第一篇 i.Sell Merchandiser 程序员指南

第一章 概 述

i.Sell Merchandiser（由 Dynamo 驱动）使用创建定制店面、支持交易和提供目标明确的促销的工具对 Personalizer（个性化定制程序）进行了扩展。基于 i.Sell 应用服务器，Merchandiser 能够满足任务-关键 Web 应用对性能、可伸缩性和可靠性的严格要求。

Merchandiser 已经准备好的应用构建模块加速了电子商业应用的开发，并且它提供的一个管理接口控制了内容的呈现方式。Merchandiser 是一种创建有效的、动态变化的电子商业站点的功能强大的工具。Merchandiser 是一种完整的店面解决方案，使企业能够在完善且部署迅速的在线销售环境中为客户提供个性化的、有吸引力的购物经历。它通过减少不断更新内容所需的工作，以及简化新店面和特性的部署任务，从而降低了成本。

Merchandiser 的优点：

- 用以创建强健、丰富的在线店面的灵活框架
- 提供现成应用基础的内置式商业服务
- 供业务经理用来控制产品呈现的易用管理接口
- 部署扩展技术以满足目前及未来的成功在线需要

Merchandiser 应用是在基于 JavaSoft JavaBean 标准的 Nucleus 架构内创建的。在该框架内，每一 Java 类均通过指定的属性、事件和方法被表示为一个组件。这种基于组件的架构使开发商能够轻松扩展现有结构或者插入新组件。

下述主题提供了 Merchandiser 的特性、结构和文档记录的概要信息：

- JavaBean 组件
- 结构
- 商店模板
- 订单处理管道。
- 用户简档
- 商店与店厅管理
  国际内容
- Merchandiser 文档

1.1 JavaBean 组件

Merchandiser 使用了“服务器页面”（Server Page）系统，该系统提供了一种通过从模板页面调用 JavaBean 组件属性将动态内容转化为 HTML 格式的独特方法。所有 Merchandiser 模板都是服务器页面，包含 HTML 和调用 servlet（服务器小应用程序）进行处理的特殊服务器页面标记。服务器页面元素附属于产品、类别和订单对象，并在 HTML 页面中显示动态对象属性。Merchandiser 的服务器页面还可调用 Order Processing Pipeline（订单处理管道）执行订单对象。默认的管道对象执行多种订单处理功能，如信用卡验证、购物成本计算以及与实现提供商的通信。通过添加、删除、替换或扩展在模板中指明的默认管道对象，可以轻松

地定制订单处理管道。

有关 Merchandiser 中服务器页面的更多信息，请参阅“服务器页面概述”。有关订单处理管道的信息，请参阅“订单处理”。

1.2 体系结构

Merchandiser 在 Nucleus 架构内运行。Merchandiser 结构还包括 i.Sell Personalizer。Merchandiser 包括两个用户接口：一。个用于“活动商店”（Live Store），另一个用于“后台商店”（Staging Store）。活动商店和后台商店分别要求使用不同的 Application Server（应用服务器）、Personalizer 和 Merchandiser 实例。

Merchandiser 管理由商店和店厅管理页面组成，这些页面提供了基于 Web 且易于使用的接口，以设计和维护在线商店（Store）。一个商店可由一个或多个店厅组成。一个店厅（Boutique）可被视为商店内的一个房间，或者商城里的一个商店。商店管理（Store Administration）用来设置 Store 的主门厅页面，该页面包含有通向每个店厅的链接。商店管理还可用于更新运行中的站点。店厅管理（Boutique Administration）用于根据目录和产品页面的层次设计店厅。店厅管理也可用于管理产品 SKU。

Merchandiser 有三种模式：Live（活动）、Staging（后台）和 Admin（管理）。在 Admin 数据库中修改店厅的设计，然后使用用户接口将这些修改移入 Live 或 Staging 数据库中。

Merchandiser 使用 Staging 数据库和 Live Site 数据库。在店厅管理中所做的所有修改都被保存在 Admin Database 中。Store指向 Live Site 数据库，后者映射 Staging 数据库。此时 Store 管理员可以下面方法之一更新运行中的站点：

- Live Store Switch ：指向数据库的这些指针使 Staging Store 变成了 Live Store。
- 拷贝商店：从 Admin Database 中将整个商店复制到 Live 或 Staging 数据库中。
- 拷贝店厅：特定店厅被复制到 Live 或 Staging 数据库中。

Merchandiser 还包括 Personalizer 数据库表，这些表可安装在一个独立数据库或者一个 Merchandiser 数据库中。在购物过程中收集的用户注册和帐户信息存放于 Personalizer 数据库表中。

Merchandiser 提供了将购买记录分开放入它们自己的数据库中的选项。将购买记录与目录存储器分开使数据库管理员能够让不同的目录数据库连机而毋需顾虑购买记录。另外，将购买记录分开使数据库管理员能够运用不同的策略以确保购买记录数据库的完整性。

下图描绘了 Merchandiser 结构中的应用、数据库及其它组件：

管道对象使用电子函件与订单实现代理商（或下述实现提供商）进行通信。 Merchandiser 还提供了通过配置可与您所安装的 CyberCash 和 TAXWARE 产品一同使用的管道对象，以分别执行信用卡授权和税额计算。有关事务功能的更多信息，请参阅“交易管道”。

Store 的定制可包括修改下列部分或者全部订单处理功能：

与实现提供商的连接

- 客户信息的验证
- 信用卡授权和收费
- 销售税计算
- 用户持续简档信息
- 订单确认
- 送货成本计算
- SKU 计算
- 价格计算
- 库存处理

1.5 用户简档收集

Merchandiser 结构中包括存储着大量用户简档信息的 Personalizer。简档信息是通过明确使用用户在订单页面输入的信息，或者通过隐含使用用户在店内的浏览和购买行为而搜集的。用户简档的收集可建立一个有关对 Store 的多次访问的永久购物车。Personalizer 控制台还使用用户简档收集建立目标明确的促销。

如果想要扩展 Merchandiser 简档收集的功能，可以使用 Personalizer 向现有 Merchandiser 简档属性中添加简档属性。例如，可以使用 Personalizer 注册和个性化表单搜集明确的简档属性，还可以将老式简档系统与 Personalizer 数据库表格相集成。

有关 Merchandiser 简档收集功能的更多信息，请参阅“个性化 Merchandiser”。

1.6 商店与店厅管理

在开发商设置并定制好 Store 之后，可以通过商店管理和店厅管理页面对 Store 的设计和产品 SKU 进行管理。

商店管理可完成下列任务：

为 Store 设置主入口页面

更新活动或者后台站点

创建新的店厅

店厅管理可完成下列任务：

设计目录和产品页面

向产品分配产品 SKU

生成 Merchandiser 销售报告

- 库存管理

有关商店和店厅管理的更多信息，请参阅“ i.Sell Merchandiser Administrator’s Gruide”。

1.6.1 查看管理页面的 ServletBeans 源代码

商店和店厅管理页面中的 ServletBeans 源代码被作为标准 Merchandiser 程序包的组成部分提供。要查看源代码文件，请访问下列目录：

<MERCHDIR>/examples/src/atg/DynamoRetailStation/jadminui/

1.7 国 际 内 容

i.Sell Merchandiser 国际化以 Java 1.1 和 i.Sell 国际化标准为基础。在设计本地化的 Merchandiser Web 应用时，可对任何站点内容进行本地化。 Web 页面元素和用户信息可根据 Merchandiser 实地情况生成。Merchandiser 应用可以一种西欧语言或日语提供内容。

在本地化的 Merchandiser Web 应用中，Web 页面元素、用户信息、开发商信息和登记文件信息都与 Java 代码分开并存储于 ResourceBundles.properties 文件和 i.Sell 服务器页面中。

第二章　设置 Merchandiser

初次设置 Merchandiser 时，必须首先完成几项工作，以便使 Merchandiser 能够运行并根据您的站点的特殊设计与订单处理要求对其进行定制。 Merchandiser 提供了默认的 Store 模板和订单处理管道对象。Store 的定制选项中包括了通过扩展或替换默认管道对象来设置新的管道对象。例如，可以对 EmailOrder 对象进行扩展或替换从而与实现提供商建立电子函件通讯系统，另外还可以对默认的类别、产品和订单处理模板进行扩展或替换。

2.1　初始化 Store 设置

我们建议在设置 Store 时，首先设置和测试 Store 的基本功能。

设置基本功能的步骤如下：

1. 安装 Merchandiser Store 和 Merchandiser Administration。详细信息请参阅《i.Sell Merchandiser 安装指南》。
2. 在 Administration 模式下启动 Application Server。
3. 运行 bin\startiSell Merchanmdiser，启动 Admin 服务器。
4. 访问下列站点查看 Merchandiser Administration:
5. http://localhost:18840/merchandiser/admin/
6. 使用商店管理和店厅管理建立 Store 店厅页面。详细信息请参阅“ i.Sell Merchandiser Administrator's Guide”。
7. 以下列任意一种方法向数据库导入 SKU:

   运行定制的 SQL 脚本或者使用 SQL 请求。向数据库导入的 SKU 数量很大时，建议使用这种方法。详细信息请参阅《Merchandiser 数据库》。

   使用店厅管理中的“添加新 SKUs ”页面。向数据库导入的 SKU 数量较少时，建议使用这种方法。详细信息请参阅《Merchandiser 管理员指南》。

8 通过店厅管理分配 SKU 并为店厅产品设定库存数量。详细信息请参阅“ i.Sell Merchandiser Administrator's Guide”。

9. 运行并测试 Admin Store。

2.2　完成 Store 设置

测试完基本功能后，必须达到站点所必需的具体的订单处理以及与实现提供商相集成的要求，以便完成 Store 的设置另外也可以对商店做进一步设置以增强 Store 的设计与功能。

2.2.1　激活电子邮件功能

通过编辑以下三个属性文件可激活电子邮件功能。这些文件应在 Informix Developer 中

进行编辑。单击 Component 标签并选择 By Path 查看组件可访问这些属性。

1. 编辑下列组件：

/atg/retail/services/EmailSender

至少应编辑下列属性：

emailHandlerHostName——SMTP 服务器的主机名

emailHandlerPort——SMTP 服务器使用的端口号

2. 打开下列组件并根据需要修改相关属性：

/atg/retail/services/op/EmailOrder

3. 打开下列组件并根据需要修改相关属性：

/atg/retail/services/op/EmailReceipt

2.2.2 关闭电子函件通知

从订单处理管道中删除 EmailOrder 和 EmailReceipt 可关闭电子函件订单确认和电子函件订单实现功能。要实现这一目的，编辑＜Docroot>/DynamoCommerceStation/op/OrderConfirm.jhtml ，从下列行中删除 EmailOrder 和 EmailReceipt：

<input name="okl" bean="OrderProcessDroplet.submit" submitvalue="RemoveItems,CalculateSubtotal,CalculateSalesTax,CalculateShippingCost,CalculateTotal,SpillOrderItems,CheckEmptyOrder,ConfirmOrder,AuthorizeCreditCard,SpillOrderPurchaseInfo,EmailOrder,EmailReceipt,CollapseBundles,LogOrder,SetPurchasedProductTraits,ClearOrder,SpillOrderItems,RedirectOnFirstError:./OrderConfirm.jhtml,RedirectOnSuccess:./OrderStatus.jhtml"
type="image" src="../images/buttons/rb-confirms2.gif" align=middle border=0 >

注意　删除时应将逗号与其前面的字一同删除。

2.2.3 定制订单处理管道

定制订单处理管道对象可修改下列功能：

客户信息验证

信用卡验证

销售税计算

永久用户简档信息

订单确认

送货成本计算

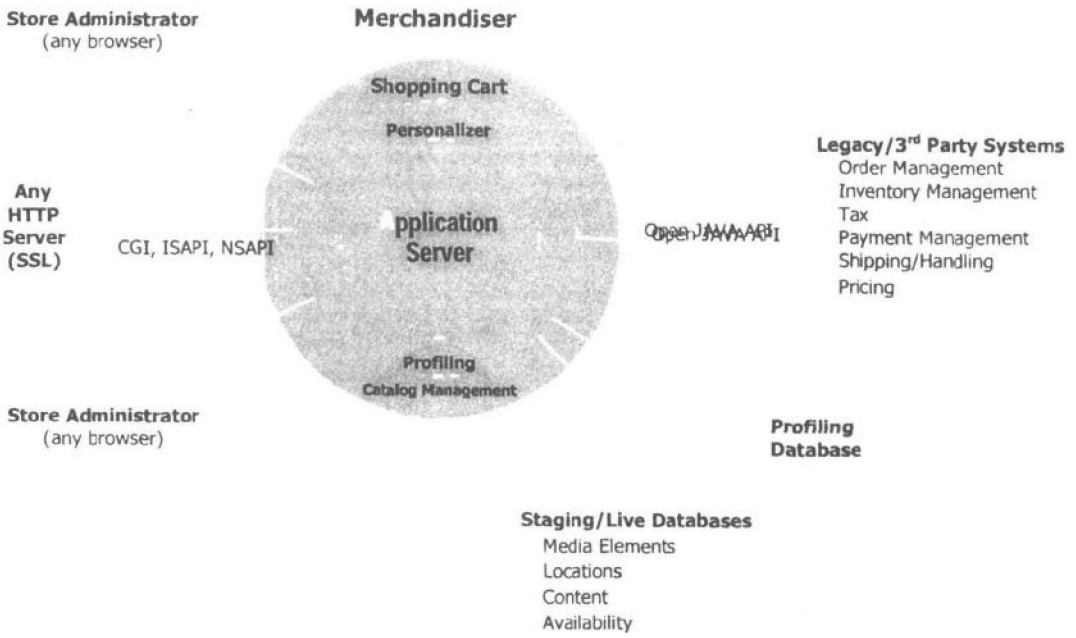
SKU 计算

价格计算

使用 Developer 编辑订单处理管道对象，这些对象位于/atg/retail/services/op。详细信息请参阅“订单处理”。

2.2.4 设置管理员登录属性

打开并行店厅管理选项可允许多名店厅管理员同时登录进入店厅（默认状态下，并行店厅管理是关闭的）。如果打开并行店厅管理并选择严格锁定强制，店厅管理员界面的行为将发生变化。



1.3 Store 模板

Merchandiser 的实时 Store 显示由目录、产品和订单处理模板定义。所有 Merchandiser 模板都是服务器页面，包含 HTML 和能访问 servlet(服务器小应用程序）的特殊标记。基于所使用的 servlet(服务器小应用程序)，Merchandiser 模板动态地在 HTML 页面中生成元素。servlet(服务器小应用程序）与产品、目录和订单对象紧密相连并在 HTML 页面中显示动态对象属性。服务器页面还访问订单处理管道（Order Processing Pipeline），以处理订单处理对象。

Merchandiser 的店厅管理允许管理员在类别模板和产品模板之间进行选择。这些模板建立了媒介、文本和 HTML 在店厅的类别和产品页面中的直观布局。订单页面是由其它模板页面定义的，如购物车模板。这些模板中的所有显示和订单处理功能均可被定制，以满足 Store 设计的需要。还可以创建自己的 Store 模板。

有关 Merchandiser 模板的更多信息，请参阅“类别与产品模板及订单处理”。

1.4 订单处理管道

订单处理管道是一种根据在.jhtml 文件中指明的一系列服务处理订单信息的机制。管道对象可在页面出现之前、之后或者在提交该页面中的表单时执行。默认的管道对象可执行许多订单处理功能，如价格计算、信用卡授权和税额计算。管道还可基于订单处理功能的成功或失败将用户重新定向到其它页面。

在设置 Merchandiser 时，如果想定制订单处理管道的某些方面，那么通过添加、删除、替换或扩展在模板中指明的默认管道对象就可以轻松地定制订单处理管道了。例如，可以创建电子函件、实时 TCP 或其它与实现提供商通信的系统。Merchandiser 默认的 EmailOrder

上述属性可在 AdminLoginManager.properties 文件中进行设置，该文件位于 atg/Retail/adminui 包含两个可更改属性：

- allowMultipleBoutiqueLogins 决定打开或关闭并行店厅管理。true 值打开并行店厅管理，false 值关闭之（默认值为 false）。
- multipleBoutiqueLoginsRestrictionLevel 当打开并行店厅管理时，控制锁定功能的行为。0 值表示在打开并行店厅管理时取消锁定限制， 1 值表示严格执行锁定限制（默认值为 1）。

详细信息请参阅“ i.Sell Merchandiser Administrator's Gruide”。

2.2.5 其它 Store 定制

- 建立 Sore 数据库至已有的以往数据库的映射。

有关 Merchandiser 数据库模式以及如何扩展 Merchandiser 数据库模式的信息请参阅“Merchandiser 数据库 ”。

- 修改任一类别与产品模板可定制产品与类别信息的显示。使用 ProductLookup 可向页面添加产品。详细信息请参阅“ 类别与产品模板 ”。
- 修改任一订单处理模板，例如购物模板，可定制订单信息的显示。详细信息请参阅“ 订单处理 ”。

第三章 服务器页面介绍

Merchandiser 应用使用服务器页面系统连接 HTML 页面与动态元素。服务器页面可清除地区 Java 代码与 HTML 内容。少量新的或经修改的 HTML 标记大大减少了 HTML 中所出现的 Java 代码的数量,同时 HTML 参数也大大减少了必须通过硬编码写入 Java 类的 HTML 的数量。服务器页面系统还为将放入 HTML 页面中的应用组件（例如 Merchandiser 组件）提供了一个框架。有关服务器页面的详细描述请参阅“ i.Sell Application Server Programmer's Guide”。本章仅简要概述如何在 Merchandiser 中使用服务器页面。

所有的 Store 类别、产品和订单处理模板都是服务器页面.jhtml 文件，其中包括 HTML 和 Servlet Beans 。这些模板不要求进行任何定制。如果想定制显示、订单处理或者 Store 用户界面，仅需编辑这些模板即可。

通过 Informix Developer 可添加新模板。详细信息请参阅“使用 Developer”。

在 Merchandiser 的模板中，Servlet Beans 包含于<droplet>标记中，后者是 Java<servlet(服务器小应用程序)>标记的扩展。与 servlet(服务器小应用程序)一样，Servlet Beans 可调用在 HTML 中使用的 Java 对象。Servlet Beans 还扩展了 servlet(服务器小应用程序)的功能。Servlet Beans 使您能显示 HTML 中的动态属性和参数值。<droplet>标记使您能够在服务器页面中嵌入服务器页面文件和 Servlet Beans ，并向这些嵌入式文件和 servlet(服务器小应用程序) beans 传递参数。

Merchandiser 使用多种类型的 Servlet Beans 显示模板页面中的元素和处理订单信息。本章介绍四种在产品、类别和订单处理模板中经常使用的 servlet Beans：ForEach、TableForEach、Switch 和 Range。有关在 Merchadiser 产品和分类模板中具体使用的 Servlet Beans 的信息，请参阅“类别与产品模板”。有关 Merchandiser 订单处理 servlet(服务器小应用程序) beans 的信息，请参阅“订单处理”。有关服务器页面与 Servlet Beans 的深入描述，请参阅“ i.Sell Application Server Programmer's Guide”。

下面几节介绍服务器页面的基本特性以及在定制 Merchandiser 时可能用到的常见 Servlet Beans：

表单处理
显示属性值
显示参数值
嵌入文件
传递参数
常用的 Merchandiser Servlet Beans

3.1 表 单 处 理

许多 Merchandiser 模板都使用表单处理用户记帐、送货和订单信息。通过将表单的输入字段直接与 Nucleus 服务或 bean 的属性相连接，服务器页面简化了 HTML 表单处理。在

处理表单时，应用服务器将使用用户所输入的值自动设置服务属性。

有关应用服务器表格处理的更多信息，请参阅“ i.Sell Application Server Programmer's Guide ”的“ 在服务器页面中使用表单 ”。

3.2 显示属性值

服务器页面允许显示 HTML 页面中的组件属性值和嵌套属性值。一般来说，可以使用 <valueof> 标记引用某一属性值。如果该属性值是 HTML 标记，例如 <body>、<a href> 或 <img src> 中的属性值，则可使用不同的语法显示该属性值。

本节介绍如何显示属性值，各小节标题分别为：

使用 valueof 显示属性值

显示标记内的属性值

显示嵌套属性值

有关显示属性值的更多信息，请参阅“ i.Sell Application Server Programmer's Guide ”的“ 服务器页面 ”。

3.2.1 使用 valueof 显示属性值

使用 valueof 标记可以显示服务器页面组件的属性。 valueof 标记语法如下：

```
<valueof bean="{component}.{property}">{default value}</valueof>
```

在页面调用过程中，该标记将被在指定组件的指定属性值所代替。如果值为 null，则显示默认值。不能引用不存在的组件或属性。请注意，无论是否想得到默认值，都必须有结束标记 </valueof>。如果没有结束标记，<valueof> 标记将无法正常工作。

3.2.2 显示标记内的属性值

如果要将属性作为 HTML 标记内的属性值显示，则不能使用 valueof 标记。在这类情况下，必须使用下列语法：

```
"bean:{component}.{property}"
```

例如，要将背景色表示为对象的属性，则指定：

```
<body bgcolor="bean:game.backgroundColor">
```

在指定了属性值的任何地方都可使用这种方法：<body> 标记、<img src> 标记、<a href> 标记等等。

3.2.3 显示嵌套属性值

在许多情况下，属性值并不是字符串。 JavaBean 属性值本身就是带有其自己的属性值的 JavaBeans 这种情况也十分常见。要引用服务器页面标记内的嵌套属性值应使用下面的语法：

```
"{component name}.{property name}.{property name}"
```

第一个 property name 表示该属性同时也是一个组件。第二个 property name 表示由第一个 property name 所指明的组件的一个属性的值。可以使用任意数量的属性间接级别。例如，一个名为 PersonManager 的 JavaBean 可能包含一种名为 currentPerson 的属性。该属性的值

可能是含有 age 属性的 Person 类对象实例。可以通过下列方法引用当前人的年龄：

```
<valueof  bean="/Employees/PersonManager.currentPerson.age">no person</valueof>
```

如果 currentPerson 值为空，则该属性被视为未设置，并显示默认文本 No Person 。如果在提交表单时中间属性值为 null ,则报错。

请注意"/"（斜杠）与"."（点）操作符的区别。"/"用来在 Nucleus 组件分层系统中分隔母元素与子元素的名称。这一般与该组件在 Nucleus 配置分层系统中的位置相对应。"."用来分隔组件与其属性值的名称。

3.3 显示参数值

在应用服务器中，参数用来完成多种目的。参数值可被传递到嵌入式 Servlet Bean 或服务器页面文件中。另外，Servlet Beans 可定义输出参数，开放式参数 Open Parameter）用这种方法显示在 HTML 页面中的值。开放式参数包含一段 HTML，而不仅仅是一个字符串。参数还能够定义开放式参数的名称。

定义参数值的语法与定义属性值的语法相同。总的来说，可以使用 <valueof>标记引用一个参数值。如果参数值为 HTML 标记，例如 <body>、<a href> 或 <img src> 中的属性值，则应使用另一种不同的语法显示参数值。另外还可以显示包含参数值和 HTML 的开放式参数。使用 <OPARAM> 标记可定义开放式参数。

本节以下述主题介绍如何显示参数值：

使用 Valueof 显示参数值

显示标记内的参数值

显示参数属性值

显示开放式参数

有关显示参数值的更多信息，请参阅“ i.Sell Application Server Programmer's Guide”。

3.3.1 使用 Valueof 显示参数值

使用 valueof标记可显示 .jhtml 页面中的参数值。valueof 标记语法如下：

```
<valueof param="{parametername}">{default value}</valueof>
```

在调用该页面时，该标记被所指定的参数值所代替。如果值为 null ，则显示默认值。不能引用不存在的参数。请注意无论是否使用默认值，都必须有结束标记 </valueof> 。没有结束标记，<valueof> 将无法正常工作。

3.3.2 显示标记内的参数值

如果要将 HTML 标记内的属性值作为参数显示，则不能使用 valueof 标记。在这些情况下应使用下列语法：

```
"param:{parametername}"
```

例如，要将背景色指定为一个参数，则可以指定：

```
<body bgcolor="param:backgroundColor">
```

在指定了属性值的任何地方都可使用这种方法： <body>标记、<img src >标记、<a href> 标记等等。

3.3.3 显示参数属性值

参数并非总是一个字符串，还可能是含有多个属性值的对象。要显示一个参数所附属的对象的具体参数，可使用下列语法：

```
<valueof param="currentPerson.name"></valueof> is
<valueof param="currentPerson.age"></valueof> years old.
```

param 标记的 param:...格式可用来显示 HTML 标记中的属性值。例如：

```
<body bgcolor="param:currentPerson.favoriteColor">
```

3.3.4 显示开放式参数

Servlet Beans 使用开放式参数定义参数值与 HTML 的组合。使用<:OPARAM>标记可以在一个 Servlet Bean 中指定一个开放式参数值，语法如下：

```
<oparam name="{parameter name}">{parameter value and HTML}</oparam>
```

在调用该页面时，Servlet Bean 使用 OPARAM 标记内的内容，并以其值替换 PARAM 标记。Servlet Bean 可含有一个以上 PARAM。Servlet Bean 决定每一 OPARAM 何时以及以何种频率被使用。如果值为 null，则显示默认值。不能引用不存在的开放式参数。请注意必须使用结束标记</OPARAM> 。如果没有结束标记，<OPARAM> 将不能正常工作。

3.4 嵌 入 文 件

服务器页面允许在 .jhtml 文件中嵌入.jhtml 文件和 Servlet Beans 。使用<:droplet src=...>标记可嵌入另一.jhtml 文件。使用<droplet bean=...>标记可嵌入 Servlet Bean。

本节包括以下内容：

嵌入.jhtml 文件

嵌入 Servlet Bean

有关嵌入.jhtml 文件和 servlet(服务器小应用程序）Beans 的更多信息，请参阅“ i.Sell Application Server Programmer's Guide”。

3.4.1 嵌入. jhtml 文件

使用<droplet> 标记可在.jhtml 文件中嵌入其它 .jhtml 文 件 这 使 HTML 元素可被其它页面包含和重复使用。

嵌入 ljhtml 文件的语法如下：

```
<droplet src="header.jhtml"></droplet>
```

请注意,使用<droplet >标记时必须有相应的</droplet>。详细信息请参阅“i.Sell Application Server Programmer's Guide ”的“ 服务器页面 ”

3.4.2 嵌入 Servlet Bean

使用<droplet>标记除可在一个.jhtml 文件中嵌入另一.jhtml 文件以外，还可在.jhtml 文件中嵌入一个 Servlet Bean。

嵌入 Servlet Bean 的语法如下：

```
<droplet bean="ItemDroplet"></droplet>
```

请注意，使用<droplet>标记时必须有相应的</droplet>。

在嵌入.jhtml 文件时，<droplet>标记使用 src 属性表示所要嵌入的 HTML 页。但当嵌入 Servlet Bean 时，则需指定 bean 属性以表示所要显示的 Nucleus 组件名。Nucleus 找到该组件，确认其实施的是 Servlet，然后才将该请求传递给该组件以满足＜droplet>标记的要求。详细信息请参阅“ i.Sell Application Server Programmer’ s Guide ”的“在服务器页面中使用组件”。

3.5 传 递 参 数

服务器页面的另一重要特性是允许从.jhtml 页向被嵌入的.jhtml 文件或 Servlet Beans 传递参数值。传递参数值使用 <param>标记。

向被嵌入的 .jhtml 文件传递参数时使用以下语法：

```
<droplet src="header.jhtml">
<param name="promotionType" value="Vacations">
</droplet>
```

向被嵌入的 Servlet Bean 传递参数时使用以下语法：

```
<droplet bean="/droplets/advertising">
<param name="adName" value="travelguide">
</droplet>
```

有关传递参数的更多信息请参阅 “i.Sell Application Server Programmer’s Guide ”。

3.6 常用 Merchandiser Servlet Beans

Merchandiser 在类别、产品和订单处理模板中使用了多种类型的 Servlet Beans。这些 Servlet Beans 用来显示模板元素并实现订单处理。下面描述的 Servlet Beans 可在 Merchandiser 模板中通用。ForEach Servlet Bean 以开放式参数指定的输出格式显示数组。 Switch Servlet Bean 显示两个以上可能输出值中的一个。

本节描述下列与 servlet Beans 相关的信息：

ForEach
Switch
Range
Cache
TableForEach

有关在产品和类别页面中分别使用的 Servlet Beans 的信息，请参阅“分类与产品模板”。有关在订单处理中使用的 Servlet Beans 的信息，请参阅“订单处理”。有关 Servlet Beans 的详细信息，请参阅 “i.Sell Application Server Programmer’s Guide”。

这些模板都是服务器页面，同时包含 HTML 和 Servlet Beans ，它们的位置如下：

<DOCROOT>/Merchandiser/catalog
<DOCROOT>/Merchandiser/catalog/templates
<DOCROOT>/Merchandiser/op

3.6.1 ForEach

ForEach Servlet Bean 对其 array 参数中的每一元素均使用其 output 参数。Array 参数可以是矢量、枚举、字典或数组。每次迭代均设置三个参数：

- index 参数被设置为当前循环计数，从 0 开始。
- count 参数被设置为索引 +1
- element 参数被设置为数组中该元素的值

ForEach Servlet Bean 使用以下参数。输入参数可被明确指定或者从包含它的 Servlet Bean 中传递。输出参数由 Servlet Bean 设定并可从开放式参数(OPARAM)的内部引用。OPARAM 参数包含 Servlet Bean 范围内的所有 HTML。

输入参数

array

该参数定义将要输出的商品列表。该参数可以是数组、字典、矢量或数组中元素的枚举。

输出参数

index

在每次为 output 参数赋值时，该参数均被设置为数组当前元素的从 0 开始计算的索引。首次迭代时，index 值为 0。

count

在每次为 output 参数赋值时，该参数均被设置为数组当前元素的基于 1 的索引。首次迭代时，count 值为 1。

key

如果 array 参数被设置为字典，则该参数被设置为字典中的关键字的值。

element

在每次 index 累加并调用 output 参数时，该参数均被设置为数组的当前元素。

sortProperties

该字符串指定如何对输出数组中的商品列表进行排序。排序可按 JavaBeans、Dynamic Beans 属性进行，也可按日期、数字或字符串进行。

要对 JavaBeans 进行排序，首先将 sortProperties 值指定为由逗号分隔的属性名称列表，第一个名称为主排序名，第二个名称为次级排序名，依次类推。如果关键字首字符为＋，则按升序排序，如果首字符为 -，则按降序排序。

示例：先以 title 属性的字母升序，再以 size 属性的字母降序对 JavaBeans 的一个输出数组排序：

```
sortProperties=+title,-size
```

对日期、数字或字符串排序时，以单个＋号或 -号为 sortProperties 赋值，表示分别以升序或降序进行排序。

示例：以字母升序对字符串的输出数组排序：

```
sortProperties=+
```

OPARAM 参数

output

该参数为每个数组元素赋值一次。

outputStart

如果数组不为空，该参数在所有输出元素之前传递。比如，这个参数可以用来传递表头。

outputEnd

如果数组不为空，该参数在所有输出元素之后传递。比如，它可以用来传递跟随在表后面的文本。

empty

如果数组不包含任何元素，则传递该可选参数。

举例

下例使用 ForEach Servlet Bean 显示一份人员列表。ForEach 对 People 数组中的每一项使用一次 output 参数。在每次迭代中，它均设定一个名为 element（表示一个人）的参数。然后，Servlet Bean 将 elment 参数传递给另一个 Servlet Bean，即 displayPerson.jhtml，以生成 HTML 格式。displayPerson.jhtml Servlet Bean 本身又使用了另一个标准 Servlet Beanswitch，为名单中的每个人赋以不同的输出值。有关 Switch Servlet Bean 的用法请参阅下一节的内容。

```
<droplet bean="ForEach">
    <param name="array" value="bean:SnowtripService.people">
    <!--
        We pass "element" to the displayPerson.jhtml Bean as the parameter
        named "person".
    -->
    <oparam name="output">
        <droplet src="displayPerson.jhtml">
            <param name="person" value="param:element">
        </droplet>
    </oparam>
</droplet>
```

3.6.2 Switch

Switch Servlet Bean 根据其 value 参数有条件地传递其参数之一。

注意 输入参数可被明确指定或者从包含它的 Servlet Bean 中传递。输出参数由 Servlet Bean 设定并可从开放式参数（OPARAM）的内部引用。OPARAM 参数包含 Servlet Bean 范围内的所有 HTML。

输入参数

- value Switch 的值。Switch 的 value 值被转换为字符串，并作为将调用的参数名使用。例如，如果 Switch 的 value 参数为一个值为“真”的布尔值，则 Switch 调用“真”