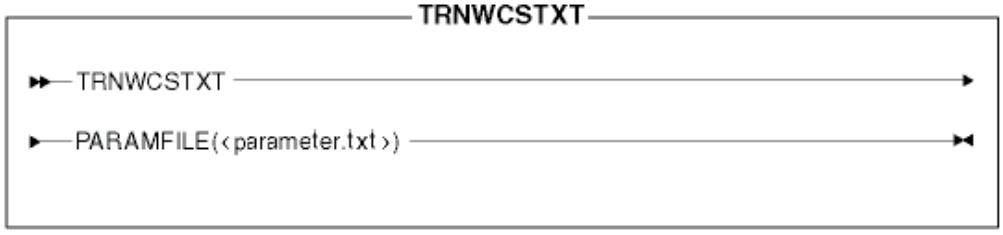
将存储转换后的数据的输出文件的名称

转换方法

要用于向输出文件添加数据的方法。如果要创建新文件，则指定**创建**；如果要在现有的数据文件后附加输出数据，则指定**附加**。

此文件也称为“清单”或“命令”文件。它可包含多行，每行有四个参数。

400



参数值:

在参数文件（*parameter.txt*）中指定了以下值且以逗号作了分隔:

输入文件

要转换的文件的名称

模式文件

要用于转换的 XML 模式文件的名称

输出文件

将存储转换后的数据的输出文件的名称

转换方法

要用于向输出文件添加数据的方法。如果要创建新文件，则指定**创建**；如果要在现有的数据文件后附加输出数据，则指定**附加**。

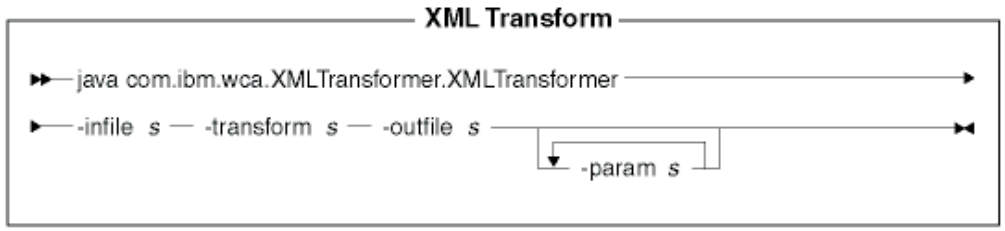
注：此文件也称为“清单”或“命令”文件。

关于此命令的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

XML 转换命令

此命令将 XML 文件转换为备用 XML 格式。

Windows AIX Solaris Linux



注:

1. 上图主要用作命令参数的参考。为此命令提供的命令文件或脚本列在第 235 页的『使用装入程序软件包命令和脚本』下，它充当实际 Java 命令的包装并接受相同参数；因此建议您使用命令文件或脚本，而不是直接调用 Java 命令。
2. 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

-infile 要转换的文件的名称

-transform

转换 XSL 映射规则文件的名称

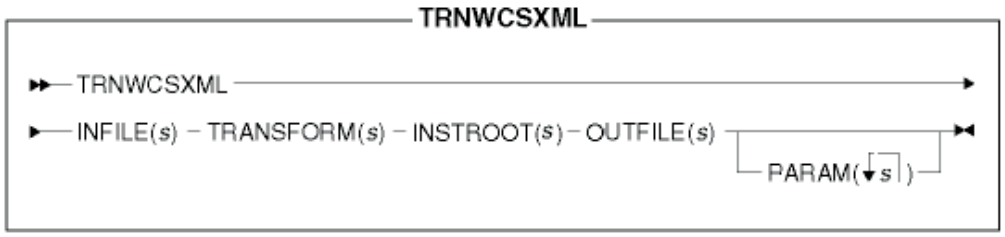
-outfile

将存储转换后的数据的输出 XML 文件的名称

-param

要传递给 XSL 映射规则文件的参数。***此参数是可选的。***此参数可以被指定多次以传递多个 name=value 对。

400



注: 指定为此命令参数的文件名可以放置在相对或绝对路径之前。

参数值:

INFILE

要转换的文件的名称

TRANSFORM

转换 XSL 映射规则文件的名称

INSTROOT

WebSphere Commerce 实例根路径的全名，例如 `/QIBM/UserData/WebCommerce/instances/`*instance_name*

OUTFILE

将存储转换后的数据的输出 XML 文件的名称

PARAM

要传递给 XSL 映射规则文件的参数。***此参数是可选的。***该字符串可包含多个值以传递多个 name=value 对。

关于此命令的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

装入程序软件包命令的相关工具

文本转换工具

文本转换工具有助于使用“文本转换”命令来处理字符定界的变量格式和 XML 格式之间的数据转换。提供了以下视图:

1. “文本模式编辑”视图让您能够创建和修改要用于转换的 XML 模式文件。
2. “转换命令编辑”视图让您能够创建和修改用于运行转换过程的实际命令。
3. “转换命令过程”视图让您能够启动转换过程。

关于此工具的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

XSL 编辑器

XML 转换程序使用 XSL 定义将一个 XML 文件转换为另一 XML 文件的规则。XSL 编辑器中的映射功能提供了可视界面，您可使用此界面建立从源 DTD 中的元素至目标 DTD 中的元素的关联。给定了两个 DTD，您可开发 XML 规则，该规则确定如何将符合第一个（源）DTD 的 XML 文件转换为符合第二个（目标）DTD 的文件。

关于此工具的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

Web 编辑器

Web 编辑器让您能够通过 Web 浏览器创建、删除和更改产品目录数据。用于查看和更新信息的数据输入表单对于 Web 编辑器处于中心位置。在最简单的情况下，表单对应于 WebSphere Commerce Server 数据库中的表。商店开发者可以选择是使用所提供的缺省表单，还是定制可用的表单。

关于此工具的更多信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。

装入商店数据

此部分提供了如何使用装入程序软件包命令行实用程序将商店数据装入 WebSphere Commerce Server 数据库的示例。

注:

1. 此部分中的示例在 Windows NT 环境中执行。关于在其它环境中运行这些命令的信息，请参阅《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本。
2. 尽管装入程序软件包命令行实用程序支持 DB2、DB2 iSeries 版，以及 Oracle 数据库，但是以下示例中仅包含了 DB2 的命令和选项。如果正在使用的数据库不是 DB2，请确保如《*IBM WebSphere Commerce 5.4 Catalog Manager 用户指南*》的最新版本所述，修改定制程序属性文件。

关于 WebSphere Commerce 数据库有用资源组的装入过程的信息，请参阅第 245 页的第 28 章，『装入 WebSphere Commerce 数据库有用资源组』和第 252 页的『装入数据库有用资源组』。

使用装入程序软件包命令和脚本

要运行装入程序软件包命令，请使用以下 WebSphere Commerce 目录中提供的脚本或命令:

- NT *drive*:\WebSphere\CommerceServer\bin
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\bin
- AIX /usr/WebSphere/CommerceServer/bin
- Solaris Linux /opt/WebSphere/CommerceServer/bin
- 400 QWEBCOMM 本机库

脚本和命令如下:

- Windows

dtdgen.cmd	DTD 生成命令
idresgen.cmd	标识解析命令
massload.cmd	装入命令
massextract.cmd	抽取命令
txttransform.cmd	文本转换命令
xmltransform.cmd	XML 转换命令

- AIX Solaris Linux

dtdgen.sh	DTD 生成外壳程序脚本
idresgen.sh	标识解析外壳程序脚本
massload.sh	装入外壳程序脚本
massextract.sh	抽取外壳程序脚本
txttransform.sh	文本转换外壳程序脚本
xmltransform.sh	XML 转换外壳程序脚本

- 400

GENWCSDTD	DTD 生成命令
RESWCSID	标识解析命令
LODWCSDTA	装入命令
EXTWCSDTA	抽取命令
TRNWCSTXT	文本转换命令
TRNWCSXML	XML 转换命令

解析标识的示例

本部分中描述的标识解析的示例使用 Business “多乐五金店”样本商店的商店有用资源文件。

因为此示例基于将新数据装入到 WebSphere Commerce Server 数据库中，所以我们将使用装入方法。

如果您稍后需要修改 XML 文档内的某些元素，您可以使用更新方法来完成。更新方法应比装入方法运行更快，因为使用更新方法不分配新标识。使用更新方法，将执行数据库查询以定位标识并在未找到标识的情况下报告错误。关于此过程如何工作的更多信息，请参阅从第 215 页开始的讨论。

如果输入 XML 文件包含已存在于数据库中的元素，也包含未存在于数据库中的元素，则使用混合方法。使用混合方法时，首先执行数据库查找，并在未找到记录的情况下将标识指定给元素。如果不能确定，请使用混合方法。尽管装入和更新方法提供了比混合方法更快的性能，但是通过使用混合方法产生的已解析 XML 文件更有可能做到装入无错误。

关于标识解析器如何工作的讨论，请参阅第 217 页的『标识解析命令的方法』。

用内部别名在 XML 文件中解析标识

要在将数据装入到 WebSphere Commerce Server 数据库之前使用内部别名来解析标识，请按以下示例所示运行标识解析命令:

1. 确保路径包含了目录，该目录包含如第 235 页的『使用装入程序软件包命令和脚本』中所列的适当的标识解析命令或脚本。

   对于此示例，您将使用标识解析命令 `idresgen.cmd`。
2. 创建工作目录。

   对于此示例，创建以下目录: `c:\WebSphere\CommerceServer\runtime\test`。
3. 确保输入 XML 文件和所有被引用的 DTD 文件位于标识解析器可找到的位置。

   对于此示例，请执行以下操作:

   a. 将 `c:\WebSphere\CommerceServer\samplestores\ToolTech\ToolTech_en_US_fr_FR.sar` 复制到 `c:\WebSphere\CommerceServer\runtime\test`。

   b. 从 Windows 命令提示符，输入以下命令:

   ```
   cd c:\WebSphere\CommerceServer\runtime\test
   ```

   c. 从 Windows 命令提示符，输入以下命令:

   ```
   jar -xvf ToolTech_en_US_fr_FR.sar
   ```

   这将 Business “多乐五金商店”样本商店的 XML 文件抽取到 `c:\WebSphere\CommerceServer\runtime\test\data`。

d. 从 Windows 命令提示符，输入以下命令:

```
cp c:\WebSphere\CommerceServer\xml\sar\store.dtd
c:\WebSphere\CommerceServer\runtime\test\data
```

这将 store.dtd 文件复制到 c:\WebSphere\CommerceServer\runtime\test\data。

e. 从 Windows 命令提示符，输入以下命令:

```
cp c:\WebSphere\CommerceServer\xml\sar\DBLoadMacros.dtd
c:\WebSphere\CommerceServer\runtime\test\data
```

这将 DBLoadMacros.dtd 文件复制到 c:\WebSphere\CommerceServer\runtime\test\data。

f. 从 Windows 命令提示符，输入以下命令:

```
cp c:\WebSphere\CommerceServer\xml\sar\fulfillment.dtd
c:\WebSphere\CommerceServer\runtime\test\data
```

这将 fulfillment.dtd 文件复制到 c:\WebSphere\CommerceServer\runtime\test\data。

4. 通过创建适当的 WebSphere Commerce Server 数据库实例，确保 WebSphere Commerce 模式已与必要的引导程序数据一起装入到数据库中。

注: 关于创建实例的信息，请参阅您的操作系统的《WebSphere Commerce 安装指南》。

此示例使用的 WebSphere Commerce Server 数据库实例称为 *mall*。将从此数据库的 KEYS 和 SUBKEYS 表中获取主键和外键；因此，标识解析器在未正确装入数据库的情况下将无法解析标识。

5. 在 store.xml 文件中，将找到以下元素:

```
<storeent
   STOREENT_ID="@storeent_id_1"
   MEMBER_ID="&MEMBER_ID;"
    TYPE="S"
    IDENTIFIER="ToolTech"
   SETCURR="USD"
/>
```

store.xml 的此元素映射到数据库中的 storeeent 表；它的 STOREENT_ID MEMBER_ID、TYPE、IDENTIFIER 和 SETCURR 属性映射到该表的列。storeent_id_1 规范是 STOREENT_ID 属性值的内部别名；&MEMBER_ID; 是实体参数。在可使用装入程序装入实体 &MEMBER_ID; 的值之前，必须对它作替换。&MEMBER_ID; 的值在 DBLoadMacros.dtd 宏文件中定义；此值是从该文件中替换的。当标识解析器遇到 @storeent_id_1 时，它查看主表高速缓存以了解 storeent 是否存在。因为它是主表，因此 storeeent 存在。标识解析器读取该表的计数器，将之递增，然后用结果来替换内部别名。以同样方式处理 store.xml 文件中的所有其它此类条目。

6. 从 Windows 命令提示符，输入以下命令:

```
idresgen -dbname mall -dbuser db2admin -dbpwd db2admin -infile store.xml
-outfile c:\WebSphere\CommerceServer\runtime\test\data\store1.xml -method load
```

store1.xml 中第一个输出 XML 片段与以下相似:

```
<storeent
   STOREENT_ID="10001"
    MEMBER_ID="-2001"
    TYPE="S"
    IDENTIFIER="ToolTech"
    SETCCURR="USD"
/>
```

store1.xml 中第二个 XML 片段与以下相似:

```
<store
   STORE_ID="10001"
    DIRECTORY="ToolTech"
   FFMCENTER_ID=""
    LANGUAGE_ID="-1"
    STOREGRP_ID="-1"
    ALLOCATIONGOODFOR="43200"
    BOPMPADFACTOR="0"
    DEFAULTBOOFFSET="2592000"
    FFMCSELECTIONFLAGS="0"
    MAXBOOFFSET="7776000"
    REJECTEDORDEXPIRY="259200"
   RTNFFMCTR_ID=""
    PRICEREFFLAGS="0"
    STORETYPE="B2B"
/>
```

未解析 FFMCENTER_ID 和 RTNFFMCTR_ID 属性。并不解析内部别名 @ffmcenter_id_1，而是用空引号（""）替换它。这是个错误。由于违反了外键约束，所以您无法使用装入程序正确装入数据。标识解析器无法解析此内部别名，因为在使用引用 FFMCENTER 表的别名之前，尚未处理该表。要解决这个问题，请选择以下三个选项之一:

- 选项 1:
  a. 输入以下命令以对照 fulfillment.xml 文件（其中定义了 FFMCENTER 表）运行标识解析器:

     ```
     idresgen -dbname mall -dbuser db2admin -dbpwd db2admin
     -infile fulfillment.xml -outfile
     c:\WebSphere\CommerceServer\runtime\test\data\fulfillment1.xml
     -method load
     ```

     fulfillment1.xml 输出文件中的已解析元素与以下相似:

     ```
     <fulfillment-asset>
     <ffmcenter
         FFMCENTER_ID="10001"
         MEMBER_ID="-2001"
        NAME="ToolTech Home"
        DEFAULTSHIPOFFSET="0"
         MARKFORDELETE="0"
     />
     </fulfillment-asset>
     ```

  b. 从生成的输出文件（fulfillment1.xml）中获取 FFMCENTER_ID 键，并对 store.xml 工作副本（位于 c:\WebSphere\CommerceServer\runtime\test\data）中所有出现

@ffmcenter_id_1 的地方以该键替换。

c. 输入以下命令:

```
idresgen -dbname mall -dbuser db2admin -dbpwd db2admin -infile store.xml
-outfile c:\WebSphere\CommerceServer\runtime\test\data\store1.xml
-method load
```

store1.xml 输出文件中的完全解析元素与以下相似:

```
<store-asset>
<storeent
    STOREENT_ID="10151"
    MEMBER_ID="-2001"
    TYPE="S"
    IDENTIFIER="ToolTech"
    SETCCURR="USD"
 />
<store
    STORE_ID="10151"
    DIRECTORY="ToolTech"
    FFMCENTER_ID="10001"
    LANGUAGE_ID="-1"
    STOREGRP_ID="-1"
    ALLOCATIONGOODFOR="43200"
    BOPMPADFACTOR="0"
    DEFAULTBOOFFSET="2592000"
    FFMCSELECTIONFLAGS="0"
    MAXBOOFFSET="7776000"
    REJECTEDORDEXPIRY="259200"
    RTNFFMCTR_ID="10001"
    PRICEREFFLAGS="0"
    STORETYPE="B2B"
 />
<vendor
    VENDOR_ID="10001"
    STOREENT_ID="10151"
    VENDORNAME="Tooltech Vendor"
    MARKFORDELETE="0"
 />
<dispentrel
   AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ProductBean"
    DEVICEFMT_ID="-1"
    DISPENTREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogProductDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispentrel
   AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ItemBean"
    DEVICEFMT_ID="-1"
    DISPENTREL_ID="10002"
    MBRGRP_ID="0"
    PAGENAME="CatalogItemDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispcgprel
    CATGROUP_ID="0"
    DEVICEFMT_ID="-1"
    DISPCGPREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogCategories.jsp"
    STOREENT_ID="10151"
```

```
    RANK="0"
 />
<invadjcode
    ADJUSTCODE="PCNT"
    INVADJCODE_ID="10001"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="SPLG"
    INVADJCODE_ID="10002"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="DISC"
    INVADJCODE_ID="10003"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<rtnreason
    REASONTYPE="C"
    RTNREASON_ID="10001"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="WPR"
 />
<rtnreason
    REASONTYPE="B"
    RTNREASON_ID="10002"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="DEF"
 />
<rtnreason
    REASONTYPE="M"
    RTNREASON_ID="10003"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="ERR"
 />
<rtnreason
    REASONTYPE="M"
    RTNREASON_ID="10004"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="WPS"
 />
</store-asset>
```

- 选项 2:
  a. 通过添加任何在 fulfillment.xml 中唯一的内容（包含对 fulfillment.dtd 的引用）来将 fulfillment.xml 和 store.xml 文件合并到 store.xml，确保下面所显示的 ffmcenter 元素在 store 元素之前。

     ```
     <ffmcenter
        FFMCENTER_ID="@ffmcenter_id_1"
        MEMBER_ID="&MEMBER_ID;"
        NAME="ToolTech Home"
        DEFAULTBOOFFSET="0"
         MARKFORDELETE="0"
     />
     ```

  b. 对照合并后的文件运行标识解析器。
- 选项 3: 使用第 252 页的『装入数据库有用资源组』中所述的过程装入商店有用资源数据组。

使用标识解析器指定属性文件

可通过使用 -propfile 参数修改标识解析器解析标识的方式。缺省属性文件是 IdResolveKeys.properties；但是可在调用标识解析命令时修改它或指定您自己的文件。

- Windows AIX Solaris Linux IdResolveKeys.properties 位于以下目录中:
  - NT *drive*:\WebSphere\CommerceServer\properties
  - 2000 *drive*:\Program Files\WebSphere\CommerceServer\properties
  - AIX /usr/WebSphere/CommerceServer/properties
  - Solaris Linux /opt/WebSphere/CommerceServer/properties

  如果运行标识解析器时未将此文件放置在当前目录中，则可将它放置在类路径环境变量中定义的目录中。也可指定到此文件的全路径。
- 400 要更改 IdResolveKeys.properties，请从 /QIBM/ProdData/WebCommerce/properties 目录复制它，将它保存到 /*instroot*/xml 目录，然后对新文件进行任何必要的更改。

  注: 上面目录位于由 RESWCSID 命令使用的类路径中。

属性文件规范具有比内部别名的使用更高的优先级。

此处是 store.xml 文件中的样本 XML 片段:

```
<store
   STORE_ID="@storeent_id_1"
    DIRECTORY="ToolTech"
   FFMCENTER_ID="@ffmcenter_id_1"
   LANGUAGE_ID="&en_US;"
    STOREGRP_ID="-1"
    ALLOCATIONGOODFOR="43200"
   BOPMPADFACTORr="0"
    DEFAULTBOOFFSET="2592000"
    FFMCSELECTIONFLAGS="0"
    MAXBOOFFSET="7776000"
    REJECTEDORDEXPIRY="259200"
   RTNFFMCTR_ID="@ffmcenter_id_1"
    PRICEREFFLAGS="0"
    STORETYPE="B2B"
/>
```

如果运行标识解析器时将 -propfile 指定为 c:\WebSphere\CommerceServer\runtime\test\data\myPropFile 且指定的文件 myPropFile.properties 包含以下条目:

```
NAMEDELIMETER=@
SELECTDELIMETER=:
FFMCENTER=@FFMCENTER_ID@MEMBER_ID:10051 -2001
```

当处理商店元素时，标识解析器使用 where 子句 10051 和 -2001 在数据库中查询 FFMCENTER 表。然后使用对此值返回的索引为 FFMCENTER_ID 解析标识。

关于使用此命令的更多信息，请参阅第 211 页的『标识解析命令』。

装入数据的示例

当解析了 XML 文件中的标识（如果需要）时，即已准备好将数据装入 WebSphere Commerce Server 数据库。

注: 如果您正确解析了 XML 数据中的标识，则源 XML 文件不应包含以下任何内容:

- 以 at（ ）符号开头的词
- 以 ampersand（&）符号开头的词
- 带有空引号（″″）的标识

如果出现了这些符号中的任何一个则表示尚未准备好装入 XML 文件。

本部分中描述的装入数据的示例使用在第 236 页的『解析标识的示例』中解析的 `fulfillment1.xml` 文件。

要将数据装入 WebSphere Commerce Server 数据库，请如以下示例所示运行装入命令:

1. 确保路径包含了目录，该目录包含如第 235 页的『使用装入程序软件包命令和脚本』中所列的适当的装入命令或脚本。

   对于此示例，请使用 `massload.cmd`。
2. 创建工作目录。

   对于此示例，使用您在第 236 页的『解析标识的示例』中创建的名为 `c:\WebSphere\CommerceServer\runtime\test\data` 的目录。
3. 确保输入 XML 文件位于装入程序可找到的位置。

   对于此示例，确保在第 236 页的『解析标识的示例』中创建的 `fulfillment1.xml` 输出文件位于 `c:\WebSphere\CommerceServer\runtime\test\data` 中。
4. 切换至您的工作目录。

   对于此示例，从 Windows 命令提示符下输入以下命令:

   ```
   cd c:\WebSphere\CommerceServer\runtime\test\data
   ```
5. 确保备份了 WebSphere Commerce Server 数据库，以便在发生不可恢复的错误时可以从备份中恢复数据库。
6. 对您解析的 XML 文件运行装入命令以将数据装入到目标数据库。

   对于此示例，从 Windows 命令提示符下输入以下命令:

   ```
   massload -dbname mall -dbuser db2admin -dbpwd db2admin -infile
   c:\WebSphere\CommerceServer\runtime\test\data\fulfillment1.xml
   -method sqlimport -commitcount 50
   ```

即使要装入的元素少于 50 个，此示例对 -commitcount 指定了值 50。这是出于性能原因。缺省情况下，提交计数是 1。使用此缺省值将引起对每个写入数据库的记录都执行提交操作。上述示例中将数值设置为 50 确保了仅当装入成功才发生数据库 I/O 且在发生错误的情况下不向数据库写入任何东西。然而，如果有大量数据要装入，建议不要将提交计数值设置得与元素数目一样大，原因如下:

- 高的提交计数值将引起高的内存消耗。
- 当提交计数值小于元素的数目时，至少会向数据库写入一些数据。根据 -maxerror 的值，较小的 -commitcount 值确保了在超出了最大错误数且工具终止之前，有一些数据写入了数据库。-maxerror 的缺省值是 1。

-noprimary 选项的缺省值是错误，这样在遗漏主键时工具会报告错误并终止。

由于这些示例不按由“商店服务”使用且在第 245 页的『数据库有用资源装入顺序』中描述的顺序装入商店有用资源，因此在第 236 页的『解析标识的示例』中创建的 store1.xml 文件可能违反一些表的完整性约束。如果尝试使用装入方法装入 store1.xml 而不加修改，则约束违反将引起数据库进入暂挂状态。因此为了简单，使用装入命令的此示例是基于 fulfillment.xml 文件的已解析版本的，该文件唯一的外键是在样本商店中定义的 MEMBER_ID 的外键。此示例装入已解析的 fulfillment1.xml 文件（该文件是第 236 页的『解析标识的示例』中的输出），并使用 SQL 导入方法。当您不能确定 XML 文件的内容是否干净时，请如此示例中所示使用带有适当设置的 -commitcount 和 -maxerror 参数的 SQL 导入方法，以便报告所有对数据库约束的违反，而不改变数据库或危及数据库的完整性。

关于使用此命令的更多信息，请参阅第 218 页的『装入命令』。

第 28 章 装入 WebSphere Commerce 数据库有用资源组

如果您不希望创建所有的数据库有用资源并在发布之前将它们封装在商店归档文件中，则您可以使用 WebSphere Commerce 装入程序软件包来装入数据库有用资源组。

本章的第一部分解释了 WebSphere Commerce 数据库有用资源组，以及如何确定分组。第二部分描述了将这些数据库有用资源组装入到 WebSphere Commerce 数据库的过程。在阅读本部分之前，应当彻底地复查第 207 页的第 27 章，『装入商店数据的概述』中的信息，这些信息将有助于您了解使用装入程序软件包装入数据库有用资源组时所需的知识。

数据库有用资源组

将数据库有用资源分成组以简化创建和装入过程。这些*数据库有用资源组*包含一个逻辑上相关的一组表。组织数据库有用资源组的顺序对于装入很重要，因为在装入数据之间的关系之前，数据必须已存在。

要装入商店的整个数据库有用资源集，需要遵循『数据库有用资源装入顺序』。要装入单组数据库有用资源，您需要确保该组在逻辑上是完整的。例如，在发布一个商店归档文件时，您可以选择忽略产品目录数据库有用资源，它可以稍后再发布。在这种情况下，任何依赖于产品目录的数据库有用资源（库存、价格列表以及一些装运和税务数据）也保持未发布。要发布省略的数据，请确保产品目录数据库有用资源在逻辑上是完整的：即，必须提供基本商品、产品目录条目和属性等。您还必须发布相关的数据库有用资源（它们必须在逻辑上对自身是完整的）。换句话说，每个库存标识都必须定义了适当的库存、价格、装运和税款。在此情况下，逻辑上完整的相关产品目录数据作为一个整体称为*产品目录数据库有用资源组*。

该指南的上一章中描述的 WebSphere Commerce 数据库有用资源可以安排在各个组中。一个组是一组逻辑上完整的数据，它可以独立装入。每个数据库有用资源组包含 WebSphere Commerce 数据库表并具有第 301 页的附录 E，『数据库有用资源组』中所述的外部相关性。表的列表是基于 WebSphere Commerce 样本商店的，然而列表适用于任何一般商店。请记住每个数据库有用资源组的表列表并非是详尽无遗的，而是作为总体指南提供。您可能要根据商店的特定需要，包含或排除一些表。

数据库有用资源装入顺序

要遵循特定顺序才能成功地装入数据库有用资源组。每个组都被看作是结构上完整的并独立于其它数据库有用资源组。然而，数据库有用资源组中有外键关系。这样的关系（带有来自其它组的数据）被称为数据库有用资源组的*外部相关性*。

在将数据库有用资源组装入 WebSphere Commerce 数据库之前，必须满足该数据库有用资源组的外部相关性。任何定义为给定数据库有用资源组的外部相关性的组都必须首先装入。可在第 301 页的『数据库有用资源组相关性』中找到外部相关性和相关表的列表。

注: WebSphere Commerce 商店需要一个商店所有者。您可以使用缺省组织（它可以作为缺省所有者可用）。Business 要装入此组，请创建新的组织，而不是使用缺省组织。

按照以下顺序装入数据库有用资源组:

1. 仅依赖于引导程序数据的数据库有用资源组。
   a. 首先装入**组织**数据库有用资源。
2. 依赖于实现所有者的数据库有用资源组。
   a. **实现**数据库有用资源。除了组织数据库有用资源组之外，其它几个数据库有用资源组对此组中定义的数据都具有直接或间接的外部相关性。
3. 依赖于商店所有者组织的数据库有用资源组。
   a. **访问控制**数据库有用资源依赖于商店所有者组织（ORGENTITY_ID）。其它所有数据库有用资源组对该组中定义的数据都不具有相关性，这意味着可以随时装入访问控制数据库有用资源。然而，访问控制所有者必须与商店所有者相同。
   b. **商店**数据库有用资源依赖于商店所有者组织（ORGENTITY_ID）。

   商店可以引用一个实现中心。商店所有者组织还可以是实现中心的所有者组织。
4. 依赖于商店数据库有用资源的数据库有用资源组。可以按任何顺序装入以下组:
   a. **竞销**数据库有用资源。
   b. **命令**数据库有用资源。
   c. **货币**数据库有用资源。
   d. **策略**数据库有用资源。
   e. **装运**数据库有用资源。
   f. **税款**数据库有用资源。
5. 其它数据库有用资源组。
   a. **产品目录**数据库有用资源依赖于装运和税款数据库有用资源组。
   b. **商店缺省**数据库有用资源对装运数据库有用资源组具有外部相关性。如果装运数据库有用资源不存在，则无需填充此组。
   c. **合同**数据库有用资源依赖于组织有用资源。不直接装入合同数据库有用资源。关于更多信息，请参阅第 258 页的『发布合同有用资源』。您应该在其它数据库有用资源组之后装入合同有用资源。

晴参阅第 301 页的附录 E, 『数据库有用资源组』以查看 WebSphere Commerce 样本商店所形成的数据库有用资源组的内容。

装入商店

为了帮助您装入数据库有用资源，可以从 WebSphere Commerce Web 站点中获得样本数据包。这些数据包基于 WebSphere Commerce 样本商店并包含以下步骤中的文件。您可以从以下地址下载这些数据包:

- Business `http://www.ibm.com/software/webservers/commerce/wc_be/downloads.html`
- Professional `http://www.ibm.com/software/webservers/commerce/wc_pe/downloads.html`

要将整个商店的 XML 数据装入 WebSphere Commerce 数据库中，请执行以下操作:

1. 复查以下信息:
   a. 第 289 页的附录 B, 『创建数据』
   b. 第 301 页的附录 E, 『数据库有用资源组』，您需要知道将会影响哪些 WebSphere Commerce 数据库有用资源文件和数据库表。
   c. 第 207 页的第 27 章, 『装入商店数据的概述』，它提供装入程序软件包的背景信息。
2. 计划您针对整个商店数据库有用资源集的装入过程。无论您是希望如第 252 页的『装入数据库有用资源组』中所示装入单个数据库有用资源组，还是希望装入整个商店，基本的过程都是相同的。在下一步中，将使用或创建以下文件用于装入过程:
   a. 每个组的一个或多个数据库有用资源文件。当您装入整个商店时，需要所有已创建的数据库有用资源文件。例如，您将需要 *database asset*.xml 文件（如 campaign.xml、catalog.xml 或 currency.xml），以及独立的特定于商店支持的语言环境的 *database asset*.xml 文件。随 WebSphere Commerce 样本商店在以下目录中附带了此类文件的示例:
      - NT *drive*:\WebSphere\CommerceServer\samplestores*sample store name*\data\
      - 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores*sample store name*\data\
      - AIX /usr/WebSphere/CommerceServer/samplestores/*sample store name*/data/
      - Solaris Linux /opt/WebSphere/CommerceServer/samplestores/*sample store name*/data/
      - 400 /QIBM/ProdData/WebCommerce/samplestores/*sample store name*/data/

      注意，不是所有的数据库有用资源组都需要特定于语言环境的信息。
   b. 一个新的 XML 文件（它巩固了商店所有的 XML 数据库有用资源文件）包含 XML 实体引用，并包含整个商店的根元素。这被称为*主数据库有用资源组 XML 文件*。您可以在样本数据包中找到此文件，它名为 store-data-assets.xml。
   c. 一个新的 DTD 文件（它定义了数据库有用资源组中的 XML 文件所需的所有数据类型），它称为*主数据库有用资源组 DTD 文件*。您可以在样本数据包中找到此文件，它名为 store-data-assets.dtd。
   d. 第二个 DTD 文件，它定义了外部相关性。您可能需要在*主数据库有用资源组 DTD 文件*中包含此文件。您可以在样本数据包中找到此文件，它名为 ForeignKeys.dtd。
   e. 第三个 DTD 文件，它包含所有 WebSphere Commerce 表的定义。wcs.dtd 文件已存在于 WebSphere Commerce 中，且位于以下目录:
      - NT *drive*:\WebSphere\CommerceServer\schema\xml\
      - 2000 *drive*:\Program Files\WebSphere\CommerceServer\schema\xml\
      - AIX /usr/WebSphere/CommerceServer/schema/xml/

- Solaris Linux /opt/WebSphere/CommerceServer/schema/xml/
- 400 /QIBM/ProdData/WebCommerce/schema/xml/

您可能需要在*主数据库有用资源组 DTD 文件*中包含此文件。如果您尚未定制 WebSphere Commerce 模式，则可以使用此文件而无需修改。

3. 如本指南先前的各章中所指导，创建数据库有用资源 XML 文件。如果你完成了有用资源章节中的任务，则这些 XML 文件已存在。数据库有用资源文件不得在文件开始处包含任何 DTD 声明或页面伪指令，因为这可能在连接文件时引起冲突。而且为了简单，您也可能决定不创建任何根元素。必须具有根元素的唯一文件是主数据库有用资源组 XML 文件。

   注: 如果有多种语言的数据库有用资源文件，则每个文件必须以 <?xml encoding = *locale specific encoding*> 开头。例如，英语数据库有用资源文件应该指定 <?xml encoding = "UTF-8"?>，而法语文件应该指定 <?xml encoding = "ISO-8859-1"?>。

4. 为整个商店数据集创建主数据库有用资源组 XML 文件。此文件包含引用实体以包含商店的各种数据库有用资源 XML 文件。通过使用外部引用实体，您可以连接 XML 文件以简化标识解析命令和装入过程。而且当同时装入多个组时，每个 XML 文件中使用的内部别名相对于该组中其它 XML 数据库有用资源文件或其它组而言可以是外部的。XML 分析程序将在外部引用的地方替换以由外部引用实体引用的文件的内容。

   将以下用于装入整个商店数据集的示例用作指南，可基于此摘要创建您的数据库有用资源组文件:

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "all-store-assets.dtd">
<import>
<!Fulfillment data group -->
&fulfillment.xml;

<!-- Store data group -->
&store.xml;
&en_US_store.xml;
&fr_FR_store.xml;

<!-- Tax data group -->
&tax.xml;
&en_US_tax.xml;
&fr_FR_tax.xml;
&taxfulfill.xml;

<!-- Shipping data group -->
&shipping.xml;
&en_US_shipping.xml;
&fr_FR_shipping.xml;
&shipfulfill.xml;

<!-- Catalog data group -->
&catalog.xml;
&en_US_catalog.xml;
&fr_FR_catalog.xml;
&storecatalog.xml;
&storefulfill.xml;
&offering.xml;
&store-catalog-tax.xml;
&store-catalog-shipping.xml;

<!-- Currency data group -->
```

```
&currency.xml;
&en_US_currency.xml;
&fr_FR_currency.xml;

<!-- Campaign data group -->
&campaign.xml;
&en_US_campaign.xml;
&fr_FR_campaign.xml;

<!-- Business policy data group -->
&businesspolicy.xml;
&en_US_businesspolicy.xml;
&fr_FR_businesspolicy.xml;

<!-- Access control data group -->
&accesscontrol.xml;
&en_US_accesscontrol.xml;
&fr_FR_accesscontrol.xml;

<!-- Other data groups -->
&command.xml;
&store-default.xml;
</import>
```

其中

- import 是 XML 文档的根元素。已在随 WebSphere Commerce 提供的 wcs.dtd 文件中定义了根元素，并且此根元素包含了所有 WebSphere Commerce 数据库表的定义。然而，如果定制了 WebSphere Commerce 模式，则可能需要使用不同的根元素。您可以生成新的 DTD 文件来反映定制的模式，或者可以更新现有的 wcs.dtd 文件。
- all-store-assets.dtd 是指将在下一步中创建的主数据库有用资源组 DTD 文件的名称。
- 注释的文本将商店不同的数据库有用资源组分隔开。
- *&database asset*.xml; 是引用数据库有用资源 XML 文件的 XML 实体。路径和位置在数据库有用资源组 DTD 文件中定义。此名称将更改以便与已为每个组创建的数据库有用资源文件相匹配。
- &*locale_database asset*.xml; 对商店支持的每种语言而言都是必需的。如果您的商店只使用一种语言，则将只引用一个文件。如果商店支持多种语言，则对每种语言都需要一个引用。上面的摘要假定商店支持英语和法语。

5. 创建一个主数据库有用资源组 DTD 文件，它定义以上实体以及数据库有用资源所需的其它 DTD 文件。

   使用以下用于整个商店数据库有用资源集的示例作为指南，您可以创建主数据库有用资源组 DTD 文件:

```
<!ENTITY % wcs.dtd SYSTEM "absolute path for WebSphere Commerce wcs.dtd file">
%wcs.dtd;

<!ENTITY % NonStoreForeignKeys.dtd SYSTEM "NonStoreForeignKeys.dtd">
%NonStoreForeignKeys.dtd;
<!ENTITY fulfillment.xml SYSTEM "data/fulfillment.xml">
<!ENTITY en_US_fulfillment.xml SYSTEM "data/en_US/fulfillment.xml">
<!ENTITY fr_FR_fulfillment.xml SYSTEM "data/fr_FR/fulfillment.xml">

<!ENTITY store.xml SYSTEM "data/store.xml">
<!ENTITY en_US_store.xml SYSTEM "data/en_US/store.xml">
<!ENTITY fr_FR_store.xml SYSTEM "data/fr_FR/store.xml">

<!ENTITY tax.xml SYSTEM "data/tax.xml">
```

```
<!ENTITY en_US_tax.xml SYSTEM "data/en_US/tax.xml">
<!ENTITY fr_FR_tax.xml SYSTEM "data/fr_FR/tax.xml">
<!ENTITY taxfulfill.xml SYSTEM "data/taxfulfill.xml">

<!ENTITY shipping.xml SYSTEM "data/shipping.xml">
<!ENTITY en_US_shipping.xml SYSTEM "data/en_US/shipping.xml">
<!ENTITY fr_FR_shipping.xml SYSTEM "data/fr_FR/shipping.xml">
<!ENTITY shipfulfill.xml SYSTEM "data/shipfulfill.xml">

<!ENTITY catalog.xml SYSTEM "data/catalog.xml">
<!ENTITY en_US_catalog.xml SYSTEM "data/en_US/catalog.xml">
<!ENTITY fr_FR_catalog.xml SYSTEM "data/fr_FR/catalog.xml">
<!ENTITY store-catalog.xml SYSTEM "data/store-catalog.xml">
<!ENTITY storefulfill.xml SYSTEM "data/storefulfill.xml">
<!ENTITY offering.xml SYSTEM "data/offering.xml">
<!ENTITY store-catalog-tax.xml SYSTEM "data/store-catalog-tax.xml">
<!ENTITY store-catalog-shipping.xml SYSTEM "data/store-catalog-shipping.xml">

<!ENTITY currency.xml SYSTEM "data/currency.xml">
<!ENTITY en_US_currency.xml SYSTEM "data/en_US/currency.xml">
<!ENTITY fr_FR_currency.xml SYSTEM "data/fr_FR/currency.xml">

<!ENTITY campaign.xml SYSTEM "data/campaign.xml">
<!ENTITY en_US_campaign.xml SYSTEM "data/en_US/campaign.xml">
<!ENTITY fr_FR_campaign.xml SYSTEM "data/fr_FR/campaign.xml">

<!ENTITY businesspolicy.xml SYSTEM "data/businesspolicy.xml">
<!ENTITY en_US_businesspolicy.xml SYSTEM "data/en_US/businesspolicy.xml">
<!ENTITY fr_FR_businesspolicy.xml SYSTEM "data/fr_FR/businesspolicy.xml">

<!ENTITY accesscontrol.xml SYSTEM "data/accesscontrol.xml">
<!ENTITY en_US_accesscontrol.xml SYSTEM "data/en_US/accesscontrol.xml">
<!ENTITY fr_FR_accesscontrol.xml SYSTEM "data/fr_FR/accesscontrol.xml">

<!ENTITY command.xml SYSTEM "data/command.xml">
<!ENTITY store-defaults.xml SYSTEM "data/store-defaults.xml">
```

其中

- wcs.dtd 是指包含了定义在其数据库有用资源组之外的数据的 DTD 文件。此文件（随 WebSphere Commerce 提供）还定义了在数据库有用资源组 XML 文件中使用的根元素。
- NonStoreForeignKeys.dtd 是指定义了除根元素之外的其它元素的 DTD 文件。此文件包含数据库有用资源组之外外部相关性所有的 XML 实体引用声明和定义。这样，XML 文件引用了外键值（这些外键值没有作为数据库有用资源组的一部分来创建，而它们必须在此组之前已装入到数据库）。

  注: 确保正确标识了路径。在此示例中，文件位于与主数据库有用资源组 DTD 文件相同的目录中。
- store.xml、en_US_store.xml 和 fr_FR_store.xml 是在主数据库有用资源组 XML 文件中使用的外部引用实体（假设您的商店支持英语和法语）。要使用引用，请遵循实体引用约定: *&alias_name;*。
- *database asset*.xml 是指从中装入数据库有用资源的 XML 文件的名称。此名称将更改以便与已为每个组创建的数据库有用资源文件相匹配。注意，特定于语言环境的 XML 文件位于以下目录中:
  - NT *drive*:\WebSphere\CommerceServer\samplestores*sample store name*\data*locale*\

– 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores*sample store name*\data*locale*\

– AIX /usr/WebSphere/CommerceServer/samplestores/*sample store name*/data/*locale*/

– Solaris Linux /opt/WebSphere/CommerceServer/samplestores/*sample store name*/data/*locale*/

– 400 /QIBM/ProdData/WebCommerce/samplestores/*sample store name*/data/*locale*/

- *path_database asset*.xml 文件对于商店支持的每种语言而言都是必需的，它位于以上目录中。如果您的商店只使用一种语言，则将只引用一个文件。如果商店支持多种语言，则每种语言都将需要一个特定于语言环境的文件。上面的摘要假定商店支持英语和法语。

6. 每个数据库有用资源组都需要在其域之外或其数据集之外定义的信息，因为每个组都可能具有外部相关性。可在 DTD 文件中提供此数据。例如，商店数据库有用资源组具有以下外部相关性:

```
  bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID,
bootstrap.SETCURR.SETCURR_ID, fulfillment.FFMCENTER.FFMCENTER_ID
```

   当装入一个数据库有用资源组或整个商店有用资源集时，必须从 WebSphere Commerce 数据库定义外部相关性。要使用此数据，请遵循对应的 XML 实体引用。例如，要使用由 ffmcenter_id 实体定义的数据，则要在 XML 文件中写入 &ffmcenter_id;。将以下用于商店数据库有用资源的示例作为指南，可基于此摘要（称为 NonStoreForeignKeys.dtd）创建 DTD 文件:

```
<!ENTITY en_US "-1"><!ENTITY fr_FR "-2">
<!ENTITY de_DE "-3">
<!ENTITY it_IT "-4">
<!ENTITY es_ES "-5">
<!ENTITY pt_BR "-6">
<!ENTITY zh_CN "-7">
<!ENTITY zh_TW "-8">
<!ENTITY ko_KR "-9">
<!ENTITY ja_JP "-10">
<!ENTITY MEMBER_ID "-2000">
<!ENTITY ffmcenter_id "10001">
```

   其中

   - MEMBER_ID 是标识商店所有者的内部引用号。
   - ffmcenter 是商店实现中心的引用号。因为商店可使用多个实现中心，因此可在 NonStoreForeignKeys.dtd 文件中定义多个实现中心。
   - *locale* 是每种语言环境的 WebSphere Commerce 引用号（由国家或地区和语言标识）。值位于 LANGUAGE 数据库表中。

   注: 如果正在将现有的商店归档文件分割成数据库有用资源组，请确保（比如说）用对应的实体引用 &ffmcenter_id; 替换所有对别名 @ffmcenter_id 的引用。

7. 一旦创建了所有数据文件，则如第 211 页的『标识解析命令』所述对照主数据库有用资源组 XML 文件运行 IDResolve 命令以解析数据。
8. 如第 218 页的『装入命令』中所述，对已解析的数据文件运行装入命令。要验证您的装入过程，请参阅日志文件:

- DB2 idresgen.db2.log 和 massload.db2.log
- Oracle idresgen.oracle.log 和 massload.oracle.log

日志文件位于以下目录中:

- NT *drive*:\WebSphere\CommerceServer\logs\
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\logs\
- AIX /usr/WebSphere/CommerceServer/logs/
- Solaris Linux /opt/WebSphere/CommerceServer/logs/
- 400 /QIBM/ProdData/WebCommerce/logs/

9. Business 如第 258 页的『发布商业帐户有用资源』中所述运行 AccountImport 命令。
10. 如果适用，如第 258 页的『发布合同有用资源』中所述发布合同。
11. 完成第 261 页的『通过复制到 WebSphere Commerce Server 发布商店前台有用资源和商店配置文件』中的任务

装入数据库有用资源组

为了帮助您装入数据库有用资源，可以从 WebSphere Commerce Web 站点中获得样本数据包。这些数据包基于 WebSphere Commerce 样本商店并包含以下步骤中的文件。您可以从以下地址下载这些数据包:

- Business http://www.ibm.com/software/webservers/commerce/wc_be/downloads.html
- Professional http://www.ibm.com/software/webservers/commerce/wc_pe/downloads.html

要将单个数据库有用资源组的 XML 数据装入 WebSphere Commerce 数据库，请执行以下操作:

1. 复查以下信息:
   a. 第 289 页的附录 B，『创建数据』
   b. 第 301 页的附录 E，『数据库有用资源组』，您需要知道将会影响哪些 WebSphere Commerce 有用资源文件和数据库表。
   c. 第 207 页的第 27 章，『装入商店数据的概述』，它提供装入程序软件包的背景信息。
2. 计划您的装入过程，并决定将要装入哪个数据库有用资源组。无论您是希望如第 246 页的『装入商店』中所示装入整个商店数据库有用资源集，还是希望装入单个数据库有用资源组，基本的过程都是相同的。在下一步中，将使用或创建以下文件用于装入过程:
   a. 一个或多个数据库有用资源文件，这取决于您选择了哪个组。例如，如果装入商店数据库组有用资源，则对于商店支持的每种语言环境，将需要 store.xml 文件和各自的 store.xml 文件。随 WebSphere Commerce 样本商店在以下目录中提供了此类文件的示例:

- NT *drive*:\WebSphere\CommerceServer\samplestores*sample store name*\data
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores*sample store name*\data
- AIX /usr/WebSphere/CommerceServer/samplestores/*sample store name*/data
- Solaris Linux /opt/WebSphere/CommerceServer/samplestores/*sample store name*/data
- 400 /QIBM/ProdData/WebCommerce/samplestores/*sample store name*/data

注意，不是所有的数据库有用资源组都需要特定于语言环境的信息。

b. 一个新的 XML 文件（它巩固了所有的 XML 数据库有用资源文件）包含 XML 实体引用，并包含数据库有用资源的根元素。这称为*主数据库有用资源组 XML 文件*。您可以在样本数据包中找到此文件，它名为 store-all-assets.xml。

c. 一个新的 DTD 文件（它定义了数据库有用资源组中的 XML 文件所需的所有数据类型），它称为*主数据库有用资源组 DTD 文件*。您可以在样本数据包中找到此文件，它名为 store-all-assets.dtd。

d. 第二个 DTD 文件，它定义了外部相关性。您可能需要在*主数据库有用资源组 DTD 文件*中包含此文件。您可以在样本数据包中找到此文件，它名为 ForeignKeys.dtd。

e. 第三个 DTD 文件，它包含所有 WebSphere Commerce 表的定义。wcs.dtd 文件已存在于 WebSphere Commerce 中，且位于以下目录:

- NT *drive*:\WebSphere\CommerceServer\schema\xml\
- 2000 *drive*:\Program Files\WebSphere\CommerceServer\schema\xml\
- AIX /usr/WebSphere/CommerceServer/schema/xml/
- Solaris Linux /opt/WebSphere/CommerceServer/schema/xml/
- 400 /QIBM/ProdData/WebCommerce/schema/xml/

您可能需要在*主数据库有用资源组 DTD 文件*中包含此文件。如果您尚未定制 WebSphere Commerce 模式，则可以使用此文件而无需修改。

3. 如本指南先前的各章中所指导，为您将装入的组创建数据库有用资源 XML 文件。如果你完成了有用资源章节中的任务，则这些 XML 文件已存在。数据库有用资源文件不得在文件开始处包含任何 DTD 声明或页面伪指令，因为这可能在连接文件时引起冲突。而且为了简单，您也可能决定不创建任何根元素。必须具有根元素的唯一文件是主数据库有用资源组 XML 文件。

   注: 如果有多种语言的数据库有用资源文件，则每个文件必须以 <?xml encoding = *locale specific encoding*> 开头。例如，英语数据库有用资源文件应该指定 <?xml encoding = "UTF-8"?>，而法语文件应该指定 <?xml encoding = "ISO-8859-1"?>。

4. 为每个您希望装入的组创建主数据库有用资源组 XML 文件。此文件包含引用实体以包含一个（或多个）数据库有用资源组中各种不同的 XML 文件。通过使用外部

引用实体，您可以连接 XML 文件以简化标识解析命令和装入过程。而且当同时装入多个组时，每个 XML 文件中使用的内部别名相对于该组中其它 XML 数据文件或其它组而言可以是外部的。XML 分析程序将在外部引用的地方替换以由外部引用实体引用的文件的内容。

将以下用于装入单个商店数据库有用资源组的示例用作指南，可基于此摘要创建您的数据库有用资源组 XML 文件:

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "store-assets.dtd">
<import>
&store.xml;
&en_US_store.xml;
&fr_FR_store.xml;
</import>
```

其中

- import 是 XML 文档的根元素。已在随 WebSphere Commerce 提供的 wcs.dtd 文件中定义了根元素，并且此根元素包含了所有 WebSphere Commerce 数据库表的定义。然而，如果定制了 WebSphere Commerce 模式，则可能需要使用不同的根元素。您可以生成新的 DTD 文件来反映定制的模式，或者您可以更新 wcs.dtd 文件。
- store-assets.dtd 是指将在下一步中创建的主数据库有用资源组 DTD 文件的名称。
- &store.xml; 是一个引用数据库有用资源组 XML 文件的 XML 实体。路径和位置在数据库有用资源组 DTD 文件中定义。此名称将更改以便与已为每个组创建的有用资源文件相匹配。
- *locale*_store.xml; 对于商店支持的每种语言而言都是必需的。如果您的商店只使用一种语言，则将只引用一个文件。如果商店支持多种语言，则对每种语言都将需要一个引用。上面的摘要假定商店支持英语和法语。

5. 创建一个主数据库有用资源组 DTD 文件，它定义以上实体以及组所需的其它 DTD 文件。

   将以下用于装入单个商店数据库有用资源组的示例用作指南，您可以为任何数据组创建主数据库有用资源组 DTD 文件:

   ```
   <!ENTITY % wcs.dtd SYSTEM "absolute path for WebSphere Commerce wcs.dtd file">
   %wcs.dtd;

   <!ENTITY % NonStoreForeignKeys.dtd SYSTEM "NonStoreForeignKeys.dtd">
   %NonStoreForeignKeys.dtd;
   <!ENTITY store.xml SYSTEM "store.xml">
   <!ENTITY en_US_store.xml SYSTEM "en_US/store.xml">
   <!ENTITY fr_FR_store.xml SYSTEM "fr_FR/store.xml">
   ```

   其中

   - wcs.dtd 是指包含了定义在其数据库有用资源组之外的数据的 DTD 文件。此文件（随 WebSphere Commerce 提供）还解析和定义了在数据库有用资源组 XML 文件中使用的根元素。
   - NonStoreForeignKeys.dtd 是指定义了除根元素之外的其它元素的 DTD 文件。此文件包含数据库有用资源组之外外部相关性所有的 XML 实体引用声明和定义。这样，XML 文件引用了外键值（这些外键值没有作为数据库有用资源组的一部分来创建，而它们必须在此组之前已装入到数据库）。

注: 确保正确标识了路径。在此示例中，文件位于与数据库有用资源组 DTD 文件相同的目录中。

- store.xml、en_US_store.xml 和 fr_FR_store.xml 是在数据库有用资源组 XML 文件中使用的外部引用实体。要使用引用，请遵循实体引用约定：*&alias_name;*。
- store.xml 是指从中装入数据库有用资源的组的数据文件。此名称将更改以便与已为每个组创建的数据库有用资源文件相匹配。注意，特定于语言环境的 XML 文件位于以下目录中:
  - NT *drive*:\WebSphere\CommerceServer\samplestores*sample store name*\data*locale*\
  - 2000 *drive*:\Program Files\WebSphere\CommerceServer\samplestores*sample store name*\data*locale*\
  - AIX /usr/WebSphere/CommerceServer/samplestores/*sample store name*/data/*locale*/
  - Solaris Linux /opt/WebSphere/CommerceServer/samplestores/*sample store name*/data/*locale*/
  - 400 /QIBM/ProdData/WebCommerce/samplestores/*sample store name*/data/*locale*/
- *path*_store.xml 对于商店支持的每种语言而言都是必需的，它位于以上目录中。如果您的商店只使用一种语言，则将只引用一个文件。如果商店支持多种语言，则每种语言都将需要一个特定于语言环境的文件。上面的摘要假定商店支持英语和法语。

6. 每个数据库有用资源组都需要在其域之外或其数据集之外定义的信息，因为每个组都可能具有外部相关性。可在 DTD 文件中提供此数据。例如，商店数据库有用资源组具有以下外部相关性:

```
  bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID,
bootstrap.SETCURR.SETCURR_ID, fulfillment.FFMCENTER.FFMCENTER_ID
```

   当装入一个数据组或整个商店数据集时，必须从 WebSphere Commerce 数据库定义以下外部相关性。要使用此数据，请遵循对应的 XML 实体引用。例如，要使用由 ffmcenter_id 实体定义的数据，则要在 XML 文件中写入 &ffmcenter_id;。将以下用于商店数据库有用资源组的示例作为指南，您可以基于此摘要（称为 NonStoreForeignKeys.dtd）创建 DTD 文件:

```
<!ENTITY en_US "-1"><!ENTITY fr_FR "-2">
<!ENTITY de_DE "-3">
<!ENTITY it_IT "-4">
<!ENTITY es_ES "-5">
<!ENTITY pt_BR "-6">
<!ENTITY zh_CN "-7">
<!ENTITY zh_TW "-8">
<!ENTITY ko_KR "-9">
<!ENTITY ja_JP "-10">
<!ENTITY MEMBER_ID "-2000">
<!ENTITY ffmcenter_id "10001">
```

   其中

   - MEMBER_ID 是标识商店所有者的内部引用号。

- `ffmcenter` 是商店实现中心的引用号。因为商店可使用多个实现中心，因此可在 `NonStoreForeignKeys.dtd` 文件中定义多个实现中心。
- *`locale`* 是每种语言环境的 WebSphere Commerce 引用号（由国家或地区和语言标识）。值位于 LANGUAGE 数据库表中。

注: 如果正在将现有的商店归档文件分割成数据库有用资源组，请确保（比如说）用对应的实体引用 `&ffmcenter_id;` 替换所有对别名 `@ffmcenter_id` 的引用。

7. 一旦创建了所有数据文件，则如第 211 页的『标识解析命令』所述对照数据库有用资源组 XML 文件运行 IDResolve 命令以解析数据。
8. 如第 218 页的『装入命令』中所述，对已解析的数据文件运行装入命令。要验证您的装入过程，请参阅日志文件:
   - DB2 `idresgen.db2.log` 和 `massload.db2.log`
   - Oracle `idresgen.oracle.log` 和 `massload.oracle.log`

   日志文件位于以下目录中:
   - NT *`drive`*`:\WebSphere\CommerceServer\logs\`
   - 2000 *`drive`*`:\Program Files\WebSphere\CommerceServer\logs\`
   - AIX `/usr/WebSphere/CommerceServer/logs/`
   - Solaris Linux `/opt/WebSphere/CommerceServer/logs/`
   - 400 `/QIBM/ProdData/WebCommerce/logs/`
9. Business 如第 258 页的『发布商业帐户有用资源』中所述运行 AccountImport 命令。
10. 如果适用，如第 258 页的『发布合同有用资源』中所述发布合同。
11. 完成第 261 页的『通过复制到 WebSphere Commerce Server 发布商店前台有用资源和商店配置文件』中的任务

第 29 章 发布商业帐户和合同

一些商店数据库有用资源（商业帐户和合同）不能由装入程序软件包装入。可以通过使用“商店服务”或从命令行发布这些数据库有用资源，作为“发布完整的商店”选项的一部分（如第 195 页的第 26 章，『发布完整的商店』中所述），或者可使用其对应的命令发布商业帐户和合同。这些命令如下:

- AccountImport — 从商店归档文件中的 `businessaccount.xml` 文件中创建商业帐户。
- ContractImportApprovedVersion — 从 `contract.xml` 文件中创建合同。如果合同处于活动状态，则此命令创建并部署合同。即使 `contract.xml file` 包含多个合同，也仅需要调用此命令一次。
- ProductSetPublish — 在创建商业帐户和合同之前，使产品集数据库表中的产品集数据与产品目录同步。“商店服务”和命令行发布调用 ProductSetPublish 命令，该命令接着调用 AccountImport 和 ContractImportApprovedVersion 命令。

注: 关于这些命令的更多信息，请参阅 WebSphere Commerce 联机帮助。

商业帐户有用资源以 XML 文件的格式包含在随 WebSphere Commerce 提供的一些样本商店归档文件中。然而，建议您使用所提供的工具创建商业帐户有用资源，而不是为这些有用资源创建 XML 文件。关于使用所提供的工具创建这些有用资源的更多信息，请参阅 WebSphere Commerce 联机帮助。用于发布商业帐户的指导包含在以下部分中，以备您选择发布随样本商店归档文件提供的对应的 XML 文件或创建自己的 XML 文件。

注: 如果您未使用“商店服务”来发布商业帐户或合同，则必须先发布商店和产品目录有用资源才能发布商业帐户和合同。特别的，您需要商店和产品目录标识，以及拥有商店的组织的标识，还有任何与该合同关联的买方组织的标识。如果合同的条款和条件未指定特定的产品目录，则您无需在发布商业帐户或合同之前发布产品目录。

如果使用“商店服务”或命令行发布来发布这些有用资源，请确保选择了产品目录选项，或者商店具有已发布的产品目录。如果使用对应的命令发布这些有用资源，请确保已将上面列出的有用资源装入数据库。

使用“商店服务”或命令行发布商业帐户和合同

您可以使用“商店服务”或从命令行使用发布实用程序来发布商业帐户和合同。为了使用“商店服务”或命令行发布商业帐户和合同，必须以商店归档文件格式封装有用资源。关于将商店前台有用资源封装为商店归档文件的更多信息，请参阅第 185 页的第 6 部分，『封装商店』。

使用“商店服务”或命令行发布时，可选择发布商店归档文件中所有类型的有用资源（包括商店前台有用资源、商店数据有用资源和资源绑定），或者可选择仅发布一种类型的有用资源。关于使用“商店服务”或命令行发布有用资源的详细的逐步指导，请参阅 WebSphere Commerce 联机帮助。

使用命令发布商业帐户和合同

如果不希望将有用资源封装为商店归档文件，则仍可使用对应的命令发布商业帐户和合同:

- AccountImport — 从商店归档文件中的 businessaccount.xml 文件中创建商业帐户。
- ContractImportApprovedVersion — 将已核准的或活动的合同从 XML 文件导入到 WebSphere Commerce Server。在导入合同之前，此命令确保正在导入的合同包含必要的条款和条件并且是有效的合同。
- ProductSetPublish — 在创建商业帐户和合同之前，使产品集数据库表中的产品集数据与产品目录同步。“商店服务”和命令行发布调用 ProductSetPublish 命令，该命令接着调用 AccountImport 和 ContractImportApprovedVersion 命令。

发布商业帐户有用资源

要发布商业帐户有用资源，请执行以下操作:

1. 使用管理控制台，更新视图注册表，或重新启动 WebSphere Commerce 实例。关于更多信息，请参阅 WebSphere Commerce 联机帮助。
2. 将 businessaccount.xml 复制到以下目录中:
   - NT `drive:\WebSphere\CommerceServer\xml\trading`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\trading`
   - AIX `/usr/WebSphere/CommereServer/xml/trading`
   - Solaris `/opt/WebSphere/CommerceServer/xml/trading`
   - Linux `/opt/WebSphere/CommerceServer/xml/trading`
   - 400 `/QIBM/UserData/WebCommerce/instances/instancename/xml/trading`
3. 打开 businessaccount.xml 并进行以下更改:
   - 使用您商店的商店标识替换出现 &STORE_IDENTIFIER; 的每一处。
   - 用商店的成员专有名称替换出现 &MEMBER_IDENTIFIER; 的每一处。

   注: 如果您正在处理 businessaccount.xml（它是使用“商店服务”创建的商店归档文件的一部分），则此步骤已完成。
4. 保存并关闭文件。
5. 打开管理控制台。以管理员身份登录。
6. 在浏览器中输入以下内容:
   - `https://hostname:8000/webapp/wcs/stores/servlet/AccountImport?fileName=businessaccount.xml&URL=URL to redirct to upon successful completion`

注: 关于命令语法和参数的更多信息，请参阅 WebSphere Commerce 联机帮助。

发布合同有用资源

要发布合同有用资源，请执行以下操作:

1. 将 contract.xml 复制到以下目录中:

注: 此路径是可以配置的。以下路径为缺省路径。

- NT `drive:\WebSphere\CommerceServer\xml\trading`
- 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\trading`
- AIX `/usr/WebSphere/CommerceServer/xml/trading`
- Solaris `/opt/WebSphere/CommerceServer/xml/trading`
- Linux `/opt/WebSphere/CommerceServer/xml/trading`
- 400 `/QIBM/UserData/WebCommerce/instances/`*`instancename`*`/xml/trading`

2. 打开 `contract.xml` 并进行以下更改:
   - 使用您商店的商店标识替换出现 &STORE_IDENTIFIER; 的每一处。
   - 用商店的成员专有名称替换出现 &MEMBER_IDENTIFIER; 的每一处。
3. 保存并关闭文件。
4. 打开管理控制台。以管理员身份登录。
5. 在浏览器中输入以下内容:
   - `https://`*`hostname:portnumber`*`/webapp/wcs/tools/servlet/ContractImportApprovedVersion?fileName=contract.xml &targetStoreId=store_id&URL=ContractDisplay`
6. 如果您的商店包含多个 `contract.xml` 文件（例如，特定于语言环境的合同文件），请对每个 `contract.xml` 文件重复步骤 1 到 5。

第 30 章 发布商店前台有用资源和商店配置文件

发布商店前台有用资源、HTML 和 JSP 文件、属性文件或资源绑定以及创建商店页面的图像和图形，是创建正常工作的商店的过程的一部分。可以使用“商店服务”或从命令行发布商店前台有用资源，作为“发布完整的商店”选项的一部分（如第 195 页的第 26 章，『发布完整的商店』中所述），或者可通过简单地将有用资源复制到 WebSphere Commerce Server 上的指定位置来发布商店前台有用资源。

如果发布包含在样本商店 Business “多乐五金店”和“新时尚”中的 JSP 文件，且计划配置商店使用协作功能，则还将需要发布作为该商店归档文件一部分的商店配置文件。这两个商店都包含以下商店配置文件:

- `tools_properties.zip`
- `tools_xml.zip`
- `runtime_xml.zip`

如果使用“商店服务”或从命令行发布完整的商店（选择所有发布选项），则也发布了商店配置文件。但是，如果选择通过将商店前台有用资源复制到 WebSphere Commerce Server 来发布它们，则还将必须将商店配置文件复制到 WebSphere Commerce Server。

使用“商店服务”或命令行发布商店前台有用资源和商店配置文件

可使用“商店服务”或从命令行使用发布实用程序来发布商店前台有用资源和商店配置文件。

注: 为了使用“商店服务”或发布实用程序发布商店配置文件，必须发布所有商店有用资源，包括 Web 有用资源和数据有用资源。不能选择仅发布商店配置文件。

为了使用“商店服务”或命令行发布商店前台有用资源和商店配置文件，必须以商店归档文件格式封装商店前台有用资源和商店配置文件。关于将商店前台有用资源封装为商店归档文件的更多信息，请参阅第 185 页的第 6 部分，『封装商店』。

使用“商店服务”或命令行发布时，可选择发布商店归档文件中所有类型的有用资源（包括商店前台有用资源、商店数据有用资源和资源绑定），或者可选择仅发布一种类型的有用资源。关于使用“商店服务”或命令行发布有用资源的详细的逐步指导，请参阅 WebSphere Commerce 联机帮助。

通过复制到 WebSphere Commerce Server 发布商店前台有用资源和商店配置文件

如果不希望将有用资源封装为商店归档文件，仍可通过将商店前台有用资源直接复制到 WebSphere Commerce Server 来发布它们。必须将 Web 有用资源（HTML、JSP 文件、图像和图形）复制到 Web 应用程序文档根路径。必须将资源绑定或属性文件复制到应用程序的属性路径。

要将商店前台有用资源和商店配置文件复制到 WebSphere Commerce Server，请执行以下操作:

1. 将 JSP 文件、HTML（包含文件、图像和图形）复制到商店 Web 应用程序文档根路径中的商店目录（*storedir*）：
   - NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstores.war*storedir*
   - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstores.war*storedir*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*

   其中 *storedir* 是 STORE 数据库表中 DIRECTORY 列的值。如果此值不存在，则您可以通过执行以下操作来添加一个值：select directory from store=*add relative directory name for publishing file assets*
2. 将资源绑定和属性文件复制到应用程序属性路径:
   - NT drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstores.war\WEB-INF\classes*storedir*
   - 2000 drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instancename*.ear\wcstores.war\WEB-INF\classes*storedir*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
3. 将商店配置文件复制到 WebSphere Commerce 配置文件 instance_name.xml 中定义的位置。此文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\instances*instancename*\xml
   - 2000 drive:\Program Files\WebSphere\CommerceServer\instances*instancename*\xml
   - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/xml
   - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/xml

- Linux /opt/WebSphere/CommerceServer/instances/*instancename*/xml
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/xml

商店配置文件被复制到以下位置:

- 将 runtime_xml.zip 复制到 StoresXMLPath。此路径是在 WebSphere Commerce 配置文件 instance_name.xml 中定义的。
- 将 tools_properties.zip 复制到 ToolsStoresPropertiesPath。此路径是在 WebSphere Commerce 配置文件 instance_name.xml 中定义的。
- 将 tools_xml.zip 复制到 ToolsStoresXMLPath。此路径是在 WebSphere Commerce 配置文件 instance_name.xml 中定义的。

4. 使用以下一种方法启动商店:
   - 使用 StoreCatalogDisplay 命令:
     StoreCatalogDisplay?storeId=*storeId*&catalogId=*catalogId*&langId=*langId*
     其中
     - storeId 是位于 STORE 数据库表 STORE_ID 列中的值。
     - catalogId 是位于 CATALOG 数据库表 CATALOG_ID 列中的值。
     - langId 是给定语言环境的 LANGUAGE 数据库表 LANGUAGE_ID 列的值。有关缺省 WebSphere Commerce 值的列表，请参阅 LANGUAGE 数据库表。
   - 如果您的商店基于 WebSphere Commerce 样本商店，请通过编辑以下目录中的 index.jsp 文件来组合商店的 URL:
     - NT *drive*:\WebSphere\CommerceServer\wc.ear\wcstores.war*storedir*
     - 2000 *drive*:\Program Files\WebSphere\CommerceServer\wc.ear\wcstores.war*storedir*
     - AIX /usr/WebSphere/CommerceServer/wc.ear/wcstores.war/*storedir*
     - Solaris Linux /opt/WebSphere/CommerceServer/wc.ear/wcstores.war/*storedir*
     - 400 /QIBM/ProdData/WebCommerce/wc.ear/wcstores.war/*storedir*

     为以下参数添加正确的值:
     - hostname 是 WebSphere Commerce 机器的全限定名称,
     - storeId 是位于 STORE 数据库表 STORE_ID 列中的值。
     - catalogId 是位于 CATALOG 数据库表 CATALOG_ID 列中的值。
     - langId 是给定语言环境的 LANGUAGE 数据库表 LANGUAGE_ID 列的值。有关缺省 WebSphere Commerce 值的列表，请参阅 LANGUAGE 数据库表。

     要在浏览器中查看您的商店，请启动以下 URL: http://*hostname*/webapp/wcs/stores/servlet/*storedir*/index.jsp

第 8 部分 向商店添加 WebSphere Commerce 功能部件

为了向您的商店添加一些在 WebSphere Commerce 中可用的功能，需要完成某些手工步骤。本部分中的各章讨论向您的商店添加以下功能:

第 31 章 向商店添加顾客关心

使用 Lotus® Sametime™ 服务器通过同步文本界面，WebSphere Commerce 中的顾客关心功能提供了实时客户服务支持。在商店中启用顾客关心时，顾客可以进入商店、单击一个链接并连接到客户服务代表（CSR）。然后，顾客可以通过因特网与 CSR 通信。

注: 本章讲述了如何在商店中启用顾客关心。然而，在商店可以启用顾客关心之前，必须首先安装 Sametime 服务器并配置它与 WebSphere Commerce 协同工作。关于更多信息，请参阅《*WebSphere Commerce 附加软件指南*》。还必须在管理控制台中注册 CSR，以使他们能够使用顾客关心。关于此任务以及顾客关心的整体概念和 CSR 如何使用顾客关心的更多信息，请参阅 WebSphere Commerce 联机帮助。

注:

如果您基于以下样本商店之一创建商店，则可以使用“商店服务”在商店中快速和简单地启用顾客关心: Business “多乐五金店”和“新时尚”。在使用“商店服务”发布了商店之后，请选择“商店”视图，然后选择商店，再选择**配置**并启用顾客关心功能。关于更详细指导，请参阅 WebSphere Commerce 联机帮助。

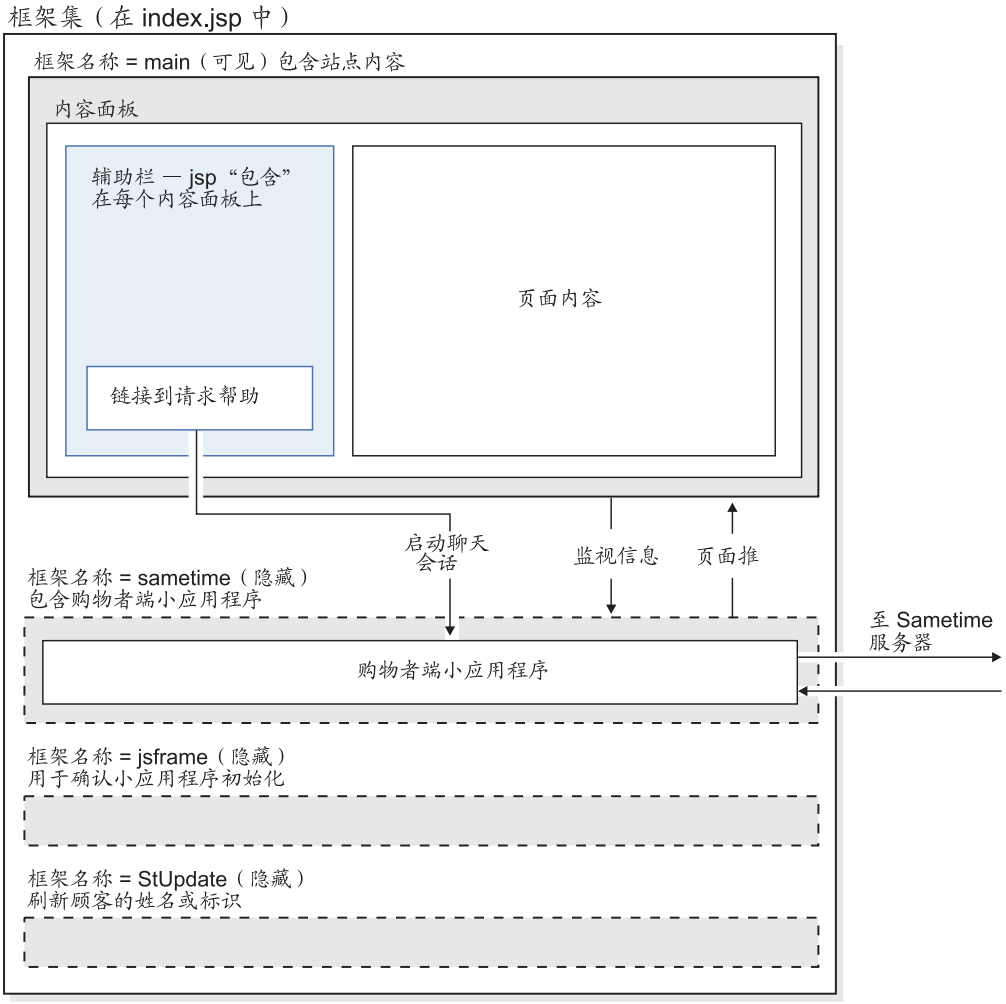
但是，如果没有使用样本商店作为基础来创建商店，则将必须执行一些操作在商店中启用顾客关心。本章的剩余部分讨论了在不是基于样本商店之一的商店中启用顾客关心所需的概念和步骤。

注: 样本商店 Business “多乐五金店”和“新时尚”演示了应如何实现顾客关心，并提供了您可以在商店中用来启用顾客关心的代码。本章将引用这两个商店的示例，以说明如何在商店中启用顾客关心。确保在阅读本章时已有了样本商店的最新版本（可从 WebSphere Commerce 产品 Web 站点获取）。

理解商店中的顾客关心

当顾客在启用了顾客关心的商店里选择顾客关心链接时（例如**与顾客助手实时聊天**），会启动包含聊天窗口的小应用程序。此小应用程序是在隐藏的框架集中运行的，这样的框架集不影响站点的外观。启动小应用程序时，它连接到 Lotus Sametime 服务器。

下图说明了框架集的构造。



框架集包含四个框架:

- 主框架: 该框架包含商店的内容，包括创建商店页面的文件（这些文件是创建页面主体的文件、头和页脚文件以及辅助栏文件）。此框架的内容是对商店访问者可见的。请注意主框架包含与 the Sametime 框架的以下连接: 指向顾客关心和监视信息的链接。在第 269 页的『使用顾客关心监视顾客』中更详细地讨论了监视信息。
- Sametime: 包含顾客关心小应用程序的框架。此框架对商店的访问者不可见。然而，当顾客单击链接启动小应用程序时，顾客将看到顾客关心窗口。此框架通过页面“推”功能，还将信息推向主框架。
- jsframe: 确认小应用程序已正确装入的框架。此框架的内容不显示给顾客。
- StUpdate: 该框架刷新顾客姓名或标识。

使用框架集

在框架集中启动顾客关心小应用程序会将小应用程序代码与商店页面中的代码分开。如上图说明，商店页面包含在框架集的主框架中，而小应用程序代码包含在 Sametime 框架中。由于在初次启动框架集时，小应用程序只下载一次，因此通过将小应用程序代码与商店页面分开，您减少了网络流量。如果顾客关心小应用程序不是框架集的一部分，则它将必须放在每个商店页面中，且每次访问新的商店页面时都要将它下载。

使用框架集还让您可以保持与 Sametime 服务器的连接。如果小应用程序是每个页面的一部分而不是框架集中的部分，则每次顾客访问新页面时将创建新的 Sametime 会话。由于顾客关心小应用程序是匿名登录到 Sametime 服务器的，因此每次顾客访问新页面时创建一个新会话会使您无法跟踪顾客在商店中的活动。使用框架集时，顾客的原始 Sametime 会话得到了保持，从而顾客活动因属性更改而发送回 Sametime 服务器。

使用框架集的问题

虽然使用框架集是在商店中实现顾客关心的建议方法，但您应了解使用框架集出现的以下问题:

- 单一入口点: 只有顾客在框架内浏览商店时，他们才可以使用顾客关心。同样，CSR 也只能通过框架集来监视顾客的行动。为确保顾客通过框架集来浏览商店，他们必须通过单一入口点来访问站点，例如通过商店主页（在样本商店中是 index.jsp）。如果顾客通过另一个页面访问商店（例如产品目录页面），则他们将不在框架集中。
- 书签: 当使用框架集时，顾客将只能将站点的主 URL 存为书签，而不是单个的页面。
- 刷新: 当顾客在框架集中并单击“刷新”时，会将他们带回框架集中编码的主框架地址，例如 index.jsp。
- 重新缩放浏览器窗口: 如果顾客在处于框架集中时重新缩放浏览器窗口，则浏览器可能自动重新装入条目地址。如果重新装入了条目地址，则可能终止与 Sametime 服务器的连接。不同的浏览器在此情况下的行为是不同的。
- 安全性: 当顾客通过框架集浏览站点时，每个单个框架以及框架集（地址栏中的 URL）都会保持自己的连接，无论是非安全的（http，缺省值是端口 80），还是安全的（https，缺省值是端口 443）。如果顾客通过非安全连接浏览商店，则框架集内的所有框架都是 HTTP 的形式。在此情况下，没有 SSL 的问题。但是如果顾客浏览到安全页面（例如注册页面），则框架集中的主框架将切换为 HTTPS，而剩余的框架仍保持为非安全的（http）。在此情况下，顾客将不能启动顾客关心小应用程序。浏览器不会授权启动小应用程序，因为小应用程序（安全的，端口 443）看起来是来自其它的服务器，而不是浏览器地址栏中的 URL（HTTP，端口 80）。在整个框架集重新变得安全之前，即，地址栏中的 URL 指向 HTTPS 之前，无法启动小应用程序。要使整个框架集重新安全起来，有以下选择:
  - 将整个框架集重定向至安全连接。但是，在这样做的时候，您将结束当前进程中的所有聊天，因为小应用程序要切换到新的 Sametime 连接。
  - 一进入站点即将整个框架集重定向至安全连接。这造成了轻微的性能损失，但对顾客会话添加了安全性和浏览保密性。

  将框架集重定向至安全连接的方法之一是将以下代码添加到 index.html 文件:

  ```
  <html>
  <head>
  <META HTTP-EQUIV=Refresh CONTENT="0;URL=https://hostname/webapp/wcs/stores/
  servlet/NewFashion/index.jsp>
  </head>
  <html>
  </head>
  ```

使用顾客关心监视顾客

顾客关心通过以下方式让您能够监视商店中正在与 CSR 通信的顾客

- 获取顾客的姓名或标识
- 确定顾客正在浏览的页面

- 跟踪购物车中的商品

将已定制代码添加到商店页面中以便获取这些信息。以下部分讨论了每个这样的监视功能是如何在样本商店中实现的。

获取顾客的姓名或标识

一旦启动了顾客关心小应用程序，且 CSR 已登录，CSR 就可以通过姓名或购物者标识来识别谁在使用小应用程序。样本商店中包含处理顾客关心小应用程序的专门代码，以确定顾客姓名或购物者标识。此代码确定顾客是临时顾客、购物车中有商品的临时顾客还是注册顾客，然后为该顾客指定一个姓名或标识，并将此名称传递回顾客关心小应用程序。然后这些名称显示给 CSR。例如，如果顾客是没有在购物车中放任何商品的临时顾客，则会为顾客指定一个已生成的标识，购物者标识为 -1002。例如，如果顾客是购物车中有商品的临时顾客，则会显示购物者标识，且如果该顾客已经注册，则他们的名字和姓氏也会显示。

样本商店通过向商店头文件添加以下代码（这将刷新 StUpdate 框架）来获取顾客姓名或标识。此代码包含在“新时尚”的 header.jsp 中，以及 Business “多乐五金店”的 NavHeader.jsp 中。

注: 每次顾客在商店中浏览新页面时，顾客姓名或标识都会刷新。

```
<script language="javascript">
    if (typeof top.updateStInfo == 'function')
       top.updateStInfo();
</script>
```

前面的代码刷新 StUpdate 框架中的以下内容。

```
//set Customer Name for LiveHelp if user is registered.

if (userRegistrationDataBean.findUser()) {
if (userRegistrationDataBean.getLastName() != null & &
userRegistrationDataBean.getLastName().length() > 0) {
     if(cmdcontext != null) {
     Long uid = cmdcontext.getUserId();
     String customerName ="";
     if (locale.toString().equals("ja_JP")||locale.toString().equals("ko_KR")
||locale.toString().equals("zh_CN")||locale.toString().equals("zh_TW"))
                {
                customerName = "" + userRegistrationDataBean.getLastName() + " "
 + userRegistrationDataBean.getFirstName();
                }
                else {
         customerName = "" + userRegistrationDataBean.getFirstName() + " "
+ userRegistrationDataBean.getLastName();
         }
         }
    else {
        customerName=userRegistrationDataBean.getUserId();
         if (customerName.equals("-1002"))
          customerName="";
              customer_name=customer_name.trim();
      }
```

在样本商店的“注销”页面中，包含了更多的定制代码，它们将顾客姓名设置为已生成的标识，并将购物车中的商品数量重新设置为零。“新时尚”中的“注销”页面是 LoginForm.jsp。在 Business “多乐五金店”中是 Logoff.jsp。定制代码如下:

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
 if (typeof parent.setCustomerName == 'function')
     parent.setCustomerName (parent.WCSGUESTID, '')
   if (typeof parent.setShoppingCartItems == 'function')
     parent.setShoppingCartItems(0);
</SCRIPT>
</HEAD>
</HTML>
```

确定顾客正在浏览的页面

顾客关心还让 CSR 能够确定商店中的顾客当前正在浏览的页面。样本商店通过向头文件（在“新时尚”中是 header.jsp，在 Business “多乐五金店”中是 NavHeader.jsp）添加以下代码来确定顾客处于什么页面:

```
<%
//Determine Page Type for LiveHelp
String headerType = (String) request.getAttribute("liveHelpPageType");
if (headerType==null)
headerType = "";

%>

<script language="javascript">
<%
    String pname = request.getRequestURI();
    int indpn = pname.lastIndexOf('/');
    indpn = pname.lastIndexOf('/', indpn-1);
    if(indpn != -1)
        pname = pname.substring(indpn+1);

    //Determine if this is a personal page or not
    if (headerType.equals("personal") ) {
%>
if (typeof parent.setPageParams == 'function')
   parent.setPageParams('PERSONAL_URL', '<%=pname%>');
<% } else { %>
if (typeof parent.setPageParams == 'function')
parent.setPageParams(location.href, '<%=pname%>');
<% } %>
</script>
```

如果页面不使用头文件，则页面中包含 StHeader1.jsp 文件。StHeader1.jsp 包含与添加到商店头文件中的相同代码。

为了维护顾客保密性，CSR 不应当具有对某些页面的访问权限。例如，CSR 不能有权访问竞销页面、包含由合同确定的价格的页面，或包含用户标识的页面，例如通讯录页面。这些页面被标记为个人的。在样本商店中，以下页面被标记为个人的:

- 新时尚
  - AddressBookForm.jsp
  - AllocationCheck.jsp
  - edit_registration.jsp
  - emptyshoppingcart.jsp
  - interestItemDisplay.jsp
  - myaccount.jsp
  - orderItemDisplay.jsp

- OrderDisplayPending.jsp
- ResultList.jsp
- shoppingcart.jsp
- TrackOrderStatus.jsp

- Business 多乐五金店
  - Address.jsp
  - Addressbook.jsp
  - AddToExistReqList.jsp
  - AdvancedSearch.jsp
  - AllocationCheck.jsp
  - CatalogMainDisplay.jsp
  - CatalogItemDisplay.jsp
  - CatalogTopCategoriesDisplay.jsp
  - Confirmation.jsp
  - OrderDisplayPending.jsp
  - OrderItemDisplay.jsp
  - OrderDetail.jsp
  - QuickOrder.jsp
  - RequisitionListCreate.jsp
  - RequisitionListDetailDisplay.jsp
  - RequsitionListDisplay.jsp
  - RequsitionListUpdate.jsp
  - Result List.jsp
  - Shipping.jsp
  - shoppingcart.jsp
  - TrackOrderStatus.jsp
  - UserAccount.jsp
  - UserRegistrationUpdate.jsp

为了将页面标记为个人页面，即对 CSR 不可见，样本商店在页面中包含了以下代码，就在包含头的位置前。

```
<%
// Set header type needed for this JSP for Customer Care.  This must
// be set before Header.jsp
request.setAttribute("liveHelpPageType", "personal");
%>

<%
String incfile;
incfile = includeDir + "Header.jsp";
%>
<jsp:include="<%=incfile%>"flush="true"/>
```

注: 尽管 CSR 无法看到标记为个人的页面的内容，但是 CSR 可以看到该页面的 URL。

跟踪购物车中的商品数

顾客关心还让 CSR 能够随时跟踪顾客的购物车中有多少商品。样本商店在以下位置获取购物车中的商品数量:

- “购物车”页面（“新时尚”：shoppingcart.jsp，Business “多乐五金店”：ShoppingCart.jsp）
- “清空购物车”页面（“新时尚”：emptyshopcart.jsp，Business “多乐五金店”：EmptyOrder.jsp）
- “订单确认”页面（“新时尚”：confirmation.jsp，Business “多乐五金店”：confirmation.jsp）
- “注销”页面（Business “多乐五金店”：Logoff.jsp）

注: 尽管将“购物车”指定为个人页面，但是 CSR 仍可跟踪购物车中的商品数。使用这些方法，他们并不能看到购物车包含的内容，仅能看到其中的商品数量。但是，CSR 可使用**查看购物车**按钮查看购物车的内容。关于更多信息，请参阅 WebSphere Commerce 联机帮助。

样本商店通过在上面的页面中添加以下代码来确定购物车中的商品数量:

- 首先定义一个 int 变量

```
int liveHelpShoppingCartItems= 0;
```

- 接下来，无论何时在购物车中添加订购商品，都使用以下代码行在 liveHelpShoppingCartItems 中添加数量:

```
liveHelpShoppingCartItems+= orderItem.getQuantityInEJBType().intValue();
```

- 然后，以下代码添加在页面的末尾，以便将顾客姓名设置为临时购物者标识，并获取顾客购物车中的商品数。

```
<script language="javascript">
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(<%=liveHelpShoppingCartItems%>);
</script>
```

以下代码用在空的购物车页面和订单确认页面中，以便将购物车中的商品数量重新设置为零:

```
<script language="javascript">
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(0);
</script
```

对商店添加顾客关心

要对不是基于样本的商店添加顾客关心，请执行以下操作:

第 1 部分: 安装先决条件

为使顾客关心可在商店中工作，必须执行以下操作:

- 安装 Sametime 服务器。关于更多信息，请参阅《*WebSphere Commerce 附加软件指南*》。
- 安装 WebSphere Commerce Sametime 集成软件包。关于更多信息，请参阅《*WebSphere Commerce 附加软件指南*》。

- 停止 WebSphere Commerce 实例，然后在配置管理器中启用 Sametime，再重新启动实例。关于更多信息，请参阅《*WebSphere Commerce 附加软件指南*》。
- 使用管理控制台创建 CSR 并对顾客关心注册 CSR。关于更多信息，请参阅 WebSphere Commerce 联机帮助。

第 2 部分: 从样本商店复制顾客关心集成文件

样本商店“新时尚”和“多乐五金店”包括以下文件，它们用来将顾客关心集成到商店中:

- `Sametime.js`: 包含对所有框架均包括的 JavaScript 功能。此文件中的功能使用 parent 前缀从主框架的页面中调用，例如 parent.setCustomerName。
- `StBlank.jsp`: 空的 JSP 文件。
- `StFrame.jsp`: 包含 JavaScript 功能并对商店前台嵌入顾客关心小应用程序。
- `StReadyJS.jsp`: 指出小应用程序已正确装入。
- `StHeader1.jsp`: 将参数传递至小应用程序的头文件，它指出包含此头的页面是否是个人页面。
- `StUpdate.jsp`: 更新顾客的姓名信息和标识。

要将 Sametime 集成文件从样本商店复制到您的商店，请执行以下操作:

1. 定位“新时尚”商店或“多乐五金店”商店的商店归档文件。商店归档文件位于以下目录中:
   - NT `drive:\WebSphere\CommerceServer\samplestores`
   - 2000 `drive:\Program Files\WebSphere\CommerceServer\samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`
2. 打开“多乐五金店”或者“新时尚”文件夹之间的任何一个，然后选择“多乐五金店”或“新时尚”商店归档文件。
3. 使用 WinZip 或类似的工具打开商店归档文件。
4. 定位 `webapp.zip` 文件。使用 WinZip 或类似的工具打开它。
5. 选择以下文件:
   - `Sametime.js`
   - `StBlank.jsp`
   - `StFrame.jsp`
   - `StReadyJS.jsp`
   - `StHeader1.jsp`
   - `StUpdate.jsp`
6. 将这些文件解压缩到包含商店 Web 有用资源的目录中。

注: StHeader1.jsp 在样本商店归档文件的一个包含目录中。如果您有用于放置包含在商店页面中的文件的单独目录，请将 StHeader1.jsp 保存在此目录中。如果没有，请将其与其它商店文件保存在相同的目录中。

第 3 部分: 向商店添加框架集

如第 268 页的『使用框架集』中所讨论的，顾客关心小应用程序在框架集中运行。此框架集应当添加到商店主页中，或添加到顾客最有可能通过它进入商店的页面。要向商店添加框架集，请执行以下操作:

1. 确定哪个页面是商店的入口点。
2. 打开“多乐五金店”或“新时尚”的商店归档文件。商店归档文件位于以下目录中:
   - NT drive:\WebSphere\CommerceServer\samplestores
   - 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores
3. 使用 WinZip 或类似的工具打开商店归档文件。
4. 定位 webapp.zip 文件。使用 WinZip 或类似的工具打开它。
5. 打开 index.jsp 文件。
6. 复制以下代码:

```
<script src="<%="Sametime.js"%></script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">
<html>
<head>
<title></title>
<script language="javascript">
var MainPageURL="";
if (MainPageURL=="")
MainPageURL="/webapp/wcs/stores/servlet/Logoff?storeId=<%=storeId%>
&langId=<%=langId%>&URL=LogonForm?
storeId=<%=storeId%>&catalogId=<%=catalogId%>";
function loadFrame()
{
 main.document.location.href=MainPageURL;
 }
</script>
</head>
<%
String sBasePath="/webapp/wcs/stores/servlet/";
String sSametimeUrl="StFrame.jsp?storeId="+storeId;
String sBlankUrl="StBlank.jsp?storeId="+storeId;
String sUpdateUrl="StUpdate.jsp?storeId="+storeId;
%>
<FRAMESET border=0 frameBorder=0 ROWS="100%,1,1,1,1,1" onLoad="loadFrame();">
<FRAME NAME="main"
 SRC="javascript:top.loadFrame();" MARGINWIDTH=0 SCROLLING="Auto"
FRAMEBORDER="no" noresize>
<FRAME NAME="JSFrame"
 SRC="<%=sBlankUrl% >" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
```

```
<FRAME NAME="sametime" SRC="<%=sSametimeUrl%>" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
<FRAME NAME="StUpdate" SRC="<%=sUpdateUrl%>" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
</FRAMESET>
</html>
```

注: 对于上面的示例，在“多乐五金店”样本商店中，“主”框架的源（SRC）是对另一页面的命令。根据您的商店建立的方式，SRC 可以是命令、JSP 文件或者 HTML 文件。

7. 将步骤 6 中复制的代码粘贴到您指定为商店入口点的页面中。对主框架的源（SRC）进行任何必要的更改。
8. 保存该文件。

第 4 部分: 添加代码以获取顾客姓名或标识

为使用顾客关心对 CSR 显示顾客的姓名或购物者标识，请执行以下操作:

1. 请复查第 270 页的『获取顾客的姓名或标识』中的信息。
2. 确定您计划从何处获取顾客姓名或购物者标识。例如,
   - 商店的入口点
   - 新建注册或更新注册信息
   - 注销
   - 头
3. 确定计划从何处获取顾客姓名或购物者标识后，确定那些页面是否包含商店的主头文件。如果不包含，请确定是希望对页面本身添加必要的代码，还是在其中包含头文件。
4. 从“新时尚”商店归档文件或“多乐五金店”商店归档文件中的 webapp.zip 文件，打开以下文件之一:
   - 新时尚: header.jsp
   - 多乐五金店: NavHeader.jsp

   注: 这两个文件都位于 webapp.zip 文件中的包含目录中。
5. 复制以下代码:

```
<script language="javascript">
if (typeof top.updateStInfo == 'function')
 top.updateStInfo()
</script>
```

6. 将步骤 5 中复制的代码粘贴到商店的头文件中，或直接粘贴到适当的页面中。

   注: 如果愿意，可以将文件 StHeader1.jsp 包含到适当的页面中，而不是将上面的代码复制到商店头文件或直接复制到文件中。
7. 保存文件。
8. （可选）要在“注销”页面中重新设置顾客标识（例如从顾客标识设置为 -1002），并将购物车中的商品数量重新设置为零，请在“注销”文件中添加以下代码:

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
 if (typeof parent.setCustomerName == 'function')
     parent.setCustomerName (parent.WCSGUESTID, '')
   if (typeof parent.setShoppingCartItems == 'function')
```

```
    parent.setShoppingCartItems(0);
</SCRIPT>
</HEAD>
</HTML>
```

第 5 部分: 添加代码以确定顾客正在浏览的页面

要确定顾客正在浏览的页面，请执行以下操作:

1. 在商店的头文件中包含 StHeader1.jsp 文件，例如:

   ```
   <%@ include file="StHeader1.jsp" %>
   ```

2. 将以下代码添加到任何应标记为个人页面的页面中，以便使其无法被 CSR 访问:

   ```
   <%
   // Set header type needed for this JSP for Customer Care.  This must
   // be set before Header.jsp
   request.setAttribute("liveHelpPageType", "personal");
   %>

   <%
   String incfile;
   incfile = includeDir + "Header.jsp";
   %>
   <jsp:include="<%=incfile%>"flush="true"/>
   ```

3. 将以下代码添加到任何不使用头、但应标记为个人页面的页面中。

   ```
   <%
   // Set header type needed for this JSP for Customer Care.  This must
   // be set before StHeader1.jsp
   request.setAttribute("liveHelpPageType", "personal");

   String incfile;
   incfile = includeDir + "StHeader1.jsp";
   %>
   <jsp:include page="<%=incfile%>" flush="true"/>
   ```

第 6 部分: 添加代码以跟踪购物车中的商品数

要让 CSR 能够跟踪顾客在购物车中放置的商品，请执行以下操作:

1. 请复查第 273 页的『跟踪购物车中的商品数』中的信息。
2. 确定您计划在何处跟踪购物车商品。例如,
   - 购物车页面
   - 空购物车页面
   - 订单确认页面
   - 注销页面
3. 在计划跟踪购物车商品的页面中，请执行以下操作:
   a. 定义 int 变量:

      ```
      int liveHelpShoppingCartItems= 0;
      ```

   b. 要在订购商品添加到购物车时对 shoppingCartItem 添加数量，请添加以下代码行:

      ```
      liveHelpShoppingCartItems+= orderItem.getQuantityInEJBType().intValue();
      ```

   c. 将以下代码添加在页面的末尾，以便将临时购物者标识（-1002）设置为实际的购物者标识，并获取购物车中的商品数:

```
<script language="javascript">
if (typeof parent.setCustomerName == 'function')
parent.setCustomerName(<%=cmdcontext.getUserId()%>, parent.CustomerName);
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(<%=liveHelpShoppingCartItems%>);
</script>
```

d. 如果计划跟踪空购物车页面和订单确认页面中的数据，请将以下代码添加到那些页面中，以便将购物车值重新设置为零:

```
<script language="javascript">
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(0);
</script
```

第 7 部分: 添加指向顾客关心的链接

要让顾客能够在商店中访问顾客关心，请执行以下操作:

1. 确定希望在何处放置指向顾客关心的链接。例如，您可能希望在导航栏中放置该链接，以便顾客始终可以使用它，或者放置在商店的某些特定页面中。
2. 将以下代码复制到将要包含该链接的页面中:

```
<a href="javascript:if((parent.sametime != null)) top.interact();">
<%=infashiontext.getString("LiveHelp")%></a>
```

第 8 部分: 更改显示给顾客的消息

当顾客初始连接到 CSR 时显示给顾客的消息（例如: “您好，我能为您做什么？”或“我们的办公时间从早上 9 点到晚上 9 点”）存储在 Sametime 服务器上的属性文件中。属性文件分为两种类型的文件: Customer.properties 和 Agent.properties。Customer.properties 文件包含显示给顾客的消息，而 Agent.properties 文件包含显示给 CSR 的信息。对于安装在 WebSphere Commerce 实例中的每种语言环境，这两种文件也都具有对应的特定于语言环境的文件，例如 Customer_de_DE.properties 和 Agent_de_DE.properties。

要更改这些文件中的消息，请执行以下操作:

1. 在 Sametime 服务器上定位这些属性文件。缺省情况下，这些属性文件位于以下目录中:
   - NT drive:\Sametime\Data\domino\html\wc\properties
2. 进行必要的更改。
3. 关闭并保存文件。

第 32 章 向商店添加电子广告位

电子广告位在商店页面上保留一定的空间，在其中显示竞销活动的个性化市场营销内容。当顾客请求页面时，该页面上显示的任何电子广告位将与规则服务器进行通信以处理与该广告位关联的基于规则的代码。每个电子广告位关联一个或多个活动。关于竞销和竞销活动的更多信息，请参阅第 103 页的第 12 章，『竞销有用资源』和 WebSphere Commerce 联机帮助。

为了在商店页面上正确显示竞销活动，必须向 JSP 文件添加一个电子广告位，然后使用 WebSphere 贸易加速器在数据库中注册它。本章将讨论如何向商店的 JSP 文件添加电子广告位。关于使用 WebSphere 贸易加速器在数据库中注册电子广告位的更多信息，请参阅 WebSphere Commerce 联机帮助。

电子广告位

以下是电子广告位的示例。

```
<!-- ==========================================================================
//*--------------------------------------------------------------------------------
//* The sample contained herein is provided to you "AS IS".
//*
//* It is furnished by IBM as a simple example and has not been thoroughly tested
//* under all conditions.  IBM, therefore, cannot guarantee its reliability,
//* serviceability or functionality.
//*
//* This sample may include the names of individuals, companies, brands and
//* products
//* in order to illustrate concepts as completely as possible.
//*All of these names
//* are fictitious and any similarity to the names and addresses used by actual
 persons
 //*or business enterprises is entirely coincidental.
//*--------------------------------------------------------------------------------
//*
==========================================================================-->
<%
/**
 * START - the following code should exist only once in a page, it initialize the
 *         command context and store data bean.
 */

// create the store bean to get the store directory
String collateralPath = "/webapp/wcs/stores/";
com.ibm.commerce.command.CommandContext emsCommandContext =
(com.ibm.commerce.command.CommandContext) request.getAttribute(
ECConstants.EC_COMMANDCONTEXT);
com.ibm.commerce.common.beans.StoreDataBean storeDataBean =
new com.ibm.commerce.common.beans.StoreDataBean();
storeDataBean.setStoreId(emsCommandContext.getStoreId().toString());
com.ibm.commerce.beans.DataBeanManager.activate(storeDataBean, request);
if (storeDataBean.getDirectory() != null) {
collateralPath += storeDataBean.getDirectory() + "/";
}
%>
<!-- ==========================================================================
// The following HTML form submits the request on the e-marketing spot to the
 ClickInfo
// command which captures the campaign statistics, and redirect to the location
//specified by the URL parameter.
========================================================================== -->
<form name="storeEmsForm" method="POST" action="/webapp/wcs/stores/servlet/ClickInfo">
```

```
 <input type="hidden" name="evtype">
 <input type="hidden" name="mpe_id">
 <input type="hidden" name="intv_id">
 <input type="hidden" name="URL">
 </form>
 <%
/**
 * END - the following code should exist only once in a page, it initialize the
 *       command context and store data bean.
 */
%>
<%
  /**
 * START - the following code can be used to drop multiple e-marketing
spots onto the page.
 *         Customize the appropriate EMarketingSpot instance name and the
e-marketing spot
 *         name before use.  Duplicate this code if more than 1 spot is
needed, do not use
 *         the same spot name.
 */

// create the e-Marketing Spot
com.ibm.commerce.marketing.beans.EMarketingSpot eMarketingSpot =
new com.ibm.commerce.marketing.beans.EMarketingSpot();

// IMPORTANT - set the correct name here
eMarketingSpot.setName("eMarketingSpotName");

// the maximum number of products/categories/ad copies that display
//through this e-marketing spot can be set here
eMarketingSpot.setMaximumNumberOfCatalogEntries(20);
eMarketingSpot.setMaximumNumberOfCategories(20);
eMarketingSpot.setMaximumNumberOfCollateral(20);

// instantiate the bean
com.ibm.commerce.beans.DataBeanManager.activate(eMarketingSpot,
 request);
%>

<%
// The following block is used to display the advertisements associated with this
// e-marketing spot.  The URL link defined with an advertisement can be referenced
//through the submittion of the HTML form attached above.

if (eMarketingSpot.getCollateral() != null && eMarketingSpot.getCollateral().
length > 0) {
%>
<TABLE>
<%  for (int i=0; i eMarketingSpot.getCollateral().length; i++) { %>
 <TR>
<%    if (eMarketingSpot.getCollateral()[i].getTypeName().equals("Image")) { %>
<TD>
   A HREF="javascript:document.storeEmsForm.evtype.value='CpgnClick'
;document.storeEmsForm.mpe_id.value=' <%= eMarketingSpot.getId() %>
';document.storeEmsForm.intv_id.value='<% = eMarketingSpot.getCollateral
()[i].getInitiativeId() %>';document.storeEmsForm.URL.value='
<%= eMarketingSpot.getCollateral()[i].getUrlLink() %>';
document.storeEmsForm.submit();">
 IMG SRC=" <%= collateralPath + eMarketingSpot.getCollateral()[i].getLocation() %>">
 </A>
 </TD>
<TD>
 <%= eMarketingSpot.getCollateral()[i].getMarketingText() %>
 </TD>
 <%  }  else if (eMarketingSpot.getCollateral()[i].getTypeName().equals("Flash")) { %>
<TD>
 EMBED src=" <% = collateralPath + eMarketingSpot.getCollateral()[i].getLocation()
 %>" quality=high bgcolor=#FFFFFF WIDTH=120 HEIGHT=90
 TYPE="application/x-shockwave-flash"> </EMBED>
 </TD>
<TD>
```

```
 A HREF="javascript:document.storeEmsForm.evtype.value='CpgnClick'
;document.storeEmsForm.mpe_id.value='<% = eMarketingSpot.getId()
 %>';document.storeEmsForm.intv_id.value='<% = eMarketingSpot.getCollateral()
[i].getInitiativeId() %>';document.storeEmsForm.URL.value='
<% = eMarketingSpot.getCollateral()[i].getUrlLink() %>';
document.storeEmsForm.submit();">
 <%= eMarketingSpot.getCollateral()[i].getMarketingText() %>
 </A>
 </TD>
<%      } %>
 </TR>
<%  } %>
 </TABLE>
<%  } %>

<%
// The following block is used to display the categories associated with this e-marketing
// spot.  The category display page which shows the selected category in the campaign will
// be referenced through the submittion of the HTML form attached above.

if (eMarketingSpot.getCategories() != null && eMarketingSpot.getCategories().length > 0) {
%>
 <TABLE>
<%  for (int i=0; i eMarketingSpot.getCategories().length; i++) { %>
 <TR>
<TD>
 A HREF="/webapp/wcs/stores/servlet/ClickInfo?evtype=CpgnClick
&mpe_id=<% = eMarketingSpot.getId() %>&intv_id=<% = eMarketingSpot.
getCategories()[i].getInitiativeId()
 %>%URL=/webapp/wcs/stores/servlet/CategoryDisplay&
 <%= ECConstants.EC_STORE_ID %>= <%= emsCommandContext.getStoreId().toString() %>&
<% = ECConstants.EC_CATEGORY_ID %>=<% =
eMarketingSpot.getCategories()[i].getCategoryId() %>&<%
 = ECConstants.EC_CATALOG_ID %>=<%= eMarketingSpot.getCategories()[i].getCatalogId() %>
&<% = ECConstants.EC_LANGUAGE_ID %>=<% = emsCommandContext.getLanguageId().toString() %>">
 <%= eMarketingSpot.getCategories()[i].getDescription
(emsCommandContext.getLanguageId()).getName() %>
 </A>
 </TD>
 <TD><% = eMarketingSpot.getCategories()[i].getDescription
(emsCommandContext.getLanguageId()).getLongDescription() %> </TD>
 </TR>
<%  } %>
 </TABLE>
<%  } %>

<%
// The following block is used to display the products associated with this e-marketing
// spot.  The product display page which shows the selected product in the campaign will
// be referenced through the submittion of the HTML form attached above.

if (eMarketingSpot.getCatalogEntries() != null &&
eMarketingSpot.getCatalogEntries().length > 0) {
%>
 <TABLE>
 <% for (int i=0; i eMarketingSpot.getCatalogEntries().length; i++) { %>
 <TR>
<TD>
 A HREF="/webapp/wcs/stores/servlet/ClickInfo?evtype=CpgnClick
&mpe_id=<%= eMarketingSpot.getId() %>&intv_id=<%=
 eMarketingSpot.getCatalogEntries()[i].getInitiativeId()
 %>&URL=/webapp/wcs/stores/servlet/ProductDisplay
&<%= ECConstants.EC_STORE_ID %>=<%= emsCommandContext.getStoreId().toString()
 %>&<%= ECConstants.EC_PRODUCT_ID %>=<%=
eMarketingSpot.getCatalogEntries()[i].getCatalogEntryID() %>&<%
= ECConstants.EC_LANGUAGE_ID %>=<%= emsCommandContext.getLanguageId().toString() %>">
 IMG SRC="<%= collateralPath + eMarketingSpot.getCatalogEntries()
[i].getDescription(emsCommandContext.getLanguageId()).
getThumbNail() %>" ALT="<%= eMarketingSpot.getCatalogEntries()[i].getDescription
(emsCommandContext.getLanguageId()).getShortDescription()%>" BORDER=0 WIDTH=60>
 </A>
 </TD>
```

```
 <TD><%= eMarketingSpot.getCatalogEntries()[i].getDescription
(emsCommandContext.getLanguageId()).getShortDescription() %> </TD>
 </TR>
 <%  } %>
 </TABLE>
<%  } %>


<%  /**
 * END - the following code is used to drop multiple e-marketing spots onto the page.
 *       Customize the appropriate e-marketing spot name before use.
 *       Duplicate this code if more than 1 spot is needed, do not use the same spot name.
 */
%>
```

以上电子广告位支持三种类型的竞销活动:

- 产品推荐
- 类别推荐
- 推广广告

注: 关于每个活动的更多详细信息，请参阅第 103 页的第 12 章，『竞销有用资源』。

电子广告位 bean

电子广告位使用电子广告位 bean 返回当前调度到广告位的竞销活动的结果。使用 bean 的不同属性将允许您定制电子广告位和相应的竞销活动。关于电子广告位 bean 及其属性的更多信息，请参阅 WebSphere Commerce 联机帮助。

向商店页面添加电子广告位

为了向商店页面添加电子广告位，请执行以下操作:

1. 确定广告位将在哪个 JSP 文件上显示。广告位可以添加到多个 JSP 文件。
2. 确定在 JSP 文件上何处放置广告位。
3. 复制第 279 页的『电子广告位』中的样本电子广告位。
4. 将样本电子广告位粘贴到 JSP 文件上的期望位置。
5. 定制样本电子广告位以符合 JSP 文件的布局。
6. 在电子广告位代码中，对电子广告位给定一个名称。

   注: 电子广告位的命名应该是描述性的，以便包含它们的位置（例如 HomePageAd 或 CheckOutPageRecommendation）。这有助于减小有关广告位出现位置以及广告位包含内容的混乱。如果必要，可以向名称添加数字以区分同一页面上出现的两个电子广告位。电子广告位名称必须是有效的 Java 标识。在使用 WebSphere 贸易加速器在数据库中注册电子广告位时，您必须使用此同一名称。
7. 如果每个 JSP 文件需要多个电子广告位，请重复步骤 2 到 6。当在 JSP 文件上添加另一个电子广告位时，请确保您复制了第 279 页的『电子广告位』中提供的样本电子广告位第二部分的全部，并将整个第二部分再次粘贴到 JSP 文件。然后，在 JSP 文件中更改电子广告位的名称。
8. 使用 WebSphere 贸易加速器在数据库中注册电子广告位。关于详细指导，请参阅 WebSphere Commerce 联机帮助。

   注:

a. 如果您计划使用以下约定 “langId=<%= languageId %>” 向 URL 添加商店标识、产品目录标识或语言标识，请注意电子广告位所嵌入的 JSP 必须使相应的标识可用。还可以通过命令上下文来检索标识，例如 getCommandContext().getLanguageId()?)。

b. 由于 “新时尚” 样本商店的结构，通过电子广告位仅可以推荐产品，而不能推荐基于 “新时尚” 样本商店的商店中的商品。

c. URL 参数 CatalogDisplay 应该以 “&” 开头而不是 “?”，因为此代码不直接引用命令。

第 9 部分 附录

附录 A. UML 图注

统一建模语言是用于表示软件设计的不同元素的标准图形语言。以下示例是一些最常用的 UML 元素。关于正式规范的进一步详细信息，请参阅 http://www.rational.com 和 http://www.omg.org。

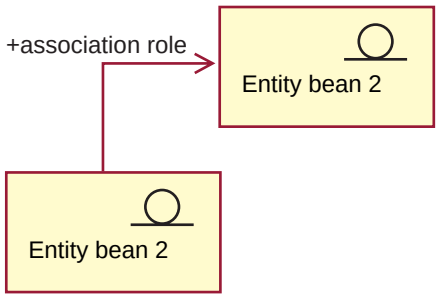
UML 图由以下项组成:

- 框: 框代表对象的类。类名出现在框的顶部。属性如果显示，则出现在类名下方。类名和属性用一条线分开。
- 线: 线代表两个类的对象之间的可能关系。线一端上类的对象可与其它类的对象相“关联”。
- 实心菱形: 线末端的实心菱形指示按值保存。线另一端上类的对象是菱形触及的一个且仅一个类的对象的组成部分。
- 空心菱形: 线末端的空心菱形指示按引用保存。线的菱形末端上的对象可以视为线另一端上类的分组对象。
- 基数: 这些基数出现在关系线的末端以指示基数限制。下表总结了基数限制:

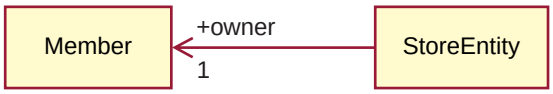
基数	关系类型
1	一种且仅一种
0..1	零种或一种
0..n	零种或多种
1..n	一种或多种

  如果未显示任何基数限制，则假定基数为 0..n，除非关系线末端出现实心菱形。在那种情况下，基数必须为 1。
- 加号: 出现在关系线末端的加号指示线末端上类的对象在关系中充当了角色。加号后面的文本指示对象在关系中的角色。
- 箭头: 关系线末端的箭头指示两个对象之间关系的方向是沿着箭头方向的。关系线上没有箭头时指示对象之间的关系通常是双向的。

下图说明了以上概念:

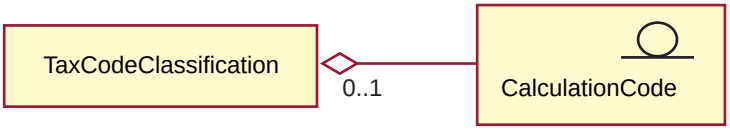


此图显示了两个实体 bean，它们以固定的装饰符号指示 Enterprise Java Bean。存在一个从第一个 bean 到第二个实体 bean 的单向关联。加号后面跟着描述实体 bean 2 在关联充当什么角色的文本。



在此图中，StoreEntity 有且仅有一个所有者，即 Member。Member 可以有零个或多个 StoreEntity。加号指示该 Member 在关系中充当了角色。在此情况下，Member 是 StoreEntity 的所有者。箭头指示您将通常通过向 StoreEntity 询问其所有者，而不是向 Member 询问其拥有的所有 StoreEntity，来找到 StoreEntity 的所有者。

在此图中，OrderItem 总是一个且仅一个 Order 的组成部分。Order 有零个或多个 OrderItem。



此图指示 CalculationCode 按零个或一个 TaxCodeClassification 分组，而 TaxCodeClassification 对零个或多个 CalculationCode 分组。

附录 B. 创建数据

在以 XML 文件格式创建商店数据之前，请执行以下操作:

- 确定正在创建的信息顺序。每个商店数据章节中的信息提示您创建数据的顺序，但是在创建 XML 文件时请切记父表的信息必须在子表的信息之前。
- 确定您希望如何使用商店。如果您正在创建样本商店归档文件（.sar），即，将被用作创建新商店的基础而被复制和使用的商店归档文件，那么将要创建的数据应当与不是正在创建样本商店时创建的数据稍微不同。关于更多信息，请参阅『创建样本商店的数据』。

创建样本商店的数据

样本商店归档文件中的数据采用组织良好的、对装入程序软件包有效的 XML 文件。商店归档 XML 文件计划成为可移植文件并且不应包含特定于数据库特定实例的已生成主键。相反，它们使用发布时由标识解析器解析的内部别名。这些约定的使用允许对样本商店归档文件进行多次复制和发布。

以 XML 文件格式创建商店的商店数据时不必使用这些约定，除非您打算创建将用于生成多个商店的样本商店归档文件，或者除非您希望创建的商店归档文件可移植，或者可以发布到另一个 WebSphere Commerce 实例中。

因此，样本商店归档文件使用以下约定:

- &，如 `member_id="&MEMBER_ID;"` 中的情况。`&XXX;` 约定是 DTD 宏（在 XML 中称为实体）。

  注: 在创建样本商店归档文件时，您必须将 MEMBER_ID 定义为 DTD 宏。WebSphere Commerce 在以下文件中定义了一组宏:

  - NT `drive:\WebSphere\CommerceServer\xml\sar\DBLoadMacros.dtd`
  - 2000 `drive:\Program Files\WebSphere\CommerceServer\xml\sar\DBLoadMacros.dtd`
  - AIX `/usr/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd`
  - Solaris `/opt/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd`
  - Linux `/opt/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd`
  - 400 `/qibm/proddata/WebCommerce/xml/sar/DBLoadMacros.dtd`

    类似 en_US 和 es_ES 的宏设置为相应的语言标识。例如:

    ```
    <!ENTITY en_US "-1">
    ```

    信息将使用“商店服务”中的工具进行指定。例如，用户可以在“商店服务”的“创建商店归档文件”页面中选择 MEMBER_ID。MEMBER_ID 宏是拥有商店的成员标识的占位符。创建商店归档文件时，您选择一个成员成为商店所有者。MEMBER_ID 宏设置为该成员的标识。例如，如果您选择成员 ID -2000，则 MEMBER_ID 按以下方式设置为 -2000:

```
<!ENTITY MEMBER_ID "-2000">
```

- @，如 ffmcenter_id="@ffmcenter_id_1" 中的情况。@ 符号的使用称为内部别名解析。标识解析器（装入程序软件包的一个实用程序）为需要标识的 XML 元素生成标识。标识解析器使用的技术之一是内部别名解析。使用内部别名解析时，以别名替换 XML 文档中的主键（标识）。然后在 XML 文件的其它位置使用此别名来引用该元素。这样就无需了解构建 XML 文件所需的唯一索引。在“商店服务”中发布或使用装入程序软件包期间，标识解析器会以唯一值替换 @ 符号。请参阅来自某 XML 文件的以下示例:
  - 使用标识解析器前

    ```
    <catalog
    catalog_id="@catalog_id_1"
    member_id="&MEMBER_ID;"
    identifer=InFashion"
    description="InFashion Catalog"/>
    ```

  - 使用标识解析器后

    ```
    <catalog
    catalog_id="10001"
    member_id="-2000"
    identifer=InFashion"
    description="InFashion Catalog"/>
    ```

    其中 10001 是由标识解析器指定的唯一标识，-2000 是用户在“商店服务”中选择的成员标识。然后使用装入程序软件包装入产生的 XML 文件。通过标识解析器运行文件确保可从一组 XML 文件创建多个商店。

商店服务和样本商店

“商店服务”中的**新建**和“创建商店归档文件”选项依赖于以上描述的约定。如果您希望通过“商店服务”将商店用作模板来创建其它商店，那么在创建数据有用资源时必须遵守这些约定。

但是，如果您不是在创建样本商店归档文件，那么仍然可以使用“商店服务”中的工具编辑或发布不遵守这些约定的商店归档文件。

附录 C. sarinfo.xml

每个商店归档文件必须包含一个 sarinfo.xml 文件。此文件（称为描述符）包含关于商店归档文件的信息（这些信息在发布商店归档文件时使用），包括文件有用资源 ZIP 文件和商店数据库 XML 文件的名称，以及它们发布的顺序。如果商店归档文件包含多语言版本的文件，那么 sarinfo.xml 文件也会包含那些信息，并确定每种语言文件发布的顺序。

注: 数据有用资源发布的顺序很重要，因为一些数据有用资源必须在其它的数据有用资源发布之前发布。因此，sarinfo.xml 文件中指定的有用资源顺序应当与样本商店的 sarinfo.xml 文件中指定的有用资源顺序一致。样本商店归档文件位于以下目录中。

- NT drive:\WebSphere\CommerceServer\samplestores
- 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

要查看商店归档文件的内容，请使用 ZIP 程序将其解压缩。sarinfo.xml 位于 SAR-INF 目录中。

sarinfo.xml 的示例

以下示例是“多乐五金店”的 sarinfo.xml 文件。关于元素、属性和属性值的更多信息，请参阅以下信息。关于商店归档文件的 XML 规范的更多信息，请参阅以下目录中的 sarinfo.dtd。

- NT drive:\WebSphere\CommerceServer\xml\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

```
<?xml version = "1.0"?>
<!DOCTYPE sarinfo SYSTEM "sarinfo.dtd">
<sarinfo complete-store="yes" multi-language="yes" version="1.0">

<store-info asset-name="store"/>

<file name="webapp.zip" type="zip">
```

```
<asset fragmented="no" name="webapp">
<file name="webapp.zip" type="zip">
<display-name>My Web App Display Name</display-name>
<description>My Web App</description>
</file>
</asset>

<asset fragmented="no" name="properties">
<file name="properties.zip" type="zip" />
</asset>

<asset fragmented="no" name="dbloadmacros">
<file name="data/DBLoadMacros.dtd" type="dtd"/>
</asset>

<asset fragmented="no" name="fulfillment">
<file name="data/fulfillment.dtd" type="dtd"/>
<file name="data/fulfillment.xml" priority="1" type="db-load"/>
<file name="data/en_US/fulfillment.xml" priority="31" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/fulfillment.xml" priority="31" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="store">
<file name="data/store.dtd" type="dtd"/>
<file name="data/store.xml" priority="2" type="db-load"/>
<file name="data/en_US/store.xml" priority="3" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/store.xml" priority="3" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="catalog">
<file name="data/catalog.dtd" type="dtd"/>
<file name="data/catalog.xml" priority="4" type="db-load"/>
<file name="data/en_US/catalog.xml" priority="5" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/catalog.xml" priority="5" type="db-load">
<locale>es_ES</locale>
</asset>

<asset fragmented="yes" name="tax">
<file name="data/tax.dtd" type="dtd"/>
<file name="data/tax.xml" priority="6" type="db-load"/>
<file name="data/en_US/tax.xml" priority="7" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/tax.xml" priority="7" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="taxfulfill">
<file name="data/taxfulfill.dtd" type="dtd"/>
<file name="data/taxfulfill.xml" priority="8" type="db-load"/>
</asset>

<asset fragmented="yes" name="shipping">
<file name="data/shipping.dtd" type="dtd"/>
<file name="data/shipping.xml" priority="9" type="db-load"/>
file name="data/en_US/shipping.xml" priority="10" type="db-load">
```

```
<locale>en_US</locale>
/file>
file name="data/es_ES/shipping.xml" priority="10" type="db-load">
<locale>es_ES</locale>
<file>
</asset>

<asset fragmented="no" name="shippingfulfill">
<file name="data/shipfulfill.dtd" type="dtd"/>
<file name="data/shipfulfill.xml" priority="11" type="db-load"/>
</asset>

<asset fragmented="no" name="store-catalog">
<file name="data/store-catalog.dtd" type="dtd"/>
<file name="data/store-catalog.xml" priority="12" type="db-load"/>
</asset>

<asset fragmented="no" name="storefulfill">
<file name="data/storefulfill.dtd" type="dtd"/>
<file name="data/storefulfill.xml" priority="13" type="db-load"/>
</asset>

<asset fragmented="yes" name="offering">
<file name="data/offering.dtd" type="dtd"/>
<file name="data/offering.xml" priority="14" type="db-load"/>
</asset>

<asset fragmented="no" name="command">
<file name="data/command.dtd" type="dtd"/>
<file name="data/command.xml" priority="16" type="db-load"/>
</asset>

<asset fragmented="yes" name="currency">
<file name="data/currency.dtd" type="dtd"/>
<file name="data/currency.xml" priority="17" type="db-load"/>
<file name="data/en_US/currency.xml" priority="18" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/currency.xml" priority="18" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="campaign">
<file name="data/campaign.dtd" type="dtd"/>
<file name="data/campaign.xml" priority="20" type="db-load"/>

<file name="data/en_US/campaign.xml" priority="24" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/campaign.xml" priority="24" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="store-catalog-tax">
<file name="data/store-catalog-tax.dtd" type="dtd"/>
<file name="data/store-catalog-tax.xml" priority="21" type="db-load"/>
</asset>

<asset fragmented="no" name="store-catalog-shipping">
<file name="data/store-catalog-shipping.dtd" type="dtd"/>
<file name="data/store-catalog-shipping.xml" priority="22" type="db-load"/>
</asset>

<asset fragmented="no" name="store-defaults">
<file name="data/store-defaults.dtd" type="dtd"/>
```

```
<file name="data/store-defaults.xml" priority="23" type="db-load"/>
</asset>

<asset fragmented="no" name="consistency_check">
<file name="data/sarrule.dtd" type="dtd"/>
<file name="data/sarrule.xml" priority="25" type="config"/>
</asset>

<asset fragmented="no" name="payment">
<file name="data/es_ES/paymentinfo.xml" type="config"/>
<file name="data/paymentinfo.dtd" type="dtd"/>
</asset>

<asset fragmented="yes" name="policy">
<file name="data/businesspolicy.dtd" type="dtd"/>
<file name="data/businesspolicy.xml" priority="26" type="db-load"/>
<file name="data/en_US/businesspolicy.xml" priority="27" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/businesspolicy.xml" priority="27" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="organization">
<file name="data/organization.dtd" type="dtd"/>
<file name="data/organization.xml" priority="28" type="db-load"/>
</asset>


asset fragmented="no" name="businessaccount">
<file name="data/businessaccount.xml" type="xml"/>
/asset>
asset fragmented="yes" name="contract">
<file name="data/contract.xml" priority="1" type="xml"/>
<file name="data/en_US/contract.xml" priority="2" type="xml">
<locale>en_US</locale>
</file>
file name="data/es_ES/contract.xml" priority="2" type="xml">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="accesscontrol">
<file name="data/accesscontrol.dtd" type="dtd"/>
<file name="data/accesscontrol.xml" priority="29" type="db-load"/>
<file name="data/en_US/accesscontrol.xml" priority="30" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/accesscontrol.xml" priority="30" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<!-- next priority should be 32 -->

</sarinfo>
```

其中

sarinfo 是 sarinfo.xml 文件中包含的所有信息。以下图表中的属性包含商店归档文件的一般信息。

属性名称	属性值
multi-language（必需）	确定此商店归档文件中是否支持多种语言: • yes • no
complete-store（必需）	确定商店归档文件是否包含完整商店必需的有用资源: • yes • no
version	商店归档文件的版本。例如: 1.0, 1.1
display-name	商店归档文件名称
description	商店归档文件的简短描述
standard-schema（必需）	商店归档文件是否遵守标准 WebSphere Commerce 数据库模式: • yes • no
store-info asset-name（必需）	作为商店归档文件锚点的有用资源。关于商店的所有信息都在属于此有用资源的文件中。例如: store
locale	商店归档文件支持的语言环境。语言环境变量（在下面列出）由 language_country 构成: • de_DE • en_US • es_ES • fr_FR • it_IT • ja_JP • ko_KR • pt_BR • zh_CN • Zh_TW

asset（必填）有用资源（asset）是相关文件的逻辑集合。例如, tax 是与商店税款相关的所有文件分组名称。

属性名称	属性值
name（必需）	有用资源类型名称。例如: • store • catalog • payment • tax

属性名称	属性值
fragmented（必需）	确定有用资源信息是否根据语言分割成多个文件。 • yes • no

`file`

属性名称	属性值
name（必需）	文件名称。
type（必需）	文件格式类型。例如: • db-load — 要装入到数据库中的文件 • dtd — 文档类型定义文件 • zip — 用于文件有用资源的 ZIP 文件，例如 webapp.zip • config — 配置文件
priority	商店归档文件中的文件发布顺序。1，2，3，4 …… **注**: 如果两个文件共享同样的优先级，装入顺序没有关系。如果文件必须按照一定的顺序装入，请确保为它们指定的优先级不同。
display-name	文件名称。
description	引用描述。
locale	语言环境。语言环境变量（在下面列出）由 language_country 构成: • de_DE • en_US • es_ES • fr_FR • it_IT • ja_JP • ko_KR • pt_BR • zh_CN • Zh_TW

附录 D. sarrule.xml

每个商店归档文件包含一个 sarrule.xml 文件，在您使用“商店服务”发布时它担当一致性检查程序的操作。在发布过程中，发布实用程序使用 sarrule.xml 文件中的规则检查商店归档文件包含 XML 文件中列出的 Web 有用资源。

sarrule.xml 的示例

以下 sarrule.xml 文件来自“多乐五金店”样本商店。

商店归档文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\samplestores
- 2000 drive:\Program Files\WebSphere\CommerceServer\samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

要查看商店归档文件中的 sarrule.xml 文件，请使用 ZIP 程序将其解压缩。sarrule.xml 文件位于 data 目录中。

sarrule.dtd 文件位于以下目录中:

- NT drive:\WebSphere\CommerceServer\xml\sar
- 2000 drive:\Program Files\WebSphere\CommerceServer\xml\sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

注: 推荐使用商店归档文件中与样本商店一同提供的 sarrule.xml 文件。但是，如果您希望，可以向现有的 sarrule.xml 文件中添加规则。

```
<?xml version="1.0"?>
<!DOCTYPE SAR-rules SYSTEM "sarrule.dtd">
<SAR-rules>
<asset name = "command">
<check type="webasset registration">
<rule entry="viewreg" attribute="properties" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
!--
   <asset name = "catalog">
<check type="webasset registration">
     <rule entry = "catalogdsc" attribute="thumbnail" type="java.lang.String"
```

```
 removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catalogdsc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="thumbnail" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="thumbnail" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
-->
   <asset name = "store">
<check type="webasset registration">
     <rule entry = "dispentrel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
     <rule entry = "dispcgprel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
</SAR-rules>
```

其中在

```
<asset name = "command">
<check type="webasset registration">
<rule entry="viewreg" attribute="properties" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
```

中“商店服务”中的发布实用程序检查 command.xml 中列出的每个 JSP 文件存在于商店归档文件的 Web 有用资源中。

其中在

```
   <asset name = "catalog">
<check type="webasset registration">
     <rule entry = "catalogdsc" attribute="thumbnail" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catalogdsc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="thumbnail" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="thumbnail" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="fullimage" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
```

中“商店服务”中的发布实用程序检查 catalog.xml 文件中列出的每个产品目录有用资源存在于商店归档文件的 Web 有用资源中。

其中在

```
<asset name = "store">
<check type="webasset registration">
     <rule entry = "dispentrel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
```

```
     <rule entry = "dispcgprel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
```

中“商店服务”中的发布实用程序检查 store.xml 文件中列出的每个商店有用资源存在于商店归档文件的 Web 有用资源中。

如果在发布过程中内存是个问题，请在试图发布前在 sarrule.xml 中注释掉产品目录有用资源规则。

附录 E. 数据库有用资源组

所有 WebSphere Commerce 数据库有用资源都分到不同的组中以供创建和装入。这些组是逻辑上相关的一组表。组织这些数据库有用资源组的顺序对于数据装入很重要，因为在装入对象之间的关系之前某些对象必须存在。

当装入 XML 格式的数据库有用资源时，可选择仅装入选定的组。这些组由先前几章中所创建的数据库有用资源（例如产品目录或实现）所组成。在如第 252 页的『装入数据库有用资源组』中所指导装入数据组之前，请执行以下操作:

- 确定正在装入哪个数据库有用资源组。每个组都包含一些相关性，在可装入有用资源之前必须满足这些相关性。请复查『数据库有用资源组相关性』中的信息。
- 确保已对选定的数据库有用资源组创建或更新了 XML 文件。每个有用资源章中的信息建议了创建数据库有用资源的顺序，但是在创建或更新 XML 文件时，请切记父表的信息必须在子表的信息之前。

数据库有用资源组相关性

每个数据库有用资源组都从 WebSphere Commerce 数据库表中抽取它的信息。在它们自己的组中，数据库有用资源具有相关性。即，数据库有用资源组无法从不同的数据组中抽取其它 XML 文件中的数据，且每个组都是独立的（除了外键之外）。然而，如果数据库有用资源需要引用其它组中定义的外部数据，则您需要手工提供该数据。这意味着，一个组中的数据对在其域之外定义的数据（即，其它数据库有用资源组上的数据）具有*外部相关性*。当数据库有用资源组对其它组中的表的主键具有外键关系时，则发生外部相关性。要装入数据库有用资源，必须满足它的外部相关性。要使用以下图表中的示例，商店数据库有用资源组的一个外部相关性是 `fulfillment.FFMCENTER.FFMCENTER_ID`，它指示在可以装入商店数据库有用资源组之前，实现数据库有用资源必须已存在于 WebSphere Commerce 数据库中。

在您开始装入过程之前，请考虑以下图表。每组数据库有用资源都依赖于其它数据库表（数据从这些数据库表中装入）。

要记住的一些要点:

- 单个组可能不满足一些外部相关性。每个商店使用的站点范围内的数据库有用资源或常规数据库有用资源在实例创建时在 *bootstrap* 中已预填充，并已准备就绪可以被访问。数据库有用资源组中包含的表具有对此类型数据的外键引用。引导程序数据被分为公共数据和特定于语言环境的数据。如果您具有多语言的商店，则需要选择公共的引导程序数据和特定于语言环境的引导程序数据。例如，您需要语言和成员引导程序数据。实例创建过程用商店支持的 WebSphere Commerce 语言填充 LANGUAGE 表，并创建根组织（`MEMBER.MEMBER_ID=-2001`）和缺省组织（`MEMBER.MEMBER_ID=-2000`）。您必须在需要的地方使用根组织，但是应该创建一个商店所有者组织，而不是使用缺省组织。关于组织及其层次结构的更多信息，请参阅 WebSphere Commerce 联机帮助。
- **外部相关性**列下列出的文件使用以下命名结构: *database asset group.database table.database column*。使用 `store.STOREENT.STOREENT_ID` 文件作为示例，数据取自

store 数据库有用资源组、*STOREENT* 表和 *STOREENT_ID* 列。以 *bootstrap* 开头的文件名表示数据是在 WebSphere Commerce 实例创建期间填充的。

- 在**外部相关性**列下列出的文件包含了对**相关表**的外键引用。必须首先填充这些表。
- 仅为了显示的目的，这些表被分割以表示包含有多语言信息（例如，产品描述）的特定于语言环境的表。
- 图表中的表代表 WebSphere Commerce 样本商店中的数据库有用资源。根据商店大小、功能和需要，这些表可能有所不同。根据您的商店的需要，确保包含了所有含有商店有用资源的数据库表，即使该特定表未列在下面。

访问控制数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
bootstrap.LANGUAGE.LANGUAGE_ID（根和商店所有者组织）， bootstrap.MEMBER.MEMBER_ID（根和商店所有者组织）	**accesscontrol.xml** ACACTACTGP， ACACTGRP， ACACTION， ACPOLICY， ACRESCGRY， ACRESGPRES， ACRESGRP	**accesscontrol.xml** ACACGPDESC， ACACTDESC， ACPOLDESC， ACRSCGDES， ACRESGPDES
业务策略数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
bootstrap.LANGUAGE.LANGUAGE_ID， boostrap.MEMBER.MEMBER_ID， store.STOREENT.STOREENT_ID（商店所有者组织）	**businesspolicy.xml** POLICY， POLICYCMD	**businesspolicy.xml** POLICYDESC
竞销数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
store.STOREENT.STOREENT_ID	**campaign.xml** CAMPAIGN， COLLATERAL， EMSPOT， STENCALUSG	**campaign.xml** COLLDESC
产品目录数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**

bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID（商店所有者组织）, store.STOREENT.STOREENT_ID, shipping.CALCODE.CALCODE_ID, tax.CALCODE.CALCODE_ID	**catalog.xml** BASEITEM, CATALOG, CATENTREL, CATENTRY, CATGROUP, CATGRPREL, CATTOGRP, ITEMSPC, ITEMVERSN, RA, RADETAIL, STOREITEM, STORITMFFC, VERSIONSPC **offering.xml** CATGRPTPC, MGPTRDPSCN, OFFER, OFFERPRICE, TRADEPOSCN **storefulfill.xml** INVENTORY **store-catalog.xml** DISPCGPREL, DISPENTREL, STORECAT, STORECENT, STORECGRP **store-catalog-shipping.xml** CATENTCALCD, CATENTSHIP **store-catalog-tax.xml** CATENTCALD	**catalog.xml** ATTRIBUTE, ATTRVALUE, BASEITMDSC, CATALOGDSC, CATENTDESC, CATGRPDESC, PKGATTR, PKGATTRVAL,
命令数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
store.STOREENT.STOREENT_ID	**command.xml** CMDREG, VIEWREG	N/A
Business **合同数据库有用资源**		
外部相关性		**数据库有用资源 XML 文件中的相关表**
store.STOREENT.STOREENT_ID		合同数据库表不是直接装入的，且遵循与其它 WebSphere Commerce 数据组不同的过程。关于更多信息，请参阅第 258 页的『发布合同有用资源』。
货币数据库有用资源		

外部相关性	数据库有用资源 XML 文件中的相关表	数据库有用资源 XML 文件中有关特定于语言环境的表
store.STOREENT.STOREENT_ID	**currency.xml** CURCVLIST	**currency.xml** CURCONVERT, CURLIST
实现数据库有用资源		
外部相关性	数据库有用资源 XML 文件中的相关表	数据库有用资源 XML 文件中有关特定于语言环境的表
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID（商店所有者组织）	**fulfillment.xml** FFMCENTER, STADDRESS	**fulfillment.xml** FFMCENTDS
组织数据库有用资源		
外部相关性	数据库有用资源 XML 文件中的相关表	数据库有用资源 XML 文件中有关特定于语言环境的表
bootstrap.LANGUAGE.LANGUAGE_ID（根和商店所有者组织）, boostrap.MEMBER.MEMBER_ID（根和商店所有者组织）	**organization.xml** ADDRBOOK, ADDRESS, MBRREL, MEMBER, ORGENTITY	N/A
装运数据库有用资源		
外部相关性	数据库有用资源 XML 文件中的相关表	数据库有用资源 XML 文件中有关特定于语言环境的表
bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID （商店所有者组织）, fulfillment.FFMCENTER.FFMCENTER_ID, store.STOREENT.STOREENT_ID	**shipping.xml** CALCODE, CALRULE, CRULESCALE, JURST, JURSTGPREL, JURSTGROUP, SHIPMODE, STENCALUSG **shipping.xml** SHPJCRULE, SHPARRANGE	**shipping.xml** CALCODEDSC, CALRANGE, CALRLOOKUP, CALSCALE, SHPMODEDSC
商店数据库有用资源		
外部相关性	数据库有用资源 XML 文件中的相关表	数据库有用资源 XML 文件中有关特定于语言环境的表

bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID （商店所有者组织）, bootstrap.SETCURR.SETCURR_ID, fulfillment.FFMCENTER.FFMCENTER_ID	**store.xml** INVADJCODE, RTNREASON, STORE, STORENT, STORELANG, VENDOR	**store.xml** FFMCENTDS, INVADJDESC, RTNRSNDESC, STADDRESS, STOREENTDS, STORLANGDS, VENDORDESC
商店缺省数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
shipping.SHIPMODE.SHIPMODE_ID（如果适用，必须首先装入此文件）, contract.CONTRACT.CONTRACT_ID, store.STOREENT.STOREENT_ID	**store-default.xml** STOREDEF	N/A
税款数据库有用资源		
外部相关性	**数据库有用资源 XML 文件中的相关表**	**数据库有用资源 XML 文件中有关特定于语言环境的表**
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID （商店所有者组织）, store.STOREENT.STOREENT_ID, fulfillment.FFMCENTER.FFMCENTER_ID, store.STOREENT.STOREENT_ID	**tax.xml** CALCODE, CALRANGE, CALRLOOKUP, CALRULE, CALSCALE, CRULESCALE, JURST, JURSTGROUP, JURSTGPREL, STENCALUSG, TAXCGRY, TAXJCRULE **taxfulfill.xml** TAXJCRULE	**tax.xml** CALCODEDSC, CALSCALEDS, TAXCGRYDS

附录 F. 声明

本信息是为在美国提供的产品和服务编写的。IBM 可能在其它国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档所述内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用联合王国或任何这样的条款与当地法律不一致的国家或地区:

国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是默示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本出版物的新版本中。IBM 可以随时对本出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。该 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue Markham, Ontario L6G 1C7
Canada

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可证协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其它操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其它可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其它关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

显示的所有 IBM 价格均是 IBM 当前的建议零售价，可随时更改而不另行通知。经销商的价格可与此不同。

此信息仅作为规划目的。其中的信息在描述的产品可用之前会得到更改。

此信息包含日常商业运作中所使用的数据和报告示例。为尽可能表述完整，这些示例包含人名及公司、品牌和产品的名称。所有这些名称均系虚构，如有实际的企业名称和地址与此雷同，纯属巧合。

版权许可证:

此信息中包含用源语言编写的样本应用程序，阐述了在各种操作平台上的编程技术。为了使开发、使用、行销或分发应用程序与编写的样本程序所在的操作平台提供的应用程序接口相一致这一目的，可以以任何形式，复制、修改并分发这些样本程序，而不必向 IBM 付费。这些示例尚未在所有条件下进行彻底地测试。因此，IBM 不能保证或暗示这些程序的可靠性、适用性或它们的功能。为了使开发、使用、行销或分发应用程序与 IBM 的应用程序的编程接口相一致，可以以任何形式复制、修改并分发这些样本程序，而不必向 IBM 付费。

这些样本程序的每个副本或任何部分，或任何派生产品都必须包含如下版权声明:

如果正在查看此信息的软拷贝，则照片和彩色图例可能不会显现。

本产品中提供的信用卡图像、商标和商品名称，仅供信用卡标记拥有者特许的商家用于接受该信用卡付款。

商标

以下术语是国际商业机器公司在美国和／或其它国家或地区的商标或注册商标:

AIX	IBM	WebSphere
AS/400	IBM Payment Manager	
DB2	iSeries	
DB2 Universal Database™	OS/400®	
eServer	VisualAge	

Microsoft®、Windows 和 Windows NT、Active Directory 以及 Windows 徽标是 Microsoft Corporation 在美国和／或其它国家或地区的商标或注册商标。

Oracle 是 Oracle Corporation 的注册商标，Oracle8 是 Oracle Corporation 的商标。

SET Secure Electronic Transaction、SET™ 和 SET 徽标是 SET Secure Electronic Transaction LLC 所拥有的商标。未经 SET Secure Electronic Transaction LLC 书面许可，严禁使用这些商标。

Solaris、Solaris Operating Environment、Java、JavaServer Pages、JavaBeans 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc. 的商标或注册商标。

UNIX® 是 The Open Group 在美国和其它国家或地区的注册商标。

其它公司、产品和服务名称可能是其它公司的商标或服务标记。

中国印刷