

版本 6.0.1



产品技术概述



产品技术概述

注意

在使用此信息及它支持的产品之前，请阅读本书结束部分的『声明』中的信息。

目录

第 1 章 业务集成	1
业务单位之间的集成	2
企业之间的集成	2
WebSphere Integration Developer	3
标准	4
集成开发者的角色	4
第 2 章 服务组件体系结构	5
服务组件	6
服务数据对象	8
服务限定词	9
模块	10
导入和导出	12
服务导入和导出绑定类型	12
选择适当的绑定	13
服务实现类型	14
Java 对象	14

BPEL 流程	14
状态机	15
业务规则	16
选择器	17
人工任务	18
接口映射	18
调解流	19
独立引用	20
相关信息	20

第 3 章 了解工具	23
“欢迎”视图概述	23
“欢迎”视图中的教程	23
“欢迎”视图中的样本	24
信息中心	25

声明	27
-----------	-----------

只有您才会面对这种情况吗？不是。这是一个普遍问题，它已出现了许多年，现在仍然存在。根据 2001 年 12 月出版的 *CIO Survey* 可以知道，应用程序集成一直是优先级最高的三种顶级技术之一。根据 2001 年冬季出版的 *The Business Integrator Journal* 可以知道，最近的调查显示，三分之二的开发者都使用集成软件来开发基于 Web 的解决方案。平均起来，每个开发者要集成三个不同的系统。

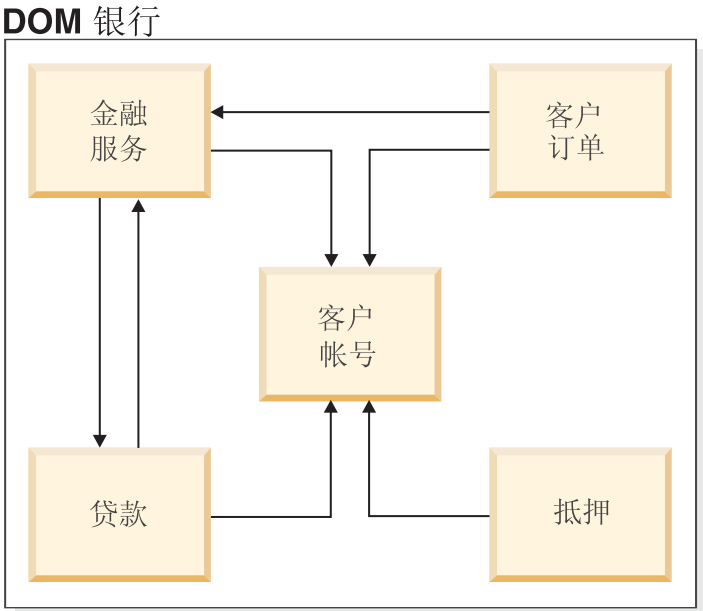
让我们考虑业务中造成该严峻情况的两个原因：企业内业务单位之间的集成和企业之间的集成。然后我们来看看 WebSphere Integration Developer 如何解决这些问题，尤其是它遵循业界标准的重要性。最后，我们将看看集成专家，他是那种使用 WebSphere Integration Developer 的工具来解决前面提出的问题的的人。

业务单位之间的集成

如今业务单位经常发现它们需要与其他人协作，这会增加更紧密地集成其应用程序的需要。

从前独立的业务单位正在被集成到一起，因为技术允许他们连接起来，且效率要求他们必须以更合作的方式运作以尽量减少开销并尽量提高产出。共同的合作目标也促使业务单位集成到一起。市场营销单位和研发单位都想生产有利可图的产品。通过将市场营销知识和产品开发信息集成到一起，生产那样畅销的产品的可能性会增加。业务单位之间的协作还让公司能够通过允许在不同业务环境中复用许多现有业务应用程序，来充分利用这些业务应用程序。

业务单位之间的集成会比企业之间的集成更容易，这是因为安全风险很低，并且管理业务单位之间的交互不会那么困难。各个业务单位可能正在使用相同的协议、操作系统和计算机语言。换句话说，它是一个相对同质的环境。然而，其关键是找到快速集成应用程序的正确工具。在下图中，DOM 银行有几个需要共享彼此的信息的业务单位。几年前，DOM 银行可以通过打印出一个业务单位的信息副本，然后将它们传递给另一个业务单位（它有自己的系统和应用程序）来进行管理。而现在，DOM 银行必须创建包含它的所有业务单位的集成应用程序，以便在与其竞争者的竞争中始终立于不败之地。



企业之间的集成

由于伙伴关系或接管关系需要共享数据和流程，因此推动在业务单位之间集成应用程序的力量也在企业之间存在。

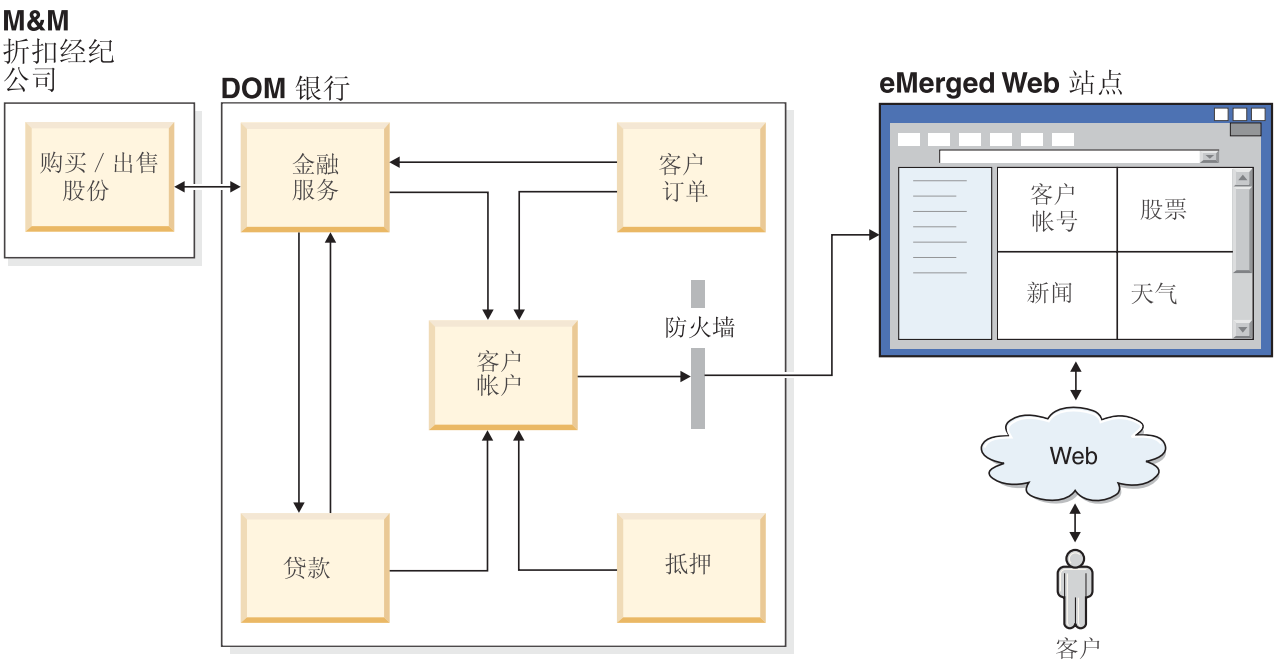
技术使企业能够在互惠的区域建立链接。例如，汽车制造商可以与轮胎供应商建立集成流程，以便当轮胎库存低时会自动通知供应商。经济的必要性促使企业之间的集成。公司之间有更紧密的联系，则意味着完成工作的延迟时间更少，开销更少。自动的流程意味着人们花费更少的时间来处理企业之间的事务，并且可大量减少出差成本和面对面会议的时间。类似地也会减少管理费用并且通知、交货和发票之间的周转时间也会减少。

但是不同的企业有不同的历史。它们的应用程序在不同的平台上以不同的语言编码并且使用不同的通信协议。面对不同的组织时，也会有更高的安全风险。不管企业之间集成有多少利益和多大的必要性，但如果没有正确的工具，开发时的成本还是很大的。

WebSphere Integration Developer

WebSphere Integration Developer 可用来解决一个组织每天都要面对的集成问题。设计此产品的目的是为构建集成应用程序的人员提供一个完整的集成开发环境。为了简化和加速集成应用程序的开发，此环境提供了一层抽象层，它将您处理的以可视方式表示的组件与底层的实现分隔开来。

集成的应用程序并不简单。它们可调用企业信息系统（EIS）上的应用程序、跨部门或企业参与业务流程、以及调用本地或远程以各种不同语言编写并在各种不同操作系统中运行的应用程序。例如，eMerged 公司是通过合并 DOM 银行和 M&M 折扣经纪公司创建的。此合并意味着所有以上内容：EIS 系统上的应用程序、业务流程、先前每个公司内的应用程序必须在这两个公司共享并以一种无缝的方式提供给一组新客户。但是，eMerged 成功地完成了此任务，并且如下图所示，先前这两个不同的企业的客户现在都能联机访问其财务信息。



eMerged 使用了 WebSphere Integration Developer 的工具来为他们自己和他们的客户构建集成的应用程序。这些工具将应用程序（包括 EIS 系统上远程存在的应用程序）和业务流程表示为组件。通过可视编辑器创建这些组件并将其组装到其他集成应用程序（也就是从一组组件创建的应用程序）。可视编辑器在组件和其实现之间放置了一层抽象层。使用这些工具的开发者可创建集成应用程序而无需对每个组件的底层实现有详细的了解。

这些工具既允许以自顶向下的设计方式构建集成应用程序（这种方式表示：一个或多个组件的实现不存在并将在以后添加）；也允许以自底向上的设计方式构建（这种方式表示：已实现组件，开发者通过在可视编辑器中

拖放这些组件来组装它们，然后开发者通过线将它们连接起来在它们之间创建逻辑流）。调试和测试环境表示在将您的应用程序部署到生产服务器之前进行的完整测试。设置监视点允许您了解应用程序是如何实际使用的，以便来调整它来优化性能。

WebSphere Integration Developer 的工具基于面向服务的体系结构。组件是服务，涉及许多组件的集成应用程序也是服务。创建的服务符合领先的业界标准。业务流程（它也变为组件）是类似地通过简单易用的可视工具创建的，它们符合业界标准的业务流程执行语言（BPEL）。在 Windows 和 Linux 平台上都提供了 WebSphere Integration Developer。

以下是 WebSphere Integration Developer 的工具的一些优点：

- 它们简单易学
- 可以将它们应用于复杂的集成情况
- 您可以快速创建符合业界标准的应用程序

标准

使用 WebSphere Integration Developer 创建的应用程序符合与面向服务的体系结构相关联的业界标准。

没有人想创建使用专利代码的应用程序，因为专利代码在几年之内可能不受支持或者是会涉及不菲的许可证费用。因此基于标准的集成是 WebSphere Integration Developer 的一个基本特征。对于连接，使用了 J2EE 连接器体系结构标准。要进行异步消息传递（经常用在需要确保数据传递的大型应用程序中），使用 Java 消息服务（JMS）标准。WebSphere Integration Developer 很容易集成基于“简单对象访问协议”（SOAP）的 Web Service。要描述服务，使用成熟的 Web 服务描述语言（WSDL）标准。要定义业务流程，使用“业务流程执行语言”（BPEL）标准。

这些基于标准的接口和组件构成了一个开放式的可插拔体系结构。但是没有包括专利元素；可使用标准化的接口来访问它们。这意味着使用 WebSphere Integration Developer 创建的应用程序可以与 .NET 应用程序（举例来说）进行交互。在体系结构部分，提供了与完整的“服务组件体系结构”的链接，而完整的“服务组件体系结构”提供了受支持的许多标准的详尽列表。

集成开发者的角色

集成开发者是 WebSphere Integration Developer 的主要用户。集成开发者通过使用可视工具，可构建复杂的集成应用程序，而不需要具备广泛的有关底层实现的知识。

WebSphere Integration Developer 将应用程序和业务流程作为组件来提供。组件的实现保持为隐藏状态，各个组件之间通过接口进行互操作。因此，集成开发者不需要有关组件的底层实现的广泛知识就能创建使用这些组件的集成应用程序。但是，集成开发者可能需要具备集成领域的广泛技术知识，因为他们需要了解 EIS 系统、业务流程和使用 Java 或其他语言编写的应用程序。例如，架构设计师广泛地了解系统如何工作，但不需要知道每个组件如何工作的细节。与架构设计师一样，集成开发者可能是一个组织中负责对应用程序进行总体设计的人，然后他又负责安排其他人编写特定组件的实现。

第 2 章 服务组件体系结构

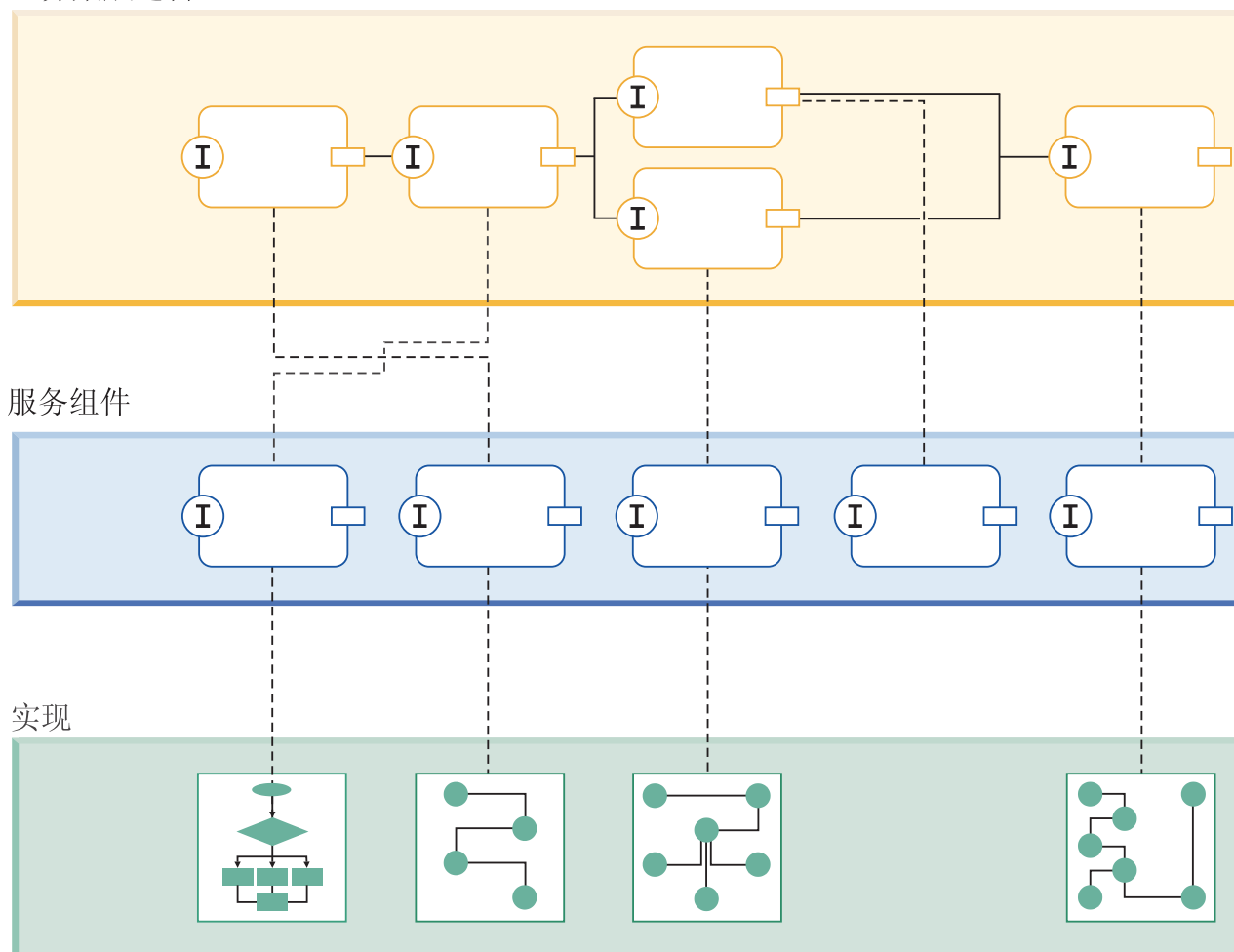
“服务组件体系结构”（它基于业界标准的面向服务的体系结构）以面向服务的方式表示所有业务流程 - Web Service、企业信息系统（EIS）服务资产、工作流程和数据库等。在本节中，我们在较高级别上检查此体系结构创建的服务和服务数据对象，服务和数据对象一起表达了业务逻辑和对业务数据的引用。

“服务组件体系结构”的目标是将业务集成逻辑与实现分隔开，以便集成开发者能专注于组装集成应用程序而不是专注于实现细节。为了，创建了包含业务流程所需的各个服务的实现的服务组件。其结果就是由以下三层组成的体系结构 - 业务集成逻辑、服务组件和实现，如下图中所示。



因为服务组件包含实现，所以集成开发者可以图形方式组装它们，而无需了解低级别的实现细节。服务组件也可以让集成开发者或者为集成开发者工作的某个人以后添加实现。将如您稍后在产品中所看到的那样，组件是以可视方式组装在一起的。换句话说，您不会面对组件内的代码。如下图所示，在业务逻辑级别，组件是独立于实现来组装的。然后，面向服务的体系结构让您通过使用和复用组件来侧重于解决业务问题，而不是让您一心钻研实现您使用的服务的技术。

业务集成逻辑

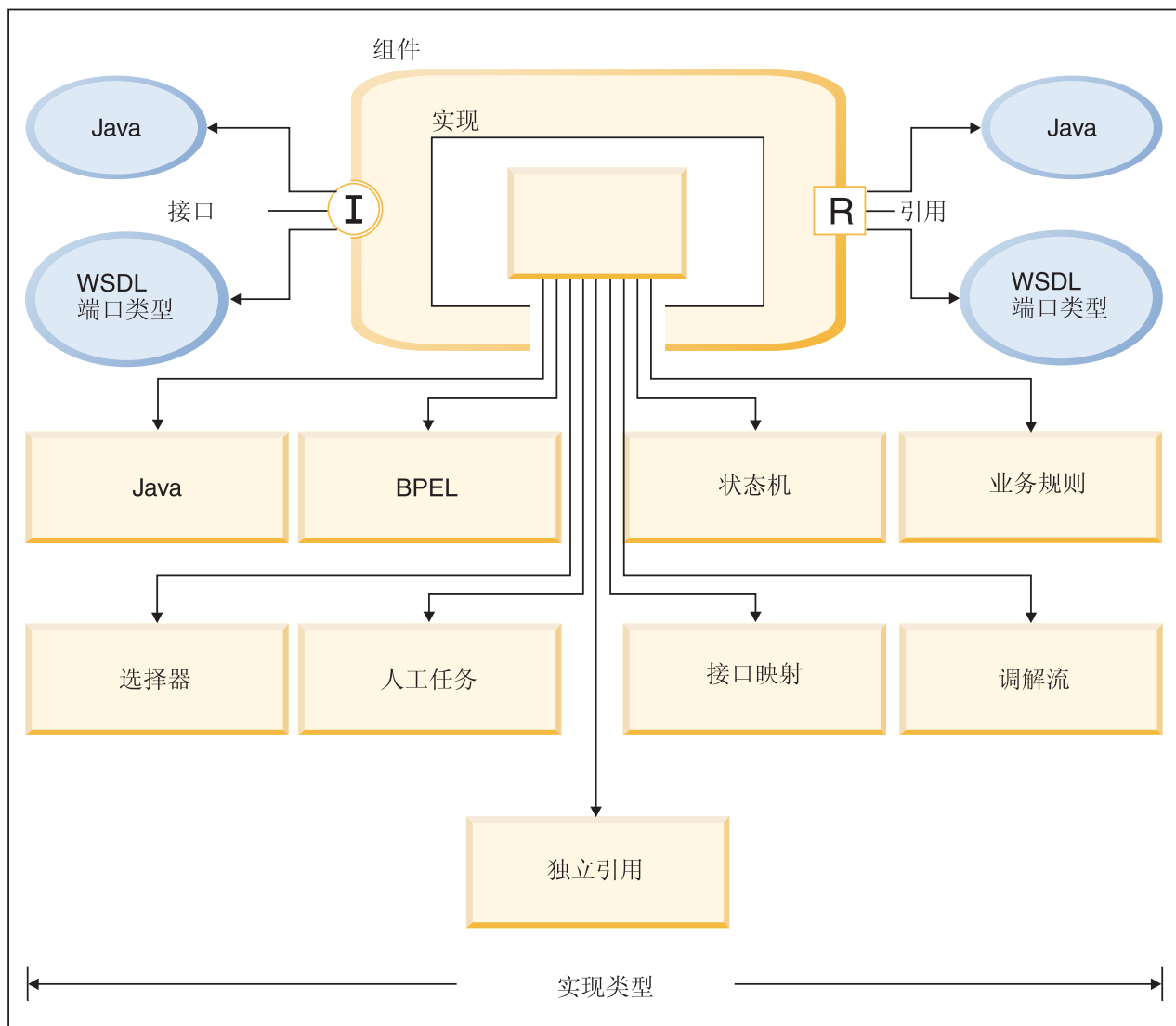


服务组件

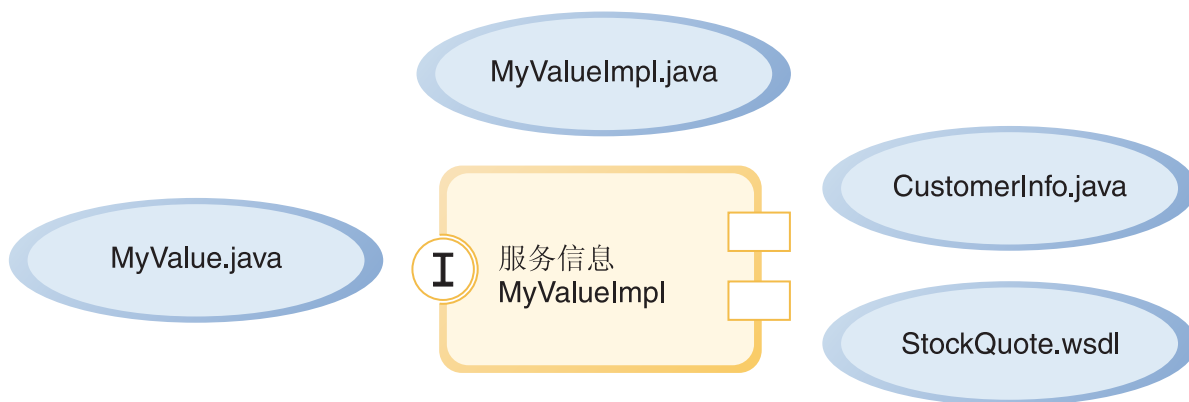
服务组件配置服务实现。服务组件是用标准的方块图来表示的。

组件由实现（使用 WebSphere® Integration Developer 的工具时它是隐藏的）、一个或多个接口（它定义组件的输入、输出和故障）以及零个或多个引用构成。引用标识此组件需要或使用的另一个服务或组件的接口。可用以下一种语言定义接口：WSDL 端口类型或 Java™。接口支持同步和异步交互样式。组件的实现可用各种不同的语言编写。

接口的类型可为 WSDL 或 Java，但如果有多接口，您不能将 WSDL 和 Java 混合起来使用。

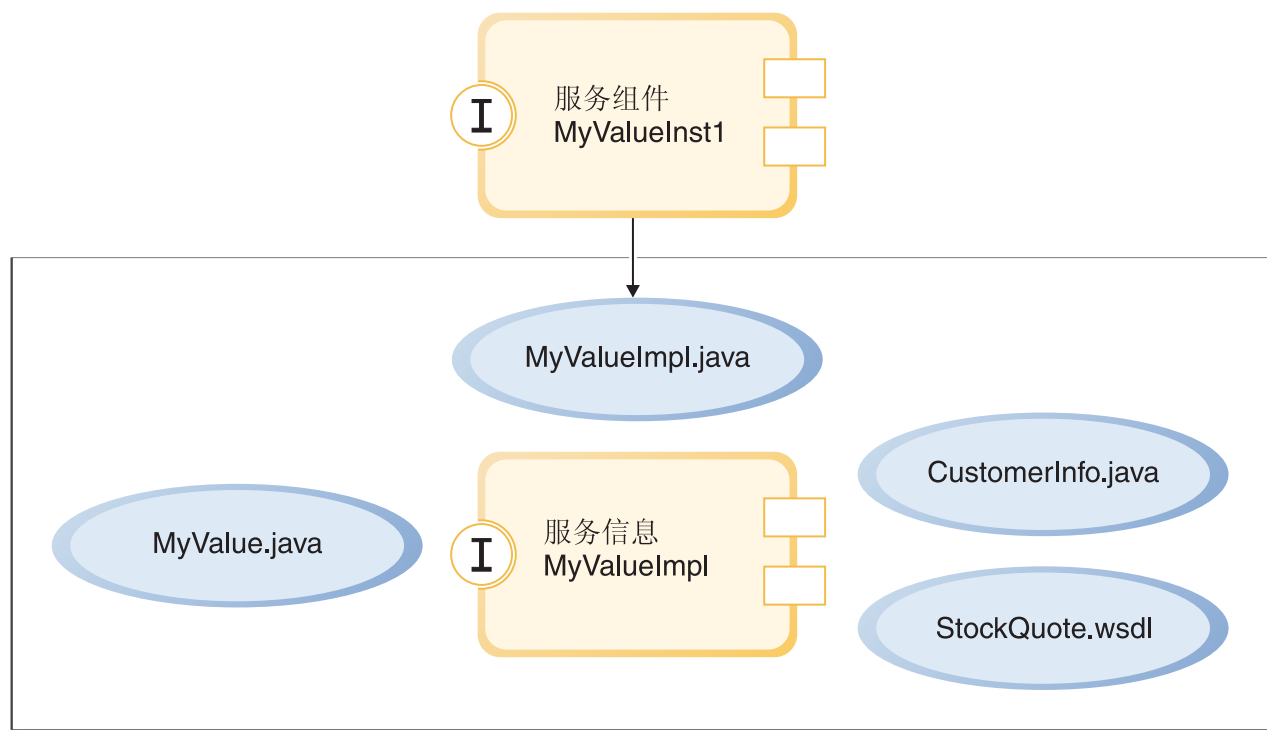


下图中，有个组件位于中心。它的实现 `MyValueImpl` 使用了 `Java` 作为接口。它有两个引用：另一 `Java` 接口和一个 `WSDL` 接口。



当使用此组件时，如下所示，您实际上只看到此组件本身。可看到有条线连到其接口，这显示有从另一组件到此组件的引用。有条线从其引用点连到另一组件的接口，这显示有从此组件到另一组件的引用。引用表示此组件使用的服务。通过命名引用并且仅指定它的接口，引用允许组件实现作者推迟到以后将该引用绑定到实际服务。这以后，集成专家将通过从引用连接到另一个组件或导入的接口来将它绑定到实际服务。这种松耦合是使用 WebSphere Integration Developer 的“服务组件体系结构”的关键原因，因为它允许推迟绑定和复用实现。

组件也可以有属性和限定词。限定词是用于运行时的有关接口和引用的服务质量（QoS）伪指令。

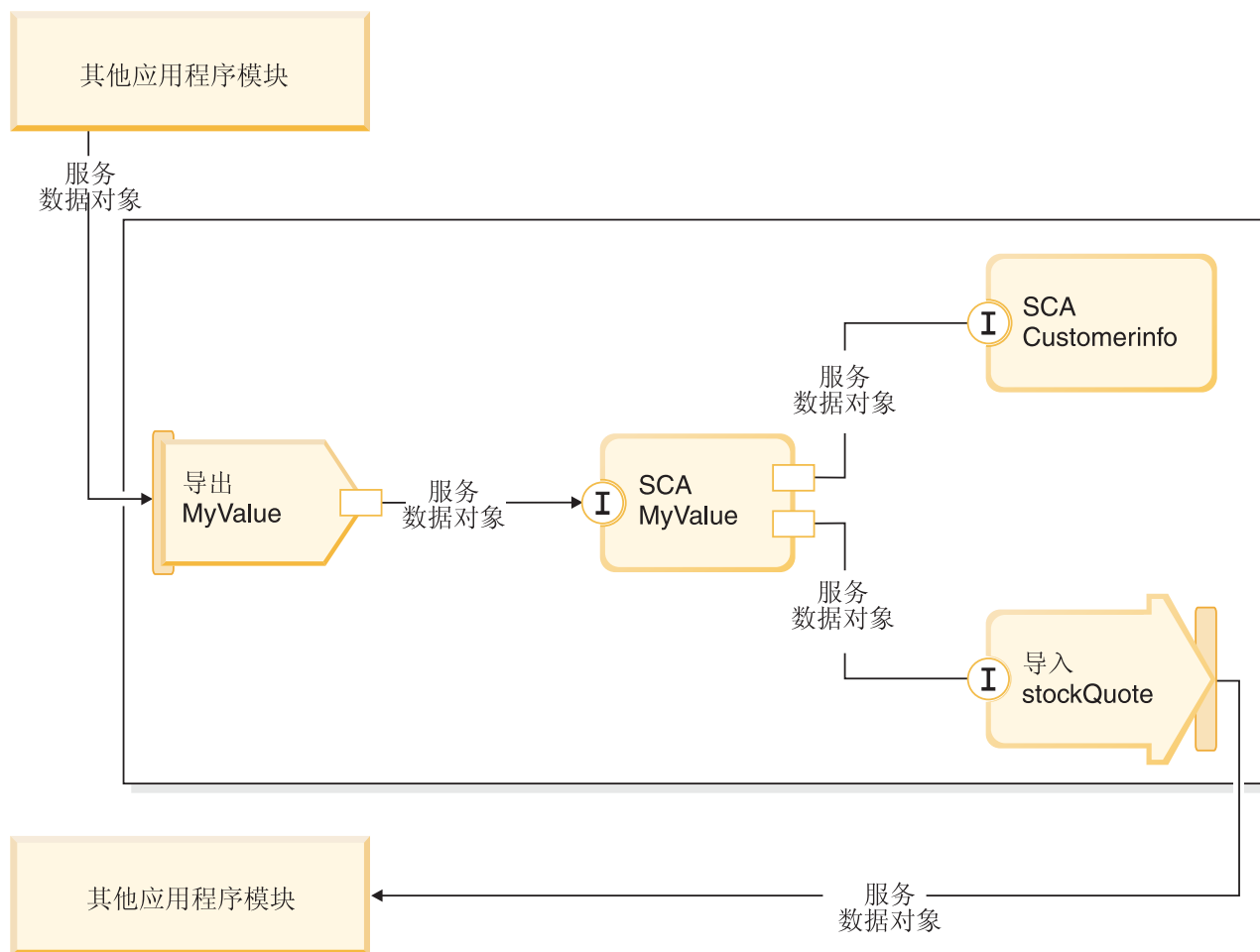


服务数据对象

服务数据对象补充了服务组件体系结构。服务组件体系结构将服务定义为组件并定义组件之间的连接。服务数据对象定义组件之间的数据流。

每个组件都以输入或输出的形式传递信息。当调用一个服务时，如果使用的是 WSDL 端口类型，则会将数据对象作为带有文档文字编码的 XML 文档来传递；如果使用的是 Java 接口，则作为 Java 对象来传递。在“服务组件体系结构”服务中，数据对象是用于数据和元数据的首选形式。与组件类似，服务数据对象将数据对象与其实现分隔开来。例如，一个组件与采购订单进行交互，而采购订单本身可能会使用 JDBC 和 EJB 等来对数据执行更新。服务数据对象让集成开发者侧重于处理业务工件。事实上，服务数据对象对于集成开发者来说是透明的。它们由服务数据对象 Java 规范请求（JSR）定义。

在下图中，服务数据对象进行了下列传递：从外部服务传递到导出，从导出传递到组件，从组件传递到组件，从组件传递到导入，并从导入传递到服务。导入和导出在后面的导入和导出一节中进行了讨论。



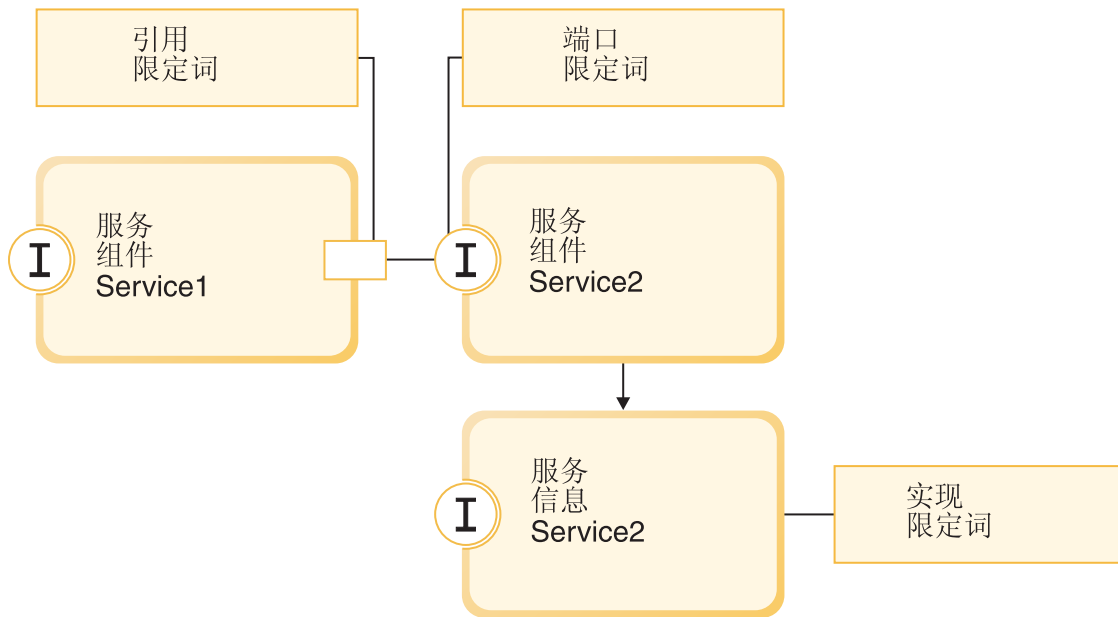
服务限定词

应用程序通过指定**服务限定词**来将其服务质量（QoS）需要告知运行时环境。服务限定词控制服务客户机和目标服务之间的交互。

可以在服务组件引用、接口和实现上指定限定词。因为 QoS 值是在实现外部声明的，所以您可更改这些值而无需更改实现，或者当在不同上下文中使用了同一实现的几个实例时，您可为这些实例设置不同值。

以下各项是限定词的类别：

- 事务 - 用于事务类型的规则
- 活动会话 - 用于连接活动会话的规则
- 安全性 - 用于许可权的规则
- 异步可靠性 - 用于异步消息传递的规则



模块

模块是一个部署单元，它确定将哪些工件打包到企业归档（EAR）文件中。为了提高性能，将一个模块内的组件进行排列，并且可以通过引用传递它们的值。可以将模块看成是一个限定作用域的机制；即，它设置工件的组织边界。

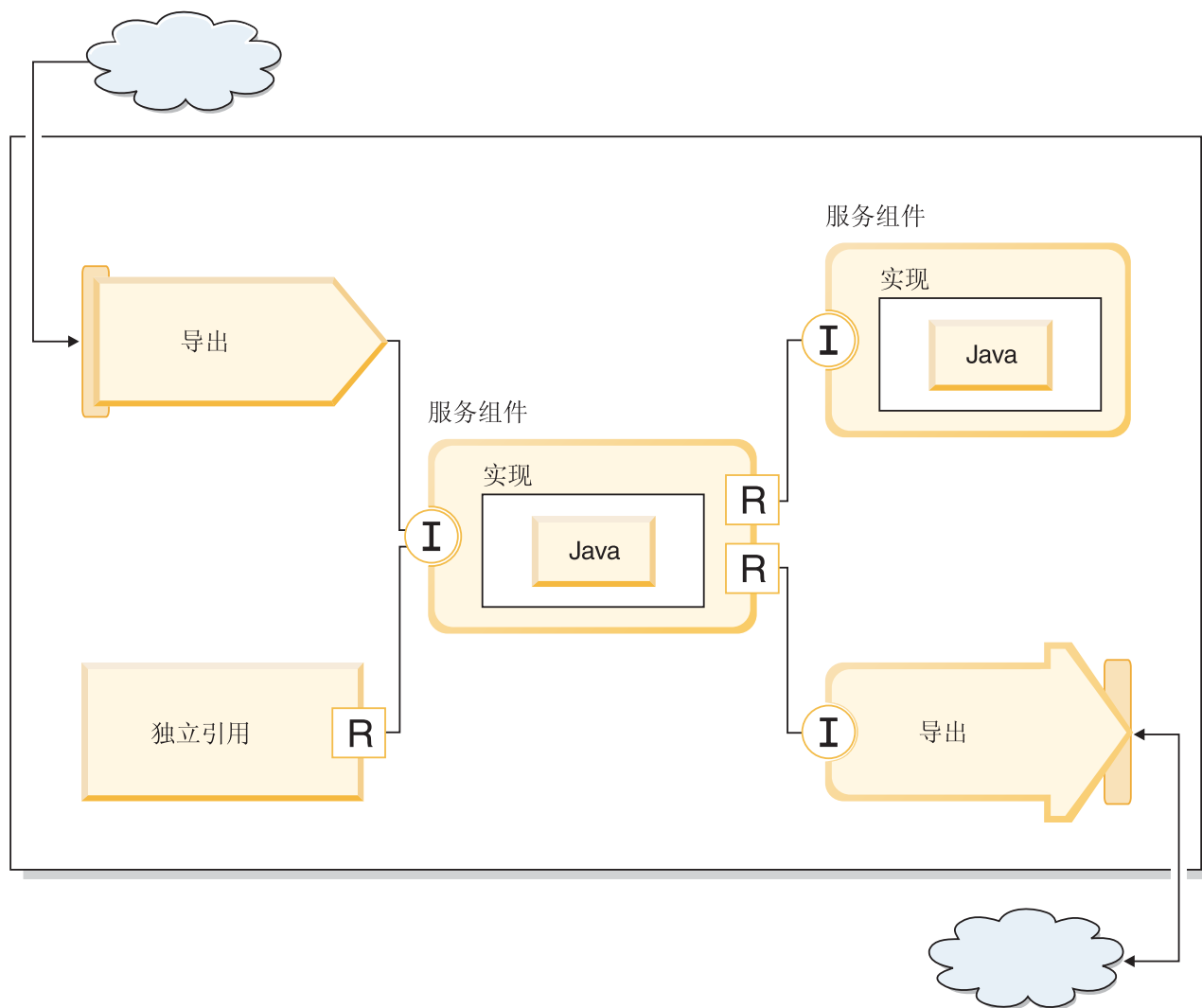
模块是服务组件、导入和导出的组合体。服务组件、导入和导出位于相同项目和根文件夹中，它们还包含链接导入和导出所需的组件和绑定的连接。模块还可以包含它的组件、导入和导出所引用的实现和接口，或放置在其他项目（如库项目）中的实现和接口。

有两种类型的模块。第一种是称为模块（有时称为业务集成模块）的模块，它包含许多组件类型以供选择，通常用来支持业务流程。第二种是称为调解模块的模块，它最多包含一个组件（即调解流组件）和零个或多个增大该调解流组件的 Java 组件。

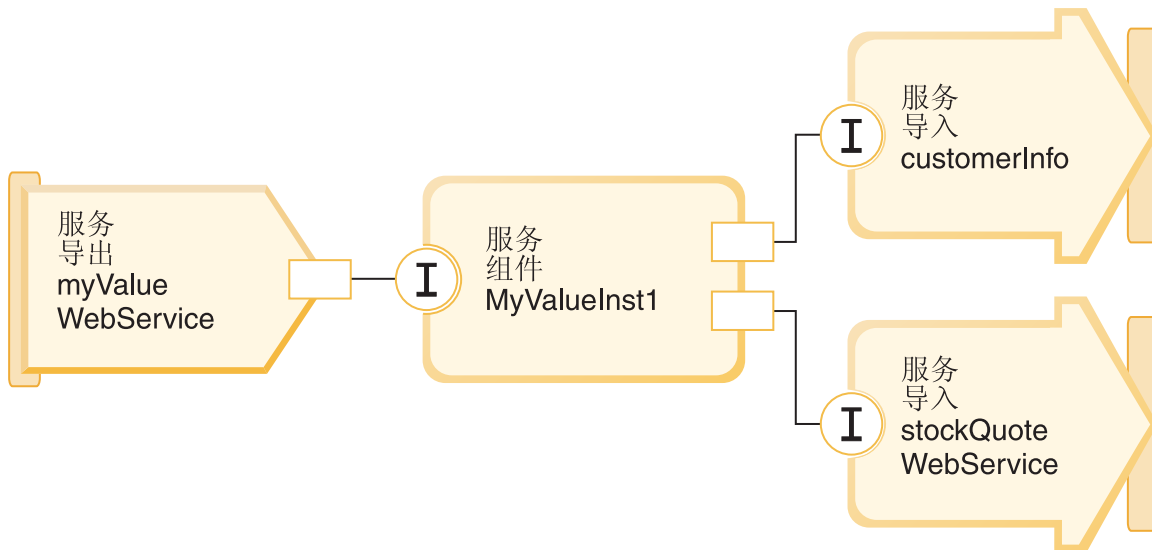
为什么有两种模块类型？第一种类型的模块主要是为业务流程设计的。调解模块就像是现有外部服务的网关，它在企业服务总线体系结构中很常见。这些外部服务或导出在调解模块中由导入或服务提供者访问。通过使用调解流从服务提供者来解耦客户机服务请求者，应用程序可以获得灵活性和弹性，这是面向服务的体系结构的目标。例如，调解流可以记录进入消息、将消息路由到在运行时确定的特定服务或变换数据以使它适合传递到另一个服务。可以随时间推移添加或更改这些功能，而不修改请求者服务或提供者服务。

模块导致测试服务应用程序并将它部署到 WebSphere Process Server。调解模块导致测试服务应用程序并将它部署到 WebSphere Process Server 或 WebSphere Enterprise Service Bus 服务器。这两种类型的模块都支持导入和导出。

通常需要在模块间共享实现、接口、业务对象、业务对象映射、角色、关系和其他工件。库是用来存储这些共享资源的项目。



在下图中，模块包含一个导出、两个导入和一个使用它们的服务组件。显示了链接接口和引用的连接。



模块和调解模块工件包括:

- 模块定义 - 定义模块。
- 服务组件 - 模块中服务的定义。模块内服务组件的名称是唯一的。但是，服务组件可以有一个任意的显示名称，它通常是对用户更有用的名称。
- 导入 - 导入的定义，它们是对此模块的外部服务的调用。导入具有绑定，在导入和导出一节中对它们进行了讨论。
- 导出 - 导出的定义，它们用来将组件显示给此模块的外部调用者。导出具有绑定，在导入和导出一节中对它们进行了讨论。
- 引用 - 模块中从一个组件到另一个组件的引用。
- 独立引用 - 引用未定义为“服务组件体系结构”组件的应用程序（例如，JavaServer Page），这使得这些应用程序能够与“服务组件体系结构”组件交互。每个模块只能有一个独立引用工件。
- 其他工件 - 这些工件包括 WSDL 文件、Java 类、XSD 文件、BPEL 流程等。

导入和导出

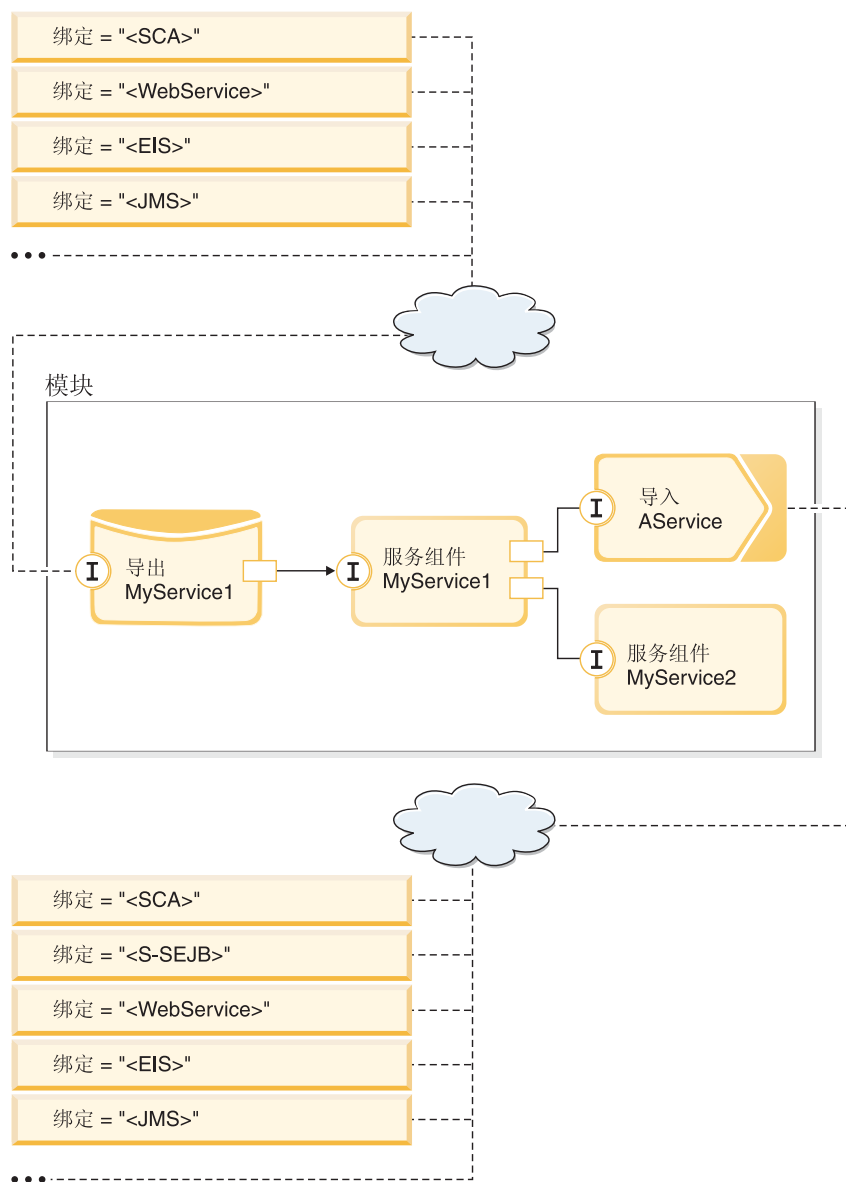
导入和导出定义模块的外部接口或访问点。导入标识模块外的服务，以便可以从模块内部调用它们。导出允许组件将他们的服务提供给外部客户机。导入或导出都需要绑定信息。提供了几种绑定，并提供一些关于哪种类型可能适用于您的应用程序的建议。

服务导入和导出绑定类型

导入和导出需要绑定信息，绑定信息指定了从模块传输数据的方式。导入绑定描述了外部服务绑定到导入组件的特定方式。导出绑定描述使模块的服务可用于客户机的特定方式。

SCA 或缺省绑定允许您的服务与其他模块中的其他服务通信。带有 SCA 绑定的导入允许您访问另一模块中的服务。带有 SCA 绑定的导出允许您为另一模块提供服务。Web service 导入绑定允许您将外部 Web service 绑定到导入。Web service 导出绑定允许您将服务作为 Web service 提供给外部客户机。企业服务发现向导创建代表 EIS 系统上的服务的导入和导出。创建的绑定为 EIS 类型或 Java 消息服务 (JMS) 类型。EIS 绑定允许与 EIS 系统上的服务进行同步通信。JMS 绑定通常用在与大型 EIS 系统的交互中，在大型 EIS 系统中通过消息队列进行的异步通信对于可靠性是非常关键的。导入（虽然不是导出）也可具有无状态会话 EJB 绑定。

组装编辑器列示支持的绑定，并在您想要创建导入或导出时简化创建过程。组装编辑器中的属性视图显示了任何导入或导出的绑定信息。



选择适当的绑定

本节讨论了某个绑定何时更符合您的应用程序需求。

WebSphere Integration Developer 中可用的绑定提供了一些选项。此列表帮助您确定一种类型的绑定何时比另一种类型的绑定更符合您的应用程序需求。

下列因素适用时，请考虑 SCA 绑定：

- 所有服务都包含在 WebSphere Integration Developer 模块中；即，没有外部服务
- 性能非常重要
- 模块是紧耦合的。

下列因素适用时，请考虑 *Web Service* 绑定：

- 需要通过因特网访问外部服务，或通过因特网提供服务
- 服务是松耦合的
- 您正在访问的外部服务协议或您希望提供的服务协议为 SOAP/HTTP 或 JMS/HTTP。

下列因素适用时，请考虑 *EIS* 绑定：

- 需要使用资源适配器访问 *EIS* 系统上的服务
- 性能比可靠性更重要；即，同步数据传输优先于异步数据传输。

下列因素适用时，请考虑 *JMS* 绑定：

- 需要访问消息传递系统
- 服务是松耦合的
- 可靠性比性能更重要；即，宁可进行异步数据传输，而不进行同步数据传输。

下列因素适用时，请考虑无状态会话 *EJB* 绑定：

- 该绑定用于本身是 *EJB* 的已导入服务
- 已导入服务是松耦合的
- *EJB* 的状态不重要。

服务实现类型

服务实现类型是服务组件的各种实现。

本节描述了服务的标准实现。这些实现将出现在组装编辑器或者 *BPEL* 流程中的服务中。

Java 对象

Java 组件的实现称为 Java 对象。

一种常见的实现是用 Java 编写的组件。有时将这种实现昵称为“传统的 Java 对象”或 *POJO*。虽然这种实现还可以具有 Java 接口，但是它通常具有 *WSDL* 接口类型。如果指定了多个接口，则不能将 *WSDL* 接口与 Java 接口混合使用。但是，您可以将使用一组 *WSDL* 接口创建的应用程序与使用一组 Java 接口创建的应用程序“连接”起来。“欢迎”视图的样本库中列示的样本说明了是如何将它们“连接”起来的。

当处理 Java 对象时，其代码隐藏在编辑器的上下文中，您看不见代码。

可以在调解模块中使用 Java 对象。可以将它部署到 WebSphere Process Server 或 WebSphere Enterprise Service Bus 服务器。

BPEL 流程

BPEL 流程组件实现业务流程。

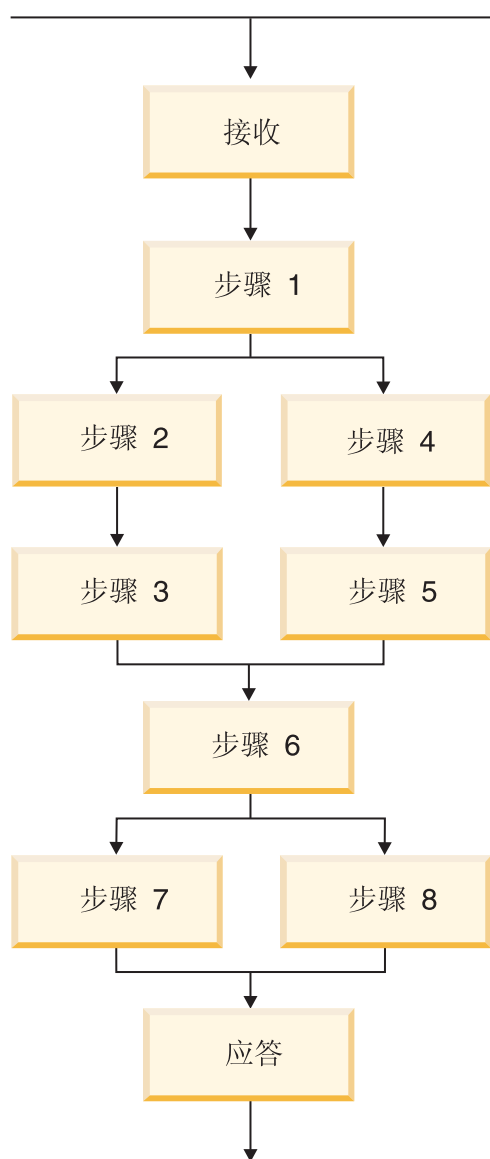
它的实现语言是业界标准的 *Web service* 的业务流程执行语言（*BPEL4WS*）及其 IBM 扩展。*BPEL* 流程通过使用多个基本服务来实现潜在的长时间运行服务。在流程编辑器中创建的 *BPEL* 流程可以执行下列操作：

- 使用控制流图描述其他服务的组合
- 使用变量来保存流程状态
- 通过故障处理进行复杂的错误处理

- 支持异步事件
- 通过使用关联集来标记请求内用来标识实例的业务数据（例如，客户标识），将入站请求与特定流程的确实实例进行关联
- 通过复杂的补偿支持提供扩展的事务

除了这些标准 BPEL 项以外，WebSphere Integration Developer 还扩展了 BPEL，使用人工任务支持将人员包括进流程。例如，此扩展可将某人批准贷款的要求添加到流程中。

流程编辑器使用 BPEL 构造的可视表示来快速和简便地构建业务流程。

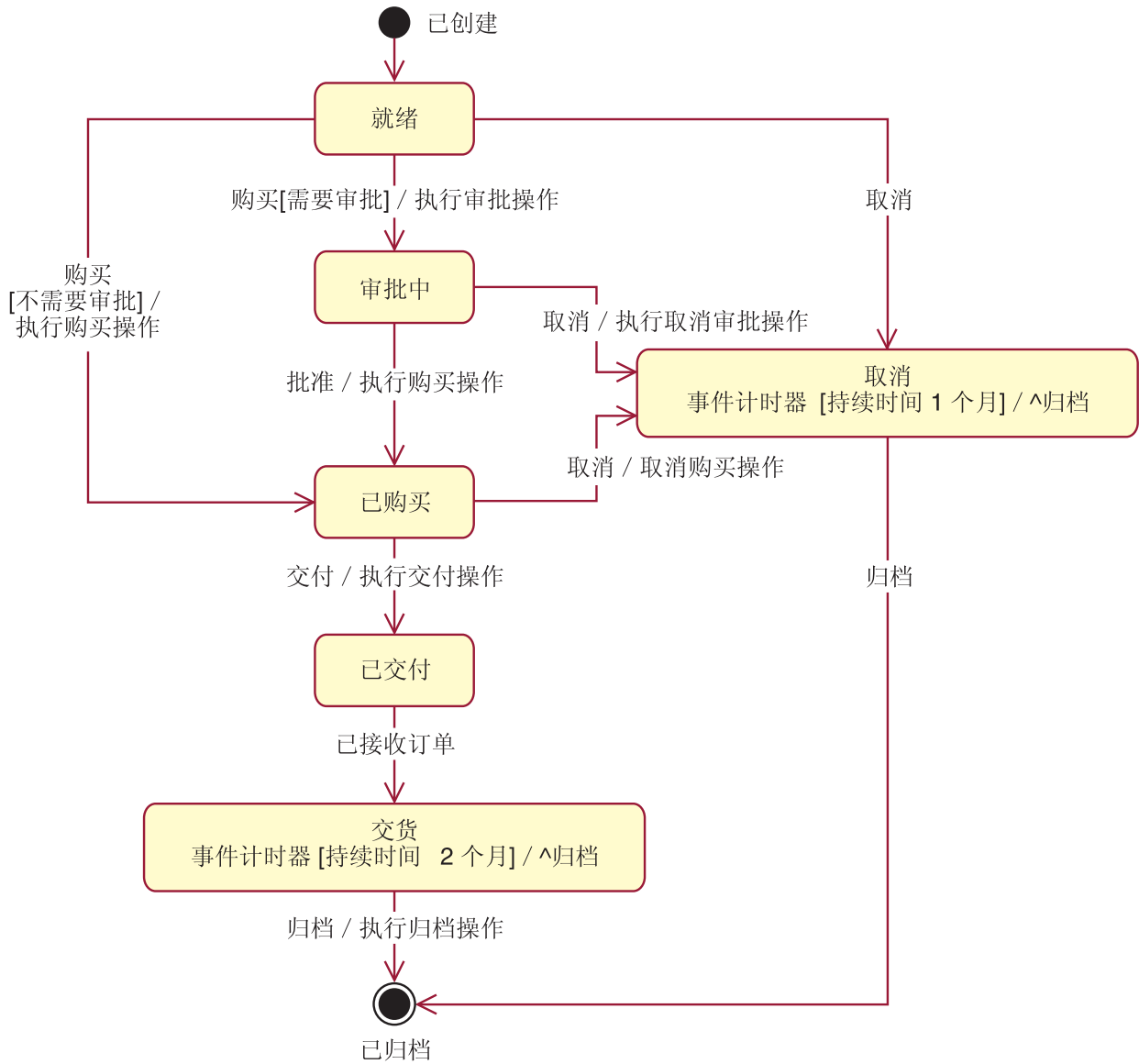


不能在调解模块中使用 BPEL 流程。只能将它部署到 WebSphere Process Server。

状态机

状态机是另一种创建业务流程的方法。状态机适合与更改状态而不是控制流相关的流程。状态定义了一个工件在某一时刻可以执行的操作。状态机是这组状态的实现。

状态机是显示流程中一组相关状态的常用方法。饮料自动售卖机就是一个大家都很熟悉的状态机。当您投入硬币投入该机器中时，它就会自动为您提供饮料，同时还会准确找回零钱，因为该状态机会根据您投入的硬币以机械方式找回需要退还给您的硬币。在下图中，显示了一个使用状态机编辑器创建的典型状态机。在该状态机中购买了一件商品，并且将该商品提供给客户。



不能在调解模块中使用状态机。只能将它部署到 WebSphere Process Server。

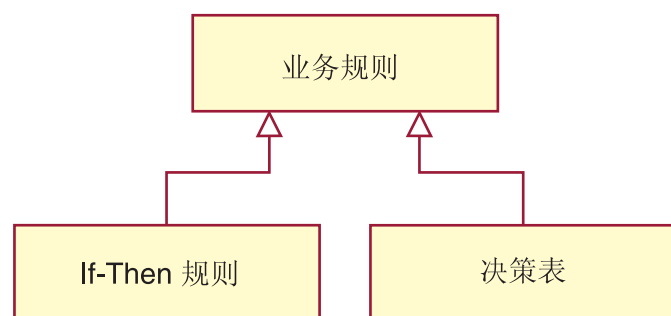
业务规则

业务规则补充了业务流程和状态机。例如，如果存在具有变量的条件，则业务规则可以在运行时更改该变量中的值。业务规则是使用可视编程语言创建的，它根据上下文作出决定。决定可以是简单的，也可以是复杂的。业务规则不是程序上的，可以独立于应用程序更改规则。

业务规则根据上下文来确定流程的结果。每天的业务情形中都使用了业务规则来根据特定的一组情况作出决定。此决定可能需要许多规则来覆盖所有情况。业务流程中的业务规则允许应用程序迅速地响应不断变化的业

务情况。例如，在保险公司中，批准申请者的车险的业务规则可以是：如果申请者是超过 25 岁的男性，车型为跑车，并且过去 5 年内也是在我们公司投保的，则批准该保险申请的保费为每月 100 美元。

WebSphere Integration Developer 提供了许多方法来创建业务规则。您可以创建 if-then 规则或决策表，所有这些方法都将会确定流程的结果。注意，这些规则是独立于流程本身的，这就意味着可以在任何时候更改这些规则而无需重做流程。例如，根据您的业务所在的位置，您可能制订了这样一条规则：如果日期是在 12 月 26 日到 1 月 1 日之间，则提供 20% 的节后销售折扣。但是，如果销售量仍然很低，则随时都可以将折扣修改为 40%。

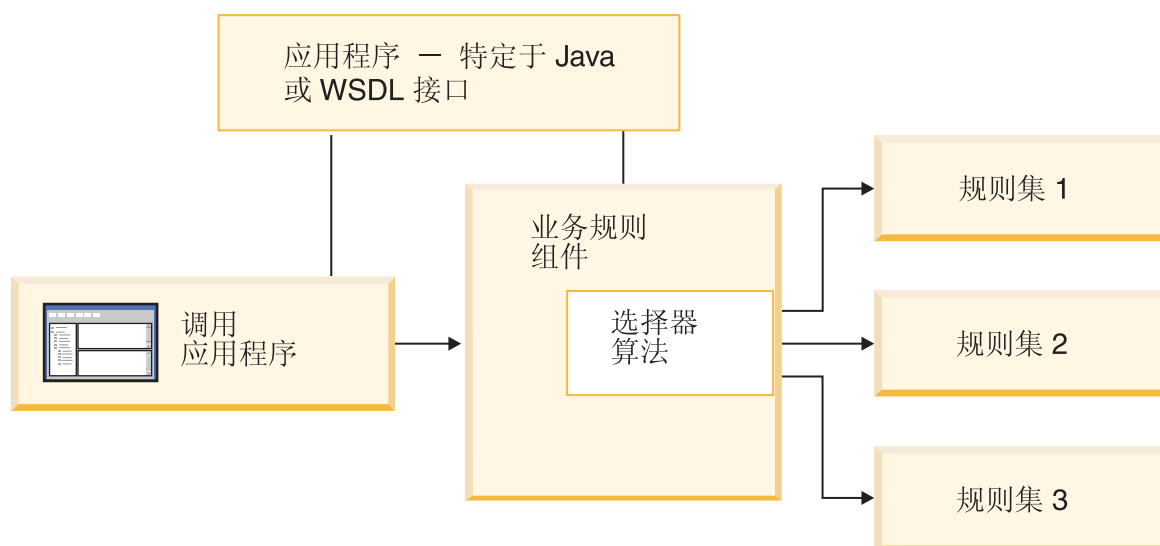


不能在调解模块中使用业务规则。只能将它们部署到 WebSphere Process Server。

选择器

集成应用程序包含多种交互方法。选择器用于将客户机应用程序中的操作路由到实现中的几个可能存在的组件。

根据日期路由到组件。例如，以下是一个根据日期的路由：在开学之前的两个星期，为与学校有关的商品提供“返校”特价。业务可能有许多这种根据日期的路由。选择器在运行时根据日期决定选择一个路由，而不选择另一个。例如，如果时间刚好是在开学之前，则会调用前面提到的“返校”特价。但是，如果是学校临近期末的季节，则提供一个让孩子们度过夏季的折扣。

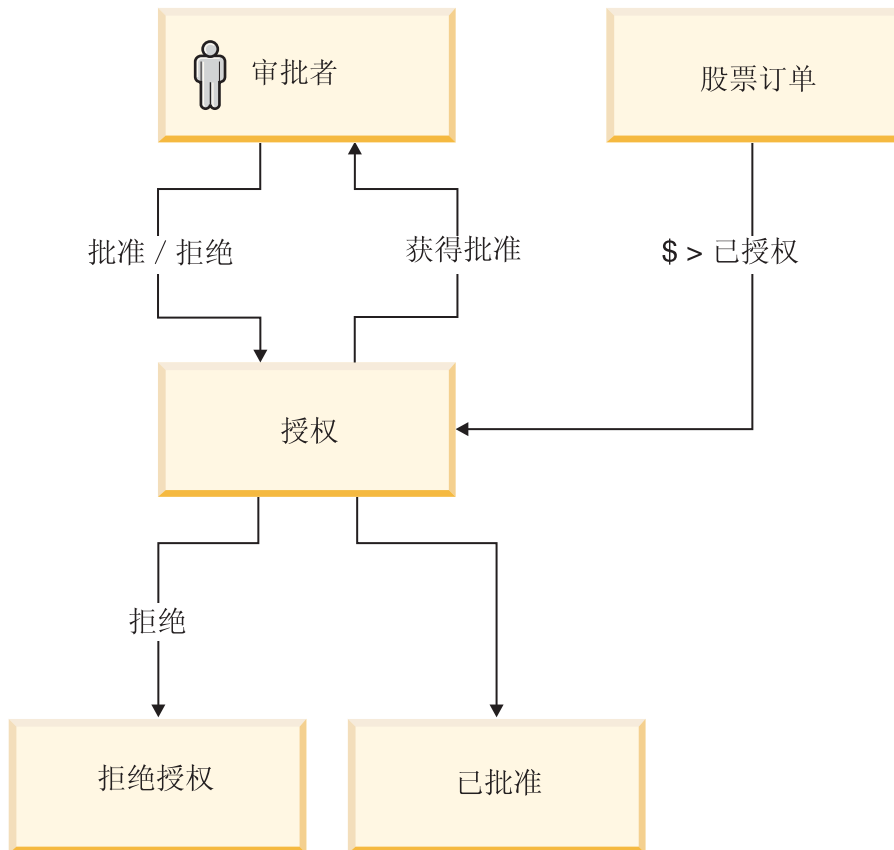


不能在调解模块中使用选择器。只能将它部署到 WebSphere Process Server。

人工任务

人工任务组件实现由某人完成的任务。它表示某人如何参与到业务流程中。

有时需要有人参与到业务流程中。例如，一个客户想购买一件超过了他的贷记限额的商品。人工任务允许您干预和覆盖将阻止客户购买商品的业务规则。人工任务可拥有属性，例如，设置任务的所有者以及在指定的人不在的情况下提供升级处理。人工任务组件认识到许多流程中的任务需要人的参与，比如，查看、搜索和审批。



不能在调解模块中使用人工任务。只能将它部署到 WebSphere Process Server。

接口映射

接口映射涉及到交互组件的接口之间的差别。

彼此间需要交互的组件的接口之间存在差别是很常见的。之所以会产生这些差别，是因为在 WebSphere Integration Developer 中通常会组装为不同应用程序创建的组件。复用组件以创建新的应用程序是 WebSphere Integration Developer 的一个优点，因为这样您就不必重新编写类似组件的代码。但是，您通常必须进行一些调整。

例如，两个组件的方法执行基本相同的操作，但具有不同的名称，如 `getCredit` 和 `getCreditRating`。它们还可以具有不同的操作名，并且操作可能具有不同的参数类型。“接口映射”映射这些方法的操作和参数，以解决差别问题，这样两个组件就可以进行交互。接口映射就像是两个组件的接口之间的桥，它允许接口连接在一起，而无论它们是否存在差别。

接口映射独立于使用它的组件存在，这表示不需要更改组件本身。

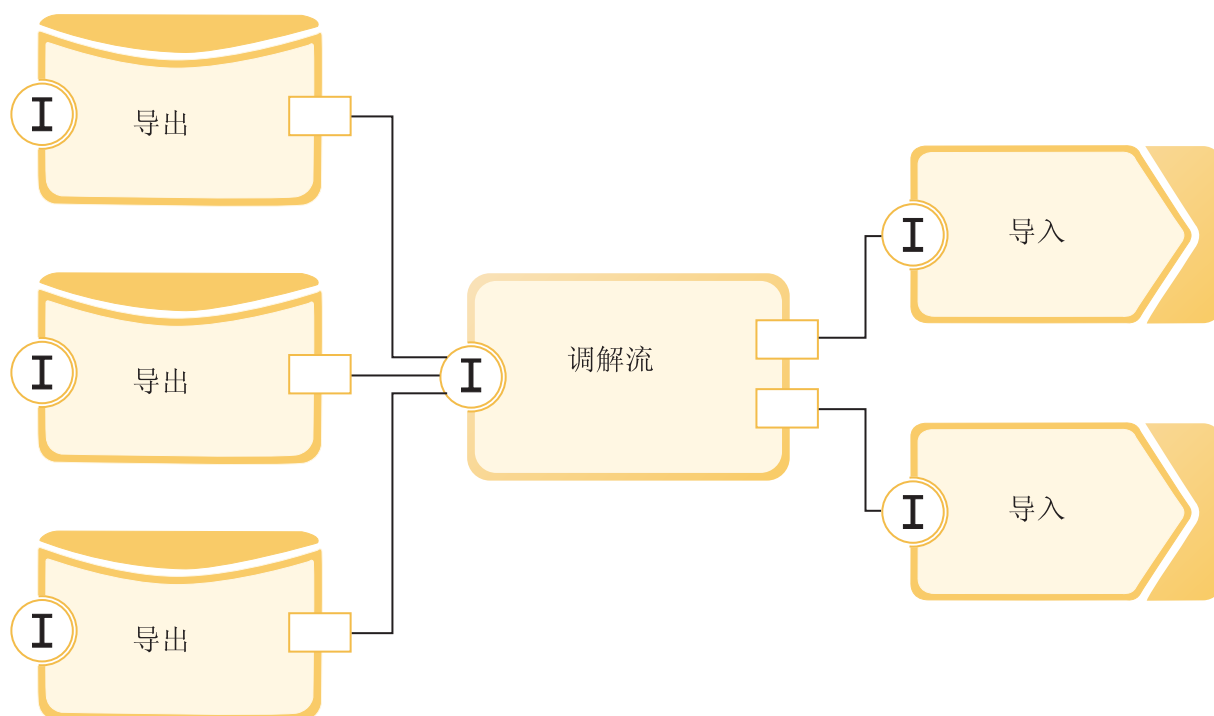
不能在调解模块中使用接口映射。只能将它部署到 WebSphere Process Server。

调解流

调解是一种动态调解或干预服务的方法。调解流实现调解。

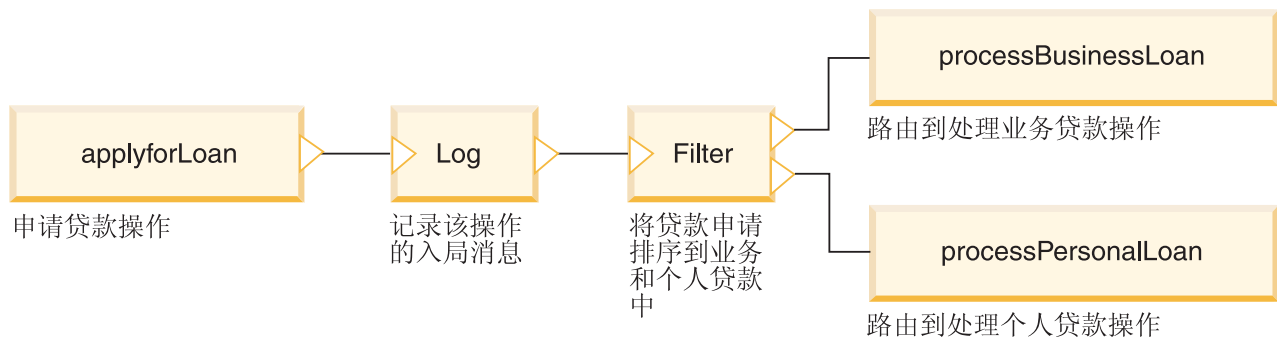
调解有一些有用的功能。例如，如果需要将一个服务中的数据变换为后续服务可接受的格式，则可以使用调解。记录功能允许您先记录来自某个服务的消息，然后再将这些消息发送到下一个服务。路由功能允许您将来自一个服务的数据路由到调解流确定的适当服务。调解的运作独立于它连接至的服务。组装编辑器中的调解显示为导出和导入之间的调解流组件。

在下图中，三个服务请求者或导出将他们的输出数据发送到调解流组件的接口。然后，调解流组件将适当的数据路由到两个服务提供者或导入。



调解流是使用调解流编辑器创建的类似于流的结构。通过在组装编辑器中选择调解流组件，可启动调解流编辑器。在调解流编辑器中，将一个服务（服务请求者或导出）中的操作映射到另一个服务（服务提供者或导入）的操作和调解流编辑器提供的功能。这些功能称为调解原始节点，它们在调解流中连接，如下图所示。调解原始节点是 IBM 提供的，您可以创建自己的定制原始节点。调解原始节点可以对消息内容和消息上下文起作用，其中上下文是特定于绑定的信息，如 SOAP 或 JMS 头，或用户定义的属性。

在下图中，操作 `applyforLoan` 先将消息发送给记录消息的记录原始节点 `Log`。`Log` 将消息发送给过滤器原始节点，然后该节点根据消息将消息路由到 `processBusinessLoan` 操作或 `processPersonalLoan` 操作。



正如在模块部分中所讨论的那样，有一个用于调解流组件的调解模块。它最多能包含一个调解流组件和零个或多个增大该调解流组件的 Java 组件。可以将调解模块部署到 WebSphere Process Server 或 WebSphere Enterprise Service Bus 服务器。

独立引用

独立引用是对未定义为“服务组件体系结构”组件的应用程序（例如，JavaServer Page 或 servlet）的引用。独立引用允许这些应用程序与“服务组件体系结构”组件交互。

独立引用既没有接口也没有实现（因为实现在模块的作用域外）。模块可以不包含独立引用，或者包含一个独立引用工件。独立引用的实用价值是允许您将现有应用程序与在 WebSphere Integration Developer 中创建的“服务组件体系结构”组件一起使用。

可以在调解模块中使用独立引用。可以将它们部署到 WebSphere Process Server 或 WebSphere Enterprise Service Bus 服务器。

相关信息

本节讨论了与体系结构相关的几个主题。

这些主题提供一些与产品的体系结构关联的其他信息。

- 『开发与 .NET 服务一起使用的服务』
- 第 21 页的『双向支持』

开发与 .NET 服务一起使用的服务

如果您打算开发与 .NET 服务一起使用的服务，则需要了解一些特殊注意事项。这些注意事项包括两个方面：一是在 .NET 开发环境中开发的 WSDL 文件引入 WebSphere Integration Developer 开发环境中，二是从 WebSphere Integration Developer 中导出 WSDL 文件，以便可以将它们与 .NET 服务一起使用。这两个开发环境的关键区别在于 .NET 环境使用内联模式，而 WebSphere Integration Developer 不使用。内联模式是将模式包括在 WSDL 文件中的方法，而不是指定将模式作为单独文件导入的方法。在 WebSphere Integration Developer 中提供了一些信息，可以帮助您处理这两种情况。

“创建接口”部分中的内联模式说明了如何导入包含内联模式的 WSDL 文件（如 .NET WSDL 文件）。

“组装业务服务”部分中的创建代理以使用 .NET 服务说明了如何用这种方法来外部发布服务从而使用 .NET 服务。本主题着重介绍如何创建使用 .NET 服务的代理。

双向支持

WebSphere Integration Developer 在多语言环境中工作。这意味着它可以显示和处理用不同语言表示的数据。此支持中包含一些具有双向脚本的语言（例如，阿拉伯语或希伯来语）。这些语言是从右到左书写的，但嵌入在此文本中的拉丁语（或 Cyrillic、希腊语等等）数字和嵌入式段是从左到右书写的。

在 WebSphere Integration Developer 中的双向脚本支持概述中讨论了对双向语言的支持，包括必需的配置、使用该支持时的一些特定技术要点和局限性。

第 3 章 了解工具

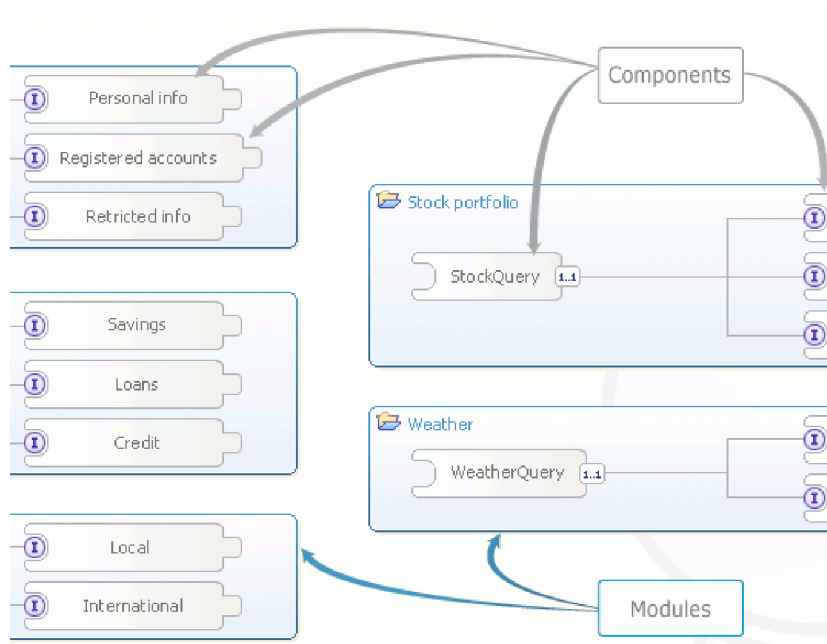
WebSphere Integration Developer 使用可视工具来根据面向服务的体系结构创建集成应用程序。

“欢迎”视图的概述部分可以让您对 WebSphere Integration Developer 的工具进行最高级别的了解。在这种情况下，最高级别意味着这些工具的最抽象表示。这些图形说明了这些工具可以完成的任务。如果您不熟悉该产品但是又想快速了解它的各个工具的用途，则请阅读概述部分。教程部分（也可在“欢迎”视图找到）则让您查看工具如何操作。样本部分让您开始使用工具（但是以一种故障自恢复的方式）。最后，当您开始使用 WebSphere Integration Developer 开发应用程序时，使用信息中心来查找有关每个工具的深入信息，包括概念性信息、您可以执行的任务和有关每个工具的参考信息。

“欢迎”视图概述

“欢迎”视图的概述部分提供了对产品中的工具和功能部件的最高级别概述。在概述中，使用了图形来说明关键工具在产品中的作用。

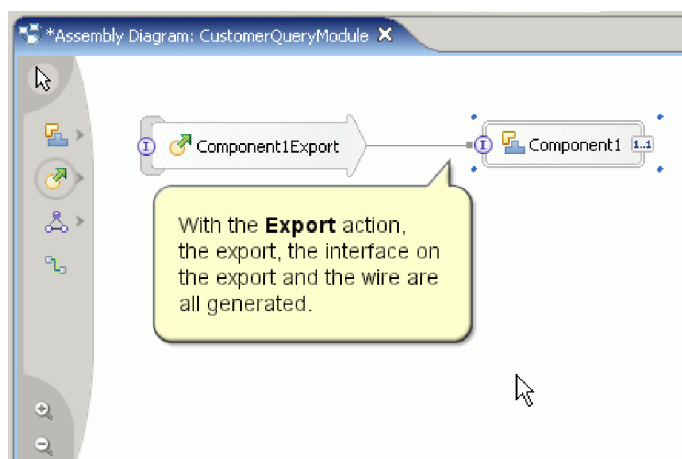
“欢迎”视图的概述部分可以让您在实际使用此产品之前快速了解它。产品的绝大多数新用户都应该从此处开始了解该产品。从概述中选择一个图标就会启动有关一个工具或一个主题的图形。例如，组件和模块的关系如下图所示。您还可以启动产品的音频视频导览。



“欢迎”视图中的教程

教程让您以一种故障自恢复的方式了解工具的操作。选择一个教程，然后在它象一个用户那样继续执行一些任务时查看它。

一旦您在概念级别上了解了此产品，您就可以通过启动教程来练习使用工具。每个教程都以电影方式演示了一个特定工具。除了查看教程外，还有冒泡文本说明了您看到的是什么内容。可以在“欢迎”视图的教程库的“边看边学”部分找到这些 WebSphere Integration Developer 教程。



“欢迎”视图中的样本

这些样本可以让您实际练习在故障安全环境中使用 WebSphere Integration Developer 来开发业务服务解决方案。

可以在“欢迎”视图的“样本库”中找到这些样本。可以通过多种方式来选择一个样本。您可以自己按照逐步的指示信息来构建它，也可以让计算机为您构建它。

各个样本的作用域是不同的。技术样本侧重于执行特定任务的特定工具。应用程序样本则使用多个工具来获得更复杂的结果。场景是较长的样本，它使用许多工具来构建一个大型的应用程序。每个样本都列示了构建它将花费的时间。

Business state machine (simple)

This sample demonstrates how a business state machine can be used to moderate a sales order transaction. Specifically, the state machine emulates an on-line brokerage that manages the selling of a share.

The following tools are used in this application:

- Business state machine editor
- Human task editor

To import the ready-made sample, click the **Import** link below and click **Finish** in the opened wizard. See **Running instructions** to run the imported code.

or

If you want to build the sample for yourself, click **Step-by-step instructions**.

Ready-made sample

[Import](#) [Running instructions](#)

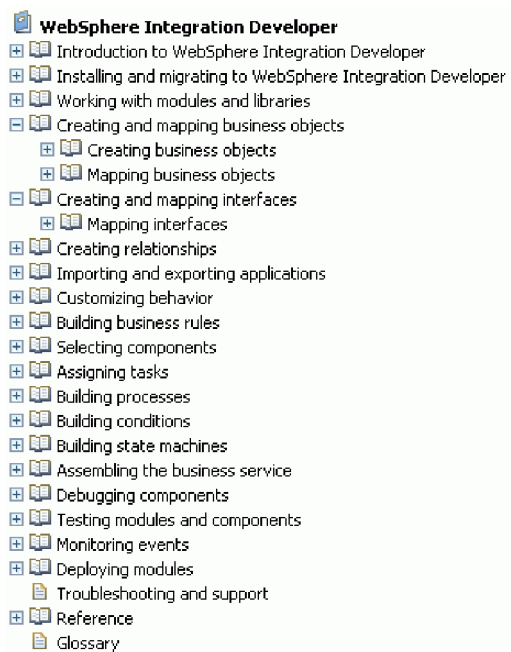
Build it yourself

[Step-by-step instructions](#)

信息中心

信息中心提供了有关产品的完整信息。除了教程和样本之外，还可以找到有关 WebSphere Integration Developer 的每个工具的更多概念、任务和参考信息。

通过在信息中心中导航可以使您快速找到想要的信息。您还可以搜索整个信息中心。选择搜索结果，以查看包含有关主题的深入信息的各种主题以及与相关信息的链接。



声明

本 IBM® 产品中包括的 XDoclet 文档经许可才能使用，并且包含在以下著作权归属声明中：Copyright (c) 2000-2004, XDoclet Team. All rights reserved.

部分内容基于由 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 合著的 *Design Patterns: Elements of Reusable Object-Oriented Software*, Copyright (c) 1995 by Addison-Wesley Publishing Company, Inc. All rights reserved.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

本信息是为在美国提供的产品和服务编写的。IBM 可能在其他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查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，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

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区：INTERNATIONAL BUSINESS MACHINES CORPORATION“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。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；这些更改将编入本资料的新版本中。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。那些 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他程序（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：

Intellectual Property Dept. for Rational Software
IBM Corporation
20 Maguire Road
Lexington, Massachusetts 02421-3112
U.S.A.

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 国际软件许可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，在其他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、其出版说明或其他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、兼容性或任何其他关于非 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。

本信息包含在日常业务经营中使用的数据和报告的示例。为了尽可能完整地说明这些示例，这些示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。用户如果是为了按照 IBM 应用程序编程接口开发、使用、经销或分发应用程序，则可以任何形式复制、修改和分发这些样本程序，而无须向 IBM 付费。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明：

(C)（贵公司的名称）（年）。此部分代码是根据 IBM 公司的样本程序衍生出来的。(C) Copyright IBM Corp. 2000, 2006. All rights reserved.

如果您正以软拷贝格式查看本信息，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息用来帮助您使用此程序来创建应用软件。

通用编程接口允许您编写获取此程序工具的服务的应用软件。

但是，此信息也可能包含诊断、修改和调整信息。这些诊断、修改和调整信息用于帮助您调试应用软件。

警告： 不要将此诊断、修改和调整信息用作编程接口，因为它是会更改的。

商标和服务标记

请参阅 <http://www.ibm.com/legal/copytrade.shtml>。



中国印刷

S151-0253-01

