IBM DB2 Information Integrator

应用程序开发者指南

版本 8.2

S152-0601-01

IBM DB2 Information Integrator

应用程序开发者指南

版本 8.2

S152-0601-01

在使用本资料及其支持的产品之前，请阅读第 261 页的『声明』中的一般信息。

本文档包含 IBM 的专利信息。它在许可证协议下提供，并受版权法保护。本出版物包含的信息不包括任何产品保证，且本手册提供的任何声明不应作如此解释。

可以在线方式或通过您当地的 IBM 代表订购 IBM 出版物。

- 要以在线方式订购出版物，可访问“IBM 出版物中心”（IBM Publications Center），网址为 www.ibm.com/shop/publications/order
- 要查找您当地的 IBM 代表，可访问“IBM 全球联系人目录”（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide

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

前言

本书显示 IBM® DB2 Information Integrator 为什么是帮助您通过统一的视图和数据放置集成数据的解决方案。它还显示如何充分地利用信息。

谁应阅读本指南?

数据管理员、信息分析员、系统集成者、Web 集成者、数据库管理员、数据结构设计者和应用程序开发者都可使用 IBM DB2 Information Integrator 解决方案来创建关键性的开放式信息集成平台。

术语

IBM DB2 Information Integrator 使用数据库、连接、结构化查询语言（SQL）和局域网（LAN）概念的标准术语。在本书中使用的所有 DB2 Information Integrator 概念都在词汇表中作了定义。除非另外指定，否则假定下列含义:

数据 原始情况。它可以是结构化的、非结构化的或半结构化的。通常会组织数据以进行分析。数据还帮助您作出决策。

信息 可用格式的数据，通常以某种方式处理或解释。

DB2 Information Integrator 产品

IBM DB2 Information Integrator 在几种产品中可用。仔细阅读许可证协议以了解用于您所安装的版本的条款和条件。有关安装和配置 Information Integrator 的信息，请参阅《*DB2 Information Integrator 安装指南*》。有关信息集成发展的更多信息，请参阅信息集成支持站点 http://www.ibm.com/software/data/integration/db2ii/support.html。

第 1 章 信息集成开发简介

本节描述用于开发集成企业中的信息的应用程序的概念和过程。

信息集成解决方案概述

企业要面对许多信息集成挑战。迅速改变的经济环境刺激了对信息的改进存取、灵活分析能力和正规信息库存的需要。

信息集成简介

在三十多年的时间里，IBM® 已经提供了世界级的数据管理技术。IBM 一直通过为大型、中型和小型公司开发信息集成来增强其企业产品的功能。信息集成在现有数据管理解决方案的坚固基础上构建。信息集成提供端到端解决方案，以便以透明方式管理现在市场中的大量且多样的数据。

开展业务的成本包括集成不同的且未连接的基础结构的需要。业务需要以下目标:

- 与新业务无缝集成并将商业应用程序与旧系统链接在一起
- 控制因为管理不同系统和在不同种类的自动化包之间集成而增加的成本
- 缓和人员和技能的短缺与快速占领新市场的矛盾

有效访问和管理信息的解决方案的需求跨产品和行业。

相关概念:

- 第 4 页的『DB2 Information Integrator – 集成解决方案』
- 第 1 页的『什么是信息集成？』
- 第 2 页的『为什么信息集成对您的企业很重要？』

什么是信息集成?

信息集成是将数据库管理系统、Web 服务、复制、联合系统和存储功能组合到一个公共平台中的技术的集合。它还包括各种编程接口和数据模型。使用信息集成技术，就可以存取不同类型的数据（结构化、非结构化和半结构化）。可将该数据变换为提供对企业中的信息方便存取的格式。

信息集成允许数据和内容源与下列功能集成:

- 提供实时读写存取
- 为业务分析和数据交换而变换数据
- 管理数据放置，并考虑到性能、流通性和可用性

相关概念:

- 第 4 页的『DB2 Information Integrator – 集成解决方案』

为什么信息集成对您的企业很重要?

IBM® 信息集成策略:

- 向用户提供处理旧数据的能力。
- 向用户提供利用熟悉的软件使用已知资产和资源的能力。
- 向用户提供获取并且简便维护新数据的能力。
- 向用户提供使用现有数据管理工具存取数据（不论数据位于何处）的能力。

IBM 已标识五类基于开放式服务基础结构的集成。可以同时使用或分别使用这些集成类型来解决业务问题。此处列示的五种集成类型表示当前面向业务的各种集成问题。信息集成是这些集成类型的核心。

用户交互作用

用户可使用单个定制用户界面，该用户界面可通过任何设备以虚拟方式获取，并且它具有完全的事务支持。用户交互作用结果集成到多个业务系统中。

进程集成

企业可通过跨人员与不同种类的系统（在企业内部和外部）对进程进行建模、使进程自动化以及监视进程来更改其运营方式。

应用程序连接

应用程序可互相连接以便共享和使用信息以便在企业级别更好地使用这些信息。

构建以进行集成

用户可通过使用 Web 服务和现有资产来构建和部署准备集成的应用程序。可以将新解决方案与现有业务资产集成在一起。

信息集成

可将不同格式的业务信息集成到企业中。集成对信息资产的统一视图启用相关搜索、存取、复制、变换和分析以满足业务需要。

IBM DB2® Information Integrator 是一种技术，它提供了综合解决方案以解决客户对信息集成的需求。DB2 Information Integrator 功能包括信息存取、信息集成和信息分析。

大部分现代集成需求是应用程序和信息集成的混合体。解决方案提供者有多种方法来集成、共享和分发信息。选择不同的方法以适应不同的业务集成需求比明确界定不同集成类型更为重要。

相关概念:

- 第 1 页的『什么是信息集成？』
- 第 3 页的『DB2 Information Integrator 解决什么问题？』

为什么信息集成会使应用程序开发更加容易

今天的企业系统包括客户、供应商、伙伴和电子市场。这些系统与数据库、应用程序服务器、内容管理系统、数据仓库、工作流系统、搜索引擎、消息队列、Web 搜寻器、挖掘和分析程序包和其它企业应用程序交互作用。企业系统需要大量的编程接口，如开放式数据库连接、Java 数据库连接、Web 服务、Java 对象和 Java 2 Platform Enterprise Edition。企业系统需要使用大量的数据模型和语言，例如“结构化查询语言”、“可扩展标记语言”、“Web 服务描述语言”和“简单对象访问协议”。

信息集成是将数据管理系统、数据仓库、Web 服务和其它企业应用程序的核心元素组合到公共平台中的技术方法。

DB2 Information Integrator 解决什么问题?

IBM® DB2® Information Integrator 为许多企业问题提供解决方案。DB2 Information Integrator 帮助您执行下列操作:

- 管理所有格式的信息。可通过使用存储、合并、归档、变换、装入、监视存取、复制安全性和随 DB2 Information Integrator 提供的清理技术来完成此任务。
- 让使用 XML、SQL、Web 服务和消息传递应用程序的本地和联合数据的查询模型保持统一。
- 使用挖掘、分类、总结和实时决策有效地分析信息。

下表概述一些典型的客户问题。在所有这些示例中，DB2 Information Integrator 及相关技术为解决方案提供软件和框架。

表 1. 典型客户问题

典型客户问题	解决方案	结果或添加的值	技术需求
提高电话销售的效率。	记录销售对话并存储为文本格式。将数据、基于销售的文本挖掘技术、事务数据和销售对话数据组合起来。	改进的销售产品和检查不可见的模式的策略让结构化数据和非结构化数据保持隔离。	结构化数据和非结构化数据的集成；需要基于集成数据的挖掘技术。
集成不同的信息组以改进业务效率。这可能包括本地应用程序、封装好的应用程序或通过合并或采集获取的应用程序。	使用“可扩展标记语言”（XML）来集成和交换企业中的信息。围绕 XML 构建企业模型以便供较新的应用程序使用。	从现有信息抽取有竞争力的值。通过可操作效率增加利润率。缩短业务周期和决策过程。	集成结构化数据和非结构化数据，包括应用程序数据。应用程序和进程集成（消息传递和工作流）。文本数据的挖掘（分类和搜索）。
提高当前客户的利润率。	分析客户帐户和行为。通过复制将业务数据合并到仓库中。使用 Intelligent Miner™ for Data 来挖掘信息和 MQ 消息传递以将信息传送给业务用户。	目标产品可以客户概要文件为基础；提高每个客户的利润。	相关信息的复制和挖掘，可靠的分配机制。
科技用户需要分布式 Oracle 源和外部非关系型源中存储的化学和生物信息的集成视图。	DiscoveryLink 专业的跨多个不同种类的数据源的数据源支持。科学挖掘算法。	对信息的实时存取。	复杂查询优化；挖掘相关信息。
改进订购过程以满足不停变化的客户需要。与现有操作系统集成。	创建新的部件订购体系结构。在数据库中输入订单。使用复制和联合系统将订单传送至较旧的系统。	改进的订购响应时间（减少写帐单周期，改进现金流）。	复制、联合系统和 WebSphere®。

表 1. 典型客户问题 (续)

典型客户问题	解决方案	结果或添加的值	技术需求
提供跨公司的几个部门的改进客户服务集成信息以提供无缝的客户关系信息。	集成跨不同业务单元的客户机信息以个人或组的形式提供客户的集成视图。	响应客户电子邮件（e-mail）的对客户的个人建议。提供跟踪整个企业的忠实点。	存储、联合系统、复制和 WebSphere。

相关概念:

- 第 12 页的『DB2 通用数据库系列 - 信息集成的基础』
- 第 4 页的『DB2 Information Integrator - 集成解决方案』
- 第 1 页的『信息集成简介』
- 第 5 页的『规划信息集成体系结构』

信息集成的基础

信息集成的基础是对不同数据源（如关系文件、“可扩展标记语言”（XML）或平面文件）建模的能力。使用信息集成的技术，可通过这些已建模的数据源获取变换功能。然后，可以公司觉得有用的方式（如使用在应用程序中嵌入的 SQL 语句或 Web 服务应用程序）表示合并视图。规划严谨的数据库体系结构在信息集成技术中是非常有用的，也是必需的，原因如下:

- “数据库管理系统”（DBMS）已证明它们能管理传统商业应用程序中激增的信息量。数据库管理员应了解有关存储、检索、变换、可伸缩性、可靠性和可用性的问题。
- 数据库行业正学习适应数据的多样性和电子商务应用程序中的存取模式。数据库行业中的应用程序和功能使用内置对象关系支持和“可扩展标记语言”（XML）功能。这些功能支持对外部数据源的联合存取。
- 数据库技术上的投入（包括数据库、支持工具、应用程序开发环境和在数据库技术方面熟练的人员）一直在增长。

相关概念:

- 第 12 页的『DB2 通用数据库系列 - 信息集成的基础』
- 第 4 页的『DB2 Information Integrator - 集成解决方案』
- 第 3 页的『DB2 Information Integrator 解决什么问题？』

DB2 Information Integrator - 集成解决方案

企业的信息集成解决方案是 IBM® DB2® Information Integrator。DB2 Information Integrator 是数据和支持该数据的进程的完整集成的框架。DB2 Information Integrator 技术可解决各种各样的复杂问题。企业的最佳解决方案可能是个别 Information Integrator 技术和姊妹产品的组合。如果有多组解决方案可供使用，能够很轻松地从一种技术转移至另一种技术就更为重要。DB2 Information Integrator 使用无缝地在一起工作的技术来交付完整的数据集成。

相关概念:

- 第 1 页的『信息集成简介』
- 第 3 页的『DB2 Information Integrator 解决什么问题？』

规划信息集成体系结构

在规划集成环境的体系结果时，可以使用 Web 服务工具、数据库管理工具和仓库工具。可以使用集成的语言，即 XML。可以使用数据库技术的语言，即 SQL。

“可扩展标记语言”（XML）是一种可用于 Web 应用程序的技术，它将数据与其元数据封装在一起。XML 在集成信息的策略中是关键元素。XML 提供了一种公共语言，它使企业间通信成为可能。它提供了通过 Web 发送半结构化数据的简便方法，使转换中不会丢失任何数据。它掩盖了端点基础结构中的差别。XML 能够为各种设备适当地呈示信息，原因是它将文档内容与文档演示分开。XML 能根据内容或决策支持灵活处理，原因是它包括数据描述以及数据如何与其它数据块相关的描述。XML 可以帮助您的公司产生更精确的搜索结果，因为它为搜索自变量提供了上下文。

SQL 是一个 ANSI（美国国家标准学会）标准，用于存取关系数据库管理系统。SQL 提供了对关系数据、联合数据和非关系数据的存取。通过在 Web 服务中使用基于 SQL 的查询，可以将 SQL 语句和存储过程调用发送至 DB2® 通用数据库。然后，Web 服务返回带有一些缺省标记的结果。当返回数据只使用 SQL 数据类型的简单映射（将列名用作元素）时，在 Web 上使用基于 SQL 的查询非常有用。当需要 SQL 数据至 XML 元素和属性的用户定义的映射时，使用 DB2 XML Extender 和基于 SQL 的查询。如果您正在使用 DB2 XML Extender 来在表的单个列中存储 XML 文档，则可以使用基于 SQL 的查询来将这些文档作为一个整体进行检索，如同检索字符大对象（CLOB）一样。同时，可以将 DB2 XML Extender 与基于 SQL 的查询配合使用，来调用抽取部分文档的用户定义的函数。

可以使用基于 SQL 的查询来调用 DB2 UDB 存储过程。存储过程通常要转换为 Web 服务，原因是存储过程本身就是编程逻辑和数据库存取的封装。将存储过程作为 Web 服务调用使动态地提供输入参数和检索结果成为可能。

信息集成的体系结构包括三层。

- 数据层

  数据层是信息集成的基础。此层提供了对不同格式的基本源中数据的存储、检索和变换。此层基于联合 DBMS 体系结构。数据是作为结构化关系表、半结构化 XML 文档或以非结构化格式存储的。混合 XML 和关系存储及检索基础结构确保了高性能和数据稳定性。此数据层使用具有灵活的包装器体系结构的联合数据库技术来集成外部数据源，这些数据源可以是传统数据服务器、企业应用程序或工作流。

- 服务层

  信息集成是关于存取数据和确定如何使用数据。此层提供了用于将数据存取服务透明地嵌入到企业应用程序和业务过程中的基础结构。这包括查询处理、文本搜索和挖掘、变换和复制。

- 应用程序层

  信息集成是关于可使用数据的编程接口。应用程序层为其它层提供了基于标准的编程模型和查询语言。可以使“接口”基于 Web 服务或传统应用程序编程接口。应用程序层增强了标准查询语言。

每一层都是一个独立的实体，但也必须依赖其它层，并与其它层配合工作来提供更完整的集成策略。第 6 页的图 1 显示了数据层、服务层和应用程序层。

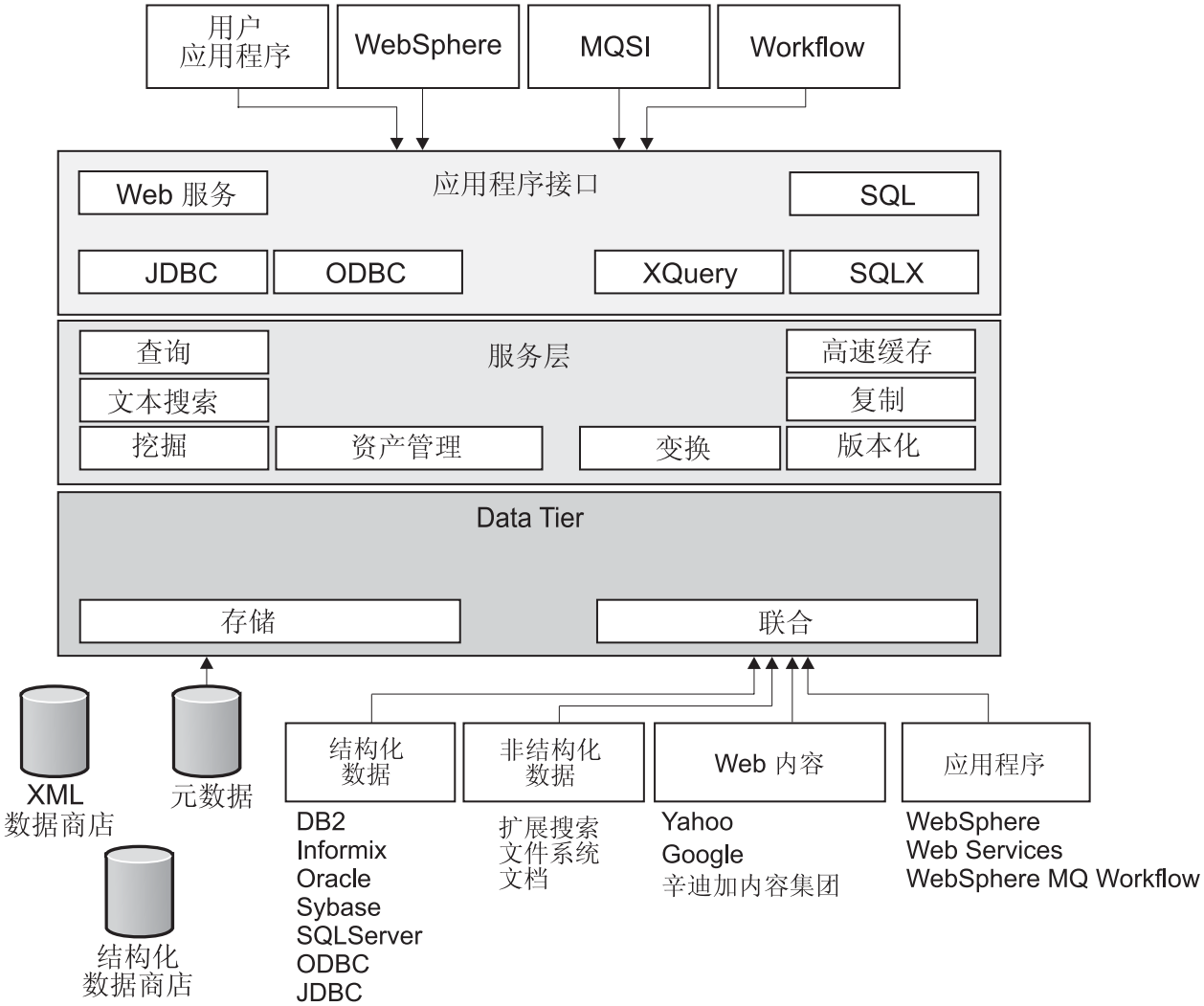


图 1. 集成层

相关概念:

- 第 4 页的『DB2 Information Integrator – 集成解决方案』
- 第 3 页的『DB2 Information Integrator 解决什么问题？』
- 第 2 页的『为什么信息集成对您的企业很重要？』

DB2 Information Integrator – 关系联合技术

联合数据库管理系统提供有关存取和处理不同数据的帮助。联合系统提供以各种格式存储或生成的分布式数据的单站点映像。联合系统提供公共接口以存取此数据。IBM® DB2® Information Integrator 的联合数据库管理系统提供扩展目录。此目录能够维护有关远程数据的统计信息并使用这些统计信息以全局方式优化数据存取。使用联合系统，可以收集有关远程数据源的统计信息并在 DB2 Universal Database™（DB2 通用数据库）提供的查询优化器技术中使用此信息。这允许您快速存取数据。联合系统使用 DB2 通用数据库的查询重写工具将执行速度较慢的查询重写为速度较快的等效查询格式。

在联合数据库引擎中，通过称为包装器的软件组件存取数据源。每个包装器包含有关数据源的信息，例如数据源数据类型与 DB2 通用数据库数据类型之间的缺省映射。要实现包装器，服务器使用存储在称为包装器模块的库中的例程。这些例程允许服务器执行一些操作，如连接至数据源并以迭代方式从中检索数据。要使用包装器查询、检索和处理数据，必须先安装这些包装器。然后，必须注册每个包装器以将其添加至联合系统。有关联合系统技术的更多信息（包括联合系统对象的定义），请参阅《*联合系统指南*》。

DB2 Information Integrator 联合技术为多个数据源提供虚拟数据库。这些数据源可执行下列操作:

- 在不同硬件和不同操作系统平台上运行
- 由不同供应商提供
- 使用不同应用程序编程接口和不同 SQL 方言

联合系统数据库使程序员能够定制其数据库管理系统以存取选择的数据源，不管该数据源是关系型还是非关系型。

关系型联合包装器可查询如第 7 页的 列示的其它关系数据库管理系统（RDBMS）中的数据。关系型和非关系型包装器将这些源映射至 DB2 UDB 以提供对这些其它数据库系统的透明存取。可使用单个查询存取这些数据源，并使用由联合系统和 DB2 UDB 提供的性能技术和查询重写功能。

在联合关系技术提供的关系型包装器中，有用于非 IBM 关系数据库和 Informix® 数据库的包装器。如果您想要存取存储在下列其中一些数据源中的数据，则需要关系型包装器:

- Oracle
- Sybase
- Microsoft® SQL Server
- IBM DB2 系列产品
- Teradata
- “开放式数据库连接”（ODBC）源

相关概念:

- 『调整查询处理』（《*联合系统指南*》）

相关任务:

- 『全局优化』（《*联合系统指南*》）

DB2 Information Integrator – 非关系联合技术

非关系联合技术由提供对非关系数据的存取的包装器组成。通过这些包装器，可以存取由“Web 服务描述语言”文件描述的表结构文件、Excel 文件、“可扩展标记语言”（XML）文档、BLAST 搜索算法、 Documentum 数据、Entrez 源、HMMER 源、BioRS、“扩展搜索”源、商业应用程序和 Web 服务提供程序。通过扩展搜索源，就可以存取各种非结构化数据源。其中一些源包括 Domino™、Microsoft® Exchange、Microsoft Index Server 和“轻量级目录访问协议”（LDAP）目录以及其它。有关联合系统的更多信息，请参阅《*联合系统指南*》。有关包装器以及如何创建包装器的更多信息，请参阅《*DB2 Information Integrator 包装器开发者指南*》。

包装器的非关系型集合包含一些可以安装的组件:

科学数据源

它们可能包括一些非结构化数据。其中一些数据可能包含为生命科学产业开发的基因组、蛋白质组、生物信息技术和化学方面的信息。这些包装器使联合系统能够从分布式源中集成基因组、化学、生物和其它研究数据。例如，可使用一个 SQL 语句来集成瑞士的数据库中的蛋白质序列数据、日本的数据库中的化学结构数据以及存储在您的局域网上的表结构平面文件中的频谱。这些数据出现时就好象它们在一个虚拟数据库中一样。

结构化文件数据源

它们包含使用定义的可重复结构存储在文件中的数据。例如，这可能是一个 Excel 电子表格或其中每个记录包含用定界符隔开的相同数目的字段的平面文件。

应用程序数据源

应用程序包装器使用应用程序来存取底层数据。原始数据可以是一些标准和非标准格式。

Web 服务提供程序源

Web 服务包装器的作用是使 DB2® UDB 和 SQL 用户访问由 “Web 服务描述语言”（WSDL）文件描述的 Web 服务提供程序。通过在 Web 服务提供程序和使用程序之间交换 SOAP 消息来调用 Web 服务。Web 服务包装器使用 Web 服务就如同 SOAP 用户定义的函数使用 Web 服务一样。可以发出具有联合昵称和视图的 SQL 语句来访问 Web 服务，也可以发出一组用户定义的函数来访问 Web 服务。可以通过指定 WSDL 来发现 Web 服务，然后可以根据 WSDL 中的信息创建必需的昵称。您可查询昵称以访问 Web 服务以便修改或存取信息。

需要非关系型包装器的数据源示例包括下列其中一些数据源。此列表仅为示例列表。有关完整的包装器信息，请参阅*《DB2 Information Integrator 数据源配置指南》*。

- BLAST
- Excel
- 表结构文件
- Documentum
- HMMER
- Entrez
- 扩展搜索
- BioRS
- Web 服务
- 可通过 WebSphere® Business Integration 访问的商业应用程序，包括 SAP、PeopleSoft 和 Siebel

相关概念:

- 『非关系数据源的数据类型映射』（*《联合系统指南》*）
- 第 135 页的『Web 服务使用程序函数』
- 『Web 服务包装器和 Web 服务描述语言文档』（*《IBM DB2 Information Integrator 数据源配置指南》*）

WebSphere Portal 示例和 DB2 Information Integrator

您可以在 WebSphere® 门户网站环境中使用 DB2® Information Integrator，并将 WebSphere Application Server 作为运行时环境来帮助实现客户目标，而无需在支持 Web 的应用程序开发技能方面进行巨额投资。DB2 Information Integrator 使用 WebSphere Portal 和 WebSphere Application Server 使 Web 应用程序或组件能够通过单个 API（SQL）存取不同的数据源，因而大大降低了存取和合并不同数据所需的复杂应用程序逻辑量。还可以使用 DB2 Information Integrator 显示源的完整数据内容以及将某些配置写入数据源中。DB2 优化能力有助于确保从远程数据源中有效检索数据。

可以将 DB2 Information Integrator 与 WebSphere Portal 配合使用以提高开发效率，在其中一个业务状况示例中，使用门户网站将不同信息和服务传递到单个视图中的职员。让我们假设有一家名为 Cotton-wood Insurance Corporation 的保险公司，该公司需要一种将与索赔有关的各种信息传递到单个视图中的客户支持代表的方法。客户支持代表需要处理的所有数据分散在不同的关系型和非关系型数据商店中，其中包括:

- DB2 UDB z/OS™ 版数据库中的客户帐户信息
- Lotus® Domino™ 服务器中的理赔师评估表和日常业务备忘录
- 作为 XML 文件存储的修正报告
- 新的保险索赔，写入 WebSphere MQ 队列以启动新的索赔处理的

Cotton-wood 的开发者创建简单的门户网站，通过在单个基于 Web 的视图中提供所有相关索赔信息来帮助服务代表向其保险客户提供服务。在门户网站中，使用若干 portlet 向门户网站传递信息或服务。DB2 Information Integrator 帮助 Cotton-wood 的开发者通过统一视图存取所有数据源的保险公司数据。Cotton-wood 开发者使用标准 WebSphere portlet 和 DB2 Information Integrator 来最小化存取不同数据源所需的代码。开发者使用 SQL 访问 DB2 Information Integrator 以查询数据，允许 DB2 Information Integrator 处理复杂访问和连接逻辑，而不是编写代码以将每个本机 API 用于不同数据源格式。

Cotton-wood 开发者创建的一个 portlet 为客户服务代表提供包含所有未决索赔的单个视图。这要求存取和合并 DB2 UDB 中存储的客户帐户信息、作为 XML 文件存储的修正报告以及 Lotus Domino 服务器存储的理赔师评估表。如果没有 DB2 Information Integrator，开发者必须存取每个数据源，然后编写将所有数据合并在一起所需的 portlet 代码，并将数据显示给用户。如果具有 DB2 Information Integrator，开发小组只需开发一个 portlet 即可利用单个 SQL 语句针对 DB2 UDB、XML 和 Lotus Domino 数据运行 portlet。

当 portlet 运行时，SQL 查询从门户网站应用程序行进到 DB2 Information Integrator 联合服务器。联合服务器存取所有必需的数据，使不同的数据源看起来像单个资源一样。这使开发者无需编写代码即可管理所需的多个连接、查询和连接逻辑。从各种源收集数据之后，联合服务器将单个结果集返回到客户机，向客户服务代表提供从 DB2 UDB、XML 和 Lotus Domino 数据源集成的所有相关索赔信息。

此示例显示 DB2 Information Integrator 如何显著减少开发应用程序逻辑所需的定制代码、技能需求和时间量、存取和合并不同关系型及非关系型源的数据时需要这些应用程序逻辑。通过提供单个 API 以供开发小组使用，并利用 DB2 UDB 的数据合并和优化功能，DB2 Information Integrator 允许开发者更多关注应用程序的可用性而不是如何存取数据及如何将数据合并到一起。

有关 WebSphere Portal 和 DB2 Information Integrator 的更多信息，请参阅WebSphere Portal 和 DB2 Information Integrator 的样本代码。

相关概念:

- 第 153 页的『联合系统的优点』
- 第 7 页的『DB2 Information Integrator – 非关系联合技术』

相关任务:

- 第 169 页的『部署联合应用程序』

本指南从头到尾使用的方案简介

本节介绍本指南从头到尾讨论的方案。这些方案以实际客户经验为基础，但表示若干公司的组合。公司名是虚构的。

Cottonwood Distributors, Inc. – 仓库示例

Cottonwood Distributors, Incorporated（CDI）是一个现有的结构严谨的物流公司。CDI 充当商品部件的经纪人。该公司经商多年且已经是忠实的 DB2® 通用数据库客户。CDI 使用关系数据库（DB2）来存储他们的信息。此信息包括成百上千种零部件、几千个部件供应商以及想要订购这些部件的客户。他们的销售代表通过他们的客户机应用程序输入订单。销售代表通过电话来接订单，并为查找不同部件的客户提供报价。客户销售跟进由外部供应商进行（他们还管理他们的人力资源流程）。CDI 对他们的通信和报告使用“可扩展标记语言”（XML）。系统是由已调整好系统并让系统保持稳定多年的经验丰富的数据库管理员来管理的。

CDI 有一些关于数据大小和位置的复杂问题。当他们收购他们在业界的两家竞争者 MyDogwood, Incorporated 和 MyOak, Incorporated 时，这些问题变得更加复杂。这两个新公司有着相同的运作方式，而且数据也和母公司一样多，但这些数据存在于不同数据库中。MyDogwood, Incorporated 的数据在 Oracle 数据库中，而 MyOak, Incorporated 的数据在 Informix® 数据库中。

在合并之后，CDI 具有各种格式的不同种类的数据源。这些数据源所在的系统已经装入且进行了配置以成功运行，但这些系统有大量不可信用户。这些进程必须处理数以百万计的部件和数以千计的供应商。所以 CDI 要保持竞争力，他们必须将通过电话联络的销售代表替换为数以千计在线请求报价和下订单的 Web 用户。此外，供应商想要能够在因特网上对部件提交报价。所以，CDI 引入了基于 Web 的经纪交易。客户在因特网上访问部件信息和订购部件。供应商将访问因特网并提供对部件的报价。销售随访和人力资源流程仍然是外部功能。CDI 现在必须处理因为多个数据库间的数据易使用性、数据流通性以及实时更新的管理问题而增加的复杂问题。

相关参考:

- 第 201 页的附录 A, 『Cottonwood Distributors, Inc. 和 YBar, Inc. 方案的脚本示例』

发现数据 – 职员技能方案

一家名为 YBar, Incorporated 的专业人力资源公司，负责找出技能和公司需求然后查找简历并分配新的工作。该公司有一个职员数据库，包含有关人员及其工作技能的信息。他们必须使用由数据库外部的无关应用程序管理的平面文件。此应用程序管理工作概要文件。此外，他们必须处理公司外的解决方案供应商所拥有的应用程序。

他们有两个主要任务。

1. 他们必须找出适合工作机会的最佳候选人。

   大部分是内部工作机会，并且他们的搜索以个人简历上的技能为基础。他们有一个 Web 应用程序，该应用程序从候选人处收集使用“可扩展标记语言”（XML）格式的简历。“人力资源”（HR）系统记录所有职员的当前工作概要文件。
2. 他们必须将职员（包括其当前工作描述）列表发送至外部教育提供商以便它们能对合适的人员提供定制类。

   它们将使用 WebSphere® MQ 在它们的应用程序中发送消息。然后，它们可使用联合系统来连接职员数据库和工作数据库，并通过 WebSphere MQ 发布该列表。

YBar, Incorporated 有一个称为 *Employee* 的表，它包含有关该公司中各个职员的信息。它还有一个称为 *Job* 的表，描述各种工作。

表 2. YBar, Inc. 的 Employee 和 Job 表和列

EMPLOYEE	JOB
Emp_ID	Job_ID
Lastname	Job_Description
Firstname	Title
Dept_ID	Responsibilities
Current_Job_ID	

相关概念:

• 第 160 页的『职员数据库方案 – 解决方案设计』

相关参考:

• 第 201 页的附录 A, 『Cottonwood Distributors, Inc. 和 YBar, Inc. 方案的脚本示例』

信息集成组件

信息集成使用已证实的数据管理技术（DB2 通用数据库系列）作为 IBM DB2 Information Integrator 基础。DB2 通用数据库在单个系统内及群集系统上使用并行性，可以支持数千并发用户和太字节。其对象关系基础结构提供了用于使用定制函数添加新数据类型的可扩展框架。信息集成基础结构提供使用消息传递和工作流工具（如 IBM WebSphere 提供的那些工具）的客户机应用程序编程接口（API）。数据库事件（如收到新的信息段）可透明地创建通知（如将新消息放置在队列上）。 WebSphere Studio 为基于 Java 的数据库和联合数据库应用程序提供了开放式集成开发环境。它提供图形功能以执行各种功能，如下列任务:

- 创建 SQL 请求
- 了解连接路径
- 开发存储过程
- 使用用户定义的函数扩展数据库
- 将数据库请求转换为 Web 服务
- 开发“可扩展标记语言”（XML）模式和文档类型定义（DTD）

DB2 通用数据库系列 - 信息集成的基础

DB2® 通用数据库是 IBM® DB2 Information Integrator 的基础技术。DB2 Universal Database™（DB2 通用数据库）可以管理各种类型的信息，无论是存储在 DB2 通用数据库、Oracle、Sybase 中还是存储在其它数据库中。

DB2 Extenders™ 可以管理图像、视频、音频或声音记录、可扩展标记语言（XML）文档、复杂文本文档和空间对象等等。DB2 通用数据库 Data Links Manager 可以管理外部文件系统中的数据。DB2 通用数据库可以处理引用完整性、访问控制、一致性和恢复。DB2 XML Extender、Net Search Extender 和 Spatial Extender 提供了特定于数据类型的扩展，可以查询、存取、更新和管理各种数据对象。例如，借助 DB2 Extenders，可以存取 XML 文档、根据图像形状或颜色进行查询或根据给定位置进行查询。

DB2 通用数据库及其关联组件提供存储和检索下列类型的数据的功能:

- 结构化关系表
- 半结构化 XML 文档
- 非结构化内容，如字节流和已扫描图像

有关 DB2 通用数据库的更多信息，请参阅 http://www.ibm.com/software/data/db2/。

DB2 XML Extender

DB2 XML Extender 充当 XML 文档及其文档类型定义（DTD）的资源库。它还提供数据管理功能，如数据完整性、安全性、可恢复性及可管理性。可将整个文档存储为 XML 用户定义的列或将该文档分解成多个表和列。XML 元素和属性可使用索引来确保搜索的速度较快。可检索整个文档或在 SQL 查询中动态抽取 XML 元素和属性。此外，XML Extender 提供存储过程以便从现有数据创建 XML 文档。

DB2 XML Extender 可以使用一些函数和存储过程，它们允许直接从 DB2 XML 应用程序存取 DB2 通用数据库消息排队函数。可以使用 WebSphere® MQ 工具来开发消息查询函数和存储过程。DB2 XML Extender 包含一些消息查询函数。有了这些函数，应用程序就可以执行下列操作:

- 使用 SQL 语句来按照应用程序消息传递接口的定义在服务点（队列）中以 XML 消息的形式发送、读取或接收文档。
- 从几个表组成一条 XML 消息并将其直接发送至消息队列，或将队列中的 XML 消息分解为关系表。

Net Search Extender

可将 IBM Net Search Extender 与 DB2 Information Integrator 的内置联合支持配合使用，以对存储在 DB2 通用数据库和 Informix® IDS 数据库中的文本数据建立索引和执行搜索。还可在联合源（如 Oracle、Sybase 和 Microsoft® SQL Server）中使用 IBM Net Search Extender。与数据库管理器优化器的智能策略的集成可确保高性能以及在 SQL 全选择中以无缝方式进行纯文本搜索。

Spatial Extender

通过 DB2 Spatial Extender，可以获取与以下对象相关的事实和数字: 这些对象可进行地理定义（如根据它们在地球上的位置或在地球上的某个地区内的位置）。这些事实和数字都是空间信息，而对象是地理特征。例如，可使用 DB2 Spatial Extender 来确定是

否有任何居民区与建议的垃圾填埋区相重叠。居民区和建议的填埋区都是特征。查找是否存在任何重叠就是空间信息的一个示例。如果存在重叠，则重叠范围也是空间信息的一个示例。

要生成空间信息，DB2 Spatial Extender 必须处理定义特征位置的数据。这种数据称为空间数据，由引用地图或类似的投影上的各个位置的坐标组成。例如，为确定一个特征是否与另一特征重叠，DB2 Spatial Extender 必须根据其中一个特征的坐标确定另一特征的坐标所在的位置。

可使用 DB2 Spatial Extender 在数据库和外部数据源之间交换空间数据。更确切地说，可通过将空间数据以文件（称为数据交换文件）形式传送至数据库来从外部源导入空间数据。还可将数据库中的空间数据导出至外部源可从中获取它的数据交换文件。

DB2 仓库管理器

数据仓库是从各种数据源获取的数据的集合。最终用户以他们可理解的方式获取该数据，然后在业务环境中使用该数据。可以通过使用仓库工具构建、管理和分析从不同种类的环境中抽取的数据来对与业务或组织结构有关的数据进行分组。

DB2 仓库管理器提供基于 SQL 的抽取、变换和装入功能以移动和变换数据。此外，DB2 仓库管理器中还包括了元数据管理解决方案信息目录中心。信息目录管理器还为第三方独立软件供应商提供了集成点以执行双向元数据和作业调度交换。DB2 仓库管理器包括分布式抽取、变换和装入作业调度系统。DB2 仓库管理器代理进程支持在源系统与目标系统之间直接移动数据，而不会增加中央服务器的成本。DB2 仓库管理器支持完全刷新、增量更新和数据移动选项，包括 IBM 的集成数据复制功能。

复制

有了复制技术，就可以在中央数据库和区域事务性数据库之间复制数据。复制使得业务数据可供区域数据库用于提示事务处理。在复制环境中，业务可以捕获源数据库中的数据更改并将这些更改传播至任何目标数据库，而不需要更改应用程序。复制提供使用标准 SQL（包括多表连接和存储过程）的数据变换。复制支持时间点和接近实时的复制（可嵌入变换），用于填充数据仓库和数据集市。

复制还支持重新构造 DB2 通用数据库事务并将这些事务作为 XML 中的消息发布的能力。复制还可以将事务作为消息发送到 WebSphere MQ 消息队列，然后可以由消息预订应用程序或 Q Apply 程序消耗。事件发布通过 Q Capture 程序（Q Replication 的另一个组件）的方式捕获数据。事件发布允许您将已落实的事务性数据或行级数据作为 WebSphere MQ 消息队列中的消息从 DB2 通用数据库表发布。这些消息可以由用户应用程序直接读取和解释，也可以首先由消息代理（例如 WebSphere Business Integration Message Broker 或 DB2 MQ 侦听器守护进程）解释。

相关概念:

- 『数据仓储提供了什么解决方案？』（*《数据仓库中心管理指南》*）
- 『Introduction to XML Extender』（*DB2 XML Extender Administration and Programming*）
- 『How XML data is handled in DB2』（*DB2 XML Extender Administration and Programming*）
- 『在数据仓库中心中复制』（*《数据仓库中心管理指南》*）

- 『Introduction to Q replication—Overview』（*IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*）
- 『Introduction to event publishing—Overview』（*IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*）
- 『The purpose of DB2 Spatial Extender』（*IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference*）
- 『How to use DB2 Spatial Extender』（*IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference*）
- 『How features, spatial information, spatial data, and geometries fit together』（*IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference*）

相关任务:

- 『Method for retrieving an XML document』（*DB2 XML Extender Administration and Programming*）
- 『对 SQL 复制作规划』（《*IBM DB2 Information Integrator SQL 复制指南与参考*》）

相关参考:

- 『Fullselect』（*SQL Reference, Volume 1*）

信息集成中的 Web 服务功能

Web 服务是集成的主要创新:

- Web 服务提升互操作性: Web 服务将服务提供程序与服务请求程序之间的交互作用设计为完全不依赖于平台并且不依赖于语言。
- Web 服务启用即时集成: 当服务请求程序使用服务代理程序来查找服务提供程序时，发现就动态发生。
- Web 服务通过封装来降低复杂性: 服务请求程序和提供程序本身关注的是彼此相互作用所必需的接口。因此，服务请求程序不知道服务提供程序如何实现其服务，服务提供程序不知道服务请求程序如何使用其服务。Web 服务将那些详细信息封装在请求程序和提供程序中。
- Web 服务技术允许您将较旧的应用程序强制转型为 Web 服务。这意味着可以有意义的新方法使用已存在于企业中的应用程序或程序包。另外，与较旧应用程序（如安全性、目录服务和事务）相关联的基础结构也可以包装为一组服务。

通过使用 "Web 服务描述语言"（WSDL），Web 服务在应用程序资源的提供程序和使用程序之间提供了简单的接口。

Web 服务提供程序

借助 "Web 服务描述语言"（WSDL）接口，Web 服务客户机应用程序可以获得对 DB2® 通用数据库的存取权。通过使用 "Web 服务对象运行时框架"（WORF）（也称为 "文档存取定义扩展"（DADX）文件），您可以创建用于存取 DB2 UDB 数据的 WSDL 接口。在定义通过 DADX 文件存取 DB2 UDB 数据的操作之后，您可以为受支持的 Java™ Web 应用程序服务器环境（Apache Jakarta Tomcat 或 IBM® WebSphere® Application Server）部署 DADX 文件及其运行时环境（Apache SOAP V2.3 或 Apache Axis V1.2）。在测试和部署 DB2 Web 服务之后，所有 Web 服务客户机都可以开始使用 DB2 Web 服务。

Web 服务使用程序 – 用户定义的函数

当 DB2 Universal Database™（DB2 通用数据库）成为使用程序时，Web 服务可以利用数据库内构建的优化。通过使用 SQL 语句，可使用和集成 Web 服务数据。通过使用 SQL 来存取 Web 服务数据，您可以减少某些应用程序的编程工作量，原因是您可以在 SQL 语句上下文中处理数据，然后才将该数据返回给客户机应用程序。您可以通过使用 WebSphere Studio V5 和更新版本中提供的工具，将现有的 WSDL 接口转换为 DB2 UDB 表或标量函数。在执行 SQL 语句期间，您与 Web 服务提供程序建立连接，然后会将响应文档接收为关系表或标量值。

Web 服务使用程序 – Web 服务包装器

在联合系统中提供了 Web 服务包装器，以允许用户使用有关调用 Web 服务的昵称和视图的 SQL 语句来访问 Web 服务。您可以使用 SELECT 语句创建 Web 服务包装器和昵称，这些昵称指定了 Web 服务的输入和访问 Web 服务的输出。

图 2 显示 DB2 通用数据库参与到 Web 服务环境中:

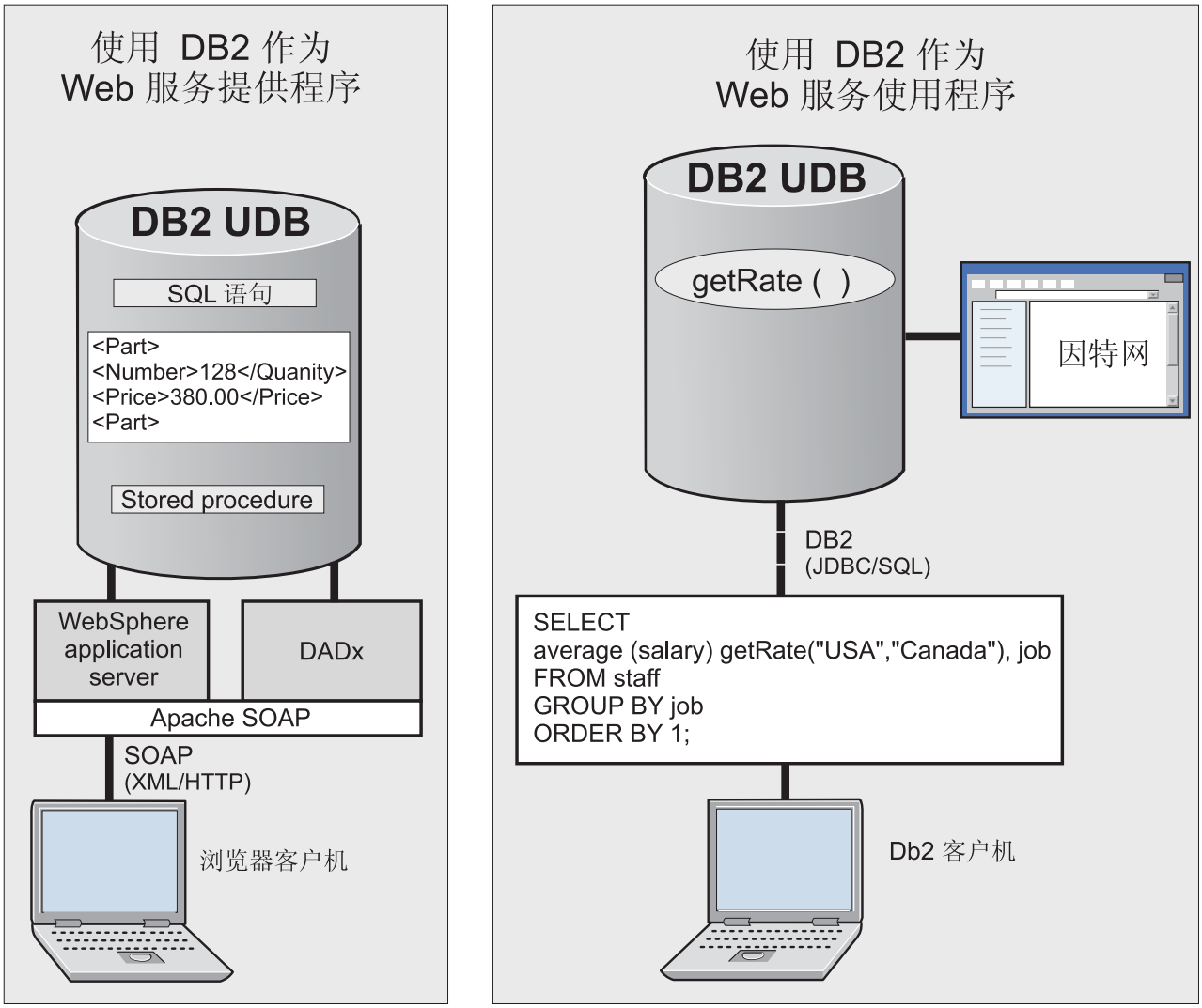


图 2. Web 服务提供程序和 SOAP 用户定义的函数

IBM DB2 Information Integrator 使用许多 Web 服务能力，如下列能力:

- 将存储过程功能显示为 Web 服务的能力。

  通过使用“Web 服务对象运行时框架”（WORF）和“文档存取定义扩展”（DADX），应用程序服务器可向客户机提供这些 Web 服务。应用程序服务器可能是 WebSphere Application Server 或 Apache Tomcat。
- DB2 UDB 执行的所有 SQL 语句的能力，包括要成为 SOAP 客户机和从 SOAP 服务器请求 Web 服务的存储过程。然后，Web 服务将数据显示为 SQL 值或与其它 SQL 数据组合在一起的表。

WebSphere Application Server 是动态电子商务的基础软件。WebSphere Studio 应用程序开发环境提供构建、部署和集成电子商务所需的工具。

WebSphere Application Server 是与 J2EE 兼容的应用程序服务器，提供用于开放式分布式计算的环境。WebSphere Application Server 提供客户机与资源管理系统（如数据库）之间的中间接地。它允许客户机（如 applet 或 C++ 客户机）与数据资源（如关系数据库或 WebSphere MQ）以及现有的应用程序相互作用。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 27 页的『Web 服务提供程序功能部件』
- 第 135 页的『Web 服务使用程序函数』
- 第 29 页的『Web 服务过程概述』
- 『Web 服务包装器和 Web 服务描述语言文档』（*《IBM DB2 Information Integrator 数据源配置指南》*）

相关任务:

- 『为 Web 服务数据源注册昵称』（*《IBM DB2 Information Integrator 数据源配置指南》*）

WebSphere MQ

IBM® WebSphere® MQ（以前称为 IBM MQSeries®）用于动态集成。它在所有主要联网系统间的各种平台上通过简单的一致编程接口或非倒转的适配器连接应用程序。WebSphere MQ 允许系统独立运行，但要保证信息的传递。它包括通过“安全套接字层”（SSL）进行的消息加密以获取额外安全性和增强性能。由于其可靠性和强大功能，所以现在可在所有行业中具有重要使命的高价值解决方案中使用 WebSphere MQ。

WebSphere MQ 提供对带有一些应用程序编程接口的应用程序的支持:

消息排队接口

消息排队是程序对程序通信的一个方法。“消息排队”允许程序发送和接收特定于应用程序的数据而不需要直接连接。程序通过将消息发送至指定队列或从指定队列检索消息来通信。程序不需要知道指定队列的位置。可以复制程序以获取可用性和性能。可以重新定位程序或队列。

应用程序消息传递接口

“MQSeries 应用程序消息传递接口”是一个简单的应用程序编程接口，它提供点到点消息传递以及发布和预订消息传递的支持。“应用程序消息传递接口”通过将功能从应用程序移至数据资源库来简化应用程序开发。“应用程序消息

传递接口”语法的三个重要部分是服务、策略和消息。服务定义将消息发送至何处。策略定义如何发送消息。消息是发送的内容。服务封装本地或远程队列。策略封装消息的选项，如*优先级*或*重试*。消息部分可能包含应用程序消息数据和属性（如格式或相关标识）。

Java™ 消息传递服务

“Java 消息传递服务”应用程序编程接口允许应用程序创建、发送、接收和读取消息。它们允许进行异步和可靠通信。

使用 WebSphere 消息传递工具来接收、处理和存储信息。然后使用 DB2® 仓库管理器来协调集合并处理消息数据。IBM DB2 Information Integrator 充当 MQSeries Integrator 处理的“可扩展标记语言”（XML）信息的源和目标。WebSphere 消息传递功能通过简化应用程序开发和测试并在所有平台间使用一致的应用程序编程接口来加速分布式应用程序的实现。

相关概念:

- 第 4 页的『DB2 Information Integrator – 集成解决方案』
- 第 186 页的『如何在 DB2 中使用 WebSphere MQ 函数』

相关任务:

- 第 177 页的『安装 DB2 WebSphere MQ 函数』

规划和测试应用程序

需要规划将使用的数据对象（例如，数据库、表和昵称）的类型以及如何更好地使用这些对象。然后需要确定如何部署这些对象。例如，可以为抽象模型定义视图。然后通过嵌套视图支持更复杂的模型。可以使用视图来掩盖使用大量数据源的多个表。最后，可在 Web 应用程序中处理视图，使它在因特网上可用。

安装规划

规划信息集成基础结构的安装涉及了解当前环境以及知道哪些组件能最好地完成业务解决方案。需要的大部分工具和环境在 Windows® 平台上至少需要 256 MB 的随机存取存储器（RAM）。还需要检查每个组件的软件和硬件配置需求，如磁盘空间、通信设置、先决条件和维护。有关安装联合系统的注意事项的更多信息，请参阅《*DB2 Information Integrator 安装指南*》。

相关概念:

- 第 4 页的『DB2 Information Integrator – 集成解决方案』
- 第 5 页的『规划信息集成体系结构』

相关参考:

- 『DB2 Information Integrator 安装工作表』（*IBM DB2 Information Integrator 安装指南 Linux、UNIX 和 Windows 版*）

配置应用程序和环境

配置数据库客户机和服务器，以便它们可以相互连接。可以将“传输控制协议／网际协议”（TCP/IP）网络用作通信网络。在 Windows® 环境中，可以将条目添加至每个系统上的 *services* 文件以指定服务名称和端口号。

在典型配置中，WebSphere® Message Queue 服务器和 DB2® 服务器驻留在同一机器上。使用 WebSphere servlet、Enterprise Java™ beans 或 Web 应用程序的 DB2 客户机可以是本地或远程的。

联合系统技术不要求在主管数据源的机器上安装任何软件。联合数据库使用数据源的普通客户机通过客户机服务器体系结构与数据源通信。这样，联合数据源对源来说就象是另一个应用程序。

通过安装 “DB2 通用数据库™” 引擎然后启用联合功能来创建联合系统。然后配置联合系统以与数据源对话。将新的数据源添加至联合系统有几个步骤。首先，必须为源安装包装器。然后，必须告知联合数据库在哪里查找此包装器。通过发出 CREATE WRAPPER 语句实现此目的。如果多个源具有相同类型，则它们只需要一个包装器。例如，即使联合系统包括五个 Oracle 数据库实例（可能在不同的机器上），也只需要一个 Oracle 包装器。发出一个 CREATE WRAPPER 语句。但是，必须使用 CREATE SERVER 语句对系统标识每个独立的源。如果有五个 Oracle 数据库实例，则发出五个 CREATE SERVER 语句。

确保在数据库上为服务器和联合系统（SVCENAME、FEDERATED）设置属性。从服务器连接至数据库。对于想要存取的每个数据源，创建必需的包装器对象、服务器对象和用户映射。

```
db2 update dbm cfg using svcename myID authentication server
db2 update dbm cfg using federated yes
db2 connect reset
db2stop
db2start
db2 connect to rdjdb user user1 using pass1word
db2 create wrapper net8 options (DB2_FENCED 'N')
db2 create server oracle8 type oracle version 8.1.5
    wrapper net8 authorization oracleuser1
    password oraclepwd
    options (node 'orafcle8.world', password 'Y', pushdown 'Y')
db2 create user mapping for user1
    server oracle8
    options (REMOTE_AUTHOID 'oracleuser1',
       REMOTE_PASSWORD 'oraclepwd')
...
```

图 3. 数据库连接和包装器设置的示例

相关概念:

- 『联合系统』（*《联合系统指南》*）
- 『如何与联合系统交互作用』（*《联合系统指南》*）

相关任务:

- 第 192 页的『配置和运行 MQListener』

相关参考:

- 『用于规划联合系统配置的核对表』（*《IBM DB2 Information Integrator 数据源配置指南》*）

性能和调整规划 - 联合系统中的具体查询表

具体查询表是根据查询结果定义的表。具体查询表机制允许管理员在一组基础表或昵称中定义数据的具体化视图。有关联合系统对象及其定义的更多信息，请参阅《*联合系统指南*》。对于某些查询类，数据库可以自动确定具体查询表是否能够应答查询，而不必存取基本表。

通过使用具体查询表，可以在透明地将更新路由至数据库时透明地将只读查询路由至数据高速缓存。复制根据用户指定的策略异步地将更改的数据传播至高速缓存。通过使用联合高速缓存和复制，用户可以更有效地请求复杂查询。

可以在电子商务环境中使用具体化视图（这些视图使用聚集表）以获得更佳性能。例如，在电子商务中，可以使用具体化视图在中间层服务器中高速缓存产品目录信息以改进浏览目录的性能，而不涉及总结表。DB2® 通用数据库允许您为用于定义远程表的昵称定义具体化视图来支持高速缓存。通过从远程表中拉出数据填充具体化视图并以本地方式存储数据，显著提高了性能。当将 REFRESH DEFERRED 参数与具体化视图包括在一起时，将获得最佳性能。具体查询表或具体化视图使经常引用的数据接近应用程序服务器。具体查询表或视图还提供了一种方法来保护经过良好调整的系统免受 Web 应用程序生成的流量的影响。

可以对昵称定义具体查询表。昵称是一个标识，用来引用位于想要存取的数据源中的对象。昵称标识对象，也称为数据源对象。通过使用昵称来引用具体查询表，本地 DB2 UDB 实例可以高速缓存远程数据。高速缓存能力使联合查询具有更好的性能，其原因在于查询以本地方式存取远程数据。如果远程表不可用，则 DB2 UDB 可以使用在远程表上定义的具体查询表（如果它满足路由条件的话）。此技术改进了可用性和性能。REFRESH IMMEDIATE 选项不适用于引用昵称的具体查询表。

通过使用具体查询表，可以避免对每个查询进行重复计算，例如由 SUM 表表示。假设您有一个称为 CUSTOMER_ORDER 的表，该表存储若干年的客户订单。该表包含超过一百万条记录，平均行宽度为 400 个字节。现在，假设您需要对 2001 年的订单运行多个查询，并且您只需要表中的三列。以下是从三个列中获取信息的典型 SQL 语句:

```
select SUM(AMOUNT), trans_dt
  from db2inst2.CUSTOMER_ORDER
  where trans_dt between '1/1/2001' and '12/31/2001'
  group by trans_dt
```

如果 CUSTOMER_ORDER 表有好的索引，则存取路径显示此语句的索引扫描。

然而，您可以创建具体查询表，它包含您需要的列和行，并且包含对总计的计算:

```
CREATE TABLE DB2INST2.SUMMARY_CUSTOMER_ORDER_2001
AS
(SELECT SUM(AMOUNT) AS
  TOTAL_SUM, TRANS_DT, STATUS
  FROM DB2INST2.CUSTOMER_ORDER
  WHERE TRANS_DT
  BETWEEN '1/1/2001' AND '12/31/2001'
  GROUP BY TRANS_DT, STATUS)
DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

如果使用子句 DATA INITIALLY DEFERRED，则数据不会作为 CREATE TABLE 语句的一部分插入到表中。发出 REFRESH TABLE 语句来填充表。表中的数据以发出 REFRESH TABLE 语句时的快照来反映查询结果。要填充所创建的具体查询表，发出以下语句:

```
REFRESH TABLE DB2INST2.SUMMARY_CUSTOMER_ORDER_2001;
```

对具体查询表运行的查询将会快得多。具体查询表的大小更小并且它的行比较短（只有 45 字节，与基本表中的 400 字节相比短得多）。

通过使用具体查询表，可以优化跨 LAN 中的若干服务器连接多个表的查询。借助具体查询表，可以在单个服务器上高速缓存结果集，并在任何服务器中的数据更改时更新该高速缓存。任何类型的联合关系数据都可以使用这些连接类型。这包括 Oracle 表、SQL Server、Sybase、消息队列、Web 服务和其它关系数据源。

请进行规划，以使用数据仓库来除去对操作数据或日常数据的直接存取。这会改进性能，其原因在于您正在为特别的查询存取仓库数据。通过将数据放在数据仓库中，可以创建信息数据的存储库，这些数据可以从操作数据中抽取并接着进行变换以用于决策。例如，创建以下表（连接客户表和帐户表）以存储坏帐的客户和帐户信息:

```
CREATE TABLE bad_account AS
     (SELECT customer_name, customer_id, a.balance
       FROM account a, customers c
       WHERE status IN ('delinquent', 'problematic', 'hot')
       AND a.customer_id = c.customer_id)
       DATA INITIALLY DEFERRED REFRESH DEFERRED
```

如果用户询问某个帐户是否存在拖欠，DB2 Universal Database™（DB2 通用数据库）优化器就会发现具体查询表已将请求的信息高速缓存。DB2 通用数据库不存取基本表 *account*，而是存取表 *bad_account*。这缩短了响应时间并且返回客户信息。

IBM® 联合系统包括基于成本的优化器。优化器不但考虑标准数据库统计信息（如基数和指数），而且考虑网络和服务器资源以及数据源引擎提供的查询能力。

相关概念:

- 『影响下推机会的昵称特征』（*《联合系统指南》*）

相关任务:

- 『创建具体查询表』（*《管理指南: 实现》*）

安全性和授权

在分布式计算系统（其中，用户、应用程序服务器和资源管理器常常分布在世界各地）中，保护计算系统资源是一项复杂的任务。好的安全服务有两个主要功能: 认证和授权。

认证在主体（用户或计算机进程）第一次尝试获取对计算资源的访问权时发生。此时，安全服务向主体提问以证明该主体是否有效。用户通常通过输入他们的用户标识和密码来证明他们的身份。进程通常提供加密密钥。如果密码或密钥有效，则安全服务给予用户一个标记或凭证。此标记标识主体并认证主体。在进行了认证之后，主体就尝试使用受安全服务保护的计算系统范围内的资源。但是，只有具有正确授权的主体才能使用特定计算资源。

授权在已认证主体请求使用资源时发生。安全服务确定用户是否可以使用该资源。通常将访问控制表（ACL）与资源关联以处理授权。资源定义哪些用户或进程（或者用户组或进程组）有权使用资源。如果安全服务授权主体，则主体就获得对资源的访问权。在分布式计算环境中，主体和资源必定互相怀疑对方的身份，直到每一方都向对方证明了身份为止。因为主体可能会伪造其身份以获取对资源的访问权，所以这是问

权。该资源必须从主体获取有价值的信息。为了解决此问题，安全服务包含充当可信第三方的安全性服务器。它对主体和资源进行认证，以便这些实体能够互相证明自己的身份。

已认证用户必须具有适当的特权（例如，SELECT、INSERT、UPDATE 或 DELETE）。特权必须对联合数据库中的昵称和远程数据源中的基础表或其它对象都起作用。这是联合数据库接受请求的方式。具有对本地 DB2® 通用数据库的特权是不够的。有关权限和特权的更多信息，请参阅*《联合系统指南》*。

相关概念:

- 第 153 页的『在 IBM DB2 Information Integrator 中设计查询的优点』
- 第 24 页的『DADX Web 服务中的安全性』

第 2 章 开发 Web 服务

本节说明 Web 服务的开发和使用。IBM DB2 Information Integrator 包含 Web 服务提供程序（WORF）和两类 Web 服务使用程序（SOAP 用户定义的函数和 Web 服务器包装器）。

Web 服务提供程序简介

Web 服务是一些业务函数的集合，应用程序或其它 Web 服务可使用 Web 服务客户机接口通过因特网对这些业务函数进行程序化调用。在 IBM DB2 Information Integrator 中，可以使用标准 SQL 语句和 DB2 XML Extender 存储过程定义基本 Web 服务。对于涉及 XML 和关系数据之间的高级变换的 Web 服务，请使用 DB2 XML Extender。

关于使用 DB2 作为 Web 服务提供程序的简介 - WORF

可以使用 Web 服务来提供对 DB2® 通用数据库信息的远程访问。Web 服务包含一组应用程序函数，这些函数（例如信息函数或事务函数）代表使用程序或请求程序执行某些有用的服务。Web 服务执行的函数多种多样，可以是简单的请求，也可以是复杂的业务进程。使用程序通常只需要知道 Web 服务的 Web 服务描述语言接口。此外，除非在接口内进行更改，否则对 Web 服务的更改通常不会影响使用程序。

Web 服务提升互操作性。互操作性的事实假定信息技术行业使用可对开发和集成 Web 服务提供指导的一组标准。Web 服务互操作性组织是一个开放式行业组织，其目的是跨平台、应用程序和编程语言促进并确保 Web 服务互操作性。“Web 服务互操作性组织”使用由万维网联盟（W3C）和 UDDI.org 制定的规范。Web 服务互操作性意味着您可以在各种 SOAP 或 Web 服务平台（包括 Apache SOAP、Apache Axis 或 Microsoft® Visual Studio.Net）上创建 Web 服务。

Web 服务应用程序员设计服务提供程序与使用程序（或服务请求程序）之间的交互作用，并且该交互作用完全独立于平台和语言。可使用即时集成，原因是服务请求程序可动态地查找服务提供程序。Web 服务通过封装减少复杂性。服务请求程序和提供程序仅关心彼此交互作用所必需的接口，而不是它们的底层实现。Web 服务为旧应用程序注入了新的生命力，原因是可以将现有的应用程序强制转型为 Web 服务。Web 服务的基本元素包括：“简单对象访问协议”（SOAP）、“统一描述、发现和集成”（UDDI）以及“Web 服务描述语言”（WSDL）。

Web 服务允许您从各种数据库和因特网位置存取数据。在存取数据之后，Web 服务使用程序可以搜索、挖掘和变换数据以便将其与数据仓库配合使用，以供进一步分析。

可以使用简单的“文档存取定义扩展”（DADX）文件来定义 Web 服务以存取数据库中的数据。可以使用简单的文本编辑器或使用 WebSphere® Studio Application Developer 和可从 WebSphere Studio 使用的向导来创建此 DADX 文件（该文件是 XML 文件）。

DADX 文件驱动 Web 服务运行时环境，该环境包括各种数据库管理工具和“Web 对象运行时框架”（WORF）。WORF 运行时环境提供了 XML 模式至 SQL 数据类型的简单映射。DADX 文件可以包含标准的 SQL 语句（例如，SELECT、INSERT、UPDATE、DELETE 和 CALL 语句）来查询和更新数据库并调用

存储过程。如果希望在运行时处理 SQL 语句，则必须使用仅包含 <DQS/> 标记的 DADX 文件来启用由 WORF 提供的动态查询服务（DQS）。

如果不使用 WORF 运行时环境，则需要编写自己的程序以处理创建 Web 服务的细节问题，例如开发自己的 WSDL。WORF 提供的一些功能包括:

- 分析 Web 服务请求
- 连接至数据库
- 执行 SQL 请求
- 对 SQL 结果中的输出消息进行编码
- 将消息返回给客户机

DADX 文件还可以包含 DB2 XML Extender 元素，例如“文档存取定义”（DAD）文件引用、用于生成和存储 XML 文档的 XML 集合操作、或用户定义的类型（UDT）以及用户定义的函数（UDF）。DAD 文件定义 XML 和关系数据之间的映射。DB2 XML Extender 允许完整地存储 XML 文档，并可选择在副表中为它建立索引。DB2 XML Extender 通过使用 XML 列存取方法来实现此操作，或将文档作为关系表的集合通过使用 XML 集合存取方法来实现此操作。

WORF 可与 IBM® DB2 Information Integrator 以及 DB2 Universal Database™（DB2 通用数据库）版本 8 和 WebSphere Studio V5 结合使用。WORF 还可用于 Informix®。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 27 页的『DADX 文件的定义』
- 第 29 页的『Web 服务过程概述』

DADX Web 服务中的安全性

可以通过使用应用程序服务器的安全性机制来保护 Web 服务。此处讨论的机制是认证、加密和保护数据库用户标识。

认证

可以通过启用认证来保护 DADX Web 服务端点。启用认证时，您确保只有经过认证的人员才能调用 Web 服务。URL 指定 Java™ 2 Enterprise Edition（J2EE）中的访问控制。每个 DADX Web 服务将 URL 用于 Web 服务的不同部分，例如，端点、生成实际 WSDL 期间和测试页中。对于 DADX 使用的每个 URL，可以指定允许哪些角色（并且，作为该定义的一部分，可以指定允许哪些用户）访问该 URL。为 Web 服务的认证用户设置权限的过程类似于授予 DB2® 通用数据库表 SELECT 特权。

以下列表显示 URL 和 URL 模式的示例，这些 URL 和 URL 模式可用于保护 Web 服务、特定操作或 WORF 提供的服务:

- 使用以下 URL 来启用对 DADX 的所有 URL、测试页和 WSDL 生成的访问控制:

  ```
  http://hostname:port/myContext/myGroup/myDadx.dadx/*
  ```

- 使用以下 URL 来只对测试页启用访问控制:

  ```
  http://hostname:port/myContext/myGroup/myDadx.dadx/TEST
  ```

- 使用以下 URL 来对某个组中的所有 DADX Web 服务启用访问控制:

```
http://hostname:port/myContext/myGroup/*
```

可以在 WebSphere® Application Server V5 的“应用程序服务器工具箱”或“应用程序汇编工具”中定义安全性约束。还可以在 WebSphere Studio Application Developer V5 的 Web 透视图中定义安全性约束。定义的约束可以包括角色名称，以便具有特定角色名称的任何人员均可以访问 Web 区域。

加密

可以通过 HTTPS 对消息进行加密来保护 DADX Web 服务。加密确保任何人都无法读取在 Web 服务客户机和 Web 服务提供程序之间交换的消息。请参阅作为应用程序服务器一部分的文档以确定如何启用 HTTPS。

数据库安全性

在 WebSphere Application Server V5 中，可以指定 JNDI 数据源在应用程序服务器上的数据库用户标识。在 group.properties 文件中，可以引用 JNDI 数据源，以便 DB2 Web 服务提供程序使用在 WebSphere 中指定的用户标识。数据库用户密码是在应用程序服务器上加密的。

有关认证、加密和数据源认证的更多信息，请参阅与特定应用程序服务器相关的信息。另请参阅以下 WebSphere 文档以了解有关安全性的特定信息:

- *IBM WebSphere Application Server V5: Security*
- *IBM WebSphere V5.0 Security WebSphere Handbook Series*

相关概念:

- 『WebSphere Studio』（*Application Development Guide: Programming Client Applications*）

相关任务:

- 第 127 页的『准备和创建 Web 归档文件』
- 第 59 页的『定制 group.properties 文件』

相关参考:

- 第 247 页的附录 E, 『Web 服务命令参考』
- 第 245 页的附录 D, 『Web 服务编码算法』

将 Web 服务提供程序与 iSeries 配合使用

Web 服务是一些业务函数的集合，应用程序或其它 Web 服务可通过因特网进行程序化调用。Web 服务的一个用途是提供对 DB2® 通用数据库信息的远程访问。Web 服务包含一组应用程序函数，这些函数代表请求程序（如信息函数或事务函数）执行某些有用的服务。Web 服务执行的函数多种多样，可以是简单的请求，也可以是复杂的业务进程。请求程序通常只需要知道 Web 服务的应用程序编程接口（API）。此外，通常对 Web 服务的更改不会影响请求程序。

Web 服务提升互操作性。Web 服务设计服务提供程序与服务请求程序之间的交互作用，并且该交互作用完全独立于平台和语言。可使用即时集成，原因是服务请求程序可动态地查找服务提供程序。Web 服务通过封装减少复杂性。服务请求程序和提供程序仅关心彼此交互所需的接口，而不是它们的底层实现。Web 服务为旧应用程序注入了新的生

命力，原因是可以将现有的应用程序强制转型为 Web 服务。Web 服务的基本元素包括: “简单对象访问协议”（SOAP）、“统一描述、发现和集成”（UDDI）以及“Web 服务描述语言”（WSDL）。

Web 服务允许您从各种数据库和因特网位置收集数据。在搜集数据之后，Web 服务使用程序可以搜索和挖掘数据，并使用存储的数据子集来变换这些数据以进行进一步分析。

可以定义实现标准 SQL 语句（例如，SELECT、INSERT、UPDATE、DELETE 和 CALL 语句）的 Web 服务。还可以通过使用 DB2 XML Extender 存储过程定义这些 Web 服务。

DB2 XML Extender 使用称为“文档存取定义”（DAD）的 XML 文档格式来定义 XML 与关系数据之间的映射。“文档存取定义扩展”（DADX）文件指定 Web 服务。这是通过使用一组操作来完成的，这些操作是由一些 SQL 语句、一系列参数以及一些 DAD 文件引用来定义的。这些操作与可以调用的编程方法相似。可以使用 XML 集合操作来生成和存储 XML 文档。可以使用 SQL 操作来查询和更新数据库并调用存储过程。

可以使用可用的数据管理工具和“Web 对象运行时框架”（WORF）从 Web 服务存取 DB2 Universal Database™（DB2 通用数据库）存储过程和数据。除了对 XML 数据指定存储器和检索操作外，这些工具还允许您将存储过程和 SQL 语句作为 Web 服务操作来调用。WORF 提供了 XML 模式至 SQL 数据类型的简单映射。

当使用存储过程时，必须对为每个 CREATE PROCEDURE 语句而创建的 *PGM 对象授权。当使用 Java™ 存储过程时，应授权用户（或 *PUBLIC）使用 Java 类文件。当使用 Java 存储过程时，将所有类文件存储在以下目录中:

```
/QIBM/UserData/OS400/SQLLib/Function
```

确保 *Spserver.class* 在此目录中。存储过程 SAMPLE.TESTRS 是仅有的 SQL 存储过程。它与 Java 类没有相关性。

要在 DB2 通用数据库中定义样本存储过程并对其编目，请使用下列步骤:

1. >qsh
2.
```
>cd  /QIBM/UserData/WebASAEs4/worf/
    installedApps/servicesApp.ear/services.war/WEB-INF/
     classes/groups/dxx_sample
```

   其中 **WebASAEs4** 是 WebSphere® 的版本，**worf** 是 WebSphere 实例的名称。
3. >db2 -f Spcreate.db2

要除去存储过程定义，执行以下命令:

```
>db2 -f Spdrop.db2
```

WORF 可用作 IBM® DB2 Information Integrator 的一部分。DB2 通用数据库版本 8 和 WebSphere Studio V4 和 V5 也提供了 WORF。当和 WebSphere Studio 一起交付时，可以使用这些工具来自动构建 DADX Web 服务。这些工具包括创建基于 SQL 语句或 DAD 文件的 DADX 文件的向导。还包括创建 DAD 文件的工具。WORF 还可使用 Informix®。

通过将 WORF 运行时环境与 DB2 配合使用，可以使用 DB2 XML Extender 来实现 Web 服务。DB2 XML Extender 由存储过程、用户定义的类型（UDT）和用户定义的

函数（UDF）组成。通过使用 DB2，可以使用这些功能部件来存储并检索 XML 数据。DB2 XML Extender 允许完整地存储 XML 文档，并可选择在副表中为它建立索引。通过使用 XML 列存取方法来实现，或将文档作为关系表的集合通过使用 XML 集合存取方法来实现。DB2 XML Extender 使用称为“文档存取定义”（DAD）的 XML 文档格式来定义 XML 与关系数据之间的映射。

相关概念:

- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 57 页的『在 iSeries 平台中定义 web.xml 和 group.properties 文件』
- 第 50 页的『在 iSeries 中安装和部署 WORF 示例』
- 第 34 页的『在 iSeries 上安装 Web 服务提供程序的软件需求』

DADX 文件的定义

“文档存取定义扩展”（DADX）文件指定如何创建 Web 服务。Web 服务是通过 Web 调用的函数。可以通过使用由 SQL 语句、存储过程调用或 DAD 文件定义的一组操作来创建 Web 服务。Web 服务存储“可扩展标记语言”（XML）文档或检索 XML 文档，包括某些由 DB2® XML Extender 管理的 XML 文档。在 DADX 文件中指定的 Web 服务称为 *DADX Web 服务*或 IBM® DB2 Information Integrator Web 服务。

WORF 为将 DADX 文档作为 Web 服务调用提供运行时支持。这些 Web 服务使用 Apache 简单对象访问协议（SOAP）（V2.3 或更新版本）引擎，或者 Apache Axis 引擎（V1.2）。这些 SOAP 引擎受 WebSphere® Application Server 和 Apache Jakarta Tomcat 支持。

DADX 文件的内容决定了 Web 服务将使用一组预定义的 SQL 操作还是动态 SQL 操作。如果 DADX 文件包含动态查询服务标记（<DQS>），则可以从浏览器指定 SQL 操作或将操作嵌入到应用程序中（如果安装了 WORF 测试 Web 应用程序）。

只需要很少的 XML 或 SQL 知识，就可以使用简单文本编辑器或 WebSphere Studio 中提供的工具来创建 DADX 文档。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』

Web 服务提供程序功能部件

WORF 提供下列功能部件:

- 第 28 页的基于资源的部署
- 自动服务重新部署，开发期间，当定义资源更改时
- 超文本传输协议（HTTP）GET 和 POST 绑定（除 SOAP 之外）
- WSDL 和 XSD 文件生成，支持 UDDI 最佳实践，UDDI 最佳实践代表行业标准

- 文档和测试页生成
- 以 HTML 和 XML 格式生成 Web 服务检查语言（WSIL）页

WORF 的一个主要功能是支持 Web 服务的基于资源的部署。资源文件（例如 DADX 文件）对 WORF 描述 Web 服务，这样 WORF 可从这些文件生成适当的 Web 服务。当请求资源文件时，WORF 装入该文件并使其成为可用的 Web 服务。如果编辑该资源文件并再次请求它，则 WORF 检测更改并自动装入新版本。这种自动重新装入资源文件的过程提高了 Web 服务开发的效率。

可以创建您自己的资源文件。资源文件必须符合特定语法和语义规则。资源文件可以相互引用（例如，DADX 文件可以包含对 DAD 文件的引用）。这些引用必须是正确的，以便您可以正确部署 Web 服务。

除了对“可扩展标记语言”（XML）数据指定存储和检索操作之外，WORF 允许存储过程和 SQL 语句作为可调用的 Web 服务操作公开。可以公开任何数据库存储过程。WORF 假定存储过程结果集具有固定的元数据。固定元数据指的是带有固定数字和固定形状的数据，它暗示具有特定列名和数据类型的特定数目的列。操作特征符包括输入和输出参数。还可以在使用由 WORF 提供的动态查询服务（DQS）时执行存储过程，而不具有所需要的固定元数据集或结果集。还可以指定 SQL 语句以选择、插入、更新和删除数据。并且，WORF 提供“可扩展标记语言”（XML）模式至 SQL 数据类型的简单映射。这些特别的功能部件不需要 XML Extender。

相关概念:

- 第 81 页的『DADX 文件的 WSDL』
- 第 104 页的『访问带有 GET、POST 和 SOAP 绑定的 Web 服务』
- 第 119 页的『Web 服务自动重新装入』
- 第 118 页的『Web 服务文档』
- 第 113 页的『Web 服务提供程序中存在的 Web 服务』

相关参考:

- 第 62 页的『DADX 文件的语法』

与 DADX 文件配合使用的 Web 服务提供程序操作

DADX 文件支持三类 Web 服务操作：非动态 SQL 操作、动态 SQL 操作和 XML 集合操作。基于 SQL 的查询可以将 SQL 语句（包括存储过程调用）发送至 DB2® 并返回包含缺省标记的结果。应用程序仅通过使用 SQL 数据类型的简单映射（将列名用作元素）返回数据。

SQL 操作：非动态

SQL 操作可以是非动态的。非动态操作是在 DADX 文件内预定义的操作。构成预定义的 SQL 操作类型的元素有三个:

<query>
查询数据库

<update>
插入到数据库中、从数据库中删除或更新数据库

<call> 调用不返回结果集或返回多个结果集的存储过程

SQL 操作: 动态

SQL 操作可以是动态操作，这取决于 DADX 文件的内容。动态操作是在不具有预定义 SQL 操作的 SOAP 消息内生成的操作。以下元素是动态操作:

<getTables>
检索可用表的描述。

<getColumns>
检索列的描述。

<executeQuery>
发出单个 SQL 语句。

<executeUpdate>
发出单个 INSERT、UPDATE 和 DELETE。

<executeCall>
调用单个存储过程。

<execute>
发出单个 SQL 语句。

可扩展标记语言（XML）集合操作（需要 DB2 XML Extender）

这些存储和检索操作帮助您将 XML 文档结构映射至 DB2 Universal Database™（DB2 通用数据库）表。可以从现有 DB2 数据构成 XML 文档，也可以将 XML 文档分解（存储未标记元素或属性内容）为 DB2 数据。此方法对于数据交换应用程序（尤其是当应用程序经常更新 XML 文档的内容时）非常有用。

有两个元素组成 XML 集合操作类型:

<retrieveXML>
生成 XML 文档

<storeXML>
存储 XML 文档

DAD 文件无论是在存储还是检索方面，对 XML 文档至 DB2 数据库的映射都提供了良好的控制。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 102 页的『测试 Web 服务应用程序 - 方案』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关参考:

- 第 74 页的『DADX 操作示例』
- 第 93 页的『Web 服务提供程序中的动态查询服务操作』

Web 服务过程概述

以下是使 DB2® 通用数据库成为 Web 服务提供程序所需步骤的概述。Web 应用程序开发者通过下列步骤来创建以下通用层次结构:

```
Web 应用程序–> 组 –>
  DADX 文件（Web 服务）–> SQL 操作
```

如果使用企业归档文件，则通用层次结构是:

```
企业应用程序 -> Web 应用程序 ->
  组 -> DADX 文件（Web 服务）-> SQL 操作
```

1. 数据库管理员建立数据库。
2. 数据库管理员可选择为 DB2 XML Extender 启用数据库。retrieveXML 和 storeXML 操作需要 DB2 XML Extender。使用 XML 列也需要 DB2 XML Extender。请参阅 *DB2 XML Extender Administration and Programming* 中的管理章节以了解如何为 XML Extender 启用数据库。
3. Web 应用程序开发者创建企业归档或 Web 归档（EAR 或 WAR）文件。当部署 EAR 或 WAR 文件时，它变为包含可进一步修改的 Web 应用程序的文件夹。 EAR 是 Java™ 2 Enterprise Edition（J2EE）规范中描述的企业应用程序。J2EE 规范中还描述了 WAR 文件。Web 应用程序文件夹是相关文件和工具的集合，位于服务器上。Web 应用程序包括界面、程序流、程序逻辑和数据存取信息，以创建在因特网上进行商业活动的基本结构。请参阅下列文档以了解更多信息:
   - WebSphere® Application Server Advanced Edition V5 文档

     可以在 WebSphere Application Server 上创建 WAR 或 EAR 文件。
   - Apache Jakarta Tomcat 文件

     可以在 Tomcat 上仅创建 WAR 文件。
4. 随后 Web 应用程序开发者创建一个组，并为此组创建 group.properties 文件。group.properties 文件包含有关数据库连接的信息和 WORF 使用的其它相关信息。组是若干访问数据库的 Web 服务操作。每个数据库可以有一个组，甚至同一数据库可以有多个组，可以在此组中定义一个或多个 DADX 文件。明确定义 Web 服务的 DADX 文件包含执行 Web 服务的操作。有关组属性的更多信息，请参阅定义 web.xml 和 group.properties 文件。
5. 数据库开发者可选择创建 DAD 来映射 XML 和关系数据转换（当使用 XML Extender 存储过程时是必需的）。
6. Web 服务开发者创建 DADX 文档。DADX 文件的内容确定是否可以在 Web 应用程序中使用动态查询。具有动态查询服务标记（</DQS>）的 DADX 文件仅包含作为启用动态查询的开关的标记。非动态 DADX 文件定义一组操作，并包含用于创建 Web 服务的信息。有关创建 DADX 文件的规则的更多信息，请参阅DADX 文件的语法。
7. 可选: 当使用 WebSphere Application Server 配置管理器 Windows® 版或 UNIX® 版时，Web 服务开发者为 Web 服务创建并部署部署描述符。部署描述符是一种 **isd** 文件（与 Apache SOAP 引擎配合使用）或 **deploy.wsdd** 文件（与 Apache Axis 引擎配合使用），它标识配置和部署信息。在 Apache Axis 引擎上，WORF 自动创建部署描述符。每个使用 Apache SOAP 的 Web 服务可以包含一个 **.isd* 文件。Web 应用程序可以包含多个 Web 服务。将所有 isd 或 wsdd 文件复制到 dds.xml 文件中。（此步骤对于 Apache Jakarta Tomcat 的用户而言是自动的）。有关部署描述符的更多信息，请参阅生成部署描述符。
8. 如果部署随 IBM® DB2 Information Integrator 附带的 WORF 示例，则可以使用可用的 DADX 测试页验证 Web 服务。可以将 Java Server Pages 从 WORF 目录复制到 apache-services.war 文件或 axis-services.war 文件，以测试应用程序中的某些 WORF 功能。

注意，在某些环境中，这些任务可能由一个人来执行。

相关概念:

- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』
- 第 53 页的『定义一组 Web 服务』

相关任务:

- 第 169 页的『部署联合应用程序』
- 第 124 页的『生成部署描述符』
- 第 54 页的『定义 web.xml 和 group.properties 文件』
- 第 40 页的『在 WebSphere Application Server V4.0.4 for z/OS 或 OS/390 上部署 WORF 示例』
- 第 37 页的『在 WebSphere Application Server for Windows and UNIX V5.1 或更新版本上部署 WORF 示例』

相关参考:

- 第 62 页的『DADX 文件的语法』

安装和配置 Web 服务提供程序

确定系统的容量并规划需要对应用程序编程接口和 Web 服务应用程序安装的软件。

UNIX 和 Windows 的 Web 服务提供程序软件需求

您可在下列任何操作系统中设置“Web 服务对象运行时框架”（WORF）或 Web 服务提供程序:

- Windows NT®
- Windows 2000®
- Linux
- AIX®
- Solaris Operating Environment

除与 IBM DB2 Information Integrator 一起打包外，WORF 还是 DB2 通用数据库扩展服务器版版本 8 和 WebSphere Studio V5 的一部分。WORF 在 DB2 通用数据库版本 8 中位于下列路径: *<DB2 UDB installed location>\samples\java\Websphere\dxxworf.zip*。

可以使用下列数据库环境:

- IBM DB2 通用数据库™版本或更新版本

  http://www.ibm.com/software/data/db2。这包括 DB2 XML Extender。
- Informix Dynamic Server（IDS）V9.3

另外，使用下列软件（视使用的服务器而定，此软件的大部分可能已是环境的一部分）:

- Java™ Java Development Kit（JDK）V1.2 或 1.3（http://java.sun.com 或 http://www.ibm.com/java）
- 下列 Web 服务器之一
  - WebSphere Application Server Advanced Edition V5（http://www.ibm.com/software/webservers/appserv/）

- Apache Web 服务器:
  - Apache Jakarta Tomcat V3.3.1 至 4.0.3 或更新版本（http://www.apache.org/）
    - Apache Jakarta Tomcat V4 standard 随适当的 Xerces 提供。
    - 对于早于 V4 的 Apache Jakarta Tomcat 版本，必须将 Xerces 解析器添加到 CLASSPATH 才能将其用作 XML 解析器
  - Apache SOAP 2.3 或更新版本的二进制，或 Apache Axis 1.2（http://xml.apache.org/（需要文档对象模型级别 2（DOM 2），它受 Xerces Java 1.4.4 或更新版本支持））。
  - Xerces Java 解析器版本 1.4.4（http://xml.apache.org/）
  - JavaMail V1.2（http://java.sun.com/）
  - JavaBeans™ Activation Framework V1.0.1（http://java.sun.com/）。Apache SOAP 需要 JavaBeans Activation Framework 版本。
  - j2ee.jar V1.3 或更新版本（http://java.sun.com/）
  - qname.jar（http://java.sun.com/）
  - *wsdl4j.jar*. 可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。

相关概念:

- 第 27 页的『DADX 文件的定义』

相关任务:

- 第 48 页的『在 Apache Jakarta Tomcat 上安装和部署 WORF 示例』
- 第 33 页的『安装 Web 服务提供程序的软件需求』

OS/390 和 z/OS 的 Web 服务提供程序软件需求

可在下列任何一个操作系统中设置 Web 服务提供程序（Web 对象运行时框架）:

- OS/390 V2.8 或更新版本
- z/OS V1.1 或更新版本

可以使用下列数据库环境:

- IBM DB2 通用数据库 OS/390 版版本 7 或 DB2 通用数据库 z/OS 版（http://www.ibm.com/software/data/db2/os390 ）
- IBM DB2 XML Extender for OS/390 V7 或更新版本（http://www.ibm.com/software/data/db2/extenders/xmlext/index.html）。存储和检索操作必需的环境

另外，使用以下软件:

- WebSphere Application Server V4.01 服务级别 W401505 或更新版本
- JavaMail V1.2（http://java.sun.com/）
- JavaBeans Activation Framework V1.0.1（http://java.sun.com/）
- j2ee.jar V1.3 或更新版本（http://java.sun.com/）
- qname.jar（http://java.sun.com/）
- 带有程序临时性修订（PTF）UW95866 的 IBM XML Toolkit for z/OS and OS/390 V1.4（http://www.ibm.com/servers/eserver/zseries/software/xml/）

- *wsdl4j.jar*. 可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。

相关任务:

- 第 102 页的『测试 Web 服务』
- 第 40 页的『在 WebSphere Application Server V4.0.4 for z/OS 或 OS/390 上部署 WORF 示例』

在 UNIX、Windows、z/OS 和 OS/390 上为 WebSphere Application Server 配置 Web 服务提供程序

可以在 WebSphere® Application Server Advanced Edition 上运行 Web 服务。“Web 服务对象运行时框架”（WORF）为下列操作提供运行时支持: 使用 Apache SOAP 2.3（或更新版本）或 Apache Axis 1.2（或更新版本）通过“超文本传输协议”（HTTP）将“文档存取定义扩展”（DADX）文档作为 Web 服务调用。WebSphere Application Server 5 或更高版本及其它 servlet 引擎支持此操作。WebSphere 允许您保护 SOAP Web 服务。有关更多信息，请参阅有关保护 SOAP 服务的 WebSphere 文档。下列各节描述 Web 服务。

安装 Web 服务提供程序的软件需求

先决条件:

确保安装了必需的软件。请参阅 UNIX 和 Windows 的 Web 服务提供者软件需求以验证 Windows 和 UNIX 中的安装。

需要 DB2® XML Extender 以用于 XML 与关系数据之间的高级映射控制。如果尚未创建 DB2 UDB SAMPLE 数据库，则通过创建此数据库来验证 DB2 UDB 安装。Web 服务需要“Java 数据库连接”（JDBC）2.0，它是 DB2 通用数据库版本 8 中的缺省配置。

请参阅 OS/390 和 z/OS 的 Web 服务提供者软件需求以验证 z/OS 安装。

过程:

在 UNIX 和 Windows 中准备 Web 服务环境的过程如下所示:

1. 停止使用 DB2 通用数据库的任何服务（例如 WebSphere Application Server）
2. 停止 DB2。
3. 对于版本 8 之前的 DB2 通用数据库版本，选择“Java 数据库连接”（JDBC）2.0。运行 *C:\SQLLIB\java12\usejdbc2.bat* 文件（假设您使用 Windows 环境，并在 C:\SQLLIB\ 中安装了 DB2）。
4. 重新启动 DB2。
5. 从 WebSphere Application Server Advanced Edition 5.1 的安装目录启动它。这些指示信息假定您在 Windows 环境 C:\WebSphere\Appserver 中安装了 WebSphere Application Server。

在 OS/390 或 z/OS 上准备 Web 服务环境的过程如下所示:

1. 创建新的目录以存储应用程序扩展（如果还没有这样的目录）

2. 在指定的 J2EE 服务器实例中将 APP_EXT_DIR 环境变量设置为此应用程序扩展目录
3. 将下列 JAR 文件添加至该应用程序扩展目录:

   xerces.jar
   此文件位于 IBM XML Toolkit for z/OS 中，可以从 http://www.ibm.com/servers/eserver/zseries/software/xml/ 下载此文件。

   mail.jar
   此文件在 JavaMail 中

   activation.jar
   此文件在“Java Bean 激活框架”中

   j2ee.jar
   可以从 http://java.sun.com/products 下载此文件

   qname.jar
   可以从 http://java.sun.com/products 下载此文件

   wsdl4j.jar
   可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。
4. 使用下列步骤验证 J2EE 服务器实例的配置:
   - 确保随 WebSphere Application Server 提供的 soap.jar 是 CLASSPATH 的一部分
   - 将下列设置添加至 J2EE 服务器实例的 jvm.properties 文件:

     ```
     com.ibm.ws390.server.classloadermode=2
     com.ibm.ws.classloader.ejbDelegationMode=false
     ```
5. 重新启动 J2EE 服务器。

相关任务:

- 第 32 页的『OS/390 和 z/OS 的 Web 服务提供程序软件需求』
- 第 47 页的『在 Apache Jakarta Tomcat 上安装或迁移 WORF』
- 第 36 页的『安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本』

相关参考:

- 第 31 页的『UNIX 和 Windows 的 Web 服务提供程序软件需求』

在 iSeries 上安装 Web 服务提供程序的软件需求

先决条件:

确保安装了必需的软件。使用以下命令从交互式 SQL 创建 SAMPLE 数据库:

```
CALL QSYS/CREATE_SQL_SAMPLE('SAMPLE')
```

需要 DB2 XML Extender 以用于 XML 与关系数据之间的高级映射控制。要使用 DB2 通用数据库 XML Extender，确保安装了该产品。可通过发出 CL 命令 **GO LICPGM** 来验证是否在系统上安装了 DB2 通用数据库 XML Extender。对于 DB2 iSeries 版 V5R2，如果安装了 DB2 通用数据库 XML Extender，则下列条目显示为 GO LICPGM 命令的结果:

- 5722DE1 *COMPATIBLE DB2 UDB Extenders

- 5722DE1 *COMPATIBLE DB2 UDB Text Extender
- 5722DE1 *COMPATIBLE DB2 UDB XML Extender
- 5722DE1 *COMPATIBLE Text Search Engine

使用以下 CL 命令启用 DB2 通用数据库 XML Extender: **CALL PGM(QDBXM/QZXMADM) PARM(enable_db LOCALRDB)**。LOCALRDB 是关系数据库目录中的 *LOCAL 数据库名称。要使用关系数据库条目，发出以下 CL 命令: **WRKRDBDIRE**。如果使用样本文件中的文档类型定义（DTD），则执行脚本 *setup-dxx.cmd*。WORF 需要 “Java 数据库连接”（JDBC）2.0，它是 DB2 通用数据库版本 8 中的缺省配置。

过程:

准备 WORF 环境的过程如下所示:

1. 停止使用 DB2 的任何服务（例如 WebSphere Application Server）
2. 对于版本 8 之前的 DB2 通用数据库版本，选择 “Java 数据库连接”（JDBC）2.0。运行 *C:\SQLLIB\java12\usejdbc2.bat* 文件（假设您使用 Windows 环境，并在 C:\SQLLIB\ 中安装了 DB2）。
3. 启动 WebSphere Application Server Advanced Edition 4.01 或 5.0（假定您在 C:\WebSphere\Appserver 中安装了 WebSphere Application Server）。

相关概念:

- 第 25 页的『将 Web 服务提供程序与 iSeries 配合使用』

相关任务:

- 第 49 页的『在 iSeries 上为 Apache Jakarta Tomcat 安装 Web 服务提供程序的软件需求』
- 第 57 页的『在 iSeries 平台中定义 web.xml 和 group.properties 文件』
- 第 50 页的『在 iSeries 中安装和部署 WORF 示例』

用于 XML Extender 的 DTD 定义

确保数据库管理员已设置了应用程序所需的任何数据库或子系统，并将启用它们以供 DB2 XML Extender 使用（如果使用 XML Extender 的话）。下表列示了 XML Extender 样本引用的缺省位置。

表 3. XML Extender 样本引用下列文档类型定义（DTD）

平台	DTD 的缺省位置
Windows 上的 DB2 UDB 版本 7.2 修订包 7 或更新版本	`c:\dxx\samples\dtd\ getstart.dtd` `c:\dxx\dtd\dad.dtd`
Windows 上的 DB2 UDB 版本 8	`c:\<DB2 UDB installed location>\samples\ db2xml\dtd\getstart.dtd` `c:\<DB2 UDB installed location>\samples\ db2xml\dtd\dad.dtd`
Solaris Operating Environment 上的 DB2 UDB 版本 8	`/opt/IBMdb2/V8.1/samples/ db2xml/dtd/dad.dtd`

表 3. XML Extender 样本引用下列文档类型定义（DTD） (续)

平台	DTD 的缺省位置
AIX 上的 DB2 UDB 版本 8	/usr/opt/db2_08_01/ samples/db2xml/dtd/dad.dtd
Linux 上的 DB2 UDB 版本 8	/usr/IBMdb2/V8.1/samples/ db2xml/dtd/dad.dtd
OS/390 和 z/OS 上 DB2 UDB 版本 7 或 OS/390 和 z/OS 上的 DB2 UDB 版本 8	/u/USER/dxx/dtd/dad.dtd

以下是引用 dad.dtd 的一些文件的列表:

- department.dad
- department2.dad
- departmentStd.dad
- order.dad
- order-public.dad
- getstart.xml
- order-10.xml
- sales_db.nst

相关任务:

- 第 80 页的『将文档类型定义转换为 XML 模式』

相关参考:

- 第 74 页的『DADX 操作示例』

安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本

先决条件:

在工作站的某个路径中安装 WebSphere Application Server, 例如 C:\WebSphere\Appserver（在 Windows 环境中）。

过程:

要从早期版本迁移至 WORF V8.2, 请参阅迁移总结部分。要安装 WORF V8.2, 请完成下列步骤:

1. 将 *dxxworf.zip* 解压到某个目录, 例如 *C:\worf*, 以便目录具有以下内容:
   - *readme.html*
   - *lib\apache-services.war* 和 *lib\axis-services.war* – 包含使用 WORF 的 Web 服务的样本 Web 应用程序。
   - *lib\worf.jar* – WORF 库。在 servlet 引擎的类路径上安装此文件
   - *lib\worf-servlets.jar*
   - *schemas* – 用于 DADX 和名称空间表（NST）XML 文件的“可扩展标记语言（XML）模式, 包括 wsdl.xsd、db2WebRowSet.xsd 和 dadx.xsd。

- *tools* – 包含 DAD 和 DADX 检查程序工具的 tools 目录。

2. 验证您正在使用的服务器（例如 WebSphere Application Server）的目录是否包含相应的 Web 服务引擎 jar 文件。如果文件未在此目录中，则从包含 Apache Axis 或 Apache SOAP 文件的目录中复制 jar 文件以便可以启用相应的 Web 服务引擎。
   - 如果要使用 Apache Axis 框架，则将 axis.jar 复制到 c:\WebSphere\AppServer\lib。然后，将 axis/lib 的内容复制到 c:\WebSphere\AppServer\lib 以存取其它 Apache Axis JAR 文件。
   - 如果要使用 Apache SOAP 框架，则将 soap.jar 复制到 WebSphere\AppServer\lib。
3. 将 *worf.jar* 复制到 *C:\WebSphere\AppServer\lib*。
4. 如果您的 WebSphere 服务器是早于 WebSphere 5.0.2 的发行版，则必须从 http://java.sun.com/xml/downloads/saaj.html 下载名为 `saaj.jar` 的文件。将 `saaj.jar` 复制到 *C:\WebSphere\AppServer\lib*。
5. 启动 WebSphere Application Server。
6. 选择“开始” —> “程序” —> IBM WebSphere —> “管理员控制台”来打开“管理员”控制台。
7. 配置 WebSphere 以便与 DB2 UDB 环境一起运行:
   a. 从左侧导航窗格，单击“服务器” —> “应用程序服务器”。
   b. 在右侧内容窗格中找到服务器的名称，然后单击服务器名称。
   c. 单击“进程定义” —> “Java 虚拟机”。
   d. 在“配置”页上，将类路径指定为 Java 数据库信息的路径。如果在目录 sqllib\ 中安装了 DB2 通用数据库，并且使用随 WORF 样本附带的 group.properties 文件，则以下示例是有效路径:
      ```
      C:\SQLLIB\java\db2java.zip
      ```
   e. 单击**应用**或**确定**。
   f. 保存配置。
8. 停止 WebSphere Application Server。

迁移总结

要从 WORF 的早期版本迁移到 WORF V8.2:

1. 将 *lib\worf.jar* 从 V8.2 dxxworf.zip 复制到 *C:\WebSphere\AppServer\lib*，以便替换 worf.jar 文件。dxxworf.zip 位于以下路径: *<DB2 UDB installed location>\samples\java\Websphere\dxxworf.zip*。
2. 对于您部署的每个应用程序，用 apache-services.war 或 axis-services.war 的 worf 目录中的文件替换该应用程序 worf 目录中的 JSP 文件。然后重新部署该应用程序。

相关概念:

- 第 53 页的『定义一组 Web 服务』

相关任务:

- 第 33 页的『安装 Web 服务提供程序的软件需求』

在 WebSphere Application Server for Windows and UNIX V5.1 或更新版本上部署 WORF 示例

先决条件:

- 在工作站的某个路径中安装 WebSphere Application Server，例如 C:\WebSphere\Appserver（在 Windows 环境中）。
- 安装 WORF。

过程:

要安装和部署 WORF 示例，完成下列步骤:

1. 启用 WebSphere Administration Server。
2. 选择“开始” —> “程序” —> IBM WebSphere —> “管理员控制台”来打开“管理员”控制台。
3. 选择“应用程序” —> “企业应用程序”。内容窗口显示在当前服务器上安装的所有企业应用程序。
4. 单击**安装**按钮将 *apache-services.war* 或 *axis-services.war* 作为企业应用程序安装。
   a. 从**本地路径**字段，单击**浏览**按钮以查找 c:\WORF\lib 目录中包括的正确服务的路径。WORF 提供两个服务文件供您在运行样本时选择。这些文件是 apache-services.war 和 axis-services.war。为要运行的 SOAP 引擎选择正确的服务文件。
   b. 在**上下文根**字段中指定 Web 应用程序的上下文名称。要执行此处讨论的示例，必须指定 `services` 作为上下文根名称。

   图 4 显示应用程序安装期间的 WebSphere Application Server 管理员控制台:

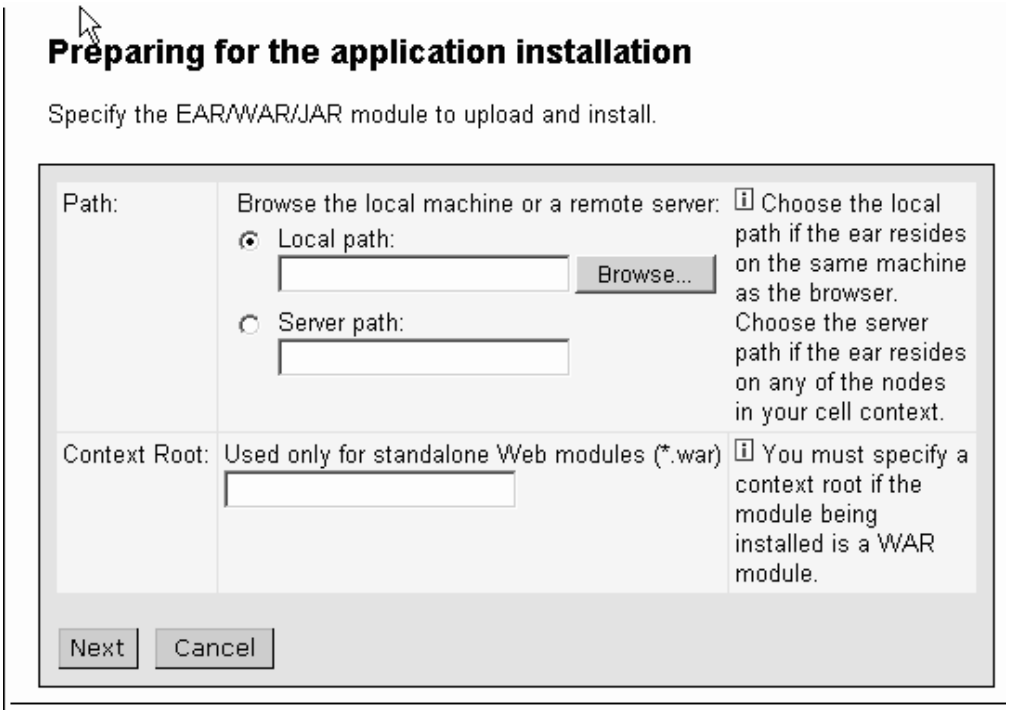


图 4. 应用程序或模块的规范

   c. 单击**下一步**。
   d. 接受其它所有缺省值，然后单击**下一步**，以转至向导的其它部分。在**映射 Web 模块的虚拟主机**窗口中，选择 `.WAR` 文件，然后单击**下一步**。在**将模块映射到应**

用程序服务器窗口中，选择 .WAR 文件，然后单击**下一步**。配置选项指定虚拟主机（例如: default_host）和应用程序服务器（例如: Default Server）。在“向导”的结尾，单击**完成**。

e. 最后的窗口显示“保存至配置”。单击**保存**。

5. 验证 group.properties 文件中的数据库设置（尤其是用户标识和密码）是否正确。
6. 在 Windows 环境的 DB2 通用数据库命令窗口（DB2 UDB 命令行处理器窗口）中发出 *setup.cmd*，或在每个数据库目录中的 UNIX 环境命令窗口中发出 *setup.sh* 以创建数据库。例如，在 *dxx_sales_db* 目录中运行 setup.cmd 以设置使用 DB2 XML Extender 的 SALES_DB 数据库。

   警告: 如果在 dxx_sample 目录中发出 setup 命令，命令将删除 SAMPLE 数据库，然后重新创建此数据库。如果使用随 DB2 通用数据库产品提供的 SAMPLE 数据库，则应清楚您将丢失对数据库所做的修改。
7. 如果部署自己的应用程序，则将 worf-servlets.jar 文件从 WORF 目录复制到 WebSphere/AppServer/installedApps/<host>/<application WAR directory>/WEB-INF/lib。
8. 停止当前服务器。
9. 重新启动服务器。
10. 通过选择“应用程序” —> “企业应用程序”，验证 services.war 已在运行。
11. 通过访问 Web 应用程序欢迎页面，打开浏览器窗口以测试安装。

    特定端口号随 WebSphere Application Server 配置的不同而有所不同。如果使用缺省值，则 “services” Web 应用程序欢迎页面可能是 *http://localhost:9080/services*。请记住: services 是在以前的步骤中创建的应用程序的名称。该页看上去应如在图 5 和第 40 页的图 6 中所显示的屏幕:

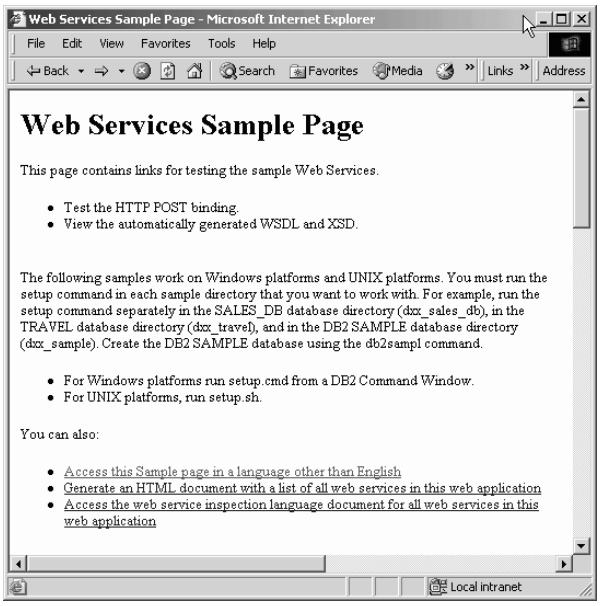


图 5. WORF 样本页

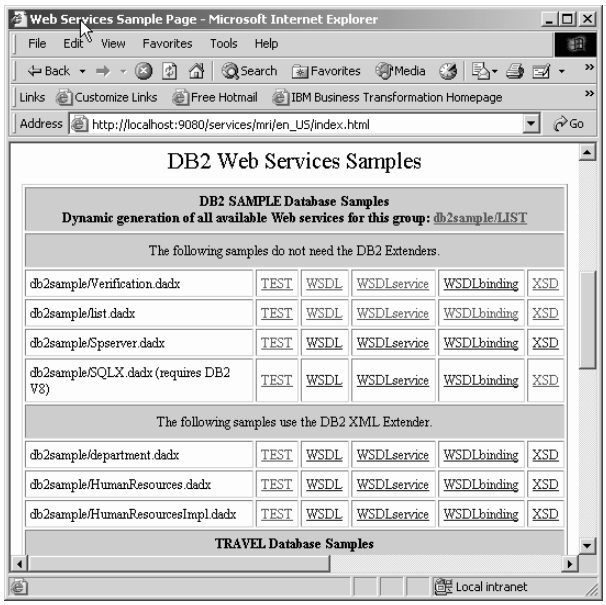


图 6. Web 服务样本页 – 测试链接

12. 单击某些链接以验证样本服务是否工作。测试页由操作的树形视图、输入视图和结果视图组成。您可以从样本“欢迎页面”内的 TEST 链接或在浏览器中输入以下内容来访问测试页面:
    <your-Web-server>:9080/<context_root_name>/<group_name>/<dadx file>/TEST。

相关任务:

- 第 36 页的『安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本』
- 第 54 页的『定义 web.xml 和 group.properties 文件』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 223 页的『在输出文本文件中指示错误和警告』
- 第 222 页的『运行 DADX 环境检查程序』

在 WebSphere Application Server V4.0.4 for z/OS 或 OS/390 上部署 WORF 示例

这些步骤用于部署将在 z/OS 或 OS/390 平台上的 WORF 中使用的 Web 应用程序。还可以验证您是否正确安装和配置了 WORF 及其先决条件。

过程:

要安装 WORF，完成下列步骤:

1. 下载并解压缩 *dxxworf.pax* 至一个空目录，如 */u/USER/worf/*。可使用以下命令来解压缩文件:

   ```
   pax -rvf dxxworf.pax
   ```

   展开文件之后，目录中就具有以下内容:

   - *readme.txt*

- *lib/apache-services.war* 和 *lib/axis-services.war* – 包含使用 WORF 的 Web 服务的样本 Web 应用程序。

  注: 注意: 此包含的命令和指令引用 services.war 或 services.ear。请将正确的 SOAP 文件用于选择运行的 SOAP 引擎。例如，示例可能引用 services.war 文件，但是如果安装了 Apache SOAP 引擎，则此文件是 apache-services.war。

- *lib/worf.jar* – WORF 库。
- lib/worf-servlets.jar
- *schemas/* – DADX 和 NST XML 文件的“可扩展标记语言”（XML）模式
- *tools/* – *该 tools 目录包含 DAD 和 DADX 检查程序工具。有关更多解释，请参阅安装 DADX 环境检查程序。*

2. 将 *worf.jar* 复制到 J2EE 服务器实例的应用程序扩展目录。
3. 启动（或重新启动）J2EE 服务器。

要安装和部署 WORF 示例，完成下列步骤:

1. 配置“系统管理脚本编制”应用程序编程接口（API）。有关在 OS/390 或 z/OS 系统上配置脚本编制 API 的更多信息，请参阅 *IBM WebSphere Application Server V4 for z/OS and OS/390: Installation and Customization* 和 *IBM WebSphere Application Server V4.0.1 for z/OS and OS/390: System Management Scripting API*。
2. 准备企业归档文件（EAR）。使用 EAR 文件来传递 Java 2 platform enterprise edition（J2EE）应用程序。EAR 文件由 Web 归档文件（WAR）和 Java 归档文件（JAR）构成。
   a. 在 UNIX 系统服务（USS）中，将 apache-services.war 或 axis-services.war 复制到可写的临时目录，并将当前目录更改为该位置。
   b. 从 USS 命令行输入以下命令（在一行上）:

      ```
      390fy -op "" -context_root "/services"
            -display_name "ServicesApp" services.war
      ```

      此命令在当前目录中创建名为 *services.ear* 的初始 EAR 文件。EAR 文件具有上下文根“/services”和显示名“ServicesApp”。“上下文根”是“统一资源定位器”（URL）的一部分，它将 WebSphere Application Server 导向至您的应用程序。“显示名”是在“系统管理最终用户界面”（SM/EUI）和 USS 中标识应用程序的字符串。可将“上下文根”和“显示名”编辑为选择的任何名称。
   c. 解析 services.ear 的“Java 命名和目录接口”（JNDI）名称映射。通过从 USS 命令行发出以下命令（在一行上）来执行此操作:

      ```
      390fy -JNDIejbp "/<Sysplex>/<J2EE Server>"
               -op "_resolved" services.ear
      ```

      上面示例中使用的术语具有以下定义:

      <J2EE Server>
      将在其上部署应用程序的 J2EE 服务器的名称

      <Sysplex>
      J2EE 服务器所在的 Sysplex 的名称

      此命令在当前目录中创建名为 *services_resolved.ear* 的新文件。
3. 部署应用程序

注: 当对此步骤执行命令时，必须使用注册为 WebSphere 的“系统管理管理员”的用户标识登录到 USS 中。

a. 将下列样本文件从 WebSphere Application Server 样本目录（*<WAS_Home>/samples/smapi/*）复制到包含刚创建的 EAR 文件的临时目录。

- *inputcreateconversation.xml*
- *inputprocessearfile.xml*
- *inputcommitconversation.xml*

b. 将环境变量 DEFAULT_CLIENT_XML_PATH 设置为包含 EAR 文件的临时目录。

c. 编辑文件 *inputcreateconversation.xml* 并指定对话名称，可选择指定名称的描述。对话名称和描述可以是想要的任何文本。但是，在以后的步骤中处理输入文件时，对话名称必须保持不变。以下是对话名称和描述的一个示例:

```
<inputcreateconversation conversationname="WORFSamples"
            conversationdescription="WORF Sample Test" />
```

保存文件 *inputcreateconversation.xml*。

d. 从 USS 命令行输入以下命令（在一行上）:

```
CB390CFG -action createconversation
   -xmlinput inputcreateconversation.xml
   -output createconv.out
```

此命令在系统的临时目录中创建文件 *createconv.out* 并且包含操作的结果。除验证应用程序部署是否成功外，不需要此文件。

e. 编辑文件 *inputprocessearfile.xml*。指定目标 J2EE 服务器和要部署的 EAR 文件。以下是指定 J2EE 服务器和 EAR 文件的一个示例:

```
<inputprocessearfile conversationname="WORFSamples"
                   j2eeservername="BBOASR2"
earfilename="/tmp/worfsamp/services_resolved.ear"
                   processingmode="standard" />
```

保存文件 *inputprocessearfile.xml*。

f. 从 USS 命令行输入以下命令（在一行上）:

```
CB390CFG -action processearfile
    -xmlinput inputprocessearfile.xml
    -output processear.out
```

此命令在系统的临时目录中创建文件 *processear.out* 并且包含操作的结果。除验证应用程序部署是否成功外，不需要此文件。

g. 编辑文件 *inputcommitconversation.xml*。指定在前面的步骤中使用的对话名称。例如:

```
<inputcommitconversation conversationname="WORFSamples" />
```

保存文件 *inputcommitconversation.xml*。

h. 从 USS 命令行输入以下命令（在一行上）:

```
CB390CFG -action commitconversation
  -xmlinput inputcommitconversation.xml
  -output commitconv.out
```

此命令在系统的临时目录中创建文件 *commitconv.out* 并且包含操作的结果。除验证应用程序部署是否成功外，不需要此文件。

4. 设置 Web 服务器。
   a. 确保 J2EE 服务器允许在步骤 第 41 页的 2 中命名的“上下文根”。在服务器的 *webcontainer.conf* 文件中，确保至少有一个主机具有如以下示例所示的规范:

      ```
      host.<host_alias>.contextroots=/somewebapp,/services
      ```

      要接受任何“上下文根”，可使用以下文本行:

      ```
      host.<host_alias>.contextroots=/*
      ```

      注: 使用此方法来指定上下文根使您可以在部署后续应用程序时跳过此步骤

   b. 如果使用 Web 服务器插件来访问 J2EE 服务器，则将 Service 语句添加至 Web 服务器的文件 *httpd.conf*。此语句指定已部署应用程序的“上下文根”。举例说明，如果在步骤 第 41 页的 2 中指定“/services”作为“上下文根”，则新的 Service 语句如以下示例所示:

      ```
      Service  /services/*
      <WAS_Home>/WebServerPlugIn/bin/was400plugin.so:service_exit
      ```

      <WAS_Home> 是 WebSphere Application Server 的安装目录。

      注: Service 语句应在一行上。

   c. 重新启动 Web 服务器。
5. 验证 WORF 配置。
   a. 访问包括在样本 WAR 文件中的“WORF Web 服务样本页”。如果在步骤 第 41 页的 2 中将“上下文根”设置为“/services”，则输入以下 URL:

      ```
      http://<hostname>/services/
      ```

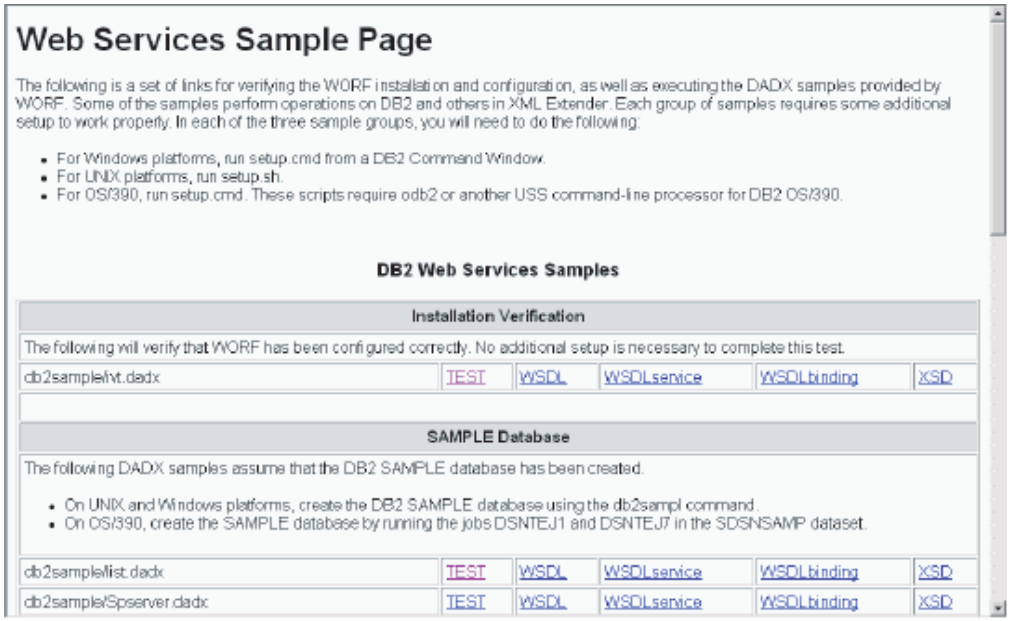


图 7. WORF 样本页

b. 在第 43 页的图 7 中，样本的第一部分（标题为 **Installation Verification**）显示了单个 DADX 文件 *ivt.dadx*。单击 **TEST** 链接。WORF 的内置测试工具打开。

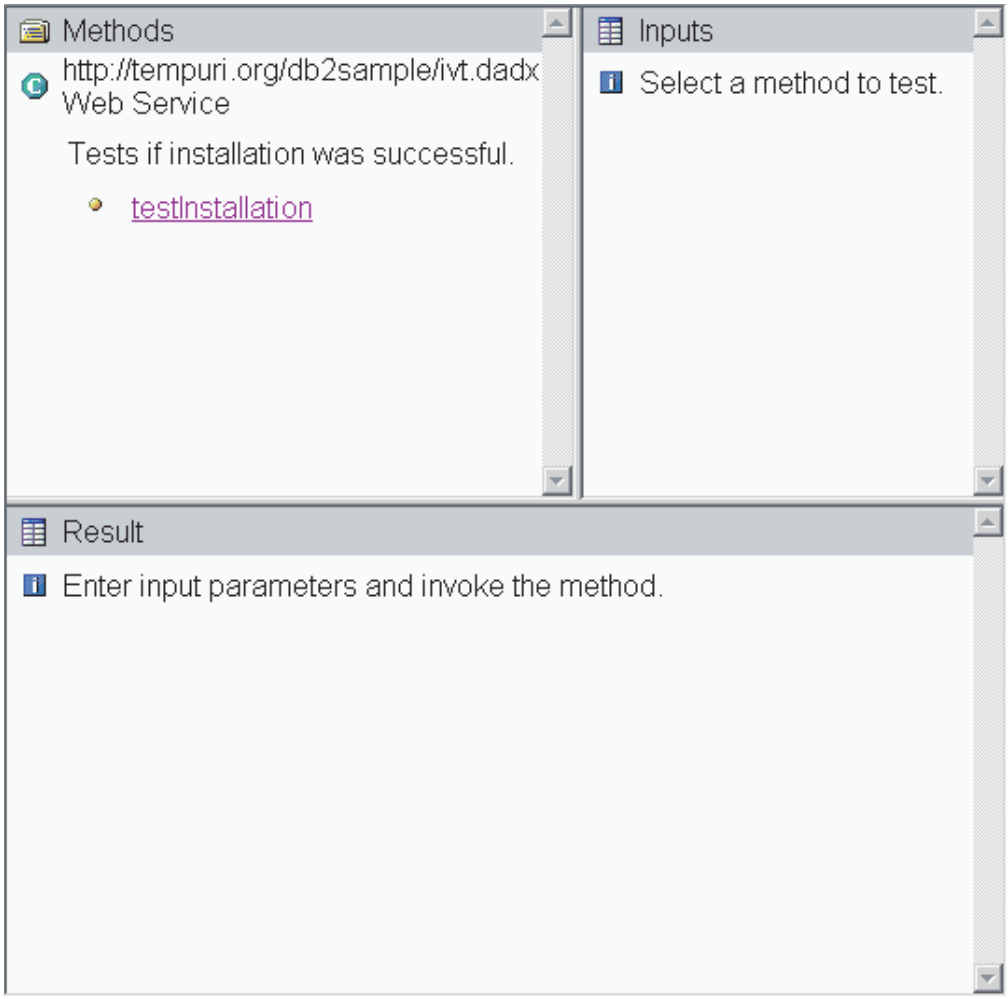


图 8. WORF 测试设施

c. 从 WORF 测试设施图 8 中，选择 “testInstallation” 操作。单击 **Invoke**。

d. XML 文档显示在窗口的底部框架中。验证当天的当前时间是否出现在文档中，如以下示例中所示:

```
<CURTIME>14:38:26.000Z</CURTIME>
```

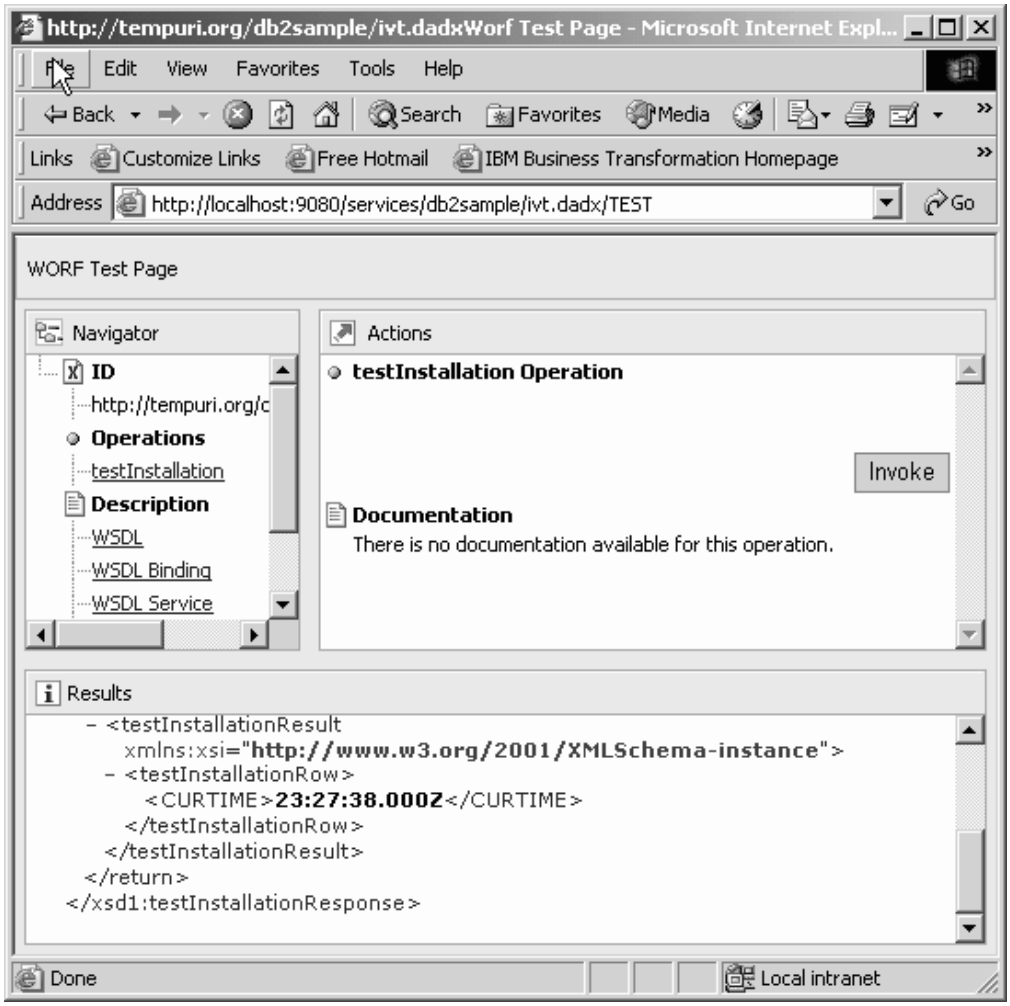


图 9. WORF 测试的结果 - 期望的输出

e. 如果此测试失败，则配置有错误。验证是否正确安装了软件需求。另外，验证您是否配置了 WebSphere Application Server 以及是否有权存取 DB2 通用数据库。

6. 准备和运行示例。

   a. 在样本页（请参阅步骤 第 43 页的 5a ）上，随 *services.war* 应用程序提供了一些 DADX 样本。在运行这些样本之前，遵循包含在每个类别中的设置指示信息。

   b. 通过为每个样本选择 **TEST** 链接来执行单独的 Web 服务。选择 **WSDL**、**WSDLservice**、**WSDLbinding** 和 **XSD** 链接以运行那些示例。

部署了应用程序之后，可以修改应用程序。还可以创建其它 WAR 文件以进行部署。在创建了 WAR 文件之后，使用步骤 第 41 页的 2 、步骤 第 41 页的 3 和步骤 第 43 页的 4 来部署 Web 应用程序。

相关概念:

- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 50 页的『在 iSeries 中安装和部署 WORF 示例』

- 第 48 页的『在 Apache Jakarta Tomcat 上安装和部署 WORF 示例』

相关参考:

- 第 221 页的『安装 DADX 环境检查程序』

在 UNIX 和 Windows 上为 Apache Jakarta Tomcat 配置 Web 服务提供程序

可以在 Apache Jakarta Tomcat 上运行 Web 服务。下列各节描述这些 Web 服务。

在 UNIX 和 Windows 上为 Apache Jakarta Tomcat 安装 Web 服务提供程序的软件需求

确保安装了必需的软件。使用特定文档来验证特殊平台的安装。

先决条件:

需要 DB2 XML Extender 以用于 XML 与关系数据之间的高级映射控制。通过创建 DB2 SAMPLE 数据库来验证安装。WORF 需要 “Java 数据库连接”（JDBC）2.0，它是 DB2 通用数据库版本 8 中的缺省配置。

过程:

准备 WORF 环境的过程如下所示:

1. 停止 DB2。
2. 如果未在运行 DB2 通用数据库版本 8，则选择 JDBC 2.0。运行 *C:\SQLLIB\java12\usejdbc2.bat*（假定您使用 Windows 环境，并在 C:\SQLLIB\ 中安装了 DB2。
3. 重新启动 DB2。
4. 安装下列因特网软件:
   - 来自 Apache:
     - Apache Jakarta Tomcat V4.0.6 或更新版本的二进制文件，位于 http://jakarta.apache.org/site/binindex.html。（Apache Jakarta Tomcat V4 Standard 提供适当的 Xerces 解析器。对于早期版本，必须将 Xerces 解析器添加至 CLASSPATH 才能将其用作 XML 解析器。）
     - Apache SOAP 2.3 或更新版本的二进制文件，位于 http://xml.apache.org/soap
     - Apache Axis 1.2，位于 http://www.apache.org/。
     - Xerces 1.4.4，位于 http://xml.apache.org/
   - 来自 Sun 公司（http://java.sun.com/products ）:
     - JavaMail 1.2
     - JavaBeans Activation Framework (JAF) 1.0 1
     - j2ee.jar V1.3 或更新版本。
     - qname.jar
   - *wsdl4j.jar*。可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。

相关任务:

- 第 32 页的『OS/390 和 z/OS 的 Web 服务提供程序软件需求』

相关参考:

- 第 31 页的『UNIX 和 Windows 的 Web 服务提供程序软件需求』

在 Apache Jakarta Tomcat 上安装或迁移 WORF

过程:

要从早期版本迁移至 WORF V8.2，请参阅迁移总结部分。要在 Apache Jakarta Tomcat 上安装 WORF V8.2，请完成以下步骤:

- 要使用 Apache SOAP 或 Apache Axis 运行 WORF，将下列 JAR 文件添加至应用程序服务器上的类路径:
  - *Apache SOAP 引擎的 soap.jar，或 Apache 轴引擎的 axis.jar*。
  - *xerces.jar*（或 Java XML 解析器的 jar）
  - *mail.jar*
  - *activation.jar*
  - *worf.jar*
  - j2ee.jar V1.3 或更新版本
  - qname.jar
  - *wsdl4j.jar*。可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。
  - *jaxrpc.jar*
  - *log4j-1.2.8.jar*
  - *commons-logging.jar*
  - *commons-logging-api.jar*
  - *commons-discovery.jar*
  - *db2java.zip* 或 *jcc.jar*（或者数据库服务器的 JDBC 实现 jar）。驱动程序类的名称取决于您使用的驱动程序包。可以在 `group.properties` 文件中修改所使用的驱动程序包。
- 修改第 47 页的表 4 中列示的文件。所做的修改取决于 Apache Jakarta Tomcat 版本和平台。为上面提到的每个 jar 文件添加一行。用 jar 文件的实际位置替换 <jarfile>。如果在集成开发环境中运行 Apache Jakarta Tomcat，请确保所有这些 jar 都在用于启动 Tomcat 的 CLASSPATH 上。您需要修改的文件都位于启动 Apache Jakarta Tomcat 的目录中。应使用 *app_server/bin* 目录中的 startup.bat 或 shutdown.bat，或者 startup.sh 或 shutdown.sh 来启动和停止服务器。

表 4. 类路径指定

平台	服务器软件	要修改的文件	要添加的命令
UNIX	Apache Jakarta Tomcat 3.2.x	\bin\tomcat.sh（在 “export CLASSPATH” 之前）	CLASSPATH = $CLASSPATH: <jarfile>
	Apache Jakarta Tomcat 3.3.x	\bin\tomcat.sh（在 “export CLASSPATH” 之前）	CLASSPATH = $CLASSPATH: <jarfile>
	Apache Jakarta Tomcat 4.x	\bin\setclasspath.sh（在 “export CLASSPATH” 之前）	CLASSPATH = $CLASSPATH: <jarfile>

表 4. 类路径指定 (续)

平台	服务器软件	要修改的文件	要添加的命令
Windows	Apache Jakarta Tomcat 3.2.x	\bin\tomcat.bat（:setClasspath 段）	set CP = %CP%; <jarfile>
	Apache Jakarta Tomcat 3.3.x	\bin\tomcat.bat	set CLASSPATH = %CLASSPATH%; <jarfile>
	Apache Jakarta Tomcat 4.x	\bin\setclasspath.bat	set CLASSPATH = %CLASSPATH%; <jarfile>

- 如果您的 WebSphere 服务器是早于 WebSphere 5.0.2 的发行版，则必须从 http://java.sun.com/xml/downloads/saaj.html 下载名为 `saaj.jar` 的文件。将 `saaj.jar` 复制到 *C:\WebSphere\AppServer\lib*。

迁移总结

要从 WORF 的早期版本迁移到 WORF V8.2:

1. 将 *lib\worf.jar* 从 V8.2 dxxworf.zip 复制到 *C:\WebSphere\AppServer\lib*，以便替换 worf.jar 文件。dxxworf.zip 位于以下路径: *<DB2 UDB installed location>\samples\java\Websphere\dxxworf.zip*。
2. 对于您部署的每个应用程序，用 apache-services.war 或 axis-services.war 的 worf 目录中的文件替换该应用程序 worf 目录中的 JSP 文件。然后重新部署该应用程序。

相关任务:

- 第 50 页的『在 iSeries 中安装和部署 WORF 示例』
- 第 48 页的『在 Apache Jakarta Tomcat 上安装和部署 WORF 示例』
- 第 36 页的『安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本』

在 Apache Jakarta Tomcat 上安装和部署 WORF 示例

过程:

要安装 WORF 示例，执行下列任务:

1. 将 *apache-services.war 或 axis-services.war* 解压到 *tomcat\webapps* 目录（这决取于安装的 SOAP 引擎）。

   如果已经安装了 *apache-services.war 或 axis-services.war* 文件，则执行下列任务:

   a. 停止 Apache Jakarta Tomcat。

   b. 删除 *webapps* 下的 *services* 子目录及其所有内容。

      注: 此操作将使先前在 “services” web 应用程序中部署的任何 Web 服务丢失，所以确保此操作是可接受的。

   c. 重新启动 Apache Jakarta Tomcat。

2. 停止并启动 Apache Jakarta Tomcat（除非在先前步骤中已删除 *services* 目录）。

   services 上下文启动:

   ```
   ContextManager: Adding context Ctx(\services)
   ```

3. 通过输入以下统一资源定位器（URL）来验证安装。此处由 *8080* 指定的端口号取决于您自己的当前机器:

   ```
   http://localhost:8080/services
   ```

   特定端口地址可能会视环境的不同而有所不同。应显示类似第 39 页的图 5 的页。要了解有关 Web 服务样本页的更多信息，请参阅测试 Web 服务。
4. 验证 group.properties 文件中的数据库设置（特别是用户标识和密码）是否正确。在系统上尝试 verification.dadx（动态测试页和 WSDL）。
5. 要显示“可扩展标记语言”（XML）文档，使用 Internet Explorer V5 或文本编辑器。
6. 在系统中的服务上下文中列示已部署的 SOAP 服务。对于您运行的每个测试，WORF 自动部署服务。从“Web 服务样本页”中单击 SOAP 管理链接。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』

相关任务:

- 第 102 页的『测试 Web 服务』
- 第 46 页的『在 UNIX 和 Windows 上为 Apache Jakarta Tomcat 安装 Web 服务提供程序的软件需求』
- 第 33 页的『安装 Web 服务提供程序的软件需求』

在 iSeries 上为 Apache Jakarta Tomcat 安装 Web 服务提供程序的软件需求

先决条件:

确保安装了必需的软件。需要 DB2 XML Extender 以用于 XML 和关系数据之间的高级映射控制。要使用 DB2 通用数据库 XML Extender，确保安装了该产品。可通过发出 CL 命令 **GO LICPGM** 来验证是否在系统上安装了 DB2 通用数据库 XML Extender。对于 DB2 通用数据库 iSeries 版 V5R2，如果安装了 DB2 通用数据库 XML Extender，则下列条目显示为 **GO LICPGM** 命令的结果:

- 5722DE1 *COMPATIBLE DB2 UDB Extenders
- 5722DE1 *COMPATIBLE DB2 UDB Text Extender
- 5722DE1 *COMPATIBLE DB2 UDB XML Extender
- 5722DE1 *COMPATIBLE Text Search Engine

使用以下 CL 命令启用 DB2 通用数据库 XML Extender: **CALL PGM(QDBXM/QZXMADM) PARM(enable_db LOCALRDB)**。LOCALRDB 是关系数据库目录中的 *LOCAL 数据库名称。要使用关系数据库条目，发出以下 CL 命令: **WRKRDBDIRE**。如果使用样本文件中的文档类型定义文档（DTD），则执行脚本 *setup-dxx.cmd*。

WORF 需要“Java 数据库连接”（JDBC）2.0，它是 DB2 通用数据库版本 8 中的缺省配置。

过程:

准备 WORF 环境的过程如下所示:

1. 如果未在运行 DB2 通用数据库版本 8，则选择 JDBC 2.0。运行 *C:\SQLLIB\java12\usejdbc2.bat*（假定您使用 Windows 环境，并在 C:\SQLLIB\ 中安装了 DB2）。
2. 安装下列因特网软件:
   - 来自 Apache:
     - Apache Jakarta Tomcat V4.0.3 或更新版本的二进制文件，位于 http://jakarta.apache.org/site/binindex.html。（Apache Jakarta Tomcat V4 Standard 提供适当的 Xerces 解析器。对于早期版本，必须将 Xerces 解析器添加至 CLASSPATH 才能将其用作 XML 解析器。）
     - Xerces 1.4.4，位于 http://xml.apache.org/
   - 来自 Sun 公司（http://java.sun.com/products ）:
     - JavaMail 1.2
     - JavaBeans Activation Framework (JAF) 1.0 1
     - j2ee.jar，版本 1.3 或更新版本。
     - qname.jar
   - *wsdl4j.jar*。可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。

相关概念:

- 第 25 页的『将 Web 服务提供程序与 iSeries 配合使用』

相关任务:

- 第 57 页的『在 iSeries 平台中定义 web.xml 和 group.properties 文件』
- 第 50 页的『在 iSeries 中安装和部署 WORF 示例』
- 第 34 页的『在 iSeries 上安装 Web 服务提供程序的软件需求』

在 iSeries 中安装和部署 WORF 示例

过程:

要安装 WORF 示例，执行下列任务:

1. 确保 worf.jar 文件位于

   ```
   /QIBM/UserData/WebASAEs4/worf/lib/app
   ```

   中，其中 **WebASAEs4** 是 WebSphere 的版本，**worf** 是 WebSphere 实例的名称。
2. 如果文件 *runtime.zip* 不在目录 /QIBM/UserData/java400/ext 中，则执行下列命令:
   a. >qsh
   b.
   ```
   >ln -s  /QIBM/ProdData/OS400/Java400/ext/runtime.zip
             /QIBM/UserData/Java400/ext/runtime.zip
   ```
3. 将 *services.war* 复制到 *tomcat\webapps* 目录中。

   如果已经安装了 *services.war* 文件，则执行下列任务:
   a. 停止 Apache Jakarta Tomcat。
   b. 删除 *webapps* 下的 *services* 子目录及其所有内容。

注意:
此操作将使先前在“services”web 应用程序中部署的任何 Web 服务丢失，所以确保此操作是可接受的。

c. 重新启动 Apache Jakarta Tomcat。

4. 停止并启动 Apache Jakarta Tomcat（除非在先前步骤中已删除 *services* 目录）。

   services 上下文启动:

   ```
   ContextManager: Adding context Ctx(\services)
   ```

5. 通过访问位于 http://<system>:<port>/services 的测试页来调用样本应用程序中的示例

   - 不带任何参数调用样本:
     http://<system>:<port>/services/travel/ZipCodes.dadx/findAll
   - 带参数调用样本:

     ```
     http://<system>:<port>/services/
         travel/ZipCodes.dadx/findCityByZipCode?zipcode=55901
     ```

6. 验证 group.properties 中的数据库设置是否正确，尤其是用户标识和密码。如果不为用户标识使用一个值，则 Web 服务代码在 QEJBSVR 下运行。因此，将此概要文件授予想要存取的任何数据库对象。在系统上尝试 verification.dadx（动态测试页和 WSDL）。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』

相关任务:

- 第 49 页的『在 iSeries 上为 Apache Jakarta Tomcat 安装 Web 服务提供程序的软件需求』
- 第 33 页的『安装 Web 服务提供程序的软件需求』

管理 Web 服务提供程序并对其进行故障排除

本章描述 Web 服务提供程序的一些性能建议和故障排除指南。

使用连接池来提高性能

每当资源尝试存取数据库时，它必须连接到该数据库。数据库连接要求资源创建连接、维持连接，并在其不再需要时释放连接。基于 Web 的应用程序所要求的数据库资源可能较高，原因是 Web 用户连接和断开连接较为频繁。可以使用 IBM WebSphere Application Server 来帮助创建和维护数据库连接池。这些数据库连接可以由应用程序服务器上的各应用程序共享，以解决资源问题。连接池可将连接开销在几个用户请求之间分布，从而保存资源以供将来的请求使用，因此提高了性能。可以为每个唯一的数据源配置一个池。可以在 WebSphere handbook 的第 10 章中阅读关于连接池的更多信息。

先决条件:

1. 创建 JDBC 提供程序（如果此程序不存在且需要要使用）。
2. 创建数据源。

过程:

已安装的应用程序使用 JDBC 提供程序从数据库存取数据。要从 WebSphere Application Server V5 为 JDBC 提供程序内的特定数据源调整某些连接池参数，请执行以下步骤:

1. 配置数据源参数。
2. 更新连接池信息，如图 10 中所示。 WebSphere Application Server 提供了 Java 命名服务（JNDI）以便于连接到 DB2 通用数据库。池由连接到同一个数据源的所有应用程序共享。

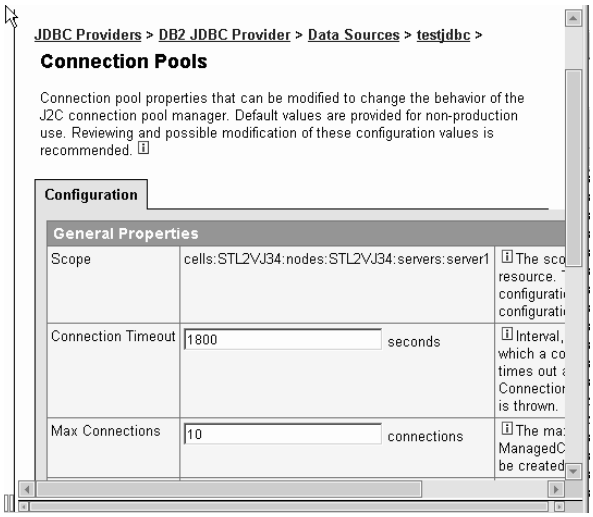


图 10. 调整连接池参数

3. 编辑 groups 子目录中的 *group.properties* 文件并添加下列文本行:

```
initialContextFactory=<your context factory>
datasourceJNDI=<your DataSource>
```

例如:

```
initialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
datasourceJNDI=jdbc/sampleDataSource
```

4. 如果已完成对 group.properties 文件的更改，则重新启动 Web 应用程序以使更改生效。

相关任务:

- 第 54 页的『定义 web.xml 和 group.properties 文件』

对 Web 服务进行故障诊断

表 5 描述了在 WebSphere Application Server 5.1 上使用 WORF 时可能会发生的问题。下表提供了建议解决方案。

表 5. 错误和解决方案

问题	解决方案
错误 500: 服务器从 servlet [isd_demos]: org.apache.soap.rpc 捕获未处理的异常。SOAPContext: 找不到方法 setClassLoader (java\lang\ClassLoader;)	缺少 SOAP 2.2 或更新版本（soap.jar）。

表 5. 错误和解决方案 (续)

问题	解决方案
在 Internet Explorer 中从 Web 服务测试页单击**调用**按钮导致“找不到该页”错误。	要查看更有帮助的错误消息，使用 Netscape 来调试问题。或者，通过执行下列步骤来编辑 Internet Explorer 环境: 1. 从 Internet Explorer 菜单栏中打开**工具**菜单。 2. 从菜单中选择 **Internet 选项**以打开“Internet 选项”窗口。 3. 单击**高级**选项卡。 4. 清除**显示友好 HTTP 错误信息**旁边的复选框。
错误 400: 服务 “http://tempuri.org/***/***.dadx” 未知	必须从 DADX 文件生成部署描述符并在调用该服务之前重新启动 Web 应用程序。
错误 400: 无法获取 DAD; 无法对 xxxxxx 获取输入流	无法使用指定的 XML Extender DAD 文件（例如，在 *.nst 文件中指定的 DAD）
错误 400: 数据库连接错误	• 数据库未启动。 • 在 DADX 文件中引用的数据库对象不存在。 • 找不到 JDBC 驱动程序。
错误 400: 无法对 /groups/xxx/yyy.dadx 获取输入流	DADX 文件未存在于组文件夹中，或者不可使用。
错误 404: 找不到文件: aaaa/abc.dadx	servlet 映射 “aaaa” 未存在于 web.xml 文件中。
空白页结果	如果使用早于 V5.0.2 的 WebSphere Application Server 版本，可能会丢失 `jaas.jar`。打开服务器目录中的 SystemErr.log 以确定是否丢失了其它 JAR 文件。

要从 Web 服务提供程序获取有关运行时事件和诊断的信息以便在部署之后对 Web 服务进行故障诊断，可以使用运行应用程序的 Web 应用程序服务器的跟踪设施。从 Web 应用程序服务器接收到的跟踪信息包括消息和事件活动。即使未启用跟踪，也会在应用程序服务器错误日志中捕获错误。要了解有关如何跟踪 Web 服务提供程序事件的更多信息，请参阅Web 服务提供者跟踪

相关概念:

• 第 128 页的『Web 服务提供程序跟踪』

相关任务:

• 第 124 页的『生成部署描述符』

相关参考:

• 第 31 页的『UNIX 和 Windows 的 Web 服务提供程序软件需求』

开发使用 Web 服务提供程序的应用程序

下列各节描述使用 Web 服务提供程序的概述和详细信息。

定义一组 Web 服务

组是共享公共配置选项的 Web 服务的容器。配置选项可以是下列项:

- 数据库配置

- 名称空间设置
- 消息编码设置

groups 目录包含所有 DADX Web 服务组的资源。WORF Web 应用程序在应用程序配置期间创建此目录。此目录位于 Web 应用程序主目录的 *WEB-INF\classes\groups* 子目录中。

DADX 文件包含 Web 服务的描述。WORF 包含 Web 服务的实现，因而类似于 Java™ 类。*classes* 目录是 Web 应用程序的 Java CLASSPATH 的一部分。这意味着 Java 类装入程序可以装入 DADX 文件。

在 *groups* 目录中，WORF 将每组 DADX Web 服务存储在名称与其 servlet 实例相同的目录中。应用程序服务器按照基于 web.xml 文件的给定的 URL 来寻找要调用的正确的 servlet 实例。

WORF 使用的另一个资源是 group.imports 文件。这是可选的资源，可帮助使用所生成的 WSDL 的各种 Web 服务使用程序或工具查找已使用的模式。如果 group.imports 文件存在，则 WSDL 会根据 group.imports 文件的内容和元素范围来生成导入元素。如果 group.imports 文件不存在，则 WSDL 不会为非动态查询服务生成导入元素。对于动态查询服务，WSDL 包含 db2WebRowSet.xsd 中的某些数据类型。在没有 group.imports 的情况下定义 db2WebRowSet.xsd 的位置，WORF 会假定此模式文件位于缺省位置，如下列示例中所示:

```
http://<server>:<port>/<contextRoot>/db2WebRowSet.xsd
```

在与 DB2® Web 服务有关的示例中，WORF 应用程序将 DADX 文件存储在 *WEB-INF\classes\groups\dxx_sample* 目录、*WEB-INF\classes\groups\dxx_sales_db* 目录以及 *WEB-INF\classes\groups\dxx_travel* 目录中。

相关任务:

- 第 127 页的『准备和创建 Web 归档文件』
- 第 59 页的『定制 group.properties 文件』
- 第 124 页的『生成部署描述符』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』

定义 web.xml 和 group.properties 文件

过程:

要定义一组新的 DADX Web 服务，请完成下列步骤: （也可以使用 WebSphere Studio V5 创建这些文件。有关更多信息，请参阅 WebSphere Studio Information Center（http://publib.boulder.ibm.com/infocenter/wsphelp/index.jsp）。如果要迁移到新版本的 WORF，则确保 web.xml 和 group.properties 文件包含预期用于您的环境的值。

1. 为 DADX 组选择一个能反映应用程序的组名。这些指示信息使用名称 `myapp_group`。
2. 编辑 *WEB-INF* 目录中的 *web.xml* 文件以定义组名。如果要使用符合 Java 2 Enterprise Edition V1.3 的 WebSphere V5 数据源（WebSphere Studio 或 WebSphere Application Server），请确保将 web.xml 文件中的 web-app_2.2.dtd 更改为 web-app_2.3.dtd。

可以在同一个 *web.xml* 文件中有多个组名。下图显示 *web.xml* 文件的示例。servlet-mapping 元素用**粗体**表示，并在下面定义了值。

```
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
<servlet>
        <servlet-name>myapp_group</servlet-name>
        <servlet-class>com.ibm.etools.
          webservice.rt.dxx.servlet.
           DxxInvoker
</servlet-class>
        <init-param id=InitParam_1076524994485>
          <param-name>faultListener</param-name>
          <param-value>
             org.apache.soap.server.DOMFaultListener
</param-value>
        </init-param>
        <init-param id=InitParam_1076524994488>
          <param-name>soap-engine</param-name>
          <param-value>apache-soap</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>myyapp_group</servlet-name>
        <url-pattern>/myapp/*</url-pattern>
</servlet-mapping>
 <welcome-file-list>
       <welcome-file>index.html</welcome-file>
 </welcome-file-list>
</web-app>
```

图 11. web.xml

<servlet> 段为该组定义新的 servlet 实例。对于每个组，至少必须存在一个 <servlet> 元素，但一个组可有多个 <servlet-mapping> 元素。请参阅 Java servlet 规范，网址为: http://java.sun.com/products/servlet/。在此示例中，<servlet-name> 元素定义名为 myapp_group 的组。

当更新此文件时，为下列元素提供信息:

<servlet-name>
这是 <servlet> 段和 <servlet-mapping> 段中的下级标记，定义组的名称。servlet 名必须是 *groups* 目录下有效的目录名称。使用此名称来存储此组 Web 服务的 DADX 资源。例如: myapp_group 是在 <servlet> 和 <servlet-mapping> 元素中定义的。

<servlet-mapping>
必须具有至少一个 <servlet-mapping> 段才能引入 URL 与组之间的映射。下级标记 <servlet-name> 定义组名，并且必须与该组的目录名相同，这也意味着的 <servlet-name> 必须与 <servlet> 组中的名称相同。<servlet-name> 标记是 <servlet> 和 <servlet-mapping> 标记之间的链接。

<url-pattern>
与组相关联的统一资源定位器（URL）。<servlet-mapping> 元素使

dxx_sales_db servlet 与形式为 */url_pattern/** 的 URL 相关联。URL 模式必须具有此形式才能使 WORF 正确操作。例如: /myapp/*。此示例中的 servlet 名是 myapp_group。

<init-param>

可以更新要使用的 SOAP 引擎的名称。用于指定 soap 引擎的参数名是 <soap-engine>。如果要使用 Apache SOAP，则参数值是 *apache-soap*。如果要使用 Apache Axis，则参数值是 *apache-axis*。如果未指定 <soap-engine> 参数，则缺省 soap 引擎是 *apache-soap*。

注: *web.xml* 文件的缺省编码是 UTF-8。当在 OS/390 或 z/OS 平台上时，通过将 *web.xml* 文件发送至 UNIX 或 Windows 系统来以 UTF-8 编码更新文件。可通过使用“文件传输协议”（FTP）二进制文件传输来发送文件。然后更新该文件，并将文件返回到原来的系统。

3. 从 groups 目录中，使用前面步骤中添加的 <servlet-name> 元素中指定的组的名称创建一个子目录。该子目录最终将包含此组的资源。
4. 在组目录中，创建一个 *group.properties* 文件，该文件定义数据库连接信息和每组 DADX Web 服务的其它公共属性。以下是新组的 group.properties 的一个示例:

```
# myapp_group group properties
dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:sample
userID=
password=
namespaceTable=myapp.nst
autoReload=true
reloadIntervalSeconds=5

for Informix, use the following database driver and URL:
dbDriver=com.informix.jdbc.IfxDriver
dbURL=jdbc\:informix-sqli://::informixserver=
For OS/390 and z/OS, use the following database driver and URL:
dbDriver=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver
dbURL=jdbc:db2os390:
```

图 12. group.properties 示例

相关概念:

- 第 108 页的『Web 服务描述语言』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关任务:

- 第 59 页的『定制 group.properties 文件』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 225 页的『检查 web.xml 文件中的错误』

在 iSeries 平台中定义 web.xml 和 group.properties 文件

过程:

要定义一组新的 DADX Web 服务，完成下列步骤:

1. 为 DADX 组选择一个能反映应用程序的组名。这些指示信息使用名称 myapp_group。
2.
   - 对于 iSeries，在 *WEB-INF* 目录中，编辑 *web.xml* 文件以定义组名。可以在同一个 *web.xml* 文件中有多个组名。如果要使用符合 Java 2 Enterprise Edition V1.3 的 WebSphere V5 数据源（WebSphere Studio 或 WebSphere Application Server），请确保将 web.xml 文件中的 web-app_2.2.dtd 更改为 web-app_2.3.dtd。

     样本应用程序包括将样式表应用于生成的“可扩展标记语言”（XML）的 servlet。该 servlet 类和源在以下目录中:

     ```
     /QIBM/UserData/WebASAEs4/worf/
     installedApps/servicesApp.ear/
     services.war/WEB-INF/classes
     ```

     样本文件包含以下设置: serveServletsByClassnameEnabled=″true″。通过执行以下文件调用 servlet:

     ```
     /QIBM/UserData/WebASAEs4/worf/
     installedApps/servicesApp.ear/
     services.war/WEB-INF/ibm-web-ext.xmi
     ```

     可在 qshell 中通过执行下列编译语句来重新编译 servlet:

     ```
     javac -J-Djava.ext.dirs=/qibm/proddata/webasaes4/lib -d . SampleXSLTServlet.java
     ```

     通过执行以下文件来从因特网上调用 servlet:

     ```
     http://<system>:<port>/services/servlet/
     SampleXSLTServlet?XML=
     http://<system>:<port>/services/travel/ZipCodes.dadx/
        findAll&XSL=
     file:///home/zipcodes.xsl
     ```

     上面的示例假定 *zipcodes.xsl* 位于 /home 目录中。可以将该文件放置在任何位置。可以将此 servlet 示例用于 XML Web 服务和“可扩展样式表语言”（XSL）样式表的任何组合。

     下图显示 *web.xml* 文件的示例。servlet-mapping 元素用**粗体**表示，并在下面定义了值。

```
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
<servlet>
        <servlet-name>myapp_group</servlet-name>
        <servlet-class>com.ibm.etools.
webservice.rt.dxx.servlet.DxxInvoker</servlet-class>
        <init-param>
          <param-name>faultListener</param-name>
          <param-value>
            org.apache.soap.server.DOMFaultListener
</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>myyapp_group</servlet-name>
        <url-pattern>/myapp/*</url-pattern>
    </servlet-mapping>
 <welcome-file-list>
       <welcome-file>index.html</welcome-file>
 </welcome-file-list>
</web-app>
```

图 13. web.xml

<servlet> 段为该组定义新的 servlet 实例。对于每个组，至少必须存在一个 <servlet> 元素，但一个组可有多个 <servlet-mapping> 元素。请参阅 Java servlet 规范，网址为: http://java.sun.com/products/servlet/。在此示例中，<servlet-name> 元素定义名为 myapp_group 的组。

当更新此文件时，为下列元素提供信息:

<servlet-name>

这是 <servlet> 段和 <servlet-mapping> 段中的下级标记，定义组的名称。servlet 名必须是 *groups* 目录下有效的目录名称。使用此名称来存储此组 Web 服务的 DADX 资源。例如: myapp_group 是在 <servlet> 和 <servlet-mapping> 元素中定义的。

<servlet-mapping>

必须具有至少一个 <servlet-mapping> 段才能引入 URL 与组之间的映射。下级标记 <servlet-name> 定义组名，并且必须与该组的目录名相同，这也意味着的 <servlet-name> 必须与 <servlet> 组中的名称相同。<servlet-name> 标记是 <servlet> 和 <servlet-mapping> 标记之间的链接。

<url-pattern>

与组相关联的统一资源定位器（URL）。<servlet-mapping> 元素使 dxx_sales_db servlet 与形式为 */url_pattern/** 的 URL 相关联。URL 模式必须具有此形式才能使 WORF 正确操作。例如: /myapp/*。此示例中的 servlet 名是 myapp_group。

<init-param>

可以更新要使用的 SOAP 引擎的名称。用于指定 soap 引擎的参数名是 <soap-engine>。如果要使用 Apache SOAP，则参数值是 *apache-soap*。如果要使用 Apache Axis，则参数值是 *apache-axis*。如果未指定 <soap-engine> 参数，则缺省 soap 引擎是 *apache-axis*。

3. 从 groups 目录中，使用前面步骤中添加的 <servlet-name> 元素中指定的组的名称创建一个子目录。该子目录包含此组的资源。
4. 在 group 目录中，创建一个 *group.properties* 文件，该文件定义数据库连接信息和每组 DADX Web 服务的其它公共属性。以下是新组的 group.properties 的一个示例:

```
# myapp_group group properties
dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:*local/SAMPLE
userID=
password=
namespaceTable=myapp.nst
autoReload=true
reloadIntervalSeconds=5
```

在第 59 页的图 14 中的示例中，在 dbURL 参数中使用的 **SAMPLE** 是您想要使用的数据库或集合。

图 14. group.properties 示例

相关任务:

- 第 59 页的『定制 group.properties 文件』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』

定制 group.properties 文件

group.properties 文件是标准 Java 属性文件。必须至少使用下列参数之一来定义 group.properties。

当将 group.properties 用于连接池时
使用带有 datasourceJNDI 的参数 initialContextFactory 来定义 group.properties。

当将 group.properties 用于常规 JDBC 数据库连接时
使用带有 dbDriver 的参数 dbURL 来定义 group.properties。

如果定义这两种类型的连接，则应用程序先尝试使用 DataSource。如果 WORF 不能获取 DataSource，则尝试使用 JDBC。下面列示了完整的属性集合。

过程:

要修改 group.properties 文件，使用环境的下列定义:

数据库配置参数

initialContextFactory
此参数与 datasourceJNDI 配合使用，且对 WebSphere 连接池是必需的。该参数指定 JNDI 初始上下文工厂（用来定位数据库的 DataSource）的 Java 类名。此属性与 datasourceJNDI 属性一起启用连接池。

datasourceJNDI
此参数与 initialContextFactory 配合使用，且对 WebSphere 连接池是必需的。此参数指定数据库的 DataSource 的 JNDI 名。当将此参数

与 initialContextFactory 结合使用时，它定义数据库连接的 DataSource。此属性启用连接池。必须定义 DataSource 或“Java 数据库连接”（JDBC）连接。

dbDriver

此参数指定用于连接至数据库的“Java 数据库连接”（JDBC）驱动程序的 Java 类名。

dbURL

此参数与 dbDriver 配合使用，并指定数据库的 JDBC 统一资源定位器（URL）。

userID

此参数是可选的。缺省值是 WORF 执行所使用的用户标识，该用户标识可以与用于连接至数据库的用户标识相同。此参数为数据库指定用户标识。

password

此参数是可选的，但要与用户标识结合使用。此参数为数据库指定密码。提供了算法以帮助您对您的密码进行编码和译码。

enableXmlClob

此参数是可选的。此参数指定 retrieveXML 操作是否将使用基于 CLOB 的 XML Extender 存储过程。缺省值为 *true*。此参数仅可用于向下兼容性。对于 OS/390 和 z/OS 平台，要么不定义此属性，要么总是将该值设置为 *true*。

Web 服务配置参数

groupNamespaceUri

此参数是可选的。此参数定义在已生成的“Web 服务描述语言”（WSDL）和“可扩展标记语言”（XML）模式文件（XSD）中使用的目标名称空间。该目标名称空间用于此组中的 Web 服务。

useDocumentStyle

此参数是可选的。缺省值为 *false*，这意味着运行时的 Web 服务使用 RPC 编码。如果将此值设置为 true，则运行时的 Web 服务使用文档样式和文字编码。IBM DB2 Information Integrator Web 服务提供程序包含设置为使用 RPC 样式的样本。对于新的应用程序，应该使用文档样式以获得最大互操作性。

namespaceTable

此参数是可选的。此参数指定名称空间表的资源名称。它引用“名称空间表”（NST）资源，该资源定义从 DB2 XML Extender DTDID 至“XML 模式”（XSD）名称空间和位置的映射。有关 NST 文件的示例，请参阅第 77 页的图 24。

运行时配置参数:

autoReload

此参数是可选的，但要与 reloadIntervalSeconds 结合使用。此参数指定是否要重新装入资源。值可以为 true 或 false。缺省值为 false。

reloadIntervalSeconds

此参数是可选的，但要与 autoReload 结合使用。此参数控制资源装入

和高速缓存。它指定整数的自动重新装入时间间隔（以秒计）。缺省值为 0，这意味着 WORF 将对每个请求检查更新的资源。

选项 *autoReload* 和 *reloadIntervalSeconds* 控制资源装入和高速缓存。如果缺少 autoReload 或它的值为 false，则不会重新装入的资源，并且应用程序忽略 reloadIntervalSeconds。如果 autoReload 为 true，则当 WORF 存取资源（如以下文件之一：DAD、DADX、文档类型定义（DTD）和 NST）时，它会将当前时间与先前装入资源的时间进行比较。如果已过去的时间超过了 `reloadIntervalSeconds` 的值，则 WORF 检查文件系统以查找更新版本并重新装入已更改的资源。自动重新装入在开发时很有用，在这种情况下将 `reloadIntervalSeconds` 设置为零。如果 Web 服务处于运营过程，则将 `autoReload` 设置为 false 或将 `reloadIntervalSeconds` 设置为很大的值以避免影响服务器性能。

相关概念:

- 第 108 页的『Web 服务描述语言』
- 第 53 页的『定义一组 Web 服务』

相关任务:

- 第 57 页的『在 iSeries 平台中定义 web.xml 和 group.properties 文件』
- 第 54 页的『定义 web.xml 和 group.properties 文件』

DADX 文件

本节描述 DADX 文件的属性、语法和操作。

使用文档存取定义扩展文件来定义 Web 服务

“文档存取定义扩展”（DADX）文件指定 Web 服务。这是通过使用定义一组操作的一些 SQL 语句、一系列参数以及一些“文档存取定义”（DAD）文件引用（可选）来完成的。可以用仅包含动态查询服务标记（<DQS>）的 DADX 文件定义一组动态 Web 服务操作。这些操作与可以调用的方法相似。可根据下列操作类型来定义 DADX Web 服务中的操作:

- SQL 操作（非动态）

  <query>
  通过使用选择操作来查询数据库

  <update>
  对数据库执行更新、插入或删除操作

  <call> 调用存储过程

- SQL 操作（动态查询服务）

  <getTables>
  检索可用表的描述。

  <getColumns>
  检索列的描述。

  <executeQuery>
  发出单个 SQL 语句。

  <executeUpdate>
  发出单个 INSERT、UPDATE 和 DELETE。

<executeCall>
调用单个存储过程。

<execute>
发出单个 SQL 语句。

- “可扩展标记语言”（XML）集合操作（需要 DB2® XML Extender）

<retrieveXML>
生成 XML 文档

<storeXML>
存储 XML 文档

相关概念:

- 第 28 页的『与 DADX 文件配合使用的 Web 服务提供程序操作』

相关参考:

- 第 62 页的『DADX 文件的语法』

DADX 文件的语法

DADX 文件是“可扩展标记语言”（XML）文档。第 63 页的 『DADX 语法定义』和第 63 页的图 15 描述了 DADX 的元素。DADX 模式在 DADX 文件的 XML 模式中。第 63 页的『DADX 语法定义』中节点和元素旁边的数字标识子分组。编号方案表示 XML 文档层次结构。例如，当标识从 1.3（result_set_metadata）更改为 1.3.1（column）时，意味着该 column 是 result_set_metadata 的子代。从 1.1（documentation）更改为 1.2（implements）意味着这些元素是兄弟元素。

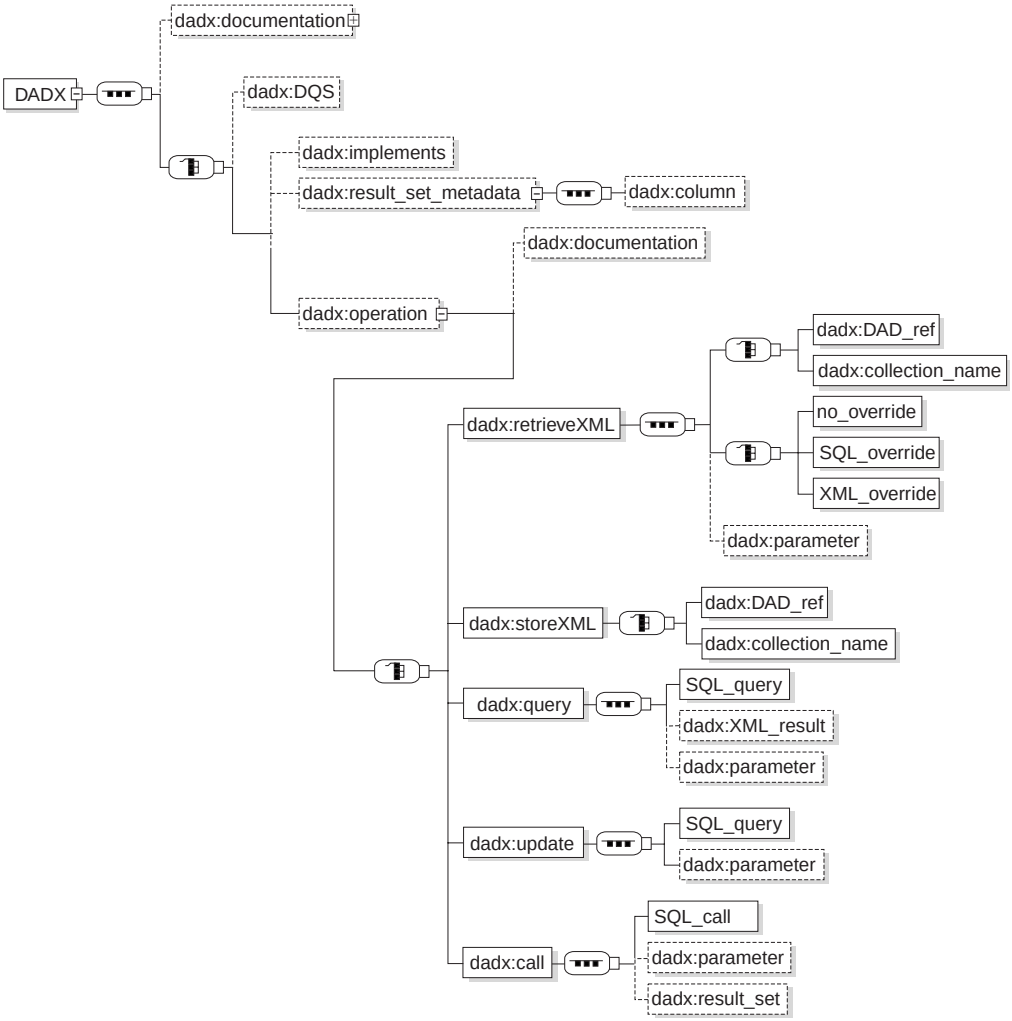


图 15. DADX 语法

0. 根元素: <DADX>

属性:

xmlns:dadx

DADX 的名称空间。

xmlns:xsd

“可扩展标记语言（XML）模式”规范的名称空间。

子代:

0.1 <documentation>

指定有关 Web 服务的用途和内容的注释或语句。可使用 XHTML 标记。

1. 指定非动态操作的 DADX 函数

1.2 <implements>

指定 Web 服务描述文件的名称空间和位置。它允许服务实现者声明 DADX Web 服务实现别处定义的可重用 WSDL 文档描述的标准 Web 服务; 例如在 UDDI 注册中心中。

1.3 <result_set_metadata>

存储过程可返回一个或多个结果集。可在输出消息中包括这些结果集。必须在非动态的 DADX 中使用 <result_set_metadata> 元素显式定义存储过程结果集的元数据。在运行时，您获取结果集的元数据。元数据必须与包含在 DADX 文件中的定义相匹配。

注: 您只能调用具有结果集（包含固定元数据）的存储过程。此限制是必需的，以对 Web 服务提供严格定义的 WSDL 文件。单结果集元数据定义可由若干 <call> 操作通过使用 <result_set> 元素来引用。结果集元数据定义对于 DADX 是全局的，并且必须在所有操作定义元素的前面。

属性:

name 标识结果集的根元素。

rowname
用作结果集每一行的元素名称。

子代:

1.3.1 <column>

定义列。列的次序必须与存储过程返回的结果集的次序相匹配。每个列都具有名称、类型和可空性，这些必须与结果集相匹配。

属性:

name 必需的。它指定列的名称。

type 如果不指定 **element**，则是必需的。它指定列的类型。

element
如果不指定 **type**，则是必需的。它指定列的元素。

as 可选。它为列提供名称。

nullable
可选。Nullable 是 true 或 false。它指示列值是否可为空。

1.4 <operation>

指定 Web 服务操作。操作元素及其子代指定操作的名称和 Web 服务执行的操作类型。Web 服务可编写 XML 文档、查询数据库或调用存储过程。单个 DADX 文件可包含对单个数据库或位置的多个操作。下表描述这些元素。

- 属性:

  name 标识操作的唯一字符串。该字符串在 DADX 文件中必须是唯一的。例如: `"findByColorAndMinPrice"`。

- 子代:

使用以下元素说明操作:

1.4.1 <dadx:documentation>

指定有关操作的用途和内容的注释或语句。可使用 XHTML 标记。

1.4.2 <retrieveXML>

此元素指定在使用 XML 集合存取方法时从一组关系表中生成零个或一个 XML 文档。根据您指定的是 DAD 文件还是 XML 集合名，操作调用适当的 XML Extender 组合存储过程。

子代:

- 指定想要使用这些存储过程中的哪一个。使用下列元素之一，可通过传递 DAD 文件的名称或集合的名称来执行此操作:

  ##### 1.4.2.1 <DAD_ref>

  此元素的内容是 DAD 文件的名称和路径。如果您为 DAD 文件指定相对路径，则应用程序假设当前工作目录是组目录。

  ##### 1.4.2.2 <collection_name>

  此元素的内容是 XML 集合的名称。通过使用 XML Extender 管理接口定义集合，如 *DB2 XML Extender Administration and Programming* 中所述。

- 使用下列元素之一指定覆盖值:

  ##### 1.4.2.3 <no_override/>

  指定不覆盖 DAD 文件中的值。如果不指定 <SQL_override> 或 <XML_override>，则是必需的。

  ##### 1.4.2.4 <SQL_override>

  指定覆盖 DAD 文件中使用 SQL 映射的 SQL 语句。

  ##### 1.4.2.5 <XML_override>

  指定覆盖 DAD 文件中使用 RDB 映射的 XML 条件。

- 通过使用以下元素定义参数:

  ##### 1.4.2.6 <parameter>

  当在 <SQL_override> 或 <XML_override> 元素中引用参数时是必需的。此元素为操作指定参数。将独立参数元素用于在操作中引用的每个参数。每个参数名在操作中必须是唯一的。参数必须使用 XML Schema 元素（复杂类型）或简单类型定义其内容。

属性:

name 参数的唯一名称。

element
使用“element”属性来指定 XML Schema 元素。

type 使用“type”属性来指定简单类型。

kind 指定参数是传递输入数据、返回输出数据还是同时执行两个操作。此属性的有效值有:

- in

1.4.3 <storeXML>

此元素指定使用 XML 集合存取方法在一组关系表中存储（分解）XML 文档。根据您指定的是 DAD 文件还是 XML 集合名，操作调用适当的 XML Extender 分解存储过程。子代:

– 指定想要使用这些存储过程中的哪一个。使用下列元素之一，可通过传递 DAD 文件的名称或集合的名称来执行此操作:

1.4.3.1 <DAD_ref>
此元素的内容是 DAD 文件的名称和路径。如果您为 DAD 文件指定相对路径，则应用程序假设当前工作目录是组目录。

1.4.3.2 <collection_name>
此元素的内容是 XML 集合的名称。通过使用 XML Extender 管理接口定义集合，如 *DB2 XML Extender Administration and Programming* 中所述。

1.4.4 <query>
指定查询操作。通过在 <SQL_select> 元素中使用 SQL SELECT 语句来定义操作。语句可不含或含有多个已命名的输入参数。如果语句具有输入参数，则每个参数由一个 <parameter> 元素描述。

此操作将每个数据库列从结果集映射至相应的 XML 元素。可在 <query> 操作中指定 XML Extender 用户定义的类型（UDT）。但是，这需要 <XML_result> 元素及支持的文档类型定义（DTD，它定义查询的 XML 列的类型）。

子代:

1.4.4.1 <SQL_query>
指定 SQL SELECT 语句。

1.4.4.2 <XML_result>

可选。这定义包含 XML 文档的已命名列。子代的根的 XML Schema 元素必须定义文档类型。

属性:

name 指定存储在列中的 XML 文档的根元素。

element
指定列中的特定元素。

1.4.4.3 <parameter>

当在 <SQL_query> 元素中引用参数时是必需的。它为操作指定参数。将独立参数元素用于在操作中引用的每个参数。每个参数名在操作中必须是唯一的。参数必须使用下列之一定义其内容: XML Schema 元素（复杂类型）或简单类型。

属性:

name 参数的唯一名称。

element
使用 “element” 属性来指定 XML Schema 元素。

type 使用 “type” 属性来指定简单类型。

kind 指定参数是传递输入数据、返回输出数据还是同时执行两个操作。此属性的有效值有:

– in

1.4.5 <update>

操作是在 <SQL_update> 元素中由 SQL INSERT、DELETE 或 UPDATE 语句定义的。语句可不含或含有多个已命名的输入参数。如果语句具有输入参数，则每个参数由一个 <parameter> 元素描述。

子代:

1.4.5.1 <SQL_update>

这指定 SQL INSERT、UPDATE 或 DELETE 语句。

1.4.5.2 <parameter>

当在 <SQL_update> 元素中引用参数时是必需的。它为操作指定参数。将独立参数元素用于在操作中引用的每个参数。每个参数名对于操作必须是唯一的。参数必须使

用下列之一定义其内容: XML Schema 元素（复杂类型）或简单类型。

属性:

name 参数的唯一名称。

element
使用 “element” 属性来指定 XML Schema 元素。

type 使用 “type” 属性来指定简单类型。

kind 指定参数是传递输入数据、返回输出数据还是同时执行两个操作。此属性的有效值有:

- in

1.4.6 <call>

指定对存储过程的调用。该处理类似更新操作，但调用操作的参数可以定义为 “in”、“out” 或 “in/out”。缺省参数类型为 “in”。“out” 和 “in/out” 参数出现在输出消息中。

1.4.6.1 <SQL_call>

指定存储过程调用。

1.4.6.2 <parameter>

当在 <SQL_call> 元素中引用参数时是必需的。它为操作指定参数。将独立参数元素用于在操作中引用的每个参数。每个参数名在操作中必须是唯一的。参数必须使用下列之一定义其内容: XML Schema 元素（复杂类型）或简单类型。

属性:

name 参数的唯一名称。

element
使用 “element” 属性来指定 XML Schema 元素。

type 使用 “type” 属性来指定简单类型。

kind 指定参数是传递输入数据、返回输出数据还是同时执行两个操作。此属性的有效值有:

- in
- out
- in/out

1.4.6.3 <result_set>

它定义结果集并且必须跟在任何

<parameter> 元素之后。结果集元素具有在操作的所有参数和结果集中必须唯一的名称。它必须引用 <result_set_metadata> 元素。必须为从存储过程返回的每个结果集定义一个 <result_set> 元素。

属性:

name SOAP 响应中结果集的唯一标识。

metadata
DADX 文件中的结果集元数据定义。该标识必须引用元素的名称。

2. **<DQS>**
   动态查询服务。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 27 页的『DADX 文件的定义』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』
- 第 231 页的附录 C, 『DADX 文件的 XML 模式』

一个简单的 DADX 文件

以下示例是一个简单 DADX 文件，该文件包含一个带有 SQL 查询的操作。此 DADX 文件用于非动态查询。有关用于启用动态查询服务的 DADX 文件的示例，请参阅配置并运行 Web 服务提供者中的动态数据库查询。

图 16 显示了定义简单 Web 服务的 DADX 文件:

```
<?xml version="1.0" encoding="UTF-8"?>
<DADX
  xmlns="http://schemas.ibm.com/db2/dxx/dadx"
  >
    <documentation>
    Simple DADX example that accesses the SAMPLE database.
  </documentation>
  <operation name="listDepartments">
    <documentation>
      Lists the departments.
    </documentation>
    <query>
      <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
    </query>
  </operation>
</DADX>
```

图 16. 简单的 DADX 文件

这个简单的 DADX 文件定义具有名为 `listDepartments` 的单个操作的 Web 服务，该操作列示 DEPARTMENT 表的内容。操作名称标识 Web 服务活动，类似编程语言中的方法名。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 27 页的『DADX 文件的定义』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关任务:

- 第 85 页的『将动态数据库查询作为 Web 服务提供程序的一部分来配置和运行』

XML 集合操作

可使用 <retrieveXML> 或 <storeXML> 操作来生成或存储 XML 文档。这些操作调用 XML Extender 存储过程并需要 DAD 文件或 XML 集合引用。这些存储过程通过在 DAD 文件中使用映射或引用启用的 XML 集合来生成或存储 XML 文档。要了解如何创建 DAD 文件，请参阅 *DB2 XML Extender Administration and Programming*。

以下示例显示一个较复杂的 DADX 文件，它从 DAD 文件生成 XML 文档。它通过使用 <RetrieveXML> 元素来引用存储过程。<DAD_ref> 元素指定 DAD 文件的名称。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation>
   Provides queries for part order information at myco.com.
   See
   <xhtml:a href="../documentation/PartOrders.html"
      target="_top">PartOrders.html
  </xhtml:a>
   for more information.
  </documentation>
  <operation name="findAll">
    <documentation>
      Returns all the orders with their complete details.
    </documentation>
    <retrieveXML>
       <DAD_ref>getstart_xcollection.dad</DAD_ref>
       <no_override/>
    </retrieveXML>
 </operation>
</DADX>
```

图 17. 生成 XML 文档的 DADX 文件

从此 DADX 文件生成的 Web 服务调用 dxxGenXML 存储过程并生成 XML 文档。该存储过程引用 getstart_xcollection.dad 文件以确定生成 XML 文档和 XML 文档结构时要使用哪些表。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 102 页的『测试 Web 服务应用程序 - 方案』

相关参考:

- 第 74 页的『DADX 操作示例』

在 DADX 文件中使用覆盖

通过使用 <XML_override> 和 <SQL_override> 元素，DADX 文件可以覆盖 DAD 文件中的“可扩展标记语言”（XML）值和 SQL 语句。覆盖的类型取决于 DAD 文件是使用 SQL 映射还是 RDB 映射。如果不需要覆盖 DAD 值，则使用 <no_override/> 元素，如第 70 页的图 17 中所示。

以下示例使用 SQL 覆盖语句。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation >
    Provides queries for part order information at myco.com.
    See <xhtml:a href="../documentation/PartOrders.html" target="_top">
         PartOrders.html</xhtml:a> for more information.
  </documentation>
  <operation name="findAll">
    <documentation >
      Returns all the orders with their complete details.
    </documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad</DAD_ref>
        <SQL_override>
           select o.order_key, customer_name, customer_email,
            p.part_key, color, quantity, price, tax, ship_id, date, mode
           from order_tab o, part_tab p,
             table(select substr(char(timestamp(generate_unique())),16)
               as ship_id, date, mode, part_key from ship_tab) s
           where p.order_key = o.order_key and s.part_key = p.part_key
              order by order_key, part_key, ship_id
        </SQL_override>
   </retrieveXML>
  </operation>
</DADX>
```

图 18. 使用 SQL 覆盖生成 XML 文档的 DADX 文件的示例

虽然您可以覆盖 SQL 语句，但是新的 SQL 语句必须产生与 DAD 文件中定义的 SQL 映射兼容的结果集。例如，出现在 DAD 文件中的列名也必须出现在 SQL 覆盖中。

如果 DAD 文件使用 RDB 节点映射，则必须使用 <XML_override> 元素来覆盖 RDB 节点。RDB 节点元素定义要包含 XML 数据的 DB2 通用数据库表、列和条件。第 72 页的图 19 中的示例显示了引用 RDB 节点 DAD 文件的 DADX 文件。<XML_override> 元素内容覆盖在 DAD 文件中指定的条件。覆盖字符串可包含使用主机变量语法的输入参数。必须定义在此操作中唯一命名的参数元素列表中所有参数的名称和类型。在此示例中，覆盖参数将价格限制为高于 $50.00 并将日期限制为迟于 1998-12-01 来覆盖查询。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation >
    Provides queries for part order information at myco.com.
    See <xhtml:a href="../documentation/PartOrders.html" target="_top">
         PartOrders.html</xhtml:a> for more information.
  </documentation>
  <operation name="findByExtendedPriceAndShipDate">
    <documentation >
      Returns all the orders with an extended price greater than $50.00
      and a ship date later than 1998-12-01.
    </documentation>
<retrieveXML>
      <DAD_ref>order_rdb.dad</DAD_ref>
        <XML_override>
           /Order/Part/ExtendedPrice > 50.00 AND
           Order/Part/Shipment/ShipDate > '1998-12-01'
       </XML_override>
    </retrieveXML>
</operation>
</DADX>
```

图 19. 使用 XML 覆盖生成 XML 文档的 DADX 文件的示例。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』

相关参考:

- 第 74 页的『DADX 操作示例』
- 第 69 页的『一个简单的 DADX 文件』

声明和引用 DADX 文件中的参数

可以在每个操作中使用参数。SQL 操作的 <SQL_query>、<SQL_update> 和 <SQL_call> 语句可以引用参数。在 <retrieveXML> 和 <storeXML> 操作中使用的“可扩展标记语言”（XML）和 SQL 覆盖也可以引用参数。通过使用 <parameter> 元素来声明参数。参数具有与内置 SQL 数据类型对应的简单“XML 模式”类型。表 6 描述了支持的类型。

表 6. 支持的 XML 模式和 SQL 类型

XML 模式简单类型	SQL 类型
字符串型	CHAR、VARCHAR、CLOB 和 LONGVARCHAR
十进制型	DECIMAL 和 NUMERIC
整型	INTEGER
短型	SMALLINT
浮点型	FLOAT
双精度型	REAL 和 DOUBLE PRECISION
日期型	DATE
时间型	TIME
时间戳记型	TIMESTAMP

表 6. 支持的 XML 模式和 SQL 类型 (续)

XML 模式简单类型	SQL 类型
长整型	BIGINT
字节型	TINYINT

要引用参数，使用冒号前缀。例如:

```
<SQL_query>
 select * from order_tab where customer_name =:customer_name
</SQL_query>
```

要定义参数，使用 <parameter> 元素，如以下示例中所示:

```
<parameter name="customer_name" type="xsd:string"/>
```

必须使用 <parameter> 元素定义引用的每个参数。此元素的名称属性标识参数，在操作中必须是唯一的。

第 73 页的图 20 中的示例显示了通过使用 SQL SELECT 语句检索一组关系数据的查询操作。通过使用参数语法，该语句包含一个输入参数。

```
<operation name="findCustomerOrders">
    <documentation>Returns all the orders for a given customer.
    </documentation>
    <query>
      <SQL_query>select * from order_tab where customer_name =
        :customer_name</SQL_query>
      <parameter name="customer_name" type="xsd:string"/>
    </query>
  </operation>
```

图 20. 具有参数的查询操作

第 74 页的图 21 中的示例显示了 retrieveXML 操作使用的 SQL 覆盖中的参数:

```
<operation name="findByColorAndMinPrice">
    <documentation>Returns all the orders that have the specified color and
       at least the specified minimum price.
    </documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad
</DAD_ref>
      <SQL_override>
       select o.order_key, customer_name, customer_email,
         p.part_key, color, quantity, price, tax, ship_id, date, mode
       from order_tab o, part_tab p,
         table(select substr(char(timestamp(generate_unique())),16)
           as ship_id, date, mode, part_key from ship_tab) s
       where p.order_key = o.order_key and s.part_key = p.part_key
         and color = :color and price >= :minprice
       order by order_key, part_key, ship_id
      </SQL_override>
      <parameter name="color" type="xsd:string">
      <parameter name="minprice" type="xsd:decimal">
    </retrieveXML>
  </operation>
```

图 21. retrieveXML 操作使用的 SQL 覆盖

可以修改 SQL 语句的 WHERE 子句以包括搜索条件。SQL 覆盖可以包括使用冒号标识的一个或多个参数。在此示例中，findByColorAndMinPrice 引用 :color 和 :minprice。使用 <parameter> 元素声明参数。参数具有与内置 SQL 数据类型对应的简单 XML 模式文件（XSD）类型。

相关概念:

- 第 112 页的『XML 模式定义』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』
- 第 62 页的『DADX 文件的语法』
- 第 231 页的附录 C, 『DADX 文件的 XML 模式』

DADX 操作示例

以下示例显示具有各种操作的 DADX 文件。

示例 1: Query 操作

此示例显示使用缺省标记的 Query 操作。此示例不需要 XML Extender。此操作选择给定客户的所有订单。要运行此样本，需要 sales_db XML Extender 样本数据库。

```
<?xml version="1.0"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
    </documentation>
  <implements namespace="http://www.poia.org/part_orders.wsdl"
       location="http://www.poia.org/part_orders.wsdl"/>
<operation name="findCustomerOrders">
    <documentation>Returns all the orders for a given customer.
    </documentation>
    <query>
      <SQL_query>select * from order_tab
                  where customer_name = :customer_name
      </SQL_query>
      <parameter name="customer_name" type="xsd:string"/>
    </query>
  </operation>
```

图 22. 具有 Query 操作的 DADX

在此操作中唯一命名的一系列参数元素必须定义输入参数。如果需要对映射进行更多的控制，则可以使用 DAD 文件。

可以使用 *Query* 操作来使用 XML Extender 用户定义的类型（UDT）和用户定义的函数（UDF）。此操作允许您查询、抽取和更新包含 XML 文档的 XML 列中的数据。这些 XML 文档要求您创建定义 <XML_result> 元素的类型的文档类型定义（DTD）。此元素指定列名和包含在列中的 XML 文档的根元素。

第 76 页的图 23 中的示例显示了一个 Query 操作，它使用 <XML_result> 元素声明的 VARCHAR UDT。retrieveOrders 操作通过使用 UDF db2xml.varchar 检索来自 SALES_TAB 表的所有 XML 订单文档。通过使用 XML Extender UDT XMLVARCHAR 存储文档。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:dtd1="http://schemas.myco.com/sales/order.dtd">
  <documentation>
       Queries part orders at myco.com.
  </documentation>

<operation name="retrieveOrders">
    <documentation>
        Retrieves all the Order documents.
    </documentation>
    <query>
      <SQL_query>
        select db2xml.varchar(order) from sales_tab
      </SQL_query>
      <XML_result name="ORDER" element="dtd1:Order"/>
    </query>
  </operation>
</DADX>
```

图 23. 使用 UDF 和 UDT 的 Query 操作

当在列中具有 XML 文档并且想要 WSDL 引用此文档的类型时，可以使用 XML_result 标记。第 76 页的图 23 中的示例指定 ORDER 列包含元素 *dtd1:Order* 的片段。元素 <XML_result name = "ORDER" element = "dtd1:Order"/> 引用名称空间声明。XML Extender 存储没有名称空间且由 DTD 定义的 XML 文档。Web 服务使用 XML 模式（XSD）而不是 DTD，并使用名称空间。通过在名称空间表中生成条目来使名称空间与 DTD 相关联。WORF 在检索 XML 文档时添加名称空间，在存储文档时除去名称空间。WORF 还自动将 DTD 转换为 XSD。行 <XML_result name = "ORDER" element = "dtd1:Order"/> 定义文件 *order.dtd* 中的列信息。以下示例显示了它引用的特定声明:

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
...
```

要指向 DTD，使用名称空间表（NST）文件。将第 77 页的图 24 作为示例进行参考。

```
<?xml version="1.0"?>
  <namespaceTable xmlns="http://schemas.ibm.com/db2/dxx/nst">
    <mapping dtdid="c:\dxx\samples\dtd\getstart.dtd"
      namespace="http://schemas.ibm.com/db2/dxx/samples/dtd/getstart.dtd"
      location="/dxx/samples/dtd/getstart.dtd/XSD"/>
    <mapping dtdid="getstart.dtd"
      namespace="http://schemas.myco.com/sales/getstart.dtd"
      location="/getstart.dtd/XSD"/>
    <mapping dtdid="order.dtd"
      namespace="http://schemas.myco.com/sales/order.dtd"
      location="/order.dtd/XSD"/>
  </namespaceTable>
```

图 24. NST 文件

必须在 *group.properties* 文件中引用此文件。请参阅第 56 页的图 12 中的示例以了解有关此文件的更多信息。

示例 2: Update 操作

第 77 页的图 25 中的示例显示了更新给定订单的客户电子邮件（e-mail）地址的操作。Update 操作可以在 <SQL_update> 元素中包含 SQL INSERT、DELETE 或 UPDATE 语句。

```
  <operation name="updateOrderEmail">
    <documentation>Updates the email address for an order.
     </documentation>
    <update>
      <SQL_update>update order_tab set customer_email = :email
                  where order_key = :key</SQL_update>
      <parameter name="key" type="xsd:int"/>
      <parameter name="email" type="xsd:string"/>
    </update>
   </operation>
 </DADX>
```

图 25. Update 操作

示例 3: Call 操作

这些示例显示调用存储过程的 Call 操作。

如果存储过程返回结果集，则必须在 DADX 文件中的 result_set_metadata 标记中定义这些结果集。这将让 WORF 为此 Web 服务操作生成 WSDL 和 XML 模式文件（XSD）。第 78 页的图 26 显示引用了两次的结果集元数据的定义。

```
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<result_set_metadata name="employeeSalaryReport" rowName="employee">
  <column name="NAME" type="VARCHAR" nullable="true" />
  <column name="JOB" type="CHAR" nullable="true" />
  <column name="3" as="SALARY" type="DOUBLE" nullable="true" />
</result_set_metadata>
<operation name="twoResultSets">
    <call>
<SQL_call>CALL TWO_RESULT_SETS (:salary, :sqlCode)
      </SQL_call>
  <parameter name="salary" type="xsd:double" kind="in" />
  <parameter name="sqlCode" type="xsd:int" kind="out" />
  <result_set name="employees1" metadata="employeeSalaryReport" />
  <result_set name="employees2" metadata="employeeSalaryReport" />
    </call>
   </operation>
 </DADX>
```

图 26. 引用了两次的结果集元数据的定义

还可以通过使用第 78 页的图 27 中所示的格式调用存储过程。

```
<operation name="callProc1">
    <documentation>Call the Proc1 stored procedure.
     </documentation>
    <call>
      <SQL_call>
        CALL Proc1 (:x, :y, :z)
      </SQL_call>
      <parameter name="x" type="xsd:string" kind="in"/>
      <parameter name="y" type="xsd:int" kind="in/out"/>
      <parameter name="z" element="dtd1:Order" kind="out"/>
    </call>
   </operation>
```

图 27. 具有备用 Call 操作的 DADX

示例 4: RetrieveXML 操作

第 79 页的图 28 中的 DADX 文件通过使用存储过程 dxxGenXMLCLOB 来实现 retrieveXML 操作。

```
<?xml version="1.0"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
     </documentation>
  <operation name="findByColorAndMinPrice">
    <documentation>Returns all the orders that have the specified color
        and at least the specified minimum price.</documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad</DAD_ref>
      <SQL_override>
       select o.order_key, customer_name, customer_email,
         p.part_key, color, quantity, price, tax, ship_id, date, mode
       from order_tab o, part_tab p,
         table(select substr(char(timestamp(generate_unique())),16)
           as ship_id, date, mode, part_key from ship_tab) s
       where p.order_key = o.order_key and s.part_key = p.part_key
         and color = :color and price >= :minprice
       order by order_key, part_key, ship_id
      </SQL_override>
      <parameter name="color" type="xsd:string"/>
      <parameter name="minprice" type="xsd:decimal"/>
    </retrieveXML>
   </operation>
 </DADX>
```

图 28. 具有 retrieveXML 操作的 DADX

第 79 页的图 28 中的操作生成基于 *getstart_xcollection.dad* 文件中的映射的 XML 文档。该操作指定 SQL 覆盖。该操作替换在 DAD 文件中定义的 SQL 语句，并在覆盖语句中引用两个参数: `:color` 和 `:minprice`。

此示例的 DAD 文件在 *DB2 XML Extender Administration and Programming* 的附录中。

示例 5: StoreXML 操作

第 80 页的图 29 中的此示例显示了一个 DADX 文件，它通过使用 RDB_node 映射 *getstart_xcollection_rdb.dad* 来引用 DAD。

```
<?xml version="1.0"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
     </documentation>

  <implements namespace="http://www.poia.org/part_orders.wsdl"
      location="http://www.poia.org/part_orders.wsdl"/>

   <operation name="storeOrder">
     <documentation>Stores an automotive part order.
     </documentation>
     <storeXML>
       <DAD_ref>getstart_xcollection_rdb.dad</DAD_ref>
     </storeXML>
   </operation>
 </DADX>
```

图 29. 具有 StoreXML 操作的 DADX

如果使用 <collection_name> 元素代替 <DAD_ref> 元素，则 storeXML 操作将由 dxxInsertXML 存储过程来实现。它与 dxxShredXML 过程执行相同的操作，但使用 XML 集合的名称而不是 DAD 文件的名称。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』

将文档类型定义转换为 XML 模式

XML Extender 当前使用文档类型定义文件（DTD）来定义文档结构，而 Web 服务描述语言（WSDL）使用 XML 模式（XSD 文件）。“Web 服务对象运行时框架”（WORF）自动创建 XML 模式（XSD）文件。必须将条目添加至名称空间表（NST 文件）以定义与 DTD 相关联的名称空间。这还使 DTD 能转换为 XSD。

过程:

通过使用以下统一资源定位器（URL）语法来请求 XSD 文件:

```
http://host/path/dtd_file.dtd/XSD
```

例如:

```
http://host_name:port/services/sample/order.dtd/XSD
```

在这种情况下，order.dtd 文件必须在 WEB-INF\groups\dxx_sample 中。

WORF 和 XML Extender 通过它们的文档类型定义标识（DTDID）来找到 DTD。DTDID 是您的数据库的 DTD_REF 表中的文件名或键值。当启用数据库时，XML Extender 创建 DTD_REF 表。最佳做法是在 DTD_REF 表中存储 DTD，这是因为当您将 Web 应用程序移至另一机器时文件位置可能会更改。在 SALES_DB 示例中从 Windows 2000 setup-xcollection.cmd 文件的以下抽取显示如何将 DTD 插入至 DTD_REF 表:

```
db2 "connect to SALES_DB"
rem Insert DTDs
db2 "insert into db2xml.dtd_ref values('getstart.dtd',
 db2xml.XMLClobFromFile('%CD%\getstart.dtd'),
 0, 'user1', 'user1', 'user1')"
db2 "insert into db2xml.dtd_ref
  values('order.dtd', db2xml.XMLClobFromFile('%CD%\order.dtd'),
         0, 'user1', 'user1', 'user1')"
```

相关概念:

- 第 112 页的『XML 模式定义』

相关参考:

- 第 231 页的附录 C, 『DADX 文件的 XML 模式』

DADX 文件的 WSDL

DADX 文档包含实现 Web 服务所需的信息。它还包含生成描述 Web 服务的 WSDL 文档所需的信息。

“Web 服务描述语言”（WSDL）文档是“可扩展标记语言”（XML）词汇表，用来描述业务服务的接口。可以使用它来将服务发布至 UDDI 注册中心。WSDL 允许开发工具使用程序创建用于绑定至 Web 服务的请求程序代码和提供程序代码。它还使前置条件应用程序能够动态绑定至 Web 服务。可以使用 WSDL 来指定请求和响应所需的数据。WSDL 将 XML 模式用于精确数据定义。

WSDL 绑定描述服务如何绑定至消息传递协议，特别是 SOAP 消息传递协议。WSDL SOAP 绑定可以是 RPC 样式绑定或文档样式绑定。SOAP 绑定还可以进行编码使用或文字使用。

要生成 WSDL，提交以下统一资源定位器（URL）。*localhost* 端口号（此处由 <*yourWebAppServer*> 指定）取决于您自己的当前机器:

```
http://yourWebAppServer:port/webapp_name/group_name/dadx_file.dadx/WSDL
```

“Web 服务对象运行时框架”（WORF）动态地生成 WSDL 文档。可以在 UDDI 或其它某个 Web 服务目录中发布此文档。

如果在安装期间使用 WORF 包括的样本，则可以提交以下 URL:

```
http://yourWebAppServer/services/sales/PartOrders.dadx/WSDL
```

相关概念:

- 第 82 页的『UDDI 注册的 WSDL』
- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 108 页的『Web 服务描述语言』

相关任务:

- 第 102 页的『测试 Web 服务』

UDDI 注册的 WSDL

“统一描述、发现和集成”（UDDI）最佳实践文档说明了如何将“Web 服务描述语言”（WSDL）与 UDDI 注册中心配合使用。此文档建议将 WSDL 文档分为两部分，部署部分和可重新使用的部分。

部署部分包括 <service> 元素，该元素包含在其中部署服务的 URL。部署部分导入包含其它顶层 WSDL 元素的可重新使用的部分。

可重新使用的部分与 UDDI <tModel> 元素相对应。部署部分与 UDDI <businessService> 相对应。在 <businessService> 元素内，每个 WSDL <port> 元素与 UDDI <bindingTemplate> 元素相对应。

要了解有关 UDDI 和 Web 服务注册的更多信息，请参阅 Web 业务的统一描述、发现和集成站点。

要生成 WSDL 部分，提交带 WSDL 路径信息的 URL:

- 要生成部署部分，提交带 WSDLservice 关键字的 URL:

  ```
  http://yourWebAppServer:port/webapp_name/
     group_name/DADX_file.dadx/WSDLservice
  ```

- 要生成可重新使用的部分，提交带 WSDLbinding 关键字的 URL:

  ```
  http://yourWebbAppServer:port/webapp_name/
      group_name/DADX_file.dadx/WSDLbinding
  ```

以下示例演示如何生成 WSDL 文档的部署部分和可重新使用的部分。要生成部署部分，提交带 WSDLservice 命令的 URL，如以下示例所示:

```
http://yourWebAppServer/sales_db/part_orders.dadx/WSDLservice
```

要生成可重新使用的部分，提交带 WSDLbinding 命令的 URL，如以下示例所示:

```
http://yourWebAppServer/sales_db/part_orders.dadx/WSDLbinding
```

以上示例处理下列情况：服务实现程序创建对公司唯一的 Web 服务。设计 UDDI 来处理的使用情况之一是标准主体或供应商定义 Web 服务接口 tModel 的情况。然后，服务实现程序使用该 Web 服务。例如，航空业界可能定义这样一种 Web 服务，该服务提供航空公司可以实现的航班时刻表。UDDI 允许用户搜索实现给定的 tModel 的所有已注册的服务。然后，旅行规划应用程序可找到所有航班时刻表服务。

使用 DADX <implements> 元素来声明该服务实现由在别处定义的可重新使用的 WSDL 文档描述的 Web 服务。在第 83 页的图 30 中显示了 <implements> 元素的一个示例。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://schemas.ibm.com/db2/dxx/dadx"
  ...
  elementFormDefault="qualified">
  <import namespace="http://schemas.xmlsoap.org/wsdl/"
   schemaLocation="wsdl.xsd"/>
        ....
   <element name="DADX">
    <annotation>
    <documentation>
        Defines a Web Service.
        The Web Service is described by an optional
         WSDL documentation element.
        ....
      </documentation>
    </annotation>
          <complexType>
            <sequence>
        <element ref="wsdl:documentation" minOccurs="0"/>
        <element ref="dadx:implements" minOccurs="0"/>
        <element ref="dadx:result_set_metadata" minOccurs="0"
                maxOccurs="unbounded"/>
        <element ref="dadx:operation" maxOccurs="unbounded"/>
            </sequence>
          </complexType>
 ...
  <element name="implements">
    <annotation>
    <documentation>
        Defines the namespace and location of a set of WSDL bindings
        defined elsewhere. This information is imported into the
        WSDL document generated for this Web Service.
      </documentation>
    </annotation>
          <complexType>
      <attribute name="namespace" type="anyURI" use="required"/>
      <attribute name="location" type="anyURI" use="required"/>
          </complexType>
        </element>
...
< /schema>
```

图 30. 元素 <implements>

以下示例显示如何在 DADX 文件中使用 <implements> 标记:

```
<?xml version="1.0"?>

  <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"

      xmlns:xsd="http://www.w3.org/2001/XMLSchema"

        xmlns:xhtml="http://www.w3.org/1999/xhtml">

  <documentation>

   Provides queries for part order information at myco.com.

  This Web Service is compliant with the Part Ordering Industry
   Association standard.

      </documentation>


<implements namespace="http://www.poia.org/PartOrders.wsdl"

  location="http://www.poia.org/PartOrders.wsdl"/>


    <operation name="findAll">

<documentation%gt;

Returns an order with its complete details.

</documentation>

    ...


    </operation>
...
```

图 31. 使用 implements 标记的示例 DADX 文件

相关概念:

- 第 81 页的『DADX 文件的 WSDL』
- 第 108 页的『Web 服务描述语言』

使用 Web 服务提供程序的动态数据库查询

使用动态查询服务，您可以在运行时动态构建和提交选择、插入、更新应用程序数据以及调用存储过程的查询，而不是运行在部署时预定义的查询。

Web 应用程序可以使用 Web 服务接口来存取数据库以及抽取有关可用表与列的信息。然后，应用程序可以通过 Web 服务来查询表并修改数据库中的数据。Web 应用程序还可以对数据库执行数据定义语言操作（例如创建表）。通过使用 Web 服务提供程序的动态查询服务，Web 应用程序可以更加灵活。

WORF 可以从包含动态查询服务标记（<DQS>）的 DADX 文件生成两种样式的 Web 服务描述语言文件（WSDL）：

- 使用面向文档的信息样式的 WSDL 文件

- 使用面向过程的信息样式（RPC）的 WSDL 文件

所生成的样式是在组级别上定义的，它取决于 group.properties 文件中是否存在 `useDocumentStyle=true`。有关 Web 服务描述语言信息样式的更多信息，请使用浏览器查看 Web 服务描述语言规范。WSDL 文件包含服务、端口和定义信息。DADX 文件中的动态查询标记不会影响静态 DADX 函数。

如果在运行应用程序之前不了解查询搜索条件，则应考虑使用动态查询服务。

Web 服务提供程序的动态查询组件支持通常由下列类别定义的 Web 服务操作:

获取元数据
您可以检索存在于数据库中的表以及那些表的列信息。

执行 DDL
您可以发出 CREATE TABLE 语句。

执行 DML
您可以发出 SELECT、INSERT、UPDATE 和 DELETE 语句以及 CALL 语句来运行存储过程。

通过在 group.properties 文件中定义具有特定用户标识与密码设置的组，服务器管理员可控制对特定数据库的存取。管理员还可以创建单独的 WORF 实例来处理对数据库的存取。

相关概念:
- 第 81 页的『DADX 文件的 WSDL』
- 第 27 页的『Web 服务提供程序功能部件』
- 第 28 页的『与 DADX 文件配合使用的 Web 服务提供程序操作』
- 第 108 页的『Web 服务描述语言』
- 第 86 页的『动态查询服务 - 示例查询』

相关任务:
- 第 59 页的『定制 group.properties 文件』

相关参考:
- 第 93 页的『Web 服务提供程序中的动态查询服务操作』

将动态数据库查询作为 Web 服务提供程序的一部分来配置和运行

通过动态查询服务，可以在访问先前部署的 Web 服务的运行时构建和执行存储过程并提交数据库查询。您不再需要在“文档存取定义扩展”（DADX）文件中定义所有数据库查询。可以根据 group.properties 文件中的信息，在组级别或在组目录的范围内运行动态查询。

先决条件:
- 确保您要在其中运行动态 Web 查询的组存在 *group.properties* 文件。
- Web 应用程序必须为“Web 服务描述语言”（WSDL）文档中定义的每个 Web 服务操作建立与目标数据库的连接。

- 除非您在 WSDL 中定义导入定义，否则必须确保 XML 模式描述文件 *db2WebRowSet.xsd* 包括在 Web 应用程序的上下文根中。文件 *db2WebRowSet.xsd* 包括在 *dxxworf.zip* 文件中。

限制:

- 当应用程序使用 Web 服务提供程序的动态查询服务时，应用程序不能使用光标或在服务器上执行导致某种状态的任何操作。必须在单个查询中获取结果。
- 标识动态查询服务操作的 XML 标记（<DQS/>）不能与任何特定于 DADX 的 Web 服务定义共存于同一个文件中。

过程:

要准备 Web 服务环境以便在具有 Web 服务提供程序的 DB2 通用数据库上运行动态查询:

1. 创建包含 XML 标记 <DQS/> 的 DADX 文件。

   此标记使组可以执行动态查询。DADX 文件中不需要其它标记。
2. 将文件保存在要对其运行动态查询的组的目录中。
3. 使用 WSDL 为应用程序开发客户机。客户机必须至少包含以下信息:
   - 组名
   - DADX 文件名称，例如 *mydqs.dadx*
   - Web 服务操作，例如 `getTables`
4. 修改客户机以发出其中一个已接受的 DQS 操作，例如 getTables 操作。
5. 运行发出 `getTables` 操作的客户机。

   查询的结果是描述表的行与列的元数据以及表中包含的数据。SQL 语句在自动落实模式下运行。客户机还可以调用其它组中的动态查询服务。唯一需要更改的信息是端点 URL。但是，客户机仅对 RPC 样式 WSDL 或文档样式 WSDL 是兼容的。不能将 RPC 样式 WSDL 定义的动态查询服务客户机用于使用文档样式 WSDL 的组。

相关概念:

- 第 108 页的『Web 服务描述语言』
- 第 86 页的『动态查询服务 - 示例查询』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』
- 第 53 页的『定义一组 Web 服务』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』

动态查询服务 - 示例查询

示例: DADX 文件

在下列示例中，DADX 文件的名称为 *mydqs.dadx*。文件 mydqs.dadx 位于要对其执行动态查询的组的目录中。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx">
    <documentation>
```

```
        This is optional documentation about DQS
    </documentation>
      <DQS/>
</DADX>
```

示例: 从浏览器运行动态查询

下列示例是可以从浏览器运行的简单动态查询。还可以在应用程序中包括此语句。Web 服务提供程序中动态查询的必需信息以**粗体**打印。此示例中的 Web 服务操作是 executeQuery。与此操作关联的参数是 queryInput。语句访存表 employee 中列 lastname 的所有行:

```
http://localhost:9080/services/<group_name>
  /somefile.dadx/executeQuery?queryInputParameter
     =select%20lastname%20from%20employee
```

示例发出 GET 绑定请求而不是完整的 SOAP 包络。有关 GET、POST 和 SOAP 绑定的更多信息，请参阅使用 GET、POST 和 SOAP 绑定访问 Web 服务。下列输出源自 executeQuery 操作，并由 db2WebRowSet 模式定义加以定义:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:executeQueryResponse
    xmlns:ns1="http://schemas.ibm.com/db2/dqs">
  <queryOutputParameter>
   <db2WebRowSet
       xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<metadata>
 <column-count>14</column-count>
 <column-definition>
  <column-index>1</column-index>
  <nullable>0</nullable>
  <column-name>EMPNO</column-name>
  <column-precision>6</column-precision>
  <column-scale>0</column-scale>
  <column-type>CHAR</column-type>
  <column-type-name>CHAR</column-type-name>
  <xml-type>string</xml-type>
 </column-definition>
 <column-definition>
  <column-index>2</column-index>
  <nullable>0</nullable>
  <column-name>FIRSTNME</column-name>
  <column-precision>12</column-precision>
  <column-scale>0</column-scale>
  <column-type>VARCHAR</column-type>
  <column-type-name>VARCHAR</column-type-name>
  <xml-type>string</xml-type>
 </column-definition>
...
 <column-definition>
  <column-index>14</column-index>
  <nullable>1</nullable>
  <column-name>COMM</column-name>
  <column-precision>9</column-precision>
  <column-scale>2</column-scale>
  <column-type>DECIMAL</column-type>
  <column-type-name>DECIMAL</column-type-name>
  <xml-type>decimal</xml-type>
```

```
 </column-definition>
...
 <column-definition>
  <column-index>14</column-index>
  <nullable>1</nullable>
  <column-name>COMM</column-name>
  <column-precision>9</column-precision>
  <column-scale>2</column-scale>
  <column-type>DECIMAL</column-type>
  <column-type-name>DECIMAL</column-type-name>
  <xml-type>decimal</xml-type>
 </column-definition>
</metadata>
</data>
</db2WebRowSet>
</queryOutputParameter>
  </ns1:executeQueryResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

示例: 导入 db2WebRowSet.xsd

当组包含动态查询服务 DADX，db2WebRowSet.xsd 文件必须可以访问 Web 服务使用程序。要确保 db2WebRowSet.xsd 文件的位置，group.imports 文件定义必需的模式位置。以下是要导入 db2WebRowSet.xsd 的 group.imports 文件的示例。此示例假定本地组目录中没有文件 db2WebRowSet.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<imports
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import namespace="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
      schemaLocation="http://myServer.myCo.com/schemas/misc/ibm/db2WRS.xsd"/>
</imports>
```

如果 group.imports 文件不存在，则 WORF 在 WSDL 中仅为动态查询服务生成缺省导入元素。在此情况下，WORF 假定 db2WebRowSets.xsd 文件位于下列位置:

```
http://<server>:<port>/<contextRoot>/db2WebRowSet.xsd
```

示例: getTables

以下是 getTables 操作的示例:

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
    <ns0:getTables
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
     <tablesInputParameter>
       <tablesInputData>
        <schemaPattern>MSCHENK</schemaPattern>
        <tableNamePattern>EMPLOYEE</tableNamePattern>
       </tablesInputData>
      </tablesInputParameter>
    </ns0:getTables>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

示例: getColumns

以下是 getColumns 操作的示例:

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
    <ns0:getColumns
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      <columnsInputParameter>
        <columnsInputData>
          <schemaPattern>MSCHENK</schemaPattern>
          <tableNamePattern>EMPLOYEE</tableNamePattern>
          <columnNamePattern>EMPNO</columnNamePattern>
        </columnsInputData>
      </columnsInputParameter>
    </ns0:getColumns>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

示例: executeQuery

以下示例查询访存表 employee 中所有行，并指定若干参数:

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
     <ns0:executeQuery
       xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      <queryInputParameter>
       select * from employee
      </queryInputParameter>
      <extendedInputParameter>
      <properties>
         <loginInfo>
           <userid>userid</userid>
           <password>some_password</password>
         </loginInfo>
         <readOnly>true</readOnly>
         <isolationLevel>READ_UNCOMMITTED</isolationLevel>
         <escapeProcessing>true</escapeProcessing>
         <startAtRow>4</startAtRow>
         <fetchSize>80</fetchSize>
         <maxFieldSize>20</maxFieldSize>
         <maxRows>100</maxRows>
         <queryTimeout>2000</queryTimeout>
      </properties>
     </extendedInputParameter>
   </ns0:executeQuery>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

示例: executeUpdate

以下示例显示动态查询服务更新语句:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
   <ns0:executeUpdate
     xmlns:ns0="http://schemas.ibm.com/db2/dqs">
```

```
      <queryInputParameter>
         update bo_events set OBJECTEVENTID=&'testestest&'
      </queryInputParameter>
      <extendedInputParameter>
        <properties/>
      </extendedInputParameter>
    </ns0:executeUpdate>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

以下示例显示返回的响应文档:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:executeUpdateResponse
   xmlns:ns1="http://schemas.ibm.com/db2/dqs">
    <updateOutputParameter xsi:type="xsd:int">
        1
    </updateOutputParameter>
   </ns1:executeUpdateResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

示例: **executeCall**

示例请求调用存储过程 multipleResultSets:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
    <ns0:executeCall
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
    <callInputParameter>
      <callInputData>
          <spName>
             multipleResultSets
          </spName>
          <parameters>
            <parameter>
              <inParam>
                 <kind>IN</kind>
                 <type>string</type>
                 <value>000130</value>
              </inParam>
            </parameter>
            <parameter>
              <inParam>
                <kind>INOUT</kind>
                <type>string</type>
                <value>000130</value>
              </inParam>
             </parameter>
             <parameter>
               <outParam>
                 <kind>OUT</kind>
                 <type>string</type>
               </outParam>
             </parameter>
          </parameters>
       </callInputData>
```

```
    </callInputParameter>
   <extendedInputParameter>
    <properties/>
   </extendedInputParameter>
  </ns0:executeCall>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

以下示例显示样本输出:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:executeCallResponse
     xmlns:ns1="http://schemas.ibm.com/db2/dqs">
  <callOutputParameter>
  <dqs:callOutputData
     xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
   <dqs:outputResultSequences>
   <db2WebRowSet
     xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <metadata>
      <column-count>5</column-count>
      <column-definition>
        <column-index>1</column-index>
        <nullable>0</nullable>
        <column-name>DEPTNO</column-name>
        <column-precision>3</column-precision>
        <column-scale>0</column-scale>
        <column-type>CHAR</column-type>
        <column-type-name>CHAR</column-type-name>
        <xml-type>string</xml-type>
      </column-definition>
        ...
      <column-definition>
        <column-index>5</column-index>
        <nullable>1</nullable>
        <column-name>LOCATION</column-name>
        <column-precision>16</column-precision>
        <column-scale>0</column-scale>
        <column-type>CHAR</column-type>
        <column-type-name>CHAR</column-type-name>
        <xml-type>string</xml-type>
 </column-definition>
</metadata>
<data>
 <row>
  <column>A00</column>
  <column>
     SPIFFY COMPUTER SERVICE DIV.
       </column>
  <column>000010</column>
  <column>A00</column>
  <column xsi:nil="true"/>
 </row>
...
 <row>
   ...
 </row>
</data>
</db2WebRowSet>
</dqs:outputResultSequences>
   <dqs:outputParameterSequences>
```

```
    <dqs:callOutputParam>
     <position>2</position>
     <type>string</type>
     <value>xxxxxx</value>
    </dqs:callOutputParam>
    <dqs:callOutputParam>
     <position>3</position>
     <type>string</type>
     <value>This is the value of name3</value>
    </dqs:callOutputParam>
    </dqs:outputParameterSequences>
    </dqs:callOutputData>
   </callOutputParameter>
  </ns1:executeCallResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

示例: execute

以下示例创建包含一个列的表:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
  <ns0:execute
    xmlns:ns0="http://schemas.ibm.com/db2/dqs">
     <queryInputParameter>
       create table temptable(in varchar(500))
     </queryInputParameter>
     <extendedInputParameter>
       <properties/>
     </extendedInputParameter>
  </ns0:execute>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

以下是 execute 操作的输出:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
   <ns1:executeResponse
     xmlns:ns1="http://schemas.ibm.com/db2/dqs">
    <executeOutputParameter>
    <dqs:executeOutputData
       xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <resultsPresent>false</resultsPresent>
     <dqs:outputResultSequences>
     </dqs:outputResultSequences>
    </dqs:executeOutputData>
    </executeOutputParameter>
  </ns1:executeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

相关概念:

- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关任务:

- 第 85 页的『将动态数据库查询作为 Web 服务提供程序的一部分来配置和运行』

相关参考:

- 第 93 页的『Web 服务提供程序中的动态查询服务操作』

Web 服务提供程序中的动态查询服务操作

下表描述了在 DB2 Web 服务提供程序中受支持的动态查询操作。

表 7. 用于元数据检索的操作

Web 服务操作	描述
getTables **tablesInputParameter** 输入；类型 = tablesInputData（请参阅第 96 页的表 11） **tablesOutputParameter** 输出；类型 = db2WebRowSet	检索指定目录和模式中的表的描述，例如目录名称、模式名称和表名。如果将模式用作输入参数，则 Java 数据库连接可能要求模式区分大小写。
getColumns **columnsInputParameter** 输入；类型 = columnsInputData（请参阅第 96 页的表 12） **columnsOutputParameter** 输出；类型 = db2WebRowSet	检索指定目录、模式和表中的列的描述。如果将模式用作输入参数，则 Java 数据库连接可能要求模式区分大小写。

表 8. 用于运行查询和存储过程的操作

操作	描述
executeQuery **queryInputParameter** 必需输入；类型 = 字符串 **extendedInputParameter** 必需输入；类型 = 属性（请参阅第 94 页的表 9） **queryOutputParameter** 输出；类型 = db2WebRowSet	在数据库服务器上发出单个 SQL SELECT 语句，然后返回单个结果集。
executeUpdate **queryInputParameter** 必需输入；类型 = 字符串 **extendedInputParameter** 必需输入；类型 = 属性（请参阅第 94 页的表 9） **updateOutputParameter** 输出；类型 = int	在数据库服务器上发出单个 INSERT、UPDATE 或 DELETE 语句，然后返回完成代码。

表 8. 用于运行查询和存储过程的操作 (续)

操作	描述
executeCall **callInputParameter** 输入；类型 = callInputData **extendedInputParameter** 输入；类型 = 属性（请参阅表 9） **callOutputParameter** 输出；类型 = callOutputData	在数据库服务器上调用单个存储过程，然后返回一组输出参数和一系列结果集。
execute **queryInputParameter** 必需输入；类型 = 字符串 **extendedInputParameter** 必需输入；类型 = 属性（请参阅表 9） **executeOutputParameter** 输出；类型 = executeOutputData	在数据库服务器上发出单个 SQL 语句，然后返回完成代码和一系列结果集。

您可以将表 9 中列示的可选参数与第 93 页的表 8 中列示的操作配合使用。

表 9. 扩展参数的输入数据类型

属性类型	描述
loginInfo • 用户标识 • 密码	loginInfo 包括传递到数据库以获取访问控制权的用户标识。它还包括与传递到数据库以获取访问控制权的用户标识相关联的密码。这些属性具有字符串类型。如果您指定用户标识，则必须指定密码。
readOnly	允许 Web 应用程序指定它将仅以只读方式使用数据库。这是二进制类型，可以为 true 或 false。
escapeProcessing	允许 Web 应用程序控制对查询字符串的转义处理。如果转义扫描已启用（true），则驱动程序执行转义替换，然后才将 SQL 发送到数据库。这是二进制类型，可以为 true 或 false。缺省值为 true。
fetchSize	指定在执行任何给定访存操作时要访存回 Web 应用程序的行数。这是整数类型。缺省值为 0。
maxFieldSize	将列中最大字节数的上限设置为指定的字节数。该值是任意列值可返回的最大字节数。这是整数类型。
maxRows	指定要访存回 Web 应用程序的最大行数。这是整数类型。如果未指定 maxRows 参数，则最多可返回 1000 行。
startAtRow	允许 Web 应用程序跳过结果集中指定的行数。这是整数类型。
queryTimeout	允许 Web 应用程序指定查询的超时值。将驱动程序等待语句对象运行所需的秒数设置为给定的秒数。如果超过该限制，则发生异常。值为 0 秒表示驱动程序可以无限期地等待。

表 9. 扩展参数的输入数据类型 (续)

属性类型	描述
isolationLevel	允许 Web 应用程序控制查询的隔离级别。 • READ_UNCOMMITTED • READ_COMMITTED • REPEATABLE_READ • SERIALIZABLE • NONE

表 10. callInputParameter 的输入数据类型

callInputData 类型	描述
spName 类型: 字符串	要调用的存储过程的名称。此参数是必选的。
模式 类型: 字符串	存储过程的模式。此参数是可选的。如果未提供参数，则该值为当前模式。
参数 类型: 一连串参数，每个都包含 inParam 或 outParam **inParam** 类型定义为: • **种类**: “IN” 或 “INOUT” • **类型**: 参数的类型（如 int 或字符串） • **值**: 参数的值 **outParam** 类型定义为: • **种类**: “IN” 或 “INOUT” • **类型**: 参数的类型	存储过程可以具有三种参数: IN、OUT 和 INOUT。此参数类型是可扩展的类型。它允许任意数量的 inParam 和 outParam 类型的所有组合。Web 应用程序必须知道它计划调用的存储过程是否需要参数。如果它需要参数，则需要知道所需的参数数目以及参数类型。 如果存储过程将其中一个不受支持的数据类型用作存储过程参数，则无法通过 WORF 执行此存储过程。 WORF 接受数种适用于存储过程参数的 XML 类型。这些参数与内置的 SQL 数据类型相对应。第 72 页的表 6 描述了支持的类型。 通过使用下列其中一个值，可以将输入参数设置为 NULL: **不使用** 不提供输入参数的 <value/> 标记。 **nil = true** 该标记带有的属性 nil 标记已经设置为 true，例如，<value xsi:nil="true"/> 输入参数的顺序必须与存储过程期望的顺序相同。

表 11. tablesInputData 类型的输入数据类型

tablesInputData 类型	描述
catalogPattern 类型 = "字符串" **schemaPattern** 类型 = "字符串" **tableNamePattern** 类型 = "字符串"	每个模式值均为可选的。如果该值未指定，则缺省设置为空值。每个值的描述和行为均在 JDBC 中指定。使用 getTables Web 服务操作，以返回符合所指定的 catalogPattern、schemaPattern 和 tableNamePattern 的表的列表。

示例（请注意，为了简单起见，此处未显示诸如名称空间定义之类的内容）：

```
<tablesInputData>
 <catalogPattern></catalogPattern>
 <schemaPattern>userSchema
 </schemaPattern>
 <tableNamePattern>EMPLOYEE
 </tableNamePattern>
</tablesInputData>
```

表 12. columnsInputData 类型的输入数据类型

columnsInputData 类型	描述
catalogPattern 类型 = "字符串" **schemaPattern** 类型 = "字符串" **tableNamePattern** 类型 = "字符串" **columnNamePattern** 类型 = "字符串"	每个模式值均为可选的。如果该值未指定，则缺省设置为空值。每个值的描述和行为均在 JDBC 中指定。使用 getColumns Web 服务操作，以接收符合所指定的目录字符串模式、schemaPattern、表名和 columnNamePattern 的列的列表。

示例（请注意，为了简单起见，此处未显示诸如名称空间定义之类的内容）：

```
<columnsInputData>
 <catalogPattern></catalogPattern>
 <schemaPattern>userSchema
 </schemaPattern>
 <tableNamePattern>EMPLOYEE
 </tableNamePattern >
 <columnNamePattern>LASTNAME
 </columnNamePattern>
</columnsInputData>
```

表 13. callOutputData 类型的输出数据类型

callOutputData 类型
outputResultSequences 包含一系列由存储过程返回的所有结果集（当类型为 db2WebRowSet 时） **outputParameterSequences**: 包含一连串 callOutputParam（从存储过程中返回的参数，可以是“种类 = INOUT”或“种类 = OUT”）

表 13. callOutputData 类型的输出数据类型 (续)

callOutputData 类型
callOutputParam 返回的参数: 包含 • <position> 类型: int – 存储过程参数列表中参数的位置 • <type> 类型: 字符串 – XML 数据类型（请参阅 callInputData 以了解类型信息） • <value> 类型: 任意 – 参数的值 如果输出参数为 NULL，则使用“不使用”方法。 该结果包含 `<value xsi:nil="true"/>`

示例（请注意，为了简单起见，此处未显示诸如名称空间定义之类的内容）:

```
<callOutputParameter>
   <dqs:callOutputData
      xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <dqs:outputResultSequences>
         <db2WebRowSet
           xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <metadata>
             ....
         </db2WebRowSet>
     </dqs:outputResultSequences>
     <dqs:outputParameterSequences>
            <dqs:callOutputParam>
                <position>1</position>
                <type>short</type>
                <value>123</value>
            </dqs:callOutputParam>
            <dqs:callOutputParam>
                <position>2</position>
                <type>int</type>
                <value xsi:nil="true" />
            </dqs:callOutputParam>
         </dqs:outputParameterSequences>
      </dqs:callOutputData>
</callOutputParameter>
```

表 14. executeOutputData 类型的输出数据类型

executeOutputData 类型	描述
resultsPresent 类型 = ″布尔″ **outputResultSequences** 0 或多次出现 db2WebRowSet	如果使用将返回结果集的查询字符串来调用 **执行** Web 服务操作，则布尔会指示此情况，并且每个 outputResultSequences 都将包含其中一个那些结果集。

表 14. executeOutputData 类型的输出数据类型 (续)

executeOutputData 类型	描述

示例（请注意，为了简单起见，此处未显示诸如名称空间定义之类的内容）：

```
<executeOutputParameter>
   <dqs:executeOutputData
      xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <resultsPresent>true</resultsPresent>
      <dqs:outputResultSequences>
         <db2WebRowSet
           xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <metadata>
              ....
         </db2WebRowSet>
      </dqs:outputResultSequences>
   </dqs:executeOutputData>
</executeOutputParameter>
```

相关概念:

- 第 86 页的『动态查询服务 - 示例查询』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关任务:

- 第 85 页的『将动态数据库查询作为 Web 服务提供程序的一部分来配置和运行』

相关参考:

- 第 98 页的『db2WebRowSet』

db2WebRowSet

动态查询服务类型 db2WebRowSet 描述从 SQL 结果集生成 XML 文档的一般方法。模式文档 db2WebRowSet.xsd 未包含有关特定结果集的任何元数据信息。它包含有关结果集元数据的一般元数据信息。实际结果集元数据和结果集数据位于 XML 实例文档中。实例文档包含元数据段和数据段。

元数据段

包含有关结果中所有列的元数据信息。

第一个标记是列计数标记。它包含结果集中列的数目。每个列均有一个列定义标记。列定义包含下列元数据信息:

表 15. 列定义元数据

标记名称	描述
<column-index>	结果集中列的位置，从 1 开始。
<nullable>	如果列可以为 NULL，则值为 1。如果列不能为 NULL，则值为 0。
<column-name>	列的名称。
<column-precision>	此标记的描述取决于 SQL 数据类型。例如，如果 SQL 数据类型是字符，则列精度为长度。如果 SQL 数据类型是十进制，则列精度为精度。
<column-scale>	列标量是十进制数据类型。

表 15. 列定义元数据 (续)

标记名称	描述
<column-type>	列类型对应于 Java 数据库连接类型，例如 BINARY、VARBINARY、CHAR 和 VARCHAR。
<column-type-name>	DB2 通用数据库数据类型名称，例如 CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、CHAR 和 VARCHAR。
<xml-type>	XML 数据类型，例如，base64binary、int、string 和 dateTime。

数据段 包含实际数据。每行映射到行标记。行标记包含的列标记数目与结果集中的列数相同，并按列索引排序。行标记包含作为 XML 数据类型的实际数据。

表 16. 数据类型映射约定

DB2 UDB 数据类型 <column-type-name>	JDBC 数据类型 <column-type>	XML 数据类型 <xml-type>
BLOB	BLOB	base64Binary
CLOB	CLOB	字符串型
LONGVARCHAR	LONGVARCHAR	字符串型
VARCHAR	VARCHAR	字符串型
CHAR	CHAR	字符串型
CHAR FOR BIT DATA	BINARY	base64Binary
VARCHAR FOR BIT DATA	VARBINARY	base64Binary
LONGVARCHAR FOR BIT DATA	LONGVARBINARY	base64Binary
DATE	DATE	日期型
TIME	TIME	时间型
TIMESTAMP	TIMESTAMP	dateTime
-	BOOLEAN	布尔值
-	BIT	布尔值
TINYINT	TINYINT	整型
SMALLINT	SMALLINT	整型
INTEGER	INTEGER	整型
BIGINT	BIGINT	整型
DOUBLE	DOUBLE	双精度型
FLOAT	FLOAT	双精度型
REAL	REAL	浮点型
DECIMAL	DECIMAL	十进制型
NUMERIC	NUMERIC	十进制型
DATALINK	DATALINK	字符串型
-	ARRAY	anyType
DISTINCT	DISTINCT	字符串型
-	JAVA_OBJECT	字符串型
-	NULL	字符串型

表 16. 数据类型映射约定 (续)

DB2 UDB 数据类型 <column-type-name>	JDBC 数据类型 <column-type>	XML 数据类型 <xml-type>
-	OTHER	字符串型
-	STRUCT	字符串型
-	REF	字符串型
	其它类型的数字	字符串型

以下是 db2WebRowSet.xsd 文件。此文件的缺省位置是 <contextRoot> 目录。

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema targetNamespace="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:db2wrs="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
  elementFormDefault="qualified">
   <xs:element name="db2WebRowSet">
  <xs:complexType>
   <xs:sequence>
      <xs:element ref="db2wrs:metadata"/>
    <xs:element name="data">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="row"
                              minOccurs="0"
                              maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element ref="db2wrs:column"
                                    minOccurs="0"
                                    maxOccurs="unbounded"/>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

图 32. db2WebRowSet.xsd (1/2)

```
<xs:element name="column"
               type="xs:anyType"
               nillable="true"/>
 <xs:element name="metadata">
  <xs:complexType>
   <xs:sequence>
    <xs:element
                name="column-count"
                type="xs:string" />
    <xs:choice>
     <xs:element name="column-definition"
                      minOccurs="0"
                      maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="column-index"
                             type="xs:string" />
        <xs:element name="nullable"
                             type="xs:string" />
        <xs:element name="column-name"
                             type="xs:string" />
        <xs:element name="column-precision"
                             type="xs:string" />
        <xs:element name="column-scale"
                             type="xs:string" />
        <xs:element name="column-type"
                             type="xs:string" />
        <xs:element name="column-type-name"
                             type="xs:string" />
        <xs:element name="xml-type"
                             type="xs:string"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:choice>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

图 32. db2WebRowSet.xsd (2/2)

相关概念:

- 第 86 页的『动态查询服务 - 示例查询』
- 第 84 页的『使用 Web 服务提供程序的动态数据库查询』

相关任务:

- 第 85 页的『将动态数据库查询作为 Web 服务提供程序的一部分来配置和运行』

相关参考:

- 第 93 页的『Web 服务提供程序中的动态查询服务操作』

验证和测试 Web 服务提供程序（WORF）

本节通过使用随 DB2 提供的 SAMPLE 数据库提供 WORF 的概述。一开始应确保应用程序服务器已准备好运行（有关更多详细信息，请参阅第 33 页的『在 UNIX、Windows、z/OS 和 OS/390 上为 WebSphere Application Server 配置 Web 服务提供程序』和第 46 页的『在 UNIX 和 Windows 上为 Apache Jakarta Tomcat 配置

Web 服务提供程序』）。现在，您就准备好创建存取 SAMPLE 数据库的 Web 服务了。此方案假设您将 WORF 样本作为名为*服务*的 Web 应用程序安装。该方案还假定您在应用程序服务器上配置了*服务*。

测试 Web 服务应用程序 - 方案

WORF 支持使用“文档存取定义扩展”（DADX）文件创建 Web 服务。DADX 文件包含创建 Web 服务所必需的信息，并且可以引用 DAD 文件。此方案将使用一个称为 *HelloSample.dadx* 的简单 DADX 文件:

```
<?xml version="1.0" encoding="UTF-8"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx">

    <operation name="listDepartments">
      <query>
        <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
      </query>
    </operation>
 </DADX>
```

图 33. 简单 DADX 文件: HelloSample.dadx

对于 OS/390® 和 z/OS™ 平台，可能需要修改表名以与安装的样本 DEPARTMENT 表相对应。此表具有缺省名称 DSN8710.DEPT。

要部署在 DADX 文件中定义的 Web 服务，将它复制至 dxx_sample 目录中由组 *db2sample* 定义的目录中的应用程序服务器。

HelloSample.dadx 定义了一个 Web 服务，它具有名为 `listDepartment` 的单个操作，该操作列示 DEPARTMENT 表的内容。子标记 <query> 指定操作的类型。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 102 页的『测试 Web 服务』

测试 Web 服务

可以通过完成以下任务来测试 Web 服务:

过程:

1. 确保已在目录 \WEB-INF\classes\groups\dxx_sample 中安装 WORF 样本。
2. 确保已在服务器（例如 WebSphere Application Server, WAS）中部署了样本应用程序。
3. 打开浏览器窗口并输入以下统一资源定位器（URL）以开始测试:

   ```
   http://<your WebAppServer>/services/db2sample/ivt.dadx/TEST
   ```

记住，*your WebAppServer* 标识取决于您的 Web 服务器配置。当输入地址时，会看到以下自动生成的文档和测试页:

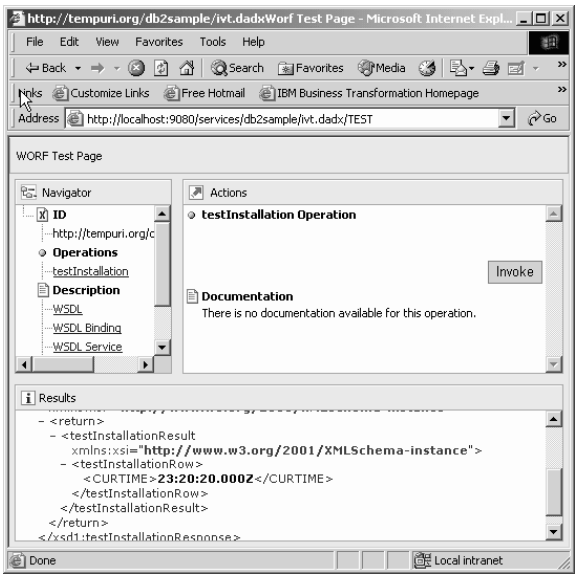


图 34. WORF 测试页

4. 测试 `listDepartments` 操作:
   a. 在**方法**窗格中单击 `listDepartments` 链接。
   b. 在**输入**窗格中单击**调用**按钮。

   可在“结果”窗格中看到操作的“可扩展标记语言”（XML）结果。

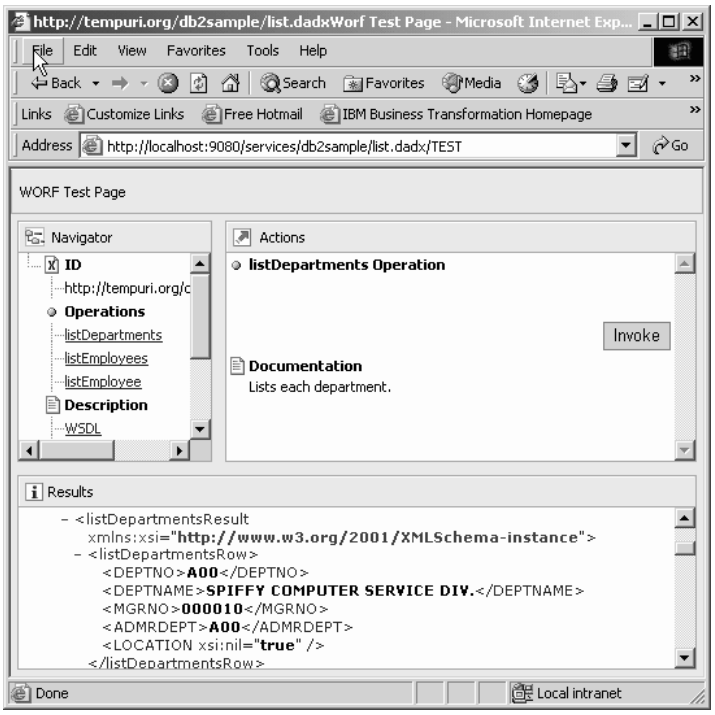


图 35. 查询的结果

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 102 页的『测试 Web 服务应用程序 - 方案』
- 第 122 页的『安装 Web 应用程序』

相关任务:

- 第 36 页的『安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本』

访问带有 GET、POST 和 SOAP 绑定的 Web 服务

"Web 对象运行时框架"（WORF）测试页充当 Web 服务的简单"超文本标记语言"（HTML）客户机并使用"超文本传输协议"（HTTP）POST 绑定。可以通过使用 HTTP GET 和 SOAP 绑定来访问 Web 服务。可以调用带有 HTTP GET 和 POST 绑定的 `listDepartments` 操作。以下示例显示 GET 或 POST 绑定的基本语法:

```
http://server:port/contextRoot/group/dadx_file/operationName
```

如果已安装了 WORF 样本，可以输入以下统一资源定位器（URL）以发出 GET 请求:

```
http://<yourWebAppServer:9080>/services/db2sample/HelloSample.dadx/listDepartments
```

localhost 端口号（此处由 *<yourWebAppServer>* 指定）取决于您自己的当前机器。WORF `listDepartments` 操作返回可保存至文件的"可扩展标记语言"（XML）响应。HTTP 响应对 GET 和 POST 是相同的。

```
<?xml version="1.0" ?>
<xsd1:listDepartmentsResponse
  xmlns:xsd1="http://schemas.ibm.com/sample/department.dadx/XSD"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<return>
 <xsd1:listDepartmentsResult
  xmlns:xsd1="http://schemas.ibm.com/sample/department.dadx/XSD"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<listDepartmentsRow>
  <DEPTNO>A00</DEPTNO>
  <DEPTNAME>SPIFFY COMPUTER SERVICE DIV.</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>B01</DEPTNO>
  <DEPTNAME>PLANNING</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>C01</DEPTNO>
  <DEPTNAME>INFORMATION CENTER</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>D01</DEPTNO>
  <DEPTNAME>DEVELOPMENT CENTER</DEPTNAME>
</listDepartmentsRow>
        ...
<listDepartmentsRow>
  <DEPTNO>E21</DEPTNO>
  <DEPTNAME>SOFTWARE SUPPORT</DEPTNAME>
</listDepartmentsRow>
</xsd1:listDepartmentsResult>
</return>
</xsd1:listDepartmentsResponse>
```

图 36. XML 响应文档

GET 绑定请求不会发送请求文档。相反，可以将查询字符串中所有必需参数连接到 URL。可以用问号（?）将查询字符串连接至 URL。任何 parameter=value 对之间的定界符是 & 符号。任何特殊字符都必须是 URL 编码的。以下示例是一个 GET 绑定请求，它使用带问号的查询字符串和定界符。

```
http://server:port/contextRoot/group/dadx/
   operationName?param1=abc&param2=1234&param3=thi&20is&20a&20parameter
```

以下示例是动态查询服务。这是 GET 绑定请求。

```
http://localhost:9080/services/db2sample/dqs.dadx/
   executeQuery?queryInputParameter=select+*+from+employee&extendedInputParameter=
      %3Cproperties%3E%0D%0A%3C%2Fproperties%3E%0D%0A
```

POST 绑定会发出 HTTP POST 请求。POST 绑定请求会发送请求文档。该文档包含请求参数，但参数不是 XML 格式。HTTP 客户机应用程序（例如 Web 浏览器）会创建请求文档。Web 浏览器通常从发送到服务器的输入表单创建请求文档。下列语法是一种典型的 POST 绑定请求:

```
http://server:port/contextRoot/group/dadx/operationName
```

以下示例是动态查询服务。这是 POST 绑定请求。

```
http://localhost:9080/services/db2sample/dqs.dadx/executeQuery
```

查询与 GET 绑定请求相同，只是问号（?）后面的信息位于请求文档中，而不是作为 URL 的一部分。在以下示例中，内容类型进行 www-url 编码:

```
queryInputParameter=
   select+*+from+employee&extendedInputParameter=
%3Cproperties%3E%0D%0A%3C%2Fproperties%3E%0D%0A
```

动态查询服务请求的 GET 和 POST 响应为:

```
<?xml version="1.0"?>
<xsd1:executeQueryResponse
  xmlns:xsd1="http://schemas.ibm.com/db2/dqs/types/soap"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <queryOutputParameter>
      ...
   </queryOutputParameter>
</xsd1:executeQueryResponse>
```

通常，HTTP GET 和 POST 绑定与任何其它 HTTP GET 和 POST 请求是相同的。HTTP GET 绑定将操作的任何输入参数添加至统一资源定位器（URL）。但是，HTTP POST 绑定在请求主体中发送参数。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 126 页的『Apache SOAP 配置』
- 第 106 页的『SOAP 绑定』

相关参考:

- 第 93 页的『Web 服务提供程序中的动态查询服务操作』
- 第 119 页的『Web 服务样本 - PartOrders.dadx』

SOAP 绑定

“简单对象访问协议”（SOAP）绑定也使用“超文本传输协议”（HTTP）POST，但它将操作名、输入参数和其它信息作为“可扩展标记语言”（XML）请求主体发送。

SOAP 请求绑定通过 HTTP 发出 SOAP 请求。SOAP 作为消息协议用于请求和响应消息。SOAP 请求指定请求和响应消息应如何显示。SOAP 绑定请求是遵循特定模式的 XML 文档。

SOAP 在 HTTP POST 请求顶部运行。HTTP 是传输协议，SOAP 是消息协议。客户机应用程序必须知道如何构建 SOAP 请求文档。参数的定义和格式以及其它信息在单独的“Web 服务描述语言”（WSDL）文档中定义。

使用以下统一资源定位器（URL）来存取 SOAP 绑定（记住，*your WebAppServer* 标识取决于您的 Web 服务器配置）:

```
http://<your WebAppServer>/services/db2sample/HelloSample.dadx/SOAP
```

请求中没有操作名。信息现在位于 SOAP 请求文档中。

以下示例（用于 RPC 样式）是使用 SOAP 绑定的动态查询服务。

```
<SOAP-ENV:Envelope
  1 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<SOAP-ENV:Body>
     2 <ns0:executeQuery
      3 xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      4 <queryInputParameter>
            select * from employee
     </queryInputParameter>
     <extendedInputParameter>
          <properties/>
     </extendedInputParameter>
   </ns0:executeQuery>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP 请求具有关键信息:

1. SOAP 包络。
2. 操作名。
3. 实际请求文档。在 WORF 环境中，这是一个 XML 文档（现在是实际 Web 服务 SOAP 请求）。
4. 参数。

SOAP 响应为:

```
<?xml version='1.0' encoding='UTF-8'?><SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
   <ns1:executeQueryResponse
      xmlns:ns1="http://schemas.ibm.com/db2/dqs"
          SOAP-ENV:encodingStyle="http://xml.apache.org/xml-soap/literalxml">
    <queryOutputParameter>
       ...
    </queryOutputParameter>
   </ns1:executeQueryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

包含在 SOAP-ENV:Body 标记内的信息是实际响应文档。在 WORF 环境中，响应文档是 XML 文档。这是实际 Web 服务 SOAP 响应。

注: 考虑将 SOAP 绑定用于 Java™ 和 JavaScript™ 客户机。WebSphere® Studio 具有生成 Java Web 服务客户机的功能。

相关概念:

- 第 104 页的『访问带有 GET、POST 和 SOAP 绑定的 Web 服务』
- 第 135 页的『Web 服务使用程序函数』
- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 102 页的『测试 Web 服务』

相关参考:

- 第 93 页的『Web 服务提供程序中的动态查询服务操作』

Web 服务描述语言

Web 服务提供程序由“Web 服务描述语言”（WSDL）文档加以描述。Web 服务的关键在于 Web 服务描述语言文档。

WSDL 是 XML 文档，将 Web 服务描述为端点或端口的集合。端点是可定址的位置，可以根据指定接口的关联绑定访问此位置的 Web 服务。一个 Web 服务可以具有多个端点。WSDL 中的端点根据消息运作。WSDL 绑定描述服务如何绑定至消息传递协议，特别是 SOAP 消息传递协议。WSDL SOAP 绑定可以是面向文档或面向过程（RPC）样式绑定。SOAP 绑定还可以进行编码使用或文字使用。

如图 37 所示，Web 服务提供程序实现服务并将接口发布到某些服务代理程序，例如 UDDI。然后，服务请求程序可以使用服务代理程序查找 Web 服务。当请求程序找到服务后，请求程序绑定至服务提供程序，以便请求程序可以使用 Web 服务。请求程序通过在请求程序与提供程序之间交换 SOAP（简单对象访问协议）消息来调用服务。

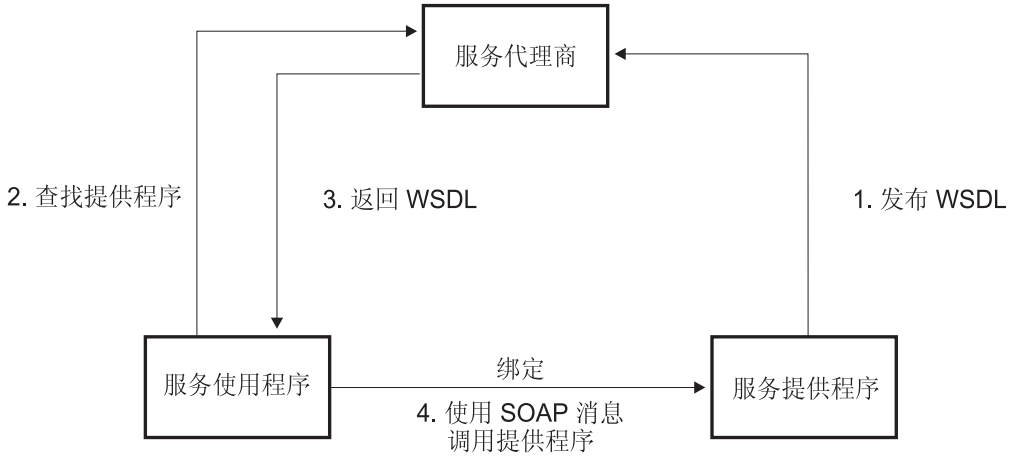


图 37. 作为面向服务的体系结构的 Web 服务

SOAP 规范定义基于 XML 的消息的布局。SOAP 消息包含在 SOAP 包络中。包络包括可选 SOAP 头和强制 SOAP 主体。SOAP 标题可以包含有关实际消息的信息，例如加密信息或认证信息。SOAP 主体包含实际消息。SOAP 规范还包含编程语言绑定的缺省编码，也称为 SOAP 编码。

WSDL 文档可以包含一个或多个 Web 服务。服务由一个或多个包含绑定的端口组成。WSDL 文档可以具有一个或多个端口类型。端口类型具有一个或多个操作并显示抽象输入和输出消息。绑定指将协议或数据格式信息与抽象实体（如消息、操作或 portType）关联的过程。绑定为特定端口类型创建具体协议和数据格式规范。端口是作为绑定和 Web 地址的端点。

第 110 页的图 38 中的示例显示提供股票报价的简单服务的 WSDL 定义。Web 服务支持名为 `GetLastTradePrice` 的单个操作。使用 SOAP 1.1 协议在 HTTP 上部署服务。该请求将字符串数据类型的订单符号作为输入读取，并返回浮点数据类型的价格。此示例中显示的类型是 XML 模式定义。可以使用 XSD 文件，将 DB2® 通用数据库表中的表和列与 Web 服务关联。

<soap:binding> 元素中指定的 WSDL 样式是文档样式。<soap:operation> 元素将操作作为一个整体提供信息。<soap:operation> 元素中的样式属性指示操作是面向 RPC（包含参数和返回值的消息）还是面向文档（包含文档的消息）。此属性的值还影响构造 SOAP 消息主体的方式。如果未指定属性，则其缺省设置为 <soap:binding> 元素中指定的值。IBM® DB2 Information Integrator Web 服务提供程序包含的样本设置为使用 RPC 样式。对于新的应用程序，应该使用文档样式以获得最大互操作性。在版本 8.2 中，Web 服务提供程序使用包含文字使用和类型节点的 RPC 样式，而不是包含文字使用和元素节点的 RPC 样式。但是，自 Web 服务提供程序的早期发行版以来，未对 SOAP 消息作任何更改。

完整的示例和 WSDL 规范位于 W3C 站点（http://www.w3.org/TR/2001/NOTE-wsdl-20010315）。

```
<?xml version='1.0'?>
<definitions name='StockQuote'
...


<types>
       <schema targetNamespace='http://example.com/stockquote.xsd'
              xmlns='http://www.w3.org/2000/10/XMLSchema'>
          <element name='TradePriceRequest'>
            <complexType>
                    <all>
                        <element name='tickerSymbol' type='string'/>
                    </all>
            </complexType>
          </element>
             <element name='TradePrice'>
            <complexType>
                    <all>
                        <element name='price' type='float'/>
                    </all>
            </complexType>
          </element>
< /schema>
    </types>

<message name='GetLastTradePriceInput'>
...
</message>

    <portType name='StockQuotePortType'>
        <operation name='GetLastTradePrice'>
           <input message='tns:GetLastTradePriceInput'/>
           <output message='tns:GetLastTradePriceOutput'/>
    </operation>
    </portType>


    <binding
       name='StockQuoteSoapBinding'
       type='tns:StockQuotePortType'>
        <soap:binding
          style='document'
            transport='http://schemas.xmlsoap.org/soap/http'/>
        <operation name='GetLastTradePrice'>
           <soap:operation
             soapAction='http://example.com/GetLastTradePrice'/>
           <input>
               <soap:body use='literal'/>
           </input>
           <output>
               <soap:body use='literal'/>
           </output>
    </operation>
    </binding>

   <service name='StockQuoteService'>
        <documentation>My first service</documentation>
        <port name='StockQuotePort'
           binding='tns:StockQuoteBinding'>
           <soap:address
             location='http://example.com/stockquote'/>
        </port>
    </service>

</definitions>
```

图 38. WSDL 的示例

因为 WSDL 文档具有一定的结构，所以 Web 服务开发者可能需要使用 WSDL 文档定义级别的外部模式的类型，也可能需要使用 WSDL 文档类型的级别类型。要使用外部模式，可以在 WSDL 中使用导入的模式定义。生成 WSDL 期间，WORF 支持两类导入:

WSDL 的 /definitions 作用域中的导入

http://schemas.xmlsoap.org/wsdl/:import

WSDL 的 /definitions/type/schema 作用域中的导入

http://www.w3.org/2001/XMLSchema:import

可以使用 group.imports 文件添加导入定义。如果 group.imports 文件存在于 Web 服务组目录的资源中，则 WORF 包括生成 WSDL 中的 group.imports 信息。以下示例是 group.imports 文件:

```
<?xml version='1.0' encoding='UTF-8'?>
<imports xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
   <wsdl:import namespace='http://some/namespace/1'
           location='schema1.xsd'/>
   <wsdl:import namespace='http://some/namespace/2'
           location='schema2.xsd'/>
   <xsd:import namespace='http://some/namespace/3'
           schemaLocation='schema3.xsd'/>
   <xsd:import namespace='http://some/namespace/4'
           schemaLocation='schema4.xsd'/>
</imports>
```

此示例在即将添加到 WSDL 中的 /definitions 作用域中定义两个导入（schema1.xsd 和 schema2.xsd）。此示例在即将添加到 WSDL 中的 /definitions/types/schema 作用域中定义两个导入（schema3.xsd 和 schema4.xsd）。WORF 生成的 WSDL 包括源自上述文件的导入定义，并具有以下结构:

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions>
  <wsdl:documentation xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
       xmlns='http://schemas.xmlsoap.org/wsdl/'>
     Documentation Text Node
  </wsdl:documentation>
  <import location='schema2.xsd'
            namespace='http://some/namespace/2'/>
  <import location='schema1.xsd'
            namespace='http://some/namespace/1'/>
  <types>
    <schema>
      <import namespace='http://some/namespace/4'
             schemaLocation='schema4.xsd'/>
      <import namespace='http://some/namespace/3'
             schemaLocation='schema3.xsd'/>
      <element name='executeQueryResponse'> .... </element>
      <element name='executeQuery'> .... </element>
< /schema>
  </types>
   ...
</definitions>
```

如果安装了 WORF 示例，并具有名为 services 的应用程序，则可以为服务 *HelloSample.dadx* 请求“Web 服务描述语言”（WSDL）文档。使用以下统一资源定位器（URL）来请求 WSDL。*localhost* 端口号（此处由 *<yourWebAppServer>* 指定）取决于您自己的当前机器:

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDL
```

WORF 自动从 DADX 生成 WSDL 文档。

相关概念:

- 第 81 页的『DADX 文件的 WSDL』

相关任务:

- 第 59 页的『定制 group.properties 文件』
- 第 102 页的『测试 Web 服务』

UDDI 业务注册表

在 “通用描述、发现和集成”（UDDI）业务注册表中注册您的 Web 服务。建议做法是将 WSDL 文档分割为服务实例文档和绑定文件。要了解有关 UDDI 和最佳实践的更多信息，请参阅 UDDI 最佳实践。

服务实例文档包含从中部署服务的地址，并且它导入绑定文档。许多服务实例可引用公共绑定文档。在 UDDI 中将绑定文档注册为可重用的 `tModel`。`tModel` 是有关 Web 服务规范的信息。

使用统一资源定位器（URL）请求 WSDL 服务实例文档。*localhost* 端口号（此处由 *<yourWebAppServer>* 指定）取决于您自己的当前机器:

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDLservice
```

使用以下 URL 请求 WSDL 绑定文档:

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDLbinding
```

相关概念:

- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 102 页的『测试 Web 服务』

XML 模式定义

XML 模式定义在 Web 服务接口中使用的数据类型。通过统一资源定位器（URL）请求该服务的 XML 模式定义。*localhost* 端口号（此处由 *<yourWebAppServer>* 指定）取决于您自己的当前机器:

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/XSD
```

WORF 生成类似第 113 页的图 39 中的示例的 XML 模式文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="http://localhost:8080/services/sample/HelloSample.dadx/XSD"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://localhost:8080/services/sample/HelloSample.dadx/XSD">
  <element name="listDepartmentsResult">
    <complexType>
      <sequence>
        <element maxOccurs="unbounded" minOccurs="0" name="listDepartmentsRow">
          <complexType>
            <sequence>
              <element name="DEPTNO" type="string"/>
              <element name="DEPTNAME" type="string"/>
              <element name="MGRNO" nillable="true" type="string"/>
              <element name="ADMRDEPT" type="string"/>
              <element name="LOCATION" nillable="true" type="string"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

图 39. XML 模式定义文件

DB2® XML Extender 可以使用文档类型定义（DTD）来定义 XML 文档的模式，因此 WORF 运行时自动将 DTD 转换为 “XML 模式”。例如，如果 DTD *order.dtd* 定义 XML 文档，则可使用以下 URL 来请求转换为 “XML 模式”：

```
http://<yourWebAppServer>/services/db2sample/order.dtd/XSD
```

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 102 页的『测试 Web 服务应用程序 - 方案』
- 第 113 页的『Web 服务提供程序中存在的 Web 服务』

相关参考:

- 第 231 页的附录 C, 『DADX 文件的 XML 模式』

Web 服务提供程序中存在的 Web 服务

在 Web 应用程序目录或 Web 服务组中，可能存在大量可在网络上使用的 Web 服务。但在可以使用这些 Web 服务之前，您必须查找并获取有关这些服务的信息。Web 服务检查语言（WSIL）使此搜索过程变得更为轻松。

Web 服务检查语言文档

DB2® Web 服务提供查找所需 Web 服务操作的方法。通过使用 WORF，您可以检查应用程序或组中可用的全部 Web 服务。检查生成器产生的 XML 文档是您可用的 Web 服务的列表。如果您从组目录中运行生成器，则该列表是组目录级别的 Web 服务的报告。如果您从应用程序目录中运行生成器，则该列表是应用程序目录级别的 Web 服务的报告。通过在浏览器中输入 *<your-Web-server>:9080/<context_root_name>/inspection.wsil*，您可以从浏览器运行检查生成器。

此主题中的示例基于 WORF 样本和名为 *services* 的 WORF 样本应用程序

下列示例创建的 Web 服务列表可从名为 *services* 的 Web 应用程序中获得:

```
http://localhost:9080/services/inspection.wsil
```

应用程序级别的 inspection.wsil 类似如下:

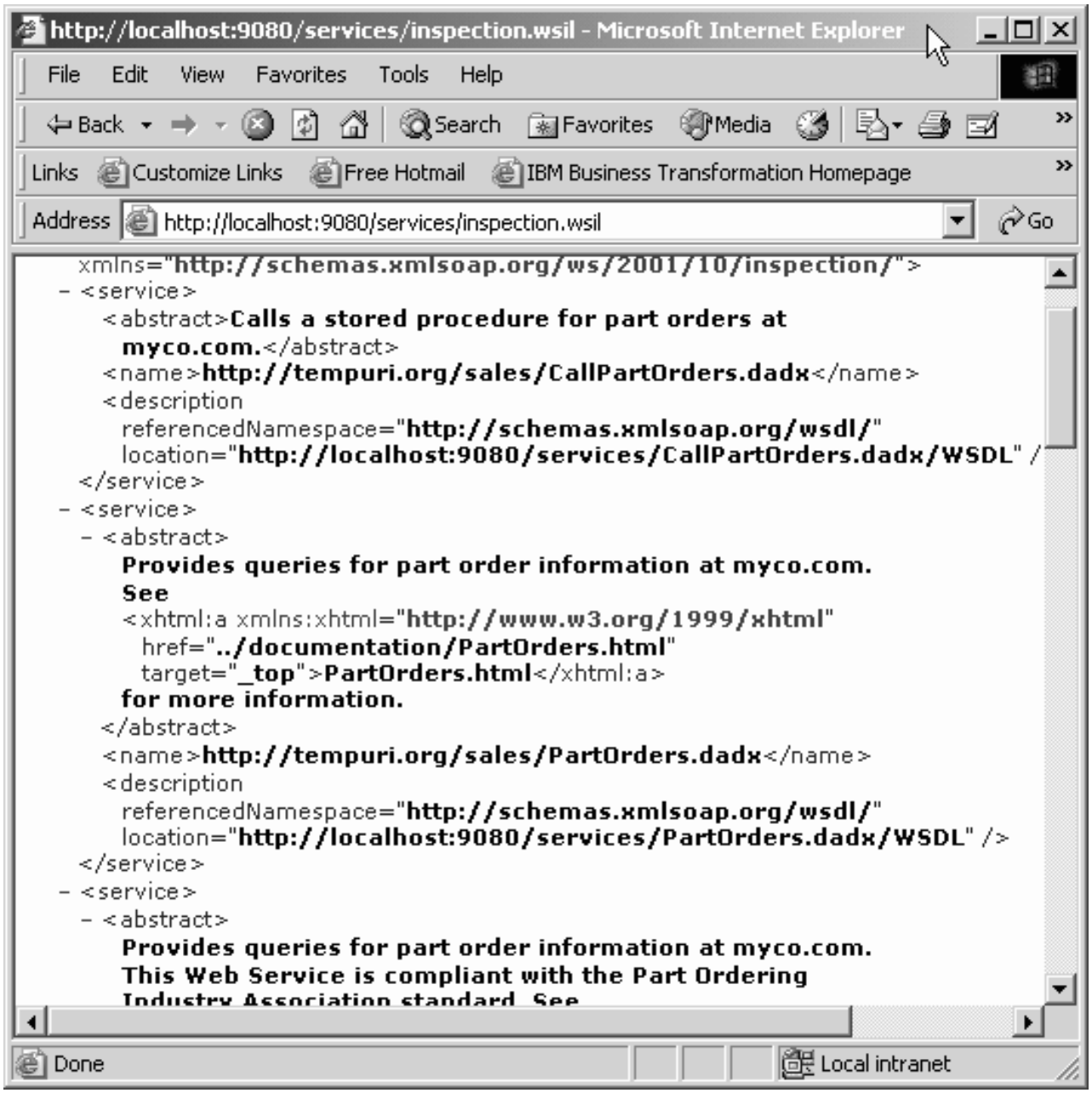


图 40. 应用程序级别的 WSIL

组级别的 inspection.wsil 类似如下:

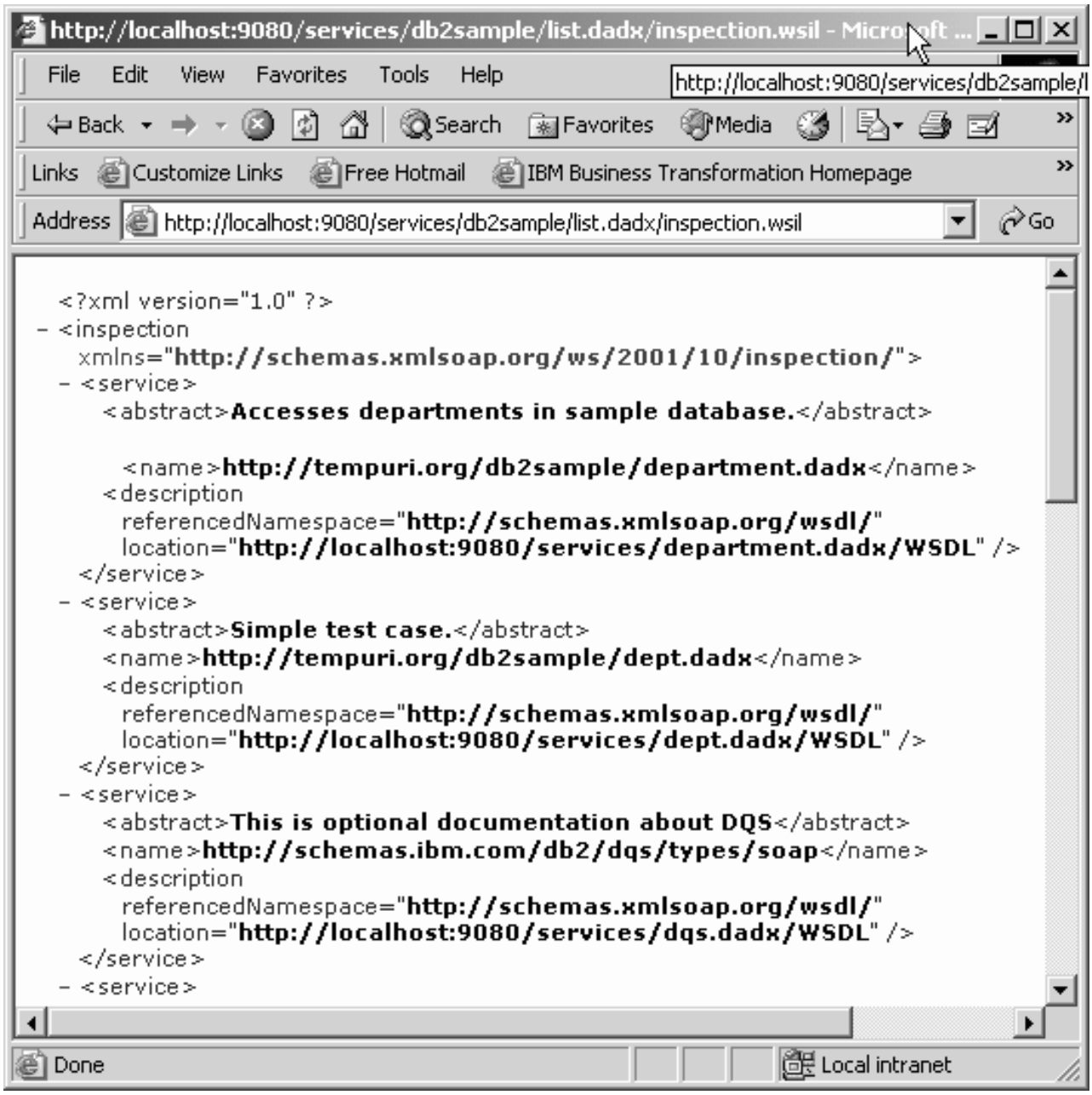


图 41. 组级别的 WSIL

要确保您生成的 WSIL 包括 Web 服务，Web 服务必须符合以下条件:

- 描述 Web 服务的 DADX 文件必须是有效的。
- 您必须定义属于 Web 服务的 group.properties 文件。
- web.xml 文件必须包含标识该组的适当的 servlet 映射。

用于 WSIL 的 web.xml 文件中的 servlet 映射类似于下列示例:

```
    <servlet>
        <servlet-name>wsil</servlet-name>
        <display-name>wsil</display-name>
        <servlet-class>
           com.ibm.etools.webservice.rt.wsil.servlet.WSILInvoker
</servlet-class>
```

```
        <init-param>
          <param-name>soap-engine</param-name>
          <param-value>apache-axis</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
<servlet-mapping>
        <servlet-name>wsil</servlet-name>
        <url-pattern>/inspection.wsil</url-pattern>
</servlet-mapping>
```

当调用 Web 服务时，WORF 会读取 web.xml 文件以获取有关如何运行该服务的信息，例如 servlet 信息和组信息。该 web.xml 文件包括了 servlet 定义，以便 WSIL 规范将正确的 Web 操作与 WORF 样本相关联。WORF 动态地生成 WSIL 文档，该文档包含 Web 应用程序服务器的组目录中的所有可用 Web 服务。在启动 Web 应用程序服务器之后，如果您将 Web 服务添加到 group.properties 文件中，则不需要重新启动服务器，即可启动 WSIL 生成器。

Web 服务列表页

您可以产生的另一种检查文档是 Web 服务列表页。Web 服务列表页是 HTML 文档，它包含应用程序目录或 Web 应用程序服务器的组目录中的所有可用 Web 服务的列表。该列表还包含指向服务的 WORF 样本和 WSDL 的链接。您可以通过在浏览器中输入下列 URL 来访问此页面:

应用程序级别

```
        <your-Web-server>:9080/<context_root_name>/LIST
```

组级别

```
        <your-Web-server>:9080/<context_root_name>/<group name>/LIST
```

列表页的 web.xml 文件中的 servlet 映射和 URL 模式应类似于下列示例:

```
<servlet>
        <servlet-name>list</servlet-name>
        <display-name>list</display-name>
        <servlet-class>
          com.ibm.etools.webservice.rt.list.servlet.ListInvoker
</servlet-class>
        <init-param>
          <param-name>soap-engine</param-name>
          <param-value>apache-axis</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>

<servlet-mapping>
        <servlet-name>list</servlet-name>
        <url-pattern>/LIST</url-pattern>
</servlet-mapping>
```

组级别的列表页类似如下:

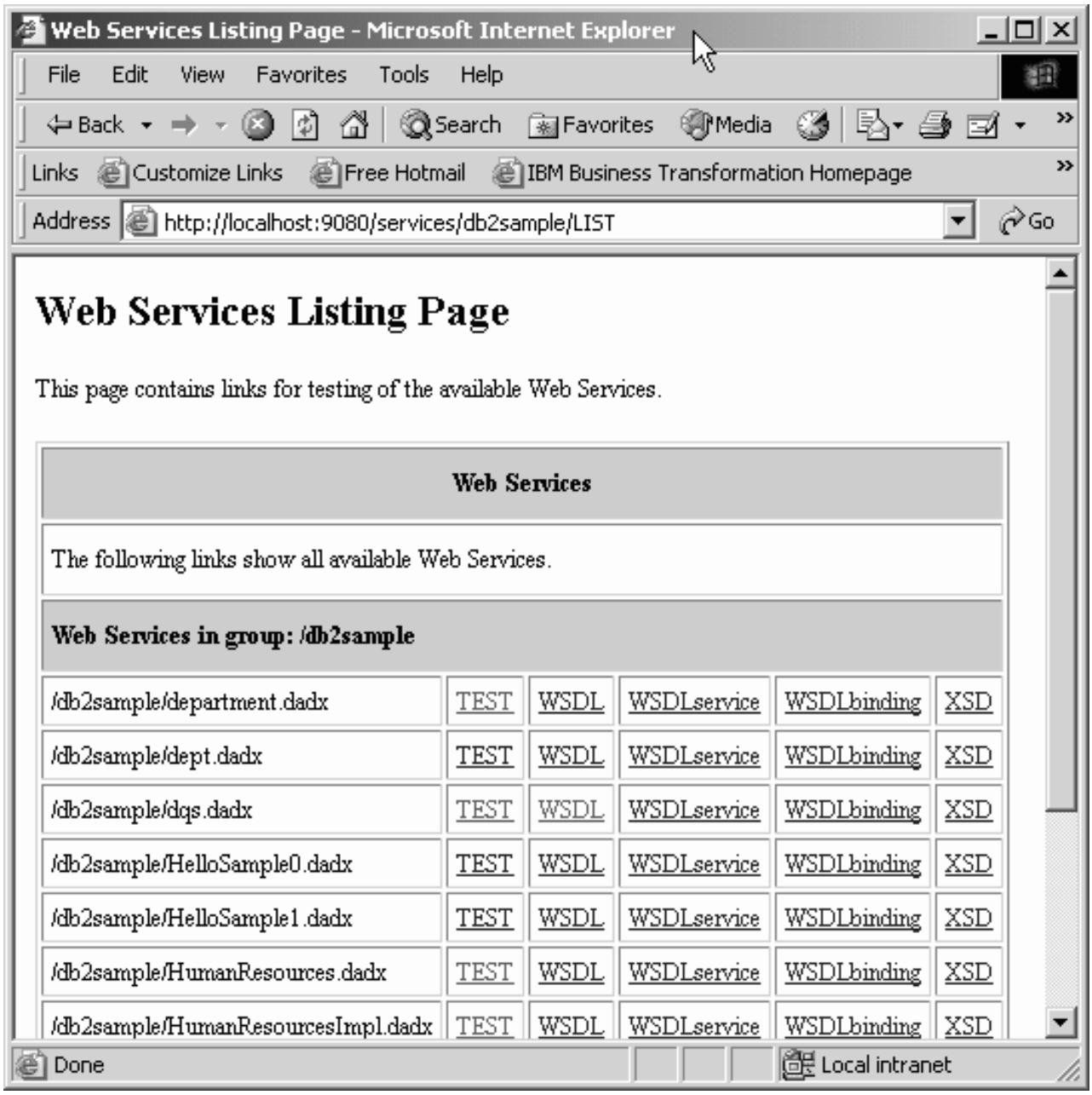


图 42. Web 服务列表页

相关概念:

- 第 112 页的『XML 模式定义』

相关任务:

- 第 102 页的『测试 Web 服务』
- 第 54 页的『定义 web.xml 和 group.properties 文件』
- 第 36 页的『安装或迁移 WORF 以使用 WebSphere Application Server for Windows and UNIX V5 或更新版本』
- 第 37 页的『在 WebSphere Application Server for Windows and UNIX V5.1 或更新版本上部署 WORF 示例』

Web 服务文档

可以对整个服务和对每个操作在 DADX 中包括文档。图 43 说明了如何添加文档:

```
<?xml version="1.0" encoding="UTF-8"?>
<DADX
  xmlns="http://schemas.ibm.com/db2/dxx/dadx"  >
  <documentation>
    Simple DADX example that accesses the SAMPLE database.
  </documentation>
  <operation name="listDepartments">
  <documentation>
      Lists the departments.
      </documentation>
    <query>
      <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
    </query>
  </operation>
</DADX>
```

图 43. HelloSample.dadx

该文档可包含任何有效的“可扩展标记语言”（XML）。为了在浏览器中正确显示，应使用 XHTML。如果使用 XHTML，则定义该文档的 XHTML 名称空间。当请求测试页时，它还包括文档:

图 44. 带有文档的 WORF 测试页

相关概念:

- 第 102 页的『测试 Web 服务应用程序 - 方案』

相关参考:

- 第 69 页的『一个简单的 DADX 文件』
- 第 62 页的『DADX 文件的语法』

Web 服务自动重新装入

在开发过程中，您可能会经常更改 DADX 文件。WORF 允许您在应用程序服务器正在运行期间更改您的 DADX 文件，并自动重新装入带有新的更新的 DADX 文件。自动重新装入使开发 DADX Web 服务与开发 Java™ Server Page 一样简单。当将 DADX Web 服务部署至生产服务器时，可关闭自动重新装入。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 59 页的『定制 group.properties 文件』
- 第 102 页的『测试 Web 服务』

Web 服务样本 - PartOrders.dadx

此主题中的示例使用名为 dxx_sales_db 的数据库样本。这是在随 DB2 XML Extender 和 WORF 提供的文档和样本中使用的样本数据库。dxx_sales_db 数据库存储有关部件订单的信息。

假设您必须提供根据下列条件检索订单的“Web 服务”：

- 查找所有订单
- 查找指定颜色的部件的所有订单
- 查找其价格高于或等于最低价格的所有订单

您创建名为 *PartOrders.dadx* 的包含下列操作的 DADX 文件:

- findAll
- findByColor
- findByMinPrice

通过将 *PartOrders.dadx* 文件部署到 services Web 应用程序来创建“Web 服务”。这是用 WORF 的 dxx_sales_db 实例配置的。此文件的部署位置是 *WEB-INF/classes/groups/dxx_sales_db/PartOrders.dadx*。

“Web 服务”支持通过下列协议进行访问:

- 超文本传输协议（HTTP）GET
- HTTP POST
- HTTP SOAP

HTTP GET 和 POST 对于从 Web 浏览器的简单访问很有用。在这种情况下，请求使用 application/x-www-form-urlencoded 的内容类型。

例如，假设您在主机 www.mycompany.com 上部署 Web 服务。下列 URL 将使用 HTTP GET 调用 Web 服务:

- `http://www.mycompany.com /services/sales/PartOrders.dadx /findAll`
- `http://www.mycompany.com /services/sales/PartOrders.dadx /findByColor?color=red`

- `http://www.mycompany.com /services/sales/PartOrders.dadx /findByMinPrice?minprice=20000`

此语法将统一资源定位器（URL）中的方法编码为额外路径信息，并将参数编码为查询字符串。对这些请求的响应具有 `text/xml` 内容类型。对于 HTTP POST，在查询的主体中发送查询字符串（而不是在 URL 中发送），但其内容类型仍然是 application/x-www-form-urlencoded。以下是使用“传输控制协议”（TCP）跟踪实用程序进行捕获时 HTTP POST 请求的一个示例。示例同时显示 HTTP 头和主体:

```
POST /services/sales/PartOrders.dadx/findByColor
HTTP/1.1
User-Agent: Java1.3.0
Host: localhost:9081
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-length: 12
color=red+++
```

由 DADX 文件定义的“Web 服务”是自描述的。它动态地生成文档和测试页、WSDL 文档以及“XML 模式”。以下 HTTP GET URL 请求文档和测试页:

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/TEST
```

以下 HTTP GET URL 请求服务的 WSDL 描述:

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/WSDL
```

对于 HTTP SOAP，通过使用 POST 将 SOAP 包络发送至以下 URL 来调用服务:

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/SOAP
```

但使用请求内容类型 `text/xml` 而不是 `application/x-www-form-urlencoded`。以下示例是使用 TCP 监视器进行跟踪的 SOAP 请求。它与构建到 WebSphere Studio 中或作为 Apache SOAP 的一部分的 SOAP 请求类似。此示例包括 HTTP 头信息和 HTTP 主体:

```
POST /services/sales/PartOrders.dadx/SOAP
HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 547
SOAPAction: "http://tempuri.org/sales/PartOrders.dadx"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:findByColor xmlns:ns1="http://tempuri.org/sales/PartOrders.dadx" SOAP-
ENV:encodingStyle="http://xml.apache.org/xml-soap/literalxml">
<color xsi:type="xsd:string" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">red </color>
</ns1:findByColor>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

.

PartOrder DADX 文件

PartOrders.dadx 使用 <retrieveXML> 运算符（它使用 XML 集合存取方法）来实现其所有三个操作。通常，每个操作可使用不同的运算符和存取方法。

```
<?xml version="1.0"?>
  <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xhtml="http://www.w3.org/1999/xhtml">
      <documentation>
      Provides queries for part order information at myco.com.
      See <xhtml:a href="../documentation/PartOrders.html" target="_top">
       PartOrders.html</xhtml:a> for more information.
      </documentation>

    <operation name="findAll">
      <documentation>

        Returns all the orders with their complete details.
      </documentation>
      <retrieveXML>
        <DAD_ref>getstart_xcollection.dad</DAD_ref>
         <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
         order by order_key, part_key, ship_id
         </SQL_override>
      </retrieveXML>
    </operation>
```

图 45. PortOrder.DADX 文件 (1/3)

```
<operation name="findByColor">
      <documentation>
Returns all  the orders that include one or
      more parts that have the specified
      color, and only shows the details for those parts.
</documentation>

      <retrieveXML>
        <DAD_ref>getstart_xcollection.dad</DAD_ref>
        <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
           and color = :color
         order by order_key, part_key, ship_id
        </SQL_override>
        <parameter name="color" type="xsd:string"/>
      </retrieveXML>
    </operation>
```

图 45. PortOrder.DADX 文件 (2/3)

```
 <operation name="findByMinPrice">
      <:documentation>
Returns all the orders that include one or more
   parts that have a price greater than
  or equal to the specified minimum price,
and only shows the details for
  those parts.
</documentation >
      <retrieveXML>
        <DAD_ref>
           getstart_xcollection.dad
       </DAD_ref>
        <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
           and p.price >= :minprice
         order by order_key, part_key, ship_id
        </SQL_override>
        <parameter name="minprice" type="xsd:decimal"/>
      </retrieveXML>
    </operation>

  </DADX>
```

图 45. PortOrder.DADX 文件 (3/3)

相关概念:

- 第 104 页的『访问带有 GET、POST 和 SOAP 绑定的 Web 服务』
- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 102 页的『测试 Web 服务应用程序 - 方案』
- 第 106 页的『SOAP 绑定』

相关任务:

- 第 102 页的『测试 Web 服务』

部署和测试 Web 应用程序

现在，可根据设计工作，编写查询和编写应用程序来实现您的业务目标。然后可在 Web 上部署这些程序。可以许多方式部署同一组文件。

安装 Web 应用程序

可使用 Web 归档文件（WAR）来封装、分发和安装 Web 应用程序。通常将组成 Web 应用程序的文件封装在单个 WAR 文件中以进行部署。WAR 文件可包含 *web.xml* 服务器配置文件、group.properties 配置文件和 DAD 及 DADX 文件。您在在用于 Windows 和 UNIX 的 WebSphere Application Server V5 或更新版本上部署 WORF 示例中部署的 WORF 样本包含两个 WAR 文件示例: *apache-services.war* 和 *axis-services.war*。某些开发环境（比如 WebSphere® Studio）提供了自动部署功能。但是，如果您想要为 Web 服务提供您自己的部署描述符文件，则可禁用此自动部署功能。

相关任务:

- 第 127 页的『准备和创建 Web 归档文件』
- 第 37 页的『在 WebSphere Application Server for Windows and UNIX V5.1 或更新版本上部署 WORF 示例』

Java 2 Enterprise Edition 应用程序

当部署 Java™ 2 Enterprise Edition (J2EE) 应用程序时，必须为电子商务应用程序构建下列组件:

Web 归档文件（WAR）
与 Web 相关的组件（HTML、JavaScript™ 和 JavaServer Pages）

Java 归档文件（JAR）
构成业务逻辑组件的 Java 类

企业归档（EAR）
构成企业解决方案的 Java 归档文件和 Web 归档文件

在 WebSphere® Application Server 5.0 中，可部署的最小单元是 Web 归档文件。如果应用程序正在开发 Enterprise JavaBeans™，则 Java 归档文件和企业归档文件是必需的。

相关概念:

- 第 153 页的『在 IBM DB2 Information Integrator 中设计查询的优点』
- 第 53 页的『定义一组 Web 服务』

在 DB2 Information Integrator 中安装 DB2 的应用程序服务器

应用程序服务器使企业能够开发、部署和集成下一代的电子商务应用程序。可以将应用程序服务器用作管理 Web 应用程序的工具。

从版本 8.1.2 开始，DB2® 通用数据库提供了称为 DB2 的应用程序服务器的嵌入式应用程序服务器。如果使用 DB2 UDB 的应用程序服务器，则无需安装单独的应用程序服务器即可在 Windows®、Linux、AIX 和 Solaris 上运行 DB2 UDB Web 应用程序。

先决条件:

- DB2 通用数据库 ESE 版本 8.1.2 或更高版本
- 必须存在至少一个 DB2 UDB 实例
- 根据环境，发出以下命令:

```
<db2instance path>/sqllib/db2profile（用于 Windows）
<db2instance path>/sqllib/db2profile（用于 UNIX 系统）
```

限制:

具有一个或多个 DB2 UDB 实例的系统中只能有一台 DB2 应用程序服务器。

过程:

从*用于 DB2 的 Java 应用程序开发和 Web 管理工具补遗* CD 安装 DB2 的应用程序服务器。DB2 Universal Database™（DB2 通用数据库）随 DB2 Universal Database™ 安装软件包一起提供此 CD。要安装 DB2 的应用程序服务器:

```
db2appserverinstall
     -asroot path
     -hostname name
```

-asroot
应用程序服务器安装的绝对路径。

-hostname
主机系统的名称。

相关任务:

- 『为 DB2 安装应用程序服务器』（《安装与配置补充手册》）
- 『为 DB2 卸载应用程序服务器』（《安装与配置补充手册》）

在 Information Integrator 中启动和停止 DB2 的应用程序服务器

可以从 DB2 目录的应用程序服务器 *bin* 子目录启动和停止 DB2 UDB 的应用程序服务器。还可以使用名为 DB2EAS.SERVER 的存储过程来启动和停止应用程序服务器。

过程:

要启动和停止 DB2 UDB 的应用程序服务器，请使用下列命令:

```
startServer <serverName>
stopServer <serverName>
```

启动和停止命令需要以下参数:

serverName
要启动的应用程序服务器的名称。

请参阅 *WebSphere Application Server System Administration* 以获取有关部署和管理应用程序的信息，以便可以将 WORF 样本与 DB2 应用程序服务器一同部署。部署 WORF 样本与 DB2 应用程序服务器之后，可以从浏览器访问 WORF 测试页。

相关任务:

- 『在本地启动 DB2 的应用程序服务器』（《安装与配置补充手册》）
- 『在本地停止 DB2 的应用程序服务器』（《安装与配置补充手册》）
- 『远程启动 DB2 的应用程序服务器』（《安装与配置补充手册》）
- 『远程停止 DB2 的应用程序服务器』（《安装与配置补充手册》）
- 第 37 页的『在 WebSphere Application Server for Windows and UNIX V5.1 或更新版本上部署 WORF 示例』
- 第 123 页的『在 DB2 Information Integrator 中安装 DB2 的应用程序服务器』

生成部署描述符

在使用 Apache SOAP 的情况下，WebSphere Application Server 5.1 使用定制 ConfigManager（com.ibm.soap.server.XMLDrivenConfigManager），它在运行时禁用部署和反部署。相反，WebSphere 读取在应用程序启动时列示已部署服务的文件。对于 OS/390 和 z/OS 平台，总是启用自动部署，因此不必生成自己的部署描述符。

过程:

要创建此文件，必须为每个 Web 服务或 DADX 文件创建部署描述符文件，它标识了配置和部署信息。将该文件插入名为 dds.xml 的“可扩展标记语言”（XML）文件，WebSphere ConfigManager 在 Web 应用程序启动时读取该 XML 文件。Apache SOAP 配置管理器在运行时按需要部署 Web 服务，因此不必将这些服务添加至部署描述符文件。

1. 验证 *worf.jar* 和 *soap.jar* 文件是否位于您的类路径中。
2. 使用 DADX2DD 命令从 DADX 文件生成部署描述符。对于 Apache Axis，不需要部署描述符文件。将类 `com.ibm.etools.webservice.rt.dadx.Dadx2Dd` 与下列参数配合使用:

   -r 与组有关的 Web 服务的资源名称。部署描述符需要此参数。

   -p 组路径。部署描述符需要此参数。

   -n 组名。部署描述符需要此参数。

   -i DADX 文件的名称，或者包含一个或多个 DADX 文件的目录。此目录可以包含一个或多个子目录，而子目录中包含一个或多个 DADX 文件。如果使用目录名，请使用 Web 应用程序的根目录，例如包含文件 dds.xml 和 WEB-INF 的目录。*–i* 参数是可选的。如果使用此参数，则 DADX 文件必须存在并且可读。如果不使用此参数，则为来自标准输入的 DADX 文件的名称。要指示标准输入，使用短划线（–）。

   -o 部署描述符文件名。此自变量是可选的。如果使用此参数，则文件必须可写。如果文件已存在，则会覆盖它。如果不使用此参数，则部署描述符文件写至标准输出。要指示标准输出，使用短划线（–）。

   -s 目标 SOAP 引擎。有效值包括 *apache-soap* 和 *apache-axis*。

   当调用 Web 服务时，WORF 会读取 web.xml 文件以确定装入哪个 SOAP 引擎类。如果不能装入 SOAP 引擎，则使用缺省 SOAP 引擎。如果未在 web.xml 文件中指定 soap-engine 参数，则缺省 soap 引擎是 Apache Axis。对于 Apache SOAP，`Dadx2Dd` 命令的输出可以插入到 ddx.xml 文件中。对于 Apache Axis，部署描述符始终在运行时动态生成。输出仅供参考。

   可以在执行 Web 服务时切换 SOAP 引擎。在切换 SOAP 引擎之前，必须确保部署描述符位于 dds.xml 文件中。dds.xml 文件仅用于 Apache SOAP。对于 Apache Axis，不需要部署描述符文件。在切换至 Apache Axis 之前，请确保 deploy.wsdd 文件对您的应用程序可用。

   可以在未连接到 SOAP 引擎的情况下运行部署描述符生成器（`Dadx2Dd`）。

   例如，如果当前目录是 *WEB-INF*，则以下命令从 dxx_travel 组读取 *ZipCity.dadx* 文件。然后，它将部署描述符写至 *dds* 子目录:

   ```
   java com.ibm.etools.webservice.rt.dadx.Dadx2Dd -r ZipCity.dadx -p \travel
      -n \dxx_travel -i classes\groups\dxx_travel\ZipCity.dadx
       -o classes\dds\dxx_travel\ZipCity.isd
   ```

3. 复制新创建的 Apache SOAP ISD 文件的内容并将其添加至 Web 应用程序目录中的 *dds.xml* 文件中。

```
<?xml version='1.0'?>

<dds>

<isd:service xmlns:isd='http://xml.apache.org/xml-soap/deployment'
  id='http://tempuri.org/travel/ZipCity.dadx'>
   <isd:provider
    type='com.ibm.etools.webservice.rt.framework.ServiceProvider'
    scope='Request'
    methods='findCityByZipCode insertZipCodeAndCity
       updateCityForZipCode deleteZipCode'>
    <isd:java class='com.ibm.etools.webservice.rt.dxx.DxxService'/>
    <isd:option key='group.name' value='/dxx_travel'/>
    <isd:option key='group.path' value='/travel'/>
    <isd:option key='group.class.name'
           value='com.ibm.etools.webservice.rt.dxx.DxxGroup'/>
   </isd:provider>
 <isd:faultListener>org.apache.soap.server.DOMFaultListener
 </id:faultListener>
 <isd:mappings
    defaultRegistryClass=
    'com.ibm.etools.webservice.rt.dxx.DxxMappingRegistry'/>
</isd:service>
...
...
</dds>
```

图 46. 具有 isd 文件的 dds.xml 文件部分的示例

4. 重新启动 Web 应用程序。

相关概念:

- 第 126 页的『Apache SOAP 配置』
- 第 122 页的『安装 Web 应用程序』
- 第 53 页的『定义一组 Web 服务』

相关任务:

- 第 102 页的『测试 Web 服务』

Apache SOAP 配置

如果您使用 Apache SOAP 引擎，则可以配置 Web 应用程序以使用 Apache 配置管理器（缺省情况）或 IBM® 配置管理器（XMLDrivenConfigManager）。Apache 配置管理器在名为 DeployedServices.ds 的串行化 Java™ 文件中存储已部署的服务。XMLDrivenConfigManager 使用 XML 格式，并在添加新的 DADX 文件时禁用自动部署 Web 服务，因此必须手工部署它们（请参阅生成部署描述符）。

为了帮助您部署样本应用程序，WORF 在随 WORF 引擎提供的 *services.war* 文件中提供了一个配置文件（*dds-example.xml*）。*dds-example.xml* 是可以与示例中的所有 DADX 文件配合使用的部署描述符文件。样本 *dds.xml* 文件部署新的服务。如果调用未部署的服务，则服务器将报告该服务未知。

如果希望使用 IBM 配置管理器，请将 *soap-ibm.xml* 重命名为 *soap.xml*，并将 *dds-example.xml* 重命名为 *dds.xml*。重新启动应用程序服务器以使用 installedApps\servicesApp.ear\services.war 子目录中的 XMLDrivenConfigManager。

相关概念:

- 第 29 页的『Web 服务过程概述』

相关任务:

- 第 124 页的『生成部署描述符』

准备和创建 Web 归档文件

要创建 Web 归档（WAR）文件，执行下列操作:

过程:

1. 为 WAR 文件创建基本目录结构，如以下示例中所示:

   ```
   WEB-INF\lib\worf-servlets.jar
   WEB-INF\web.xml

   files from the worf\ that you downloaded
   ```

   可以在 apache-services.war 文件或 axis-services.war 文件中查找此 WORF 目录层次结构。当运行 TEST 页时，将使用这些文件。如果不打算使用 WORF 的内置测试工具，则 *worf* 子目录中的文件不是必需的。worf-servlets.jar 文件在安装 WORF 的 *lib* 子目录中。web.xml 是标准 J2EE web.xml。空的 web.xml 将与图 47 中的示例相似。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
</web-app>
```

图 47. 空 web.xml 文件

2. 对于每个组，
   a. 创建您自己的组子目录（例如，WEB-INF\classes\groups\myGroup）并在该子目录中包括 group.properties 和 DADX 文件。
   b. 编辑 WEB-INF\web.xml 文件以添加 servlet 和 servlet 映射（例如，添加名为 *myGroup* 的 servlet 和名为 *myURLPath* 的 URL 映射）。
   c. 可选: 生成部署描述符（对于 OS/390 和 z/OS 平台可以跳过此步骤）。只有 IBM 配置管理器需要此步骤。在以下示例中，第一条指令在 Windows 平台上将当前目录更改为 WEB-INF 子目录。然后，运行应用程序。当应用程序运行完成后，可以更改回根目录。

      ```
      cd WEB-INF
      java com.ibm.etools.webservice.rt.dadx.Dadx2Dd
        -r ivt.dadx
        -p \myURLPath
        -n \myGroup
       -i classes\groups\myGroup\ivt.dadx
       -o classes\dds\myGroup\ivt.dadx
      cd ..
      ```

注: ivt.dadx 是交付的特定样本；在新的 WAR 文件中可能没有此文件。

d. 可选: 创建或修改文件 *dds.xml* 以添加生成的描述符的内容（对于 OS/390 和 z/OS 平台，跳过此步骤）。只有 IBM 配置管理器需要此步骤。空 dds.xml 文件看起来类似以下内容:

```
<?xml version='1.0'?>
<dds>
</dds>
```

3. 使用下列任一种方法创建 WAR 文件:
   - 通过命令行发出以下命令:

     ```
     jar -cvf minWORFwar.war WEB-INF worf
     ```

   - 从 WebSphere Studio 通过选择菜单中的**文件**；然后选择**导出**，之后选择 **WAR 文件**。选择 Web 应用程序所在的项目名并指定文件名。
4. 如为 WebSphere Application Server 或 Apache Jakarta Tomcat 安装的样本中描述的那样部署 WAR 文件（例如，将 myContext 作为 Web 应用程序上下文）。
5. 通过运行 TEST 页来验证已正确创建了 WAR 文件。例如，TEST 的 URL 可能类似以下 URL:

   ```
   http://<your WebAppServer>/myContext/myURLPath/ivt.dadx/TEST
   ```

   注: ivt.dadx 是交付的特定样本；在新的 WAR 文件中可能没有此文件。

相关概念:

- 第 53 页的『定义一组 Web 服务』

相关任务:

- 第 59 页的『定制 group.properties 文件』
- 第 57 页的『在 iSeries 平台中定义 web.xml 和 group.properties 文件』

Web 服务提供程序跟踪

部署 Web 服务之后，可能需要从 Web 服务提供程序获取有关运行时事件和诊断的信息。要调试 Web 服务应用程序和排除该应用程序的故障，DB2® Web 服务使用应用程序在其上运行的 Web 应用程序服务器的跟踪设施。从 Web 应用程序服务器接收到的跟踪信息包括消息和事件活动。

DB2 Web 服务提供程序支持两种跟踪系统:

log4j Apache Jakarta Tomcat 4.0.6 和更新版本上的 Jakarta-log4j-1.2.8 和 commons-logging-1.0.3

JRas WebSphere® Application Server V5 和更新版本使用的跟踪和日志记录系统

通过这些跟踪系统，可以将消息日志记录和跟踪设施并入 Java™ 应用程序。

从应用程序启用的跟踪生成的输出出现在您使用的 Web 应用程序服务器的根目录中。表 17 显示输出日志文件的位置:

表 17. 跟踪输出位置

服务器	输出日志文件位置
WebSphere Application Server	${SERVER_LOG_ROOT}/trace.log
Apache Jakarta Tomcat	<tomcat>/logs/worf_log4j.log

所有跟踪消息和事件均由 Web 服务的操作名称、servlet 的名称或 DADX 文件的名称标识。

DB2 Web 服务提供程序可以跟踪下列类型的事件:

参考消息
指示 Web 服务请求事件或 Web 服务响应事件何时成功完成（例如，何时成功地对 DADX 文件进行语法分析）的消息。

警告消息
指示在处理 Web 服务请求或 Web 服务响应期间何时检测到警告条件的消息，例如 XML 解析器对 DADX 文件发出的警告消息。

错误消息
指示在处理 Web 服务请求或 Web 服务响应期间何时检测到错误（例如，应用程序何时产生异常）的消息。

跟踪事件
指示应用程序何时进入或退出方法、异常、调用堆栈或变量值的事件。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』

相关任务:

- 第 129 页的『为 DB2 Web 服务提供程序 Apache Tomcat V4.0 或更新的 Web 应用程序服务器启用跟踪』
- 第 130 页的『为 DB2 Web 服务提供程序 WebSphere Application Server 启用跟踪』
- 第 132 页的『为 DB2 Web 服务提供程序 WebSphere Studio Application Developer 启用跟踪』

相关参考:

- 第 52 页的『对 Web 服务进行故障诊断』

为 DB2 Web 服务提供程序 Apache Tomcat V4.0 或更新的 Web 应用程序服务器启用跟踪

您可以配置 Apache Tomcat 服务器以跟踪 DB2 Web 服务。

先决条件:

需要具有修改所用服务器的配置的权限。

过程:

要修改 DB2 Web 服务提供程序的缺省 `log4j` 跟踪:

1. 创建名为 `log4j.configuration` 的配置文件，该文件具有下列示例中所示的条目:

```
log4j.rootCategory=DEBUG, console, rollingFile
log4j.logger.com.ibm.etools.rt.webservice.*=INFO
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%n
```

```
log4j.appender.rollingFile=org.apache.log4j.RollingFileAppender");
log4j.appender.rollingFile.File=<servletContext>\..\..\logs\worf_log4j.log
log4j.appender.rollingFile.MaxFileSize=100KB
log4j.appender.rollingFile.layout=org.apache.log4j.TTCCLayout
log4j.appender.rollingFile.layout.layout.ConversionPattern=%p %t %c - %m%n
```

2. 修改配置文件中的设置以便仅显示某些类型的消息:

表 18. log4j 配置文件的消息设置

消息类型	配置设置
log4j 警告消息或更高级别的消息	log4j.logger.com.ibm.etools.webservice.*
log4j 参考消息	log4j.logger.com.ibm.etools.webservice.*=INFO
log4j 错误消息	log4j.logger.com.ibm.etools.webservice.*=ERROR

3. 将配置文件 `log4j.configuration` 放置在 Web 应用程序的 WEB-INF/classes 目录中。

可以在 <installed Web server location>\AppServer\logs\<local server name> 处看到跟踪事件的日志。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 128 页的『Web 服务提供程序跟踪』

相关任务:

- 第 48 页的『在 Apache Jakarta Tomcat 上安装和部署 WORF 示例』

为 DB2 Web 服务提供程序 WebSphere Application Server 启用跟踪

可以将 WebSphere Application Server 配置为从管理控制台跟踪 DB2 Web 服务。

先决条件:

需要具有修改所用服务器的配置的权限。

过程:

要修改 DB2 Web 服务提供程序的缺省 jRAS 跟踪:

1. 启动 WebSphere Application Server 管理控制台。
2. 在“导航”树中，单击**故障排除 —> 日志和跟踪**。“日志记录和跟踪”窗口打开。
3. 在“日志记录和跟踪”窗口中，单击服务器名称，然后选择**诊断跟踪**。
4. 在**跟踪规范**字段中，输入跟踪字符串:

   ```
   com.ibm.etools.webservice.*=all=enabled
   ```

5. 如果服务器停止，则转至“配置”页。如果服务器在运行，则转至“运行时”页。
   - 可选: 从“运行时”页，选择**保存跟踪**复选框，将所做的更改写入服务器配置中。如果清除**保存跟踪**复选框，您所做的更改仅应用于当前运行的服务器进程列表。
   - 可选: 从“配置”页中，选择**启用跟踪**复选框。

图 48 中包含服务器未运行的情况下启用跟踪的示例。

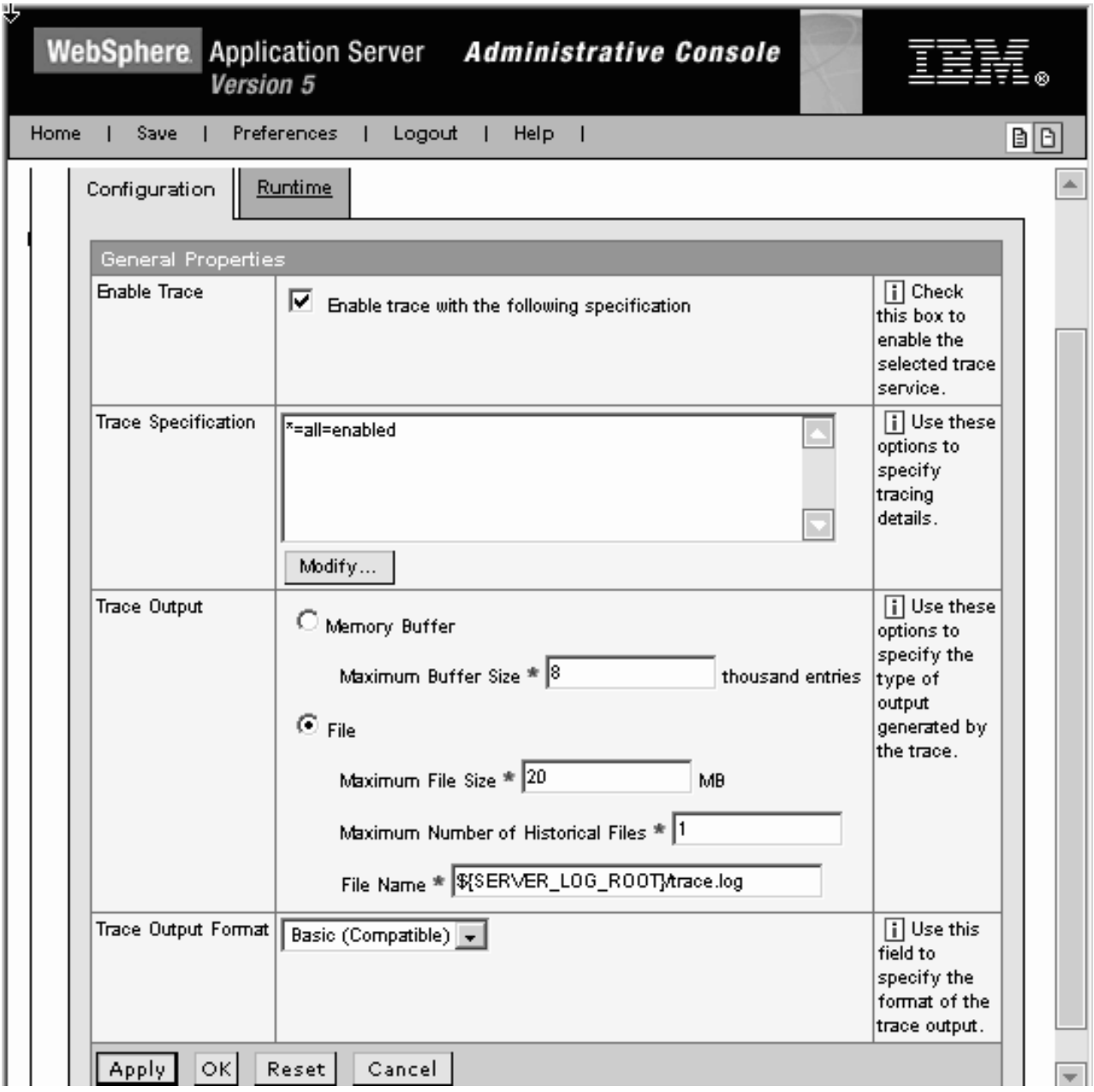


图 48. 启用 Web 服务提供程序跟踪

6. 保存所做的更改，然后重新启动服务器。

可以在 <installed Web server location>\AppServer\logs\<local server name> 处看到跟踪事件的日志。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 128 页的『Web 服务提供程序跟踪』

相关任务:

- 第 132 页的『为 DB2 Web 服务提供程序 WebSphere Studio Application Developer 启用跟踪』

为 DB2 Web 服务提供程序 WebSphere Studio Application Developer 启用跟踪

可以将 WebSphere Studio 配置为从管理控制台跟踪 DB2 Web 服务。

先决条件:

需要具有修改所用服务器的配置的权限。

过程:

要修改 DB2 Web 服务提供程序的缺省 jRAS 跟踪:

1. 启动 WebSphere Studio 管理控制台。
2. 从主菜单中，单击**窗口 —> 显示视图 —> 服务器配置**以打开“服务器配置”视图
3. 在**服务器**菜单上，双击 **WebSphere v5.0 测试环境**以打开服务器编辑器。
4. 转至“跟踪”页。
5. 在**跟踪规范**字段中，输入下列跟踪字符串:

   ```
   com.ibm.etools.webservice.*=all=enabled
   ```

6. 选择**启用跟踪**复选框。
7. 保存所做的更改，然后重新启动服务器。

可以在 <installed Web server location>\AppServer\logs\<local server name> 处看到跟踪事件的日志。

相关概念:

- 第 27 页的『Web 服务提供程序功能部件』
- 第 128 页的『Web 服务提供程序跟踪』

相关任务:

- 第 130 页的『为 DB2 Web 服务提供程序 WebSphere Application Server 启用跟踪』

发布 Web 服务

Web 服务提供者发布其 Web 服务，以便客户机可基于“超文本传输协议”（HTTP）使用“简单对象访问协议”（SOAP）来访问它们。这与 Enterprise Java™ Bean（EJB）客户机形成对比，EJB 客户机基于“因特网 ORB 间协议”（IIOP）使用远程方法调用（RMI）来存取 bean。Web 服务通过调用适当的业务函数并通常返回响应来处理 Web 客户机的请求。“Web 服务描述语言”（WSDL）文档描述 Web 服务。在资源库（例如，UDDI 注册中心）中或 Web 服务提供程序的服务器上存储 WSDL。在适当的资源库中存储 Web 服务描述将使感兴趣的客户可能发现 Web 服务存在，这样就可能为 Web 服务提供者带来新的业务。

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』

相关任务:

• 第 102 页的『测试 Web 服务』

安装和使用 Web 服务使用程序

DB2 通用数据库可作为使用程序优化对 Web 服务提供程序的存取。通过使用 SQL 语句，可使用和集成 Web 服务数据。通过使用 SQL 来存取 Web 服务数据，可大大减少工作量，原因是您可以处理 SQL 语句上下文中的数据。然后，您可以将该语句返回至客户机应用程序。Web 服务使用程序工具组帮助从 SQL 存取 Web 服务数据。Web 服务使用程序将现有 WSDL 接口转换为 DB2 表或标量函数。本节描述 IBM 为将 WSDL 转换为 DB2 SQL 函数而提供的 Web 服务使用程序独立工具和 WebSphere Studio 插件。

安装 Web 服务使用程序用户定义的函数

先决条件:

Web 服务使用程序用户定义的函数（UDF）在下列平台（所有平台都是 32 位）上可用:

- Windows 2000
- Linux
- AIX
- Solaris Operating Environment（带有 DB2 通用数据库版本 8 修订包 2）

在执行 SOAP UDF 之前，应安装以下软件:

- DB2 通用数据库
  - 版本 8（包含 Xerces 解析器和 XML Extender）
- 可选: WebSphere Studio Application Developer（WSAD）V5.1.1

  插件要求 WebSphere Studio 从 WSDL 生成 Web 服务 UDF。可以直接调用 Web 服务使用程序，并且可以开发自己的 SQL 函数。

还必须使用 **dxxadm enable_db sample** 命令启用 DB2 XML Extender。有关 DB2 XML Extender 命令的更多选项，请参阅 *DB2 XML Extender Administration and Programming*。

过程:

要启用（或安装）和禁用 Web 服务使用程序:

1. 执行以下实用程序来注册五个用户定义的函数:

   ```
   db2enable_soap_udf  -n dbName [-u uID] [-p password] [-force]
   ```

   参数定义如下:

   dbName
   数据库名称

   uID 可选: 用户标识

   password
   可选: 与用户标识关联的密码

   –force 尝试删除任何现有函数

enable 命令使数据库可以使用 SOAP 请求程序函数。

2. 当禁用 Web 服务使用程序时，将删除这些函数。执行以下实用程序:

```
db2disable_soap_udf  -n dbName [-u uID] [-p password]
```

参数的含义与上面描述的启用实用程序的相同。

3. 还可以使用 DB2 CLP（命令行处理器）来创建和删除用户定义的函数。

```
CREATE SCHEMA db2xml;

   CREATE FUNCTION db2xml.soaphttpv (
               endpoint_url VARCHAR(256),
               soap_action VARCHAR(256),
               soap_body VARCHAR(3072))
         RETURNS VARCHAR(3072)
      LANGUAGE C PARAMETER STYLE DB2SQL
      SPECIFIC soaphttpvivo EXTERNAL NAME 'db2soapudf!soaphttpvivo'
      SCRATCHPAD FINAL CALL FENCED
      NOT DETERMINISTIC CALLED ON NULL INPUT
      NO SQL EXTERNAL ACTION DBINFO;

   CREATE FUNCTION db2xml.soaphttpv (
               endpoint_url VARCHAR(256),
               soapaction VARCHAR(256),
               input_message CLOB(1M))
         RETURNS VARCHAR(3072)
      LANGUAGE C PARAMETER STYLE DB2SQL
      SPECIFIC soaphttpcivo EXTERNAL NAME 'db2soapudf!soaphttpcivo'
      SCRATCHPAD FINAL CALL FENCED
      NOT DETERMINISTIC CALLED ON NULL INPUT
      NO SQL EXTERNAL ACTION DBINFO;

   CREATE FUNCTION db2xml.soaphttpc (
               endpoint_url VARCHAR(256),
               soapaction VARCHAR(256),
               input_message CLOB(1M))
         RETURNS clob(1M)
      LANGUAGE C PARAMETER STYLE DB2SQL
      SPECIFIC soaphttpcico EXTERNAL NAME 'db2soapudf!soaphttpcico'
      SCRATCHPAD FINAL CALL FENCED
      NOT DETERMINISTIC CALLED ON NULL INPUT
      NO SQL EXTERNAL ACTION DBINFO;

   CREATE FUNCTION db2xml.soaphttpc (
               endpoint_url VARCHAR(256),
               soapaction VARCHAR(256),
               soap_body varchar(3072))
         RETURNS clob(1M)
      LANGUAGE C PARAMETER STYLE DB2SQL
      SPECIFIC soaphttpvico EXTERNAL NAME 'db2soapudf!soaphttpvico'
      SCRATCHPAD FINAL CALL FENCED
      NOT DETERMINISTIC CALLED ON NULL INPUT
      NO SQL EXTERNAL ACTION DBINFO;

   CREATE FUNCTION db2xml.soaphttpcl (
               endpoint_url VARCHAR(256),
               soapaction VARCHAR(256),
               soap_body varchar(3072))
         RETURNS CLOB(1M) as locator
      LANGUAGE C PARAMETER STYLE DB2SQL
      SPECIFIC soaphttpviclo EXTERNAL NAME 'db2soapudf!soaphttpviclo'
      SCRATCHPAD FINAL CALL NOT FENCED
      NOT DETERMINISTIC CALLED ON NULL INPUT
      NO SQL EXTERNAL ACTION DBINFO;
```

4. Web 服务使用程序 WebSphere Studio 插件是 WebSphere Studio Application Developer（WSAD）V5.1.1 的一个组件。

相关概念:

- 第 136 页的『Web 服务使用程序用户定义的函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

Web 服务使用程序函数

IBM® DB2® Information Integrator 和 DB2 Universal Database™（DB2 通用数据库）由于能够从“结构化查询语言”（SQL）语句中调用 Web 服务，从而扩展了 DB2 通用数据库和 Web 服务的功能。通过调用一组用户定义的函数（UDF）来实现此目的，这些函数通过“超文本传输协议”（HTTP）接口为可使用的 Web 服务提供高速客户机“简单对象访问协议”（SOAP）。可直接从 SQL 语句调用这些函数。

可以根据 Web 服务的“Web 服务描述语言”（WSDL）来构造 SOAP 主体。还可以在 WebSphere® Studio Application Developer（WSAD）中使用 Web 服务“用户定义的函数”（UDF）工具来自动生成特定 UDF。这些 UDF 可以调用那些由用户指定的“Web 服务描述语言”文件定义的操作。生成的 UDF 是执行以下操作的 DB2 UDB 函数:

- 为 Web 服务请求提供参数。
- 调用 SOAP 客户机函数。
- 将 Web 服务调用的结果映射至用户指定的返回类型。

在某些网络上，访问因特网必须通过防火墙。通信可能会限于特定的机器和允许发送网络通信的特定端口。某些系统允许应用程序穿过防火墙。SOAP UDF 支持使用 SOCKS 客户机和 HTTP 代理来穿过防火墙。要使用 SOCKS 服务器穿过防火墙，必须在系统上安装 SOCKS 客户机软件。要使用 HTTP 代理，必须设置两个环境变量以便配置 DB2 通用数据库。设置 `DB2SOAP_PROXY` 以包括计算机的主机名以及 HTTP 代理。将 `DB2SOAP_PORT` 设置为 HTTP 代理的端口，例如 8080。在这两种情况下，SOAP 通信都会通过穿越防火墙的系统。

可以使用下列步骤来测试随 DB2 Information Integrator 提供的样本应用程序:

1. 启动数据库管理器（使用 db2start 命令）。
2. 创建“sample”数据库（使用 db2sampl 命令）。
3. 建立与“sample”数据库的连接。
4. 使用以下命令调用示例文件（在 Windows® 环境中，这些示例文件位于 <DB2 UDB 安装路径>\samples\soap 中）：*db2 -vf filename -t*。

相关任务:

- 『声明注册表和环境变量』（*《管理指南: 实现》*）
- 第 133 页的『安装 Web 服务使用程序用户定义的函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

Web 服务使用程序用户定义的函数

“简单对象访问协议”（SOAP）是一种具有下列特征的“可扩展标记语言”（XML）协议:

- 一个包络，它定义用于描述消息内容的框架以及如何处理消息
- 一组编码规则，用于表示应用程序定义的数据类型的实例
- 一个约定，用于表示 SOAP 请求和响应

DB2® 通用数据库需要以下信息来构建 SOAP 请求和接收 SOAP 响应。

- 服务端点，例如 *http://services.xmethods.net/soap/servlet/rpcrouter*
- SOAP 主体的一些“可扩展标记语言”（XML）内容，包括具有请求的名称空间 URI 的操作的名称、编码样式和输入自变量。
- 可选: SOAP 操作 URI 引用。引用可以是空的（如以下示例中所示）、*http://tempuri.org/* 或只是 "

DB2 UDB 函数 db2xml.soaphttp() 执行下列操作:

1. 编写 SOAP 请求
2. 将请求向服务端点公布
3. 接收 SOAP 响应
4. 返回 SOAP 主体的内容

以下是用于 VARCHAR() 或 CLOB()（取决于 SOAP 主体）的过载函数。

```
db2xml.soaphttpv returns VARCHAR():
    db2xml.soaphttpv (endpoint_url VARCHAR(256),
                      soap_action VARCHAR(256),
                      soap_body VARCHAR(3072))
                  RETURNS VARCHAR(3072)
db2xml.soaphttpv returns VARCHAR():
    db2xml.soaphttpv (endpoint_url VARCHAR(256),
                      soap_action VARCHAR(256),
                      soap_body CLOB(1M))
                  RETURNS VARCHAR(3072)


db2xml.soaphttpc returns CLOB():
       db2xml.soaphttpc (endpoint_url VARCHAR(256),
                         soapaction VARCHAR(256),
                         soap_body VARCHAR(3072))
                       RETURNS CLOB(1M)
db2xml.soaphttpc returns CLOB():
       db2xml.soaphttpc (endpoint_url VARCHAR(256),
                         soapaction VARCHAR(256),
                         soap_body CLOB(1M))
                       RETURNS CLOB(1M)

db2xml.soaphttpcl returns CLOB() as locator:
       db2xml.soaphttpcl(endpoint_url VARCHAR(256),
                          soapaction VARCHAR(256),
                          soap_body varchar(3072))
                       RETURNS CLOB(1M) as locator
```

DB2 UDB 构造的 SOAP 请求包络示例

第 137 页的图 49 中的示例显示“超文本传输协议”（HTTP）公布头来将 SOAP 请求包络向主机公布。粗体区域显示 web 服务端点（公布路径和主机）和 SOAP 主体的内容。SOAP 主体显示针对邮政编码 95120 的温度请求。

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: services.xmethods.net
Connection: Keep-Alive User-Agent: DB2SOAP/1.0
Content-Type: text/xml; charset="UTF-8"
SOAPAction: ""
Content-Length: 410

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
                   xmlns:SOAP-ENC=http://schemas.xmlsoap.org/soap/encoding/
                   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                   xmlns:xsd=http://www.w3.org/2001/XMLSchema >
<SOAP-ENV:Body>
       <ns:getTemp xmlns:ns="urn:xmethods-Temperature">
         <zipcode>95120</zipcode>

       </ns:getTemp>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

图 49. DB2 UDB 构造的 SOAP 请求包络

使用 DB2 UDB 来抽取 SOAP 响应包络内容的示例

第 137 页的图 50 中的示例显示具有 SOAP 响应包络的 HTTP 响应头。SOAP 主体的**粗体**内容显示温度请求的结果。此处未显示 SOAP 包络的名称空间定义，但也应包含在内。

```
HTTP/1.1 200 OK
Date: Wed, 31 Jul 2002 22:06:41 GMT
Server: Enhydra-MultiServer/3.5.2
Status: 200
Content-Type: text/xml; charset=utf-8
Servlet-Engine: Lutris Enhydra Application Server/3.5.2
  (JSP 1.1; Servlet 2.2; Java™ 1.3.1_04;
   Linux 2.4.7-10smp i386; java.vendor=Sun Microsystems Inc.)
Content-Length: 467
Set-Cookie:JSESSIONID=JLEcR34rBc2GTIkn-0F51ZDk;Path=/soap
X-Cache: MISS from www.xmethods.net
Keep-Alive: timeout=15, max=10
Connection: Keep-Alive
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
                   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                   xmlns:xsd=http://www.w3.org/2001/XMLSchema >
<SOAP-ENV:Body>
           <ns1:getTempResponse xmlns:ns1="urn:xmethods-Temperature"
           SOAP-ENV:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/ >
              <return xsi:type="xsd:float">85<return>
           </ns1:getTempResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

图 50. 使用 DB2 UDB 来抽取 SOAP 响应包络的内容

相关任务:

- 第 133 页的『安装 Web 服务使用程序用户定义的函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

跟踪 Web 服务使用程序事件

可以使用 DB2 通用数据库跟踪实用程序来跟踪 Web 服务使用程序用户定义的函数。另外，当使用 Windows 平台时，可以跟踪“超文本传输协议”（HTTP）SOAP 请求和响应，并记入文件。

过程:

要在 DB2 通用数据库中跟踪 SOAP 组件，请使用以下跟踪掩码:

```
db2trc on -m *.*.147.*.*
```

相关概念:

- 第 135 页的『Web 服务使用程序函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

Web 服务使用程序 – 使用 WebSphere Studio 用户定义的函数工具

Web 服务使用程序“用户定义的函数”向导在 WebSphere® Studio V5 中生成并测试用户定义的函数。

与 WebSphere Studio 配合使用的向导读取“Web 服务定义语言”（WSDL）文件。然后，它生成“用户定义的函数”（UDF），这些函数提供从数据库应用程序对 Web 服务的容易的访问。还可以在 SQL 语句中使用生成的 UDF 来将关系数据与从 Web 服务中检索到的动态数据进行组合。可以在 SQL 中直接调用 Web 服务使用程序函数（请参阅Web 服务使用者函数）。但是，任务可能会需要一些高级编程技巧，并且它可能是非常耗时的。在生成和部署 UDF 之后，可以在 SQL 中使用这些函数来将关系数据与从 Web 服务中检索到的动态数据进行组合。生成的 UDF 的结构如下:

1. 构造 SOAP 主体
2. 调用 SOAP 使用程序（提交 SOAP 请求包络）
3. 从 SOAP 响应中抽取值

相关概念:

- 第 135 页的『Web 服务使用程序函数』
- 第 136 页的『Web 服务使用程序用户定义的函数』

相关任务:

- 第 133 页的『安装 Web 服务使用程序用户定义的函数』
- 第 138 页的『跟踪 Web 服务使用程序事件』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

如何从 WebSphere Studio 生成用户定义的函数

先决条件:

1. 启用 DB2 XML Extender 数据库
2. 为数据库启用 Web 服务使用程序 UDF
3. 创建想要与 Web 服务 UDF 配合使用的项目
4. 创建与刚启用的数据库的连接
5. 将数据库导入到 WebSphere Studio V5 项目中。有关更多信息，请参阅 *WebSphere Studio Application Developer Programming Guide*。

过程:

在 WebSphere Studio 中，可使用三种不同的方法来启动生成用户定义的函数（UDF）的向导。

- 可从*文件* > *新建* > *其它*菜单中调用向导。然后选择**数据**。文件夹展开，然后从菜单中选择 **Web 服务用户定义的函数**。单击**下一步**按钮以继续。
- 可在 Web 服务客户机向导中启动它，其中，它显示为除生成 Java 代理之外的另一个选项。
- 它是在生成测试客户机时 Web 服务向导中的一个选项。

使用下列步骤来生成 UDF:

1. 从向导的第一页指定 WSDL 文件（请参阅第 140 页的图 51）。使用此 WSDL 文件来生成 UDF。

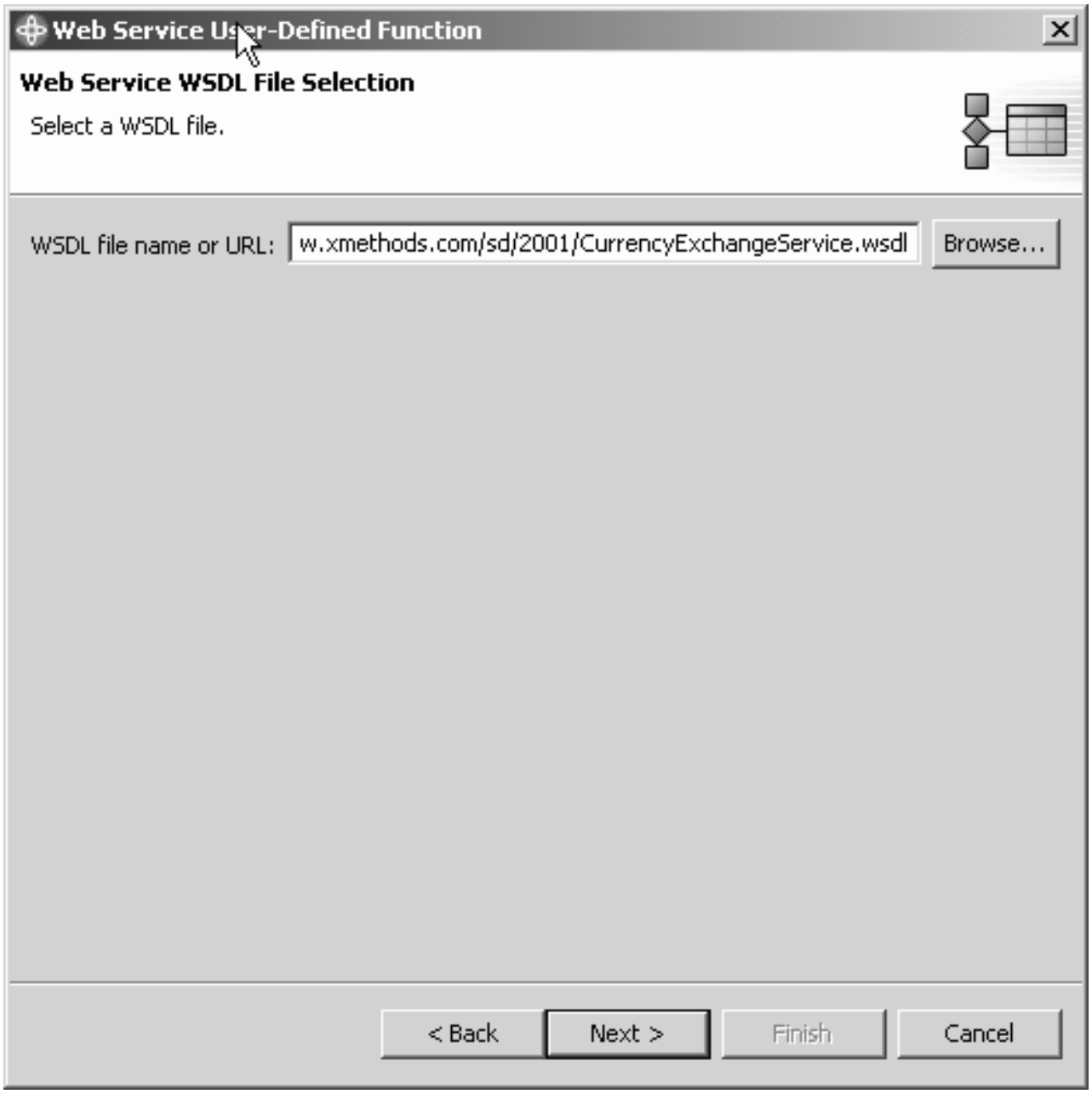


图 51. 选择 WSDL 文件

从工作空间选择 WSDL 文件或指定适当的统一资源定位器（URL）。例如，货币汇率 Web 服务将两个国家或地区作为输入参数，并返回它们之间的货币汇率。WSDL 文件位于 *www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl*。

2. 选择数据库。在第 141 页的图 52 中，会看到数据库连接和为其生成 UDF 的模式。单击**浏览**按钮以从 WebSphere Studio 工作空间中选择数据库模式。向导需要对 Web 服务使用程序 UDF 和 DB2 XML Extender 启用数据库。如果当前没有与指定数据库的可用连接，则一个附加消息窗口会要求输入连接信息。可以选择立即将生成的 UDF 部署到数据库中，也可以选择仅在 WebSphere Studio 工作空间中生成 UDF。可在稍后部署 UDF。

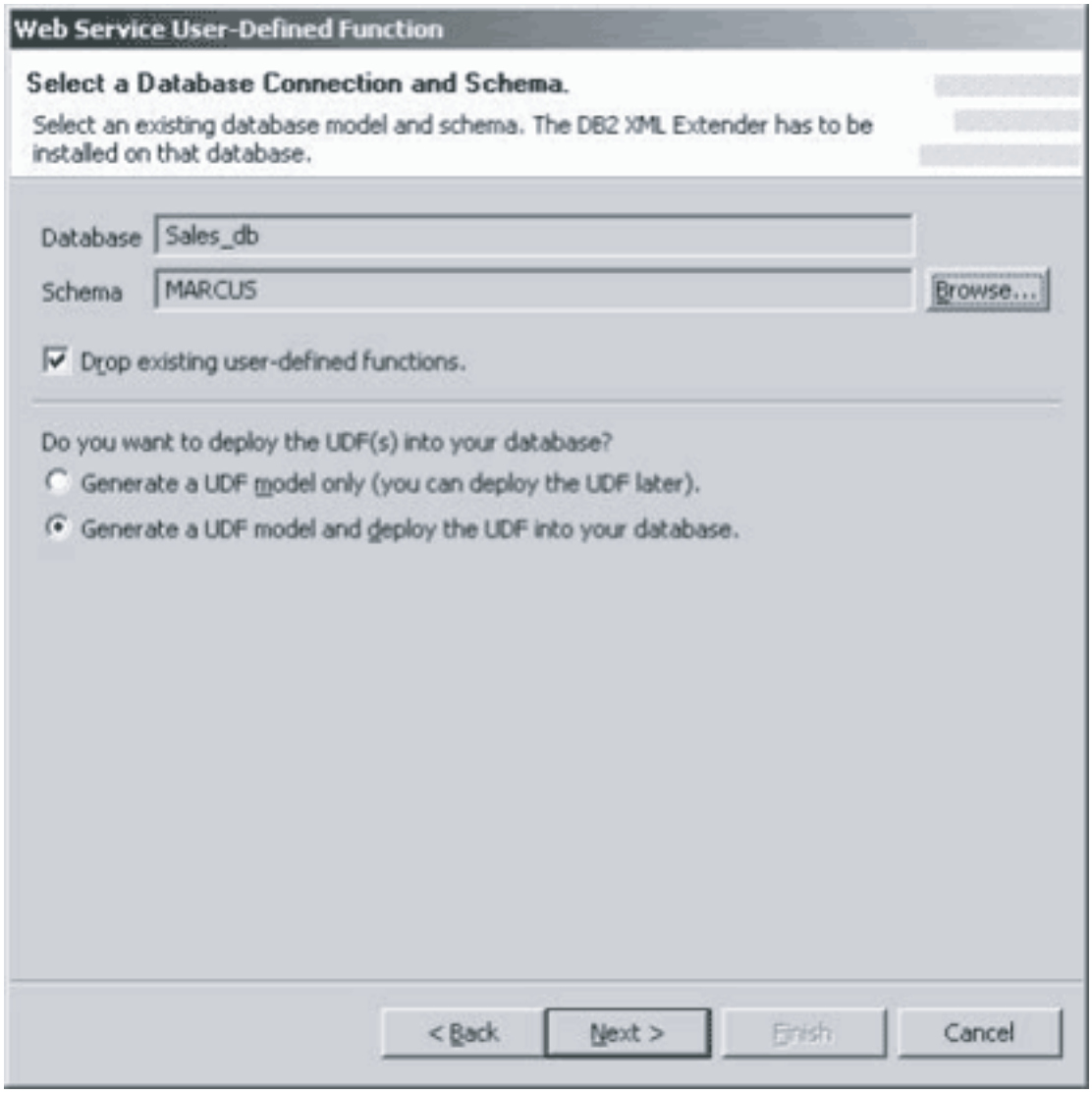


图 52. WSDL 第 2 页

3. 选择想要创建的 UDF。从第 142 页的图 53 中描述的操作列表中，选择想要创建的操作。向导会为选择的每个操作生成一个 UDF。由于用于此示例的 Web 服务只提供了一个操作，所以向导自动选择该操作。进入下一页。

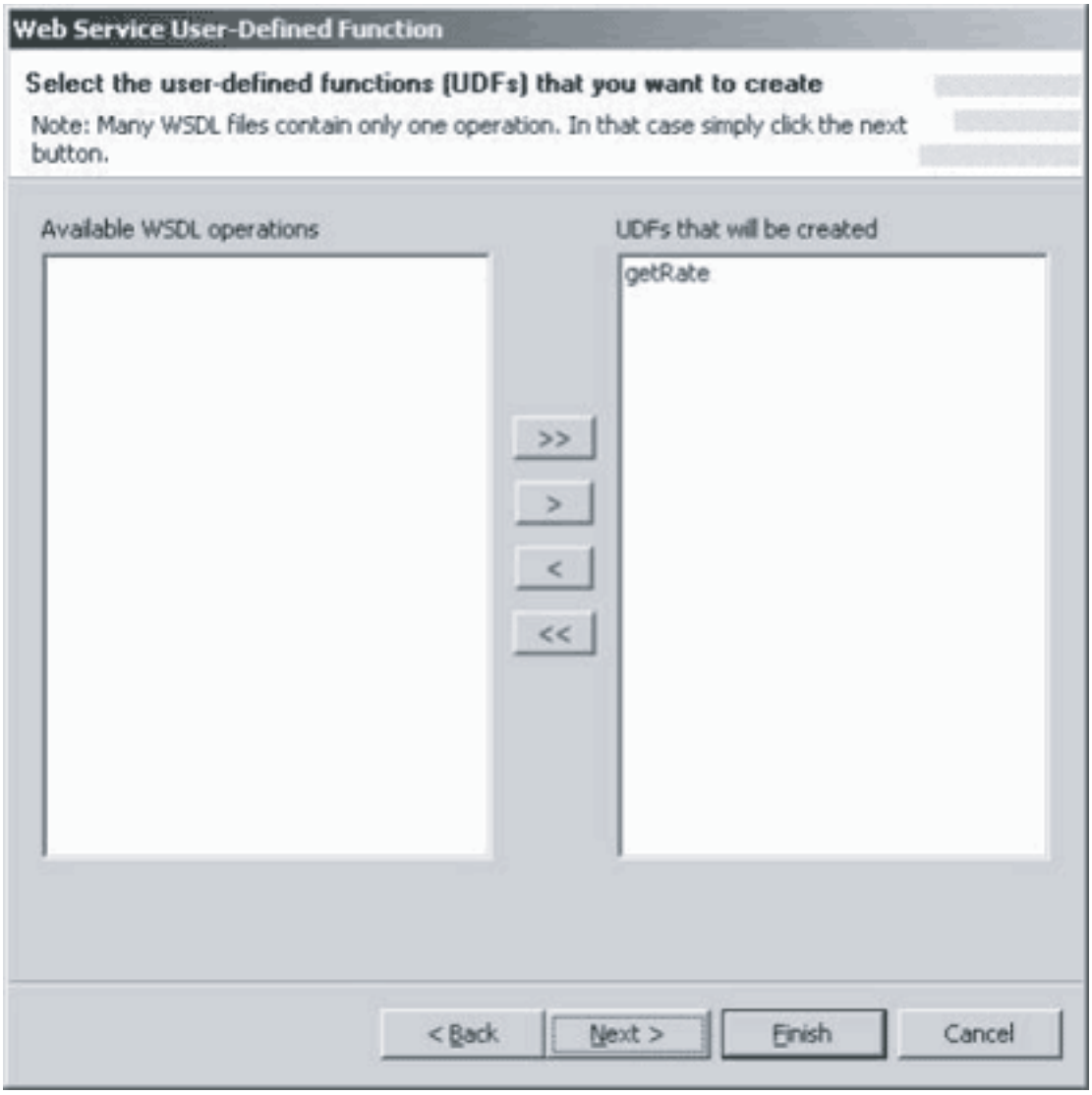


图 53. 向导页 3

4. 为 UDF 选择选项。对于在前一步骤中选择的每个操作，可为那些 UDF 定义选项，如更改函数名或提供对函数的注释。请参阅第 143 页的图 54 和第 144 页的图 55。

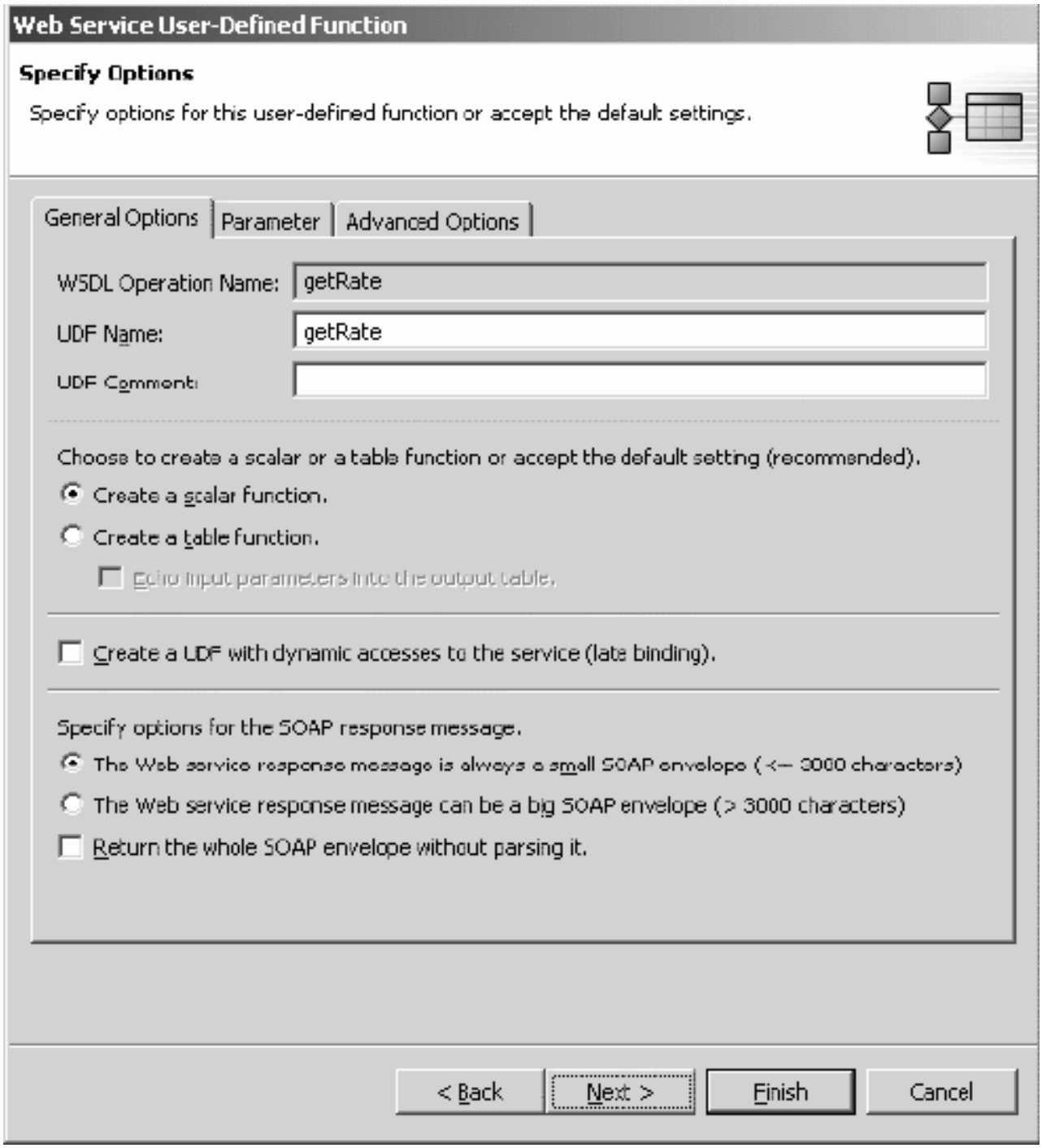

Web Service User-Defined Function
Specify Options
Specify options for this user-defined function or accept the default settings.
General Options
Parameter
Advanced Options
W5DL Operation Name:
getRate
UDF Name:
getRate
UDF Comment:
Choose to create a scalar or a table function or accept the default setting (recommended).
Create a scalar function.
Create a table function.
Echo input parameters into the output table.
Create a UDF with dynamic accesses to the service (late binding).
Specify options for the SOAP response message.
The Web service response message is always a small SOAP envelope (<= 3000 characters)
The Web service response message can be a big SOAP envelope (> 3000 characters)
Return the whole SOAP envelope without parsing it.
< Back
Next >
Finish
Cancel

图 54. 选择选项页 1

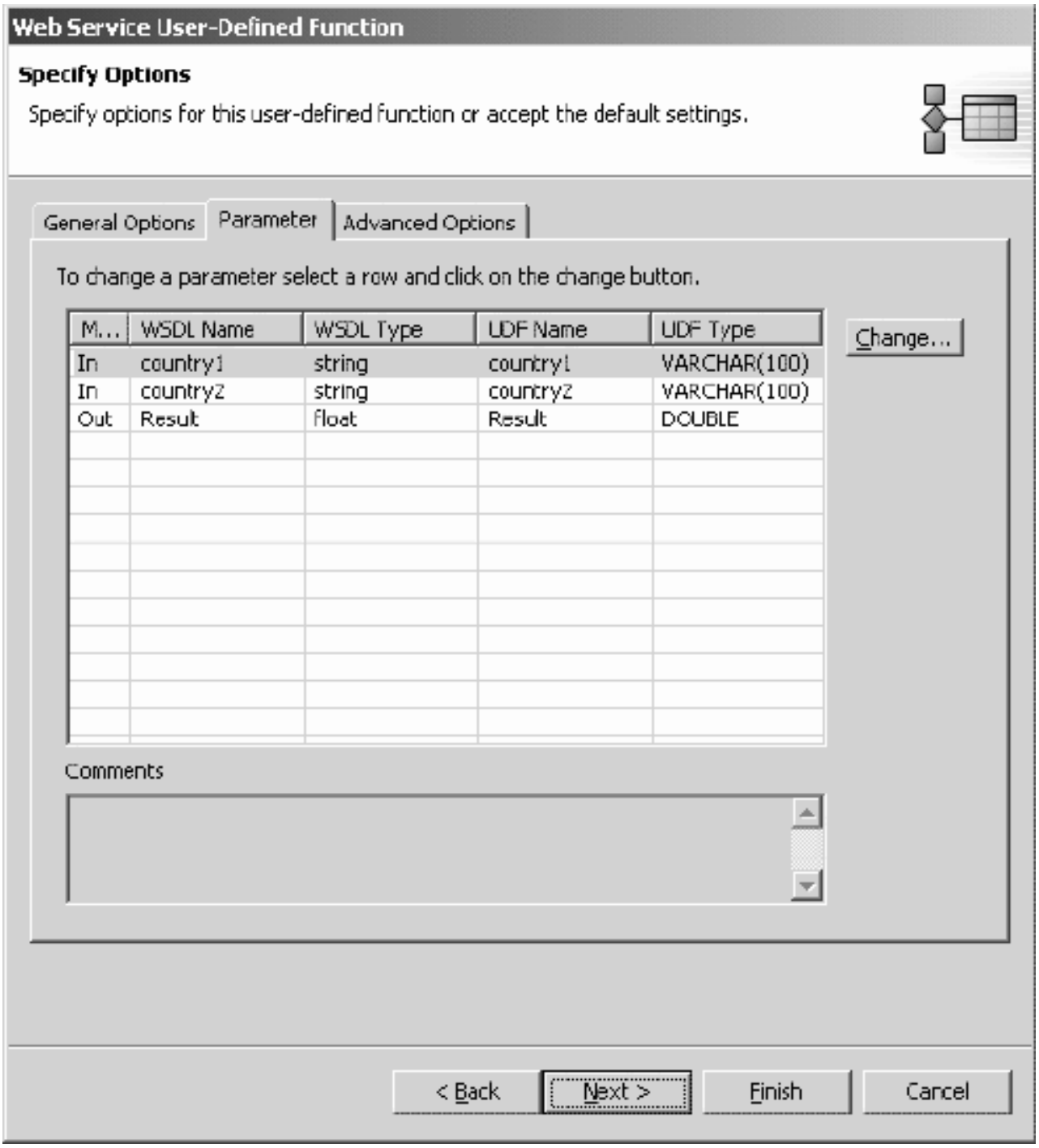


图 55. 选择选项页 2

- 可以选择构建标量函数或表函数。
  - 向导在 Web 服务返回简单 XML 类型时生成标量函数。
  - 向导在返回复杂 XML 类型时生成表函数。表函数自动将复杂 XML 类型映射到多个列中。

在向导不应自动映射返回的类型时，从表函数切换到标量函数是有意义的。在这种情况下，向导应将类型返回为 XML 段。能够从标量函数切换至表函数允许您在 FROM 子句中使用 UDF。

- 可以选择**将输入参数回传到输出表中**复选框来将输入参数作为列包含在输出表中。
- 可以选择生成能够动态访问 Web 服务的 UDF。

  当不在 WSDL 文档中指定服务位置（soap:address 元素的位置属性）时，可生成动态函数。甚至当已指定服务位置时也可选择**创建动态访问服务的 UDF** 复选框来使用最新绑定。当生成动态函数时，在运行时将服务位置指定为 UDF 的参数。
- 当使用的 Web 服务可返回 3000 多个字符的响应时，指定 **Web 服务响应消息可以是大 SOAP 包络**单选按钮。缺省情况下，指定 **Web 服务响应消息总是小 SOAP 包络**单选按钮，原因是这使大多数 Web 服务的性能更好。如果指定小 SOAP 响应选项并且向导返回具有大于 3000 个字符的 SOAP 包络，则生成的 Web 服务 UDF 返回描述性错误消息。
- 选择**返回整个 SOAP 包络而不语法分析它**复选框来帮助调试 Web 服务使用程序 UDF。

5. 从选项窗口的“参数”页中，可查看和更改从 WSDL 类型至 SQL 类型的参数映射（请参阅第 144 页的图 55）。
6. 从“高级选项”页中，可指定 UDF 的名称。如果不指定名称，则在部署 UDF 时数据库会自动生成唯一的名称。
7. 查看生成 UDF 的设置。检查数据库和模式以及将对数据库发出的 CREATE 语句（请参阅第 146 页的图 56）。
8. 单击**下一步**或**完成**按钮。由于先前选择生成和部署，此操作将生成 UDF 并将其部署到数据库中。

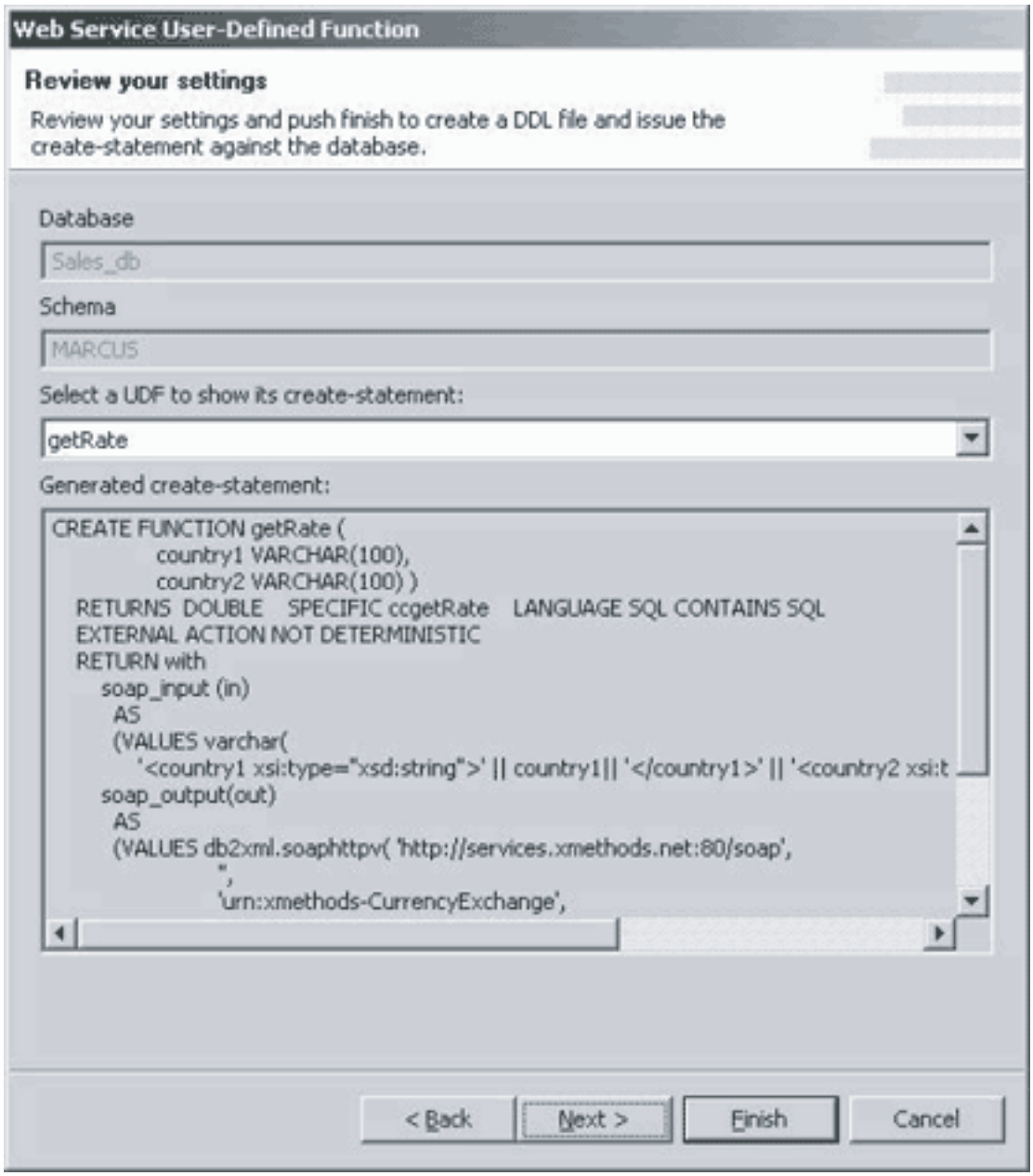


图 56. 查看

9. 可直接从工作空间运行 Web 服务使用程序 UDF。要运行已部署的 UDF:
   a. 右键单击 UDF。
   b. 选择**运行**（请参阅第 147 页的图 57）。“运行设置”窗口打开（第 148 页的图 58）。
   c. 可以从“运行设置”窗口中指定参数值。
   d. 单击**确定**按钮来查看测试的结果（第 148 页的图 59）。

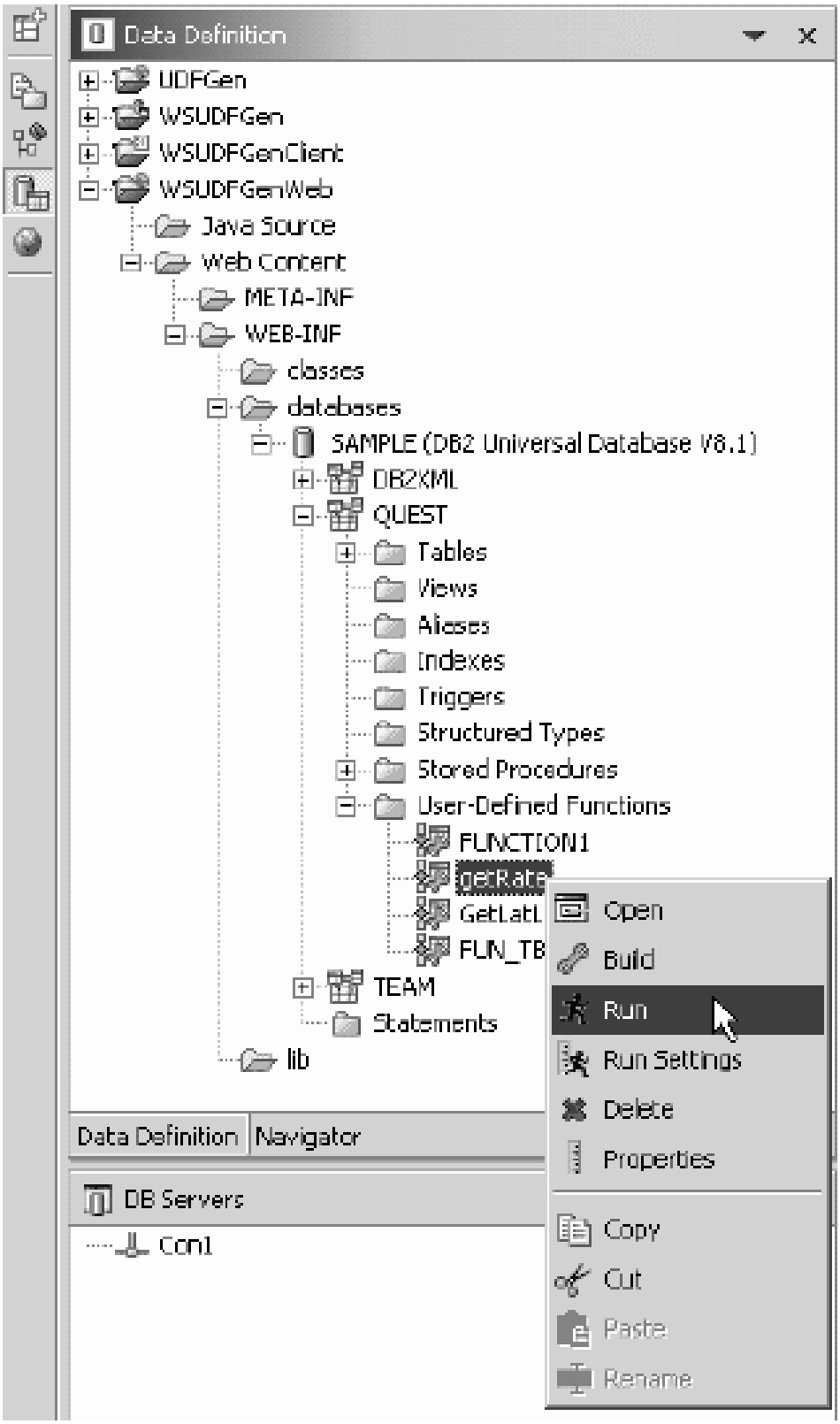

Data Definition
UDFGen
WSUDFGen
WSUDFGenClient
WSUDFGenWeb
Java Source
Web Content
META-INF
WEB-INF
classes
databases
SAMPLE (DB2 Universal Database V8.1)
DB2XML
QUEST
Tables
Views
Aliases
Indexes
Triggers
Structured Types
Stored Procedures
User-Defined Functions
FUNCTION1
GetRate
GetLatL
FUN_TB
TEAM
Statements
lib
Data Definition
Navigator
DB Servers
Con1
Open
Build
Run
Run Settings
Delete
Properties
Copy
Cut
Paste
Rename

图 57. 测试

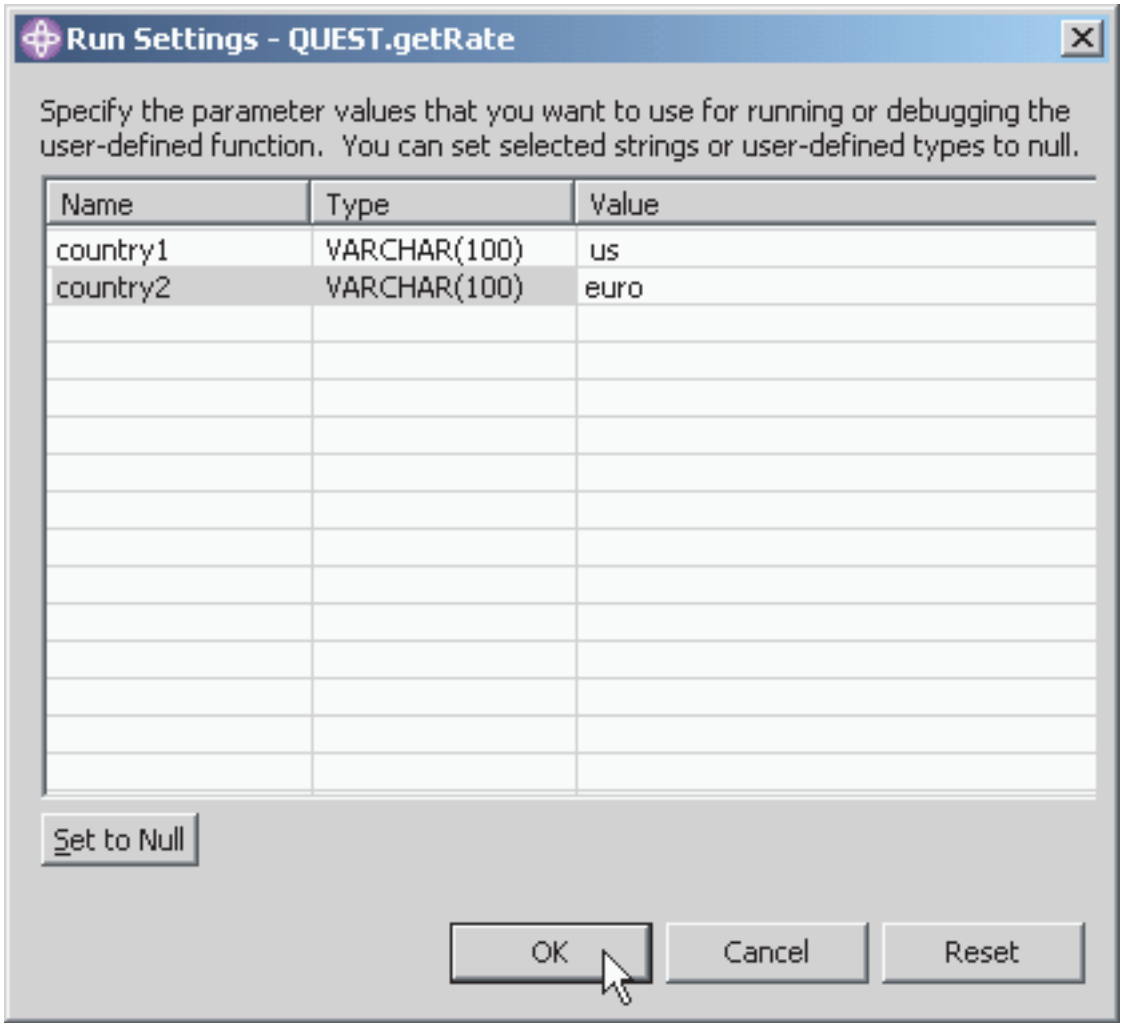


图 58. 运行设置

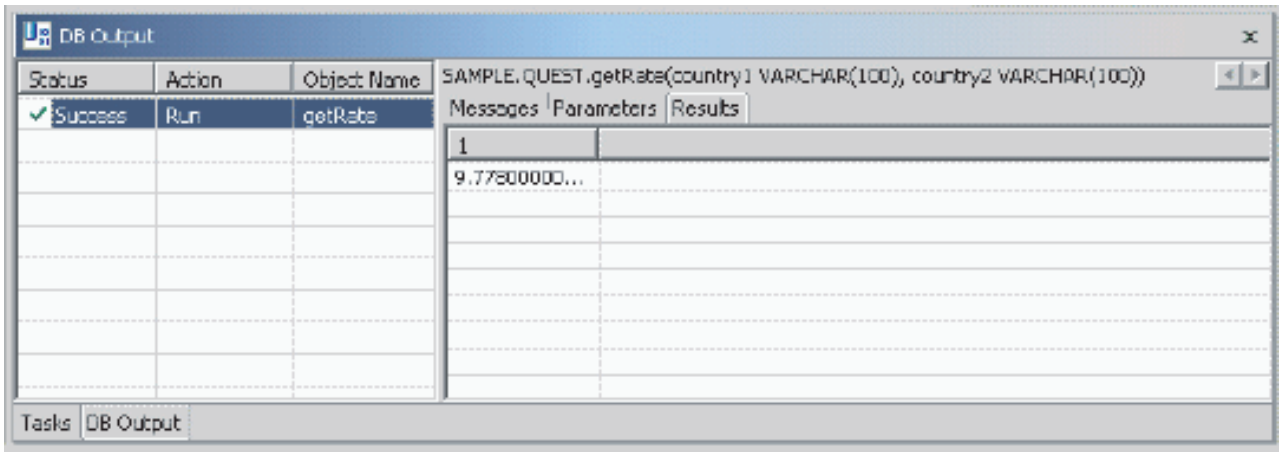


图 59. 结果

相关概念:

- 第 135 页的『Web 服务使用程序函数』

- 第 136 页的『Web 服务使用程序用户定义的函数』
- 第 138 页的『Web 服务使用程序 – 使用 WebSphere Studio 用户定义的函数工具』

相关任务:

- 第 133 页的『安装 Web 服务使用程序用户定义的函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』
- 『dxxEnableDB() stored procedure』（*DB2 XML Extender Administration and Programming*）

使用 Web 服务使用程序 UDF

使用 UDF 来在关系表和 Web 服务之间共享信息。假设在关系数据库中有具有以下数据的表:

表 19. Products 表

Product	Price
Gear	950.00
Nut	25.00
Bolt	35.00

并假设远程表中存在有关货币类型的信息。

表 20. 货币表

地区
US
EURO
UK

使用 第 149 页的 的以下 SQL 语句来确定如何使用货币汇率函数来用欧元而不是美元显示价格信息。此操作会存取实时汇率。注意，该语句使用内置 decimal 函数来转换价格信息。

```
SELECT product, decimal(getRate('us', 'euro') * price, 10, 2)
  as 'EUR_Price'
  FROM products
```

第 149 页的 的语句的结果是:

表 21. 使用实时汇率

Product	EUR_Price
Gear	1019.82
Nut	26.84
Bolt	37.57

使用以下 SQL 语句来显示如何将关系数据用作 Web 服务的输入。第 150 页的 示例显示货币汇率函数可如何以不同货币显示价格信息。

```
SELECT p.product, c.area,
    decimal(getRate('us', c.area) * price, 10, 2)
      as Price
  FROM products, areas
```

结果是:

表 22. 以不同货币显示价格信息

Product	Area	Price
Gear	us	950.00
Nut	us	25.00
Bolt	us	35.00
Gear	euro	1019.82
Nut	euro	26.84
Bolt	euro	37.57
Gear	uk	650.84
Nut	uk	17.12
Bolt	uk	23.97

如果经常使用此查询，则可能想要定义一个视图来提供一个更简单的界面。此视图的一个示例如下:

```
CREATE VIEW prices AS
  SELECT p.product, c.area,
      decimal(getRate('us', c.area) * price, 10, 2)
      as Price
  FROM p.products, c.areas
```

通过使用此视图，可对以下更简单的查询进行编码:

```
SELECT * FROM prices
```

使用以下 SQL 语句来显示如何将生成的 UDF 用作 FROM 子句中的表函数。此示例将 getRate-UDF 重新生成为表函数。输入参数回送到输出表中。

```
SELECT t.*
FROM countries c,
table( getRate('us', c.countries) ) t
```

结果是:

表 23. 使用 getRate 作为表函数

AREA1	AREA2	RESULT
us	us	+1.00000000000000E+000
us	euro	+1.07280000000000E+000
us	uk	+6.84800000000000E-001

相关概念:

- 第 135 页的『Web 服务使用程序函数』
- 第 136 页的『Web 服务使用程序用户定义的函数』

相关任务:

- 第 138 页的『跟踪 Web 服务使用程序事件』

Web 服务使用程序示例

此处列示的示例适用于 DB2 通用数据库版本 8。文件 *<DB2_installed path>/samples/soapsample.sql* 描述如何运行样本。文件 *soapsample.sql* 包含以下示例和样本查询列表:

- getTemp – 检索温度（以华氏温度表示）
- getRate – 返回任何两种货币之间的汇率

相关概念:

- 第 135 页的『Web 服务使用程序函数』
- 第 136 页的『Web 服务使用程序用户定义的函数』
- 第 138 页的『Web 服务使用程序 – 使用 WebSphere Studio 用户定义的函数工具』

相关任务:

- 第 133 页的『安装 Web 服务使用程序用户定义的函数』

相关参考:

- 第 149 页的『使用 Web 服务使用程序 UDF』

第 3 章 开发联合应用程序、仓库应用程序和消息队列应用程序

本节说明如何开发联合应用程序、仓库应用程序，以及如何使用消息队列（MQ）功能。

开发使用联合服务器的应用程序

联合系统中的 “DB2 通用数据库” 服务器称为联合服务器。要允许透明数据存取，可为想要存取的远程数据创建昵称。可创建函数或使用补偿不同数据源之间差异和模拟本机不支持的功能的已定义函数。多站点连接和联合提升了来自多个源的数据集成。

联合系统的优点

联合系统是一种分布式数据库管理系统，使您可以在单个 SQL 语句中将分布式请求发送至多个数据源。IBM® DB2® Information Integrator 的联合系统与 Web 应用程序服务器提供的内置数据库支持相辅相成。

若没有联合系统，则访问不同的数据源需要多个步骤:

1. 必须分别连接至各个数据源。
2. 必须通过使用不同的本机应用程序编程接口来抽取必需的数据。
3. 必需手工对数据进行过滤、排序及合并。

通过联合系统，只需使用 SELECT、INSERT、UPDATE 和 DELETE 语句即可查询昵称。

在联合系统中，还可以对跨多个不同种类源的数据进行透明存取。通过联合系统，可以增强 Web 应用程序服务器的使用和功能以支持远程数据（物理存储的或动态生成的）。可以将联合系统与应用程序服务器组件（例如企业 bean 和 Web 服务）配合使用。使用企业 bean 和联合系统对象，程序员可以执行某些数据库操作或事务工作、访问多个数据源并创建集成不同数据的应用程序。

相关概念:

- 『联合系统』（*《联合系统指南》*）
- 第 155 页的『联合系统中的企业 bean』
- 第 7 页的『DB2 Information Integrator - 非关系联合技术』
- 第 19 页的『性能和调整规划 - 联合系统中的具体查询表』
- 第 6 页的『DB2 Information Integrator - 关系联合技术』

在 IBM DB2 Information Integrator 中设计查询的优点

DB2® 通用数据库使数据库管理员能够创建跨越不同数据源中的多个表的数据视图。在联合系统中，视图可能包括在多个服务器上以不同格式存储的数据。要构建这种视图，请为远程数据对象创建昵称。然后，使用 SQL 来创建连接或联合这些昵称的视图。连接或联合多个数据源的视图是只读视图。可以构建映射到这些视图的容器管理的持久性实体 bean，这些 bean 是只读 bean。

除联合视图之外，联合系统还包括用于关系数据库管理器的 SQL 数据定义语言（DDL）透明度。DDL 透明度意味着您可以使用 OPTIONS 子句创建 DB2 UDB 表，并使用一个语句执行两个单独的任务。如图 60 中的示例所示，您在远程数据源创建了一个表，并为此表创建了相应的 DB2 UDB 昵称。

```
    CREATE TABLE orarest (
id int primary key not null,
name varchar(20),
cuisine varchar(20),
budget int)
OPTIONS (remote_server 'ORACLE8',
         remote_schema 'ORACLEUSER1')

          CREATE TABLE msrest (
id int primary key not null,
name varchar(20),
cuisine varchar(20),
budget int)
OPTIONS (remote_server 'MSSQL',
         remote_schema 'MSUSER1')
```

图 60. DDL 透明度示例

具有 OPTIONS 子句的创建语句创建下列数据库对象:

- 远程 Oracle 数据库中名为 ORAREST 的表。
- DB2 UDB 联合数据库中名为 ORAREST 的昵称。
- 远程 Microsoft® SQL Server 数据库中名为 MSREST 的表。
- DB2 UDB 联合数据库中名为 MSREST 的昵称。

然后可以使用视图中生成的表和昵称来联合或集成两个不同数据源中的信息。

```
CREATE VIEW multirest
   (id, name, cuisine, budget)
AS
  SELECT id, name, cuisine, budget
  FROM orarest UNION
   SELECT id, name, cuisine, budget FROM msrest;
```

使用 DDL 透明度时，数据库管理器执行大多数 SQL 转换以正确构造远程表。您不需要了解某个数据源的特定 SQL 语法就可以创建有效的表。

还可以将具体查询表合并到应用程序中以增强性能。具体查询表允许您预先计算每个查询的全部或一部分，然后使用计算的结果来回答将来的查询。具体查询表使您可以保存先前查询的结果，然后在后续查询中重用普通查询结果。具体查询表有助于避免多余的扫描、聚集和连接。

- 即使在远程数据源不可用（例如网络不可用）时，具体查询表也允许进行查询处理。远程表上的具体查询表可以被当作该表的高速缓存，这可以增强系统可用性。
- 具体查询表提高了整个系统的可伸缩性。具体查询表可以从主数据库中卸载工作。您可以具有多个本地数据库，每个都带有常用主数据的一部分的副本。主数据库对企业的约束较少。
- 通过使用具体查询表，可以避免因某些查询连接到远程系统。总系统吞吐量可能会增加，且总响应时间可能会降低。
- 具体查询表在访问远程数据源时使用 DB2 UDB 的完整功能。即使远程数据源不支持具体查询表，您也可以体验具体查询表的优点。

在数据库环境中开发的支持 Web 的应用程序可以使用 Java™ 2 Enterprise Edition（J2EE）服务器环境的若干组件。某些 J2EE 组件包括对企业 bean、servlet、JavaServer Pages 代码以及 Java 命名和目录接口扩展的支持。J2EE 还支持连接到数据库管理器并进行访问，而数据库管理器支持 “Java 数据库连接” 代码和 Java 事务应用程序编程接口。

相关概念:

- 『什么是透明 DDL？』（*《联合系统指南》*）
- 『具体查询表和联合系统 - 概述』（*《联合系统指南》*）
- 『调整查询处理』（*《联合系统指南》*）
- 第 155 页的『联合系统中的企业 bean』
- 第 19 页的『性能和调整规划 - 联合系统中的具体查询表』

相关任务:

- 『创建联合具体查询表』（*《联合系统指南》*）

联合系统中的企业 bean

企业 bean 是在 Web 服务器上运行的 Java™ 组件。可以创建容器管理的持久性实体 bean 以映射至使用 IBM® DB2® Information Integrator 联合系统创建的昵称。容器管理的持久性实体 bean 可以存取关系数据库中的数据。只读容器管理的持久性实体 bean 可以存取非关系数据库中的数据。可以使用实体 bean 通过 Enterprise JavaBean 体系结构集成不同的数据。

Enterprise JavaBean 体系结构

企业 bean 组件是 Enterprise JavaBean 体系结构的一部分，并在 Enterprise JavaBean 容器内执行。容器在 Enterprise JavaBean 服务器内运行。企业 bean 实现业务逻辑。Enterprise JavaBean 容器向企业 bean 组件提供诸如事务和资源管理、持久性和安全性等服务。数据库处理的详细信息可在 Enterprise JavaBean 容器中找到。

实体 bean 和会话 bean

两类企业 bean 对联合系统很重要 - 会话 bean 和实体 bean。

会话 bean

会话 bean 通常与单个客户机关联，且通常不具有持久性。会话 bean 的用途不是显示或更新现有数据库内容。相反，会话 bean 的用途是充当在服务器上执行某些操作的单个客户机。

实体 bean

实体 bean 显示在数据库中持久存储的信息。实体 bean 与数据库事务相关联。实体 bean 可以向多个用户提供数据存取。实体 bean 可以表示底层的数据库行，或者单个行中的 SELECT 语句的结果。

联合系统支持单个容器管理的持久性实体 bean（其属性映射至多个资源中的数据）的自动开发和部署。将容器管理的持久性实体 bean 映射到联合数据库对象时，联合服务器透明地将数据库存取转换为适合于数据源的数据存取请求。当部署企业 bean 时，bean 驻留在提供服务（例如对持久性的支持）的容器中。当部署企业 bean 时，实体 bean 自

动生成实现持久性的代码。相比之下，构建存取持久数据的会话企业 bean 时，您必须编写自己的 Java 数据库连接语句才能建立数据库连接并发出 SQL 语句。

容器管理的持久性实体 bean 使所有与数据库的交互作用延迟至 Enterprise JavaBean 容器。通常，企业 bean 从数据库读取数据并将数据放入容器管理的持久性实体 bean 中的字段内。可以引用或更新实体 bean 中的数据。事务结束时，Enterprise JavaBean 容器会存取实体 bean 中的数据并更新表中的底层行。

相关概念:

- 『Enterprise Java Beans』（*Application Development Guide: Programming Client Applications*）

相关任务:

- 第 163 页的『创建和部署容器管理的持久性 bean』

职员技能方案 - 解决方案设计

YBar, Incorporated 在 DB2® 通用数据库表和 XML 文档中存储职员信息。YBar, Incorporated 可以处理 DB2 UDB 表和 XML 文档中的数据。逻辑解决方案是创建将职员简历以“可扩展标记语言”（XML）文档格式存储在职员数据库中的应用程序。然后公司可以使用 XML 搜索功能来查找与特定项目所需的技能相匹配的简历。

YBar, Incorporated 使用 IBM® DB2 Information Integrator 的联合系统来访问作为关系数据的 XML 格式简历文件，以解决其业务问题。

1. 为改进存取平面文件数据源中数据的性能，YBar, Incorporated 对 DAD 文件中定义的副表创建索引。DB2 XML Extender 根据 DAD 文件中的模式定义创建副表。

```
CREATE INDEX KEY_Skill ON
      resume_skills_sidetable(skill)
```

2. YBar, Inc. 数据库管理员在关系表的职员数据中插入三行。名为 Resume 的列是 XML 列。XMLVARCHARFROMFILE 是一个 DB2 XML Extender 函数，它从服务器文件中读取 XML 文档并将文档作为 XMLVARCHAR 类型返回。列 Resume 的值从位于 'Some_Path'\<the resume file contents> 的简历文件填充。

```
INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(12,'Douglas','Laurie',
      123, 1,
      db2xml.XMLVarcharFromFile
      ('Some_Path'\resume_ld.xml'))

INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(13,'Smith','John',
      123, 2,
      db2xml.XMLVarcharFromFile
      ('Some_Path'\resume_js.xml'))

INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(14,'Jackson','George',
      123, 3,
      db2xml.XMLVarcharFromFile
      ('Some_Path'\resume_gl.xml'))
```

3. 然后 YBar, Incorporated 开发者使用 XML 列和副表查询基本表。此 SQL 语句查找具有 Java™ 技能的所有职员。它充分利用副表，从而可以使用索引存取。

```
SELECT e.firstname, e.lastname,e.resume
    FROM employee
   AS e, resume_skills_sidetable AS r
    WHERE e.emp_id=r.emp_id
      AND r.skill LIKE '%Java%'
```

YBar, Inc. 开发者可以通过几种方法查询 XML 内容:

- 以下示例使用标量函数来返回某个职员的经验:

```
SELECT db2xml.EXTRACTVARCHAR
   (resume,'/resume/experience')
    FROM employee
   WHERE emp_id=13
```

  用户定义的函数 extractVarchar 使用列 resume 作为输入，使用位置路径 /resume/experience 作为选择标识。标量函数返回一个职员的经验。通过 WHERE 子句，抽取函数仅对标识为 “13” 的简历求值。

- 以下示例使用表函数返回几行:

```
SELECT e.firstname, e.lastname, e.resume
    FROM employee
   AS e,
    TABLE(db2xml.EXTRACTVARCHARS(e.resume,'/resume/skill'))
    AS t
   WHERE t.returnedvarchar LIKE '%Java%'
```

  用户定义的函数 extractVarchars 使用列 resume 作为输入，使用位置路径 /resume/skill 作为选择标识。表函数返回简历文件的技能节点中包含字符串 “Java” 的职员的技能。以下示例是原始 resume.xml 文件的一部分:

```
<resume emp_id="12" email_address="some_person@email_address.com">
  <experience start_date="12/01/1999" end_date="06/30/2002">
    Develop partner and customer demos.
   </experience>
  <experience start_date="12/01/1999" end_date="06/30/2002">
    IBM DB2 Information Integrator development
  </experience>
  <skill>
    Databases
  </skill>
  <skill>
    Java
  </skill>
  <skill>
    C++
  </skill>
  <skill>
    FORTRAN
  </skill>
</resume>
```

解决外部教育问题

人力资源部门需要将职员及其当前工作描述的列表发送至外部教育提供商。然后教育提供商可以向 YBar, Incorporated 的职员提供定制的类。 YBar, Inc. 开发者连接职员和工作数据库，然后使用 WebSphere® MQ 来发布消息队列中的列表。以下步骤演示了示例解决方案:

1. 工作表创建为外部平面文件的联合系统昵称。昵称创建为平面文件包装器的一部分。

```
CREATE NICKNAME job
(Job_ID INTEGER,
Job_description VARCHAR(255),
Title VARCHAR(50),
responsibilities VARCHAR(100))
FOR SERVER local_flat_files OPTIONS
(FILE_PATH 'C:\SPC\job_table.txt',
COLUMN_DELIMITER ',',
KEY_COLUMN 'job_id',
VALIDATE_DATA_FILE 'y')"
```

   FILE_PATH 选项定义名为 job_table.txt 的外部文件的位置。

2. 开发者运行选择职员信息和工作信息的查询。

```
SELECT 1 as x, emp_id , firstname, lastname,
Job_description
    FROM employee
    as e, job as j
WHERE e.Current_Job_ID = j.Job_ID
and j.Job_ID != 1000
ORDER BY x, emp_id
```

3. YBar 开发者生成新的 XML 文档，该文档包含 DB2 UDB 表、职员和外部平面文件信息的连接信息。他们将新生成的文件命名为 All_employee.xml。他们使用 employee.dad 文件将 xml 信息映射到 DB2 UDB 关系表信息。
4. 使用下列语句创建 DB2 UDB 临时表以保存 XML 文档:

```
create table tmpTable
    (x_doc DB2XML.XMLCLOB not logged)
```

5. 分解 employee.dad 和填充的 tmpTable 中的 XML 信息。然后，
6. 使用以下示例将 XML 数据写入并存储至文件 employee.xml 中:

```
select DB2XML.Content
    (x_doc, 'c:\YourName\employee.XML')
    from tmpTable
```

7. 现在准备将信息作为消息发送至外部教育提供商。将连接信息放入 WebSphere MQ 消息队列。:
8. 以下示例将消息作为 XML 文档放在队列上，显示消息队列的内容并使消息不受干扰。

   使用 MQSENDXML 函数将 XML 消息发送至队列。此消息是职员编号 12 的简历信息。

```
db2 "SELECT db2xml.MQSENDXML('DB2.DEFAULT.SERVICE',
    'DB2.DEFAULT.POLICY', resume)
    FROM employee
    WHERE emp_id=12"
```

   创建 DB2 UDB 表，此表具有一个可保存整个 XML 文档的列。

```
db2 "CREATE TABLE temporaryXML
    (XML_doc DB2XML.XMLVARCHAR)"
```

   用 XML 信息填充此表，XML 信息是 DB2 UDB Employee 表和外部工作文件的结合:

```
db2 "INSERT INTO temporaryXML
    SELECT CONCAT('<FullName>'||firstname
    ||' '||lastname||'</FullName>          ','<Job_Desc>'||
    job_description||
    '</Job_Desk>')
    FROM employee
    AS e, job AS j
    WHERE e.current_job_id=j.job_id
    AND  j.Job_ID = 1"
```

使用 MQSENDXML 函数将 XML 消息发送至队列。此消息是 XML 列中的职员和工作信息。

```
db2 "SELECT db2xml.MQSENDXML('DB2.DEFAULT.SERVICE',
    'DB2.DEFAULT.POLICY', XML_doc)
    FROM temporaryXML"
```

可在图 61 中查看 YBar, Incorporated 使用的一组完整组件。

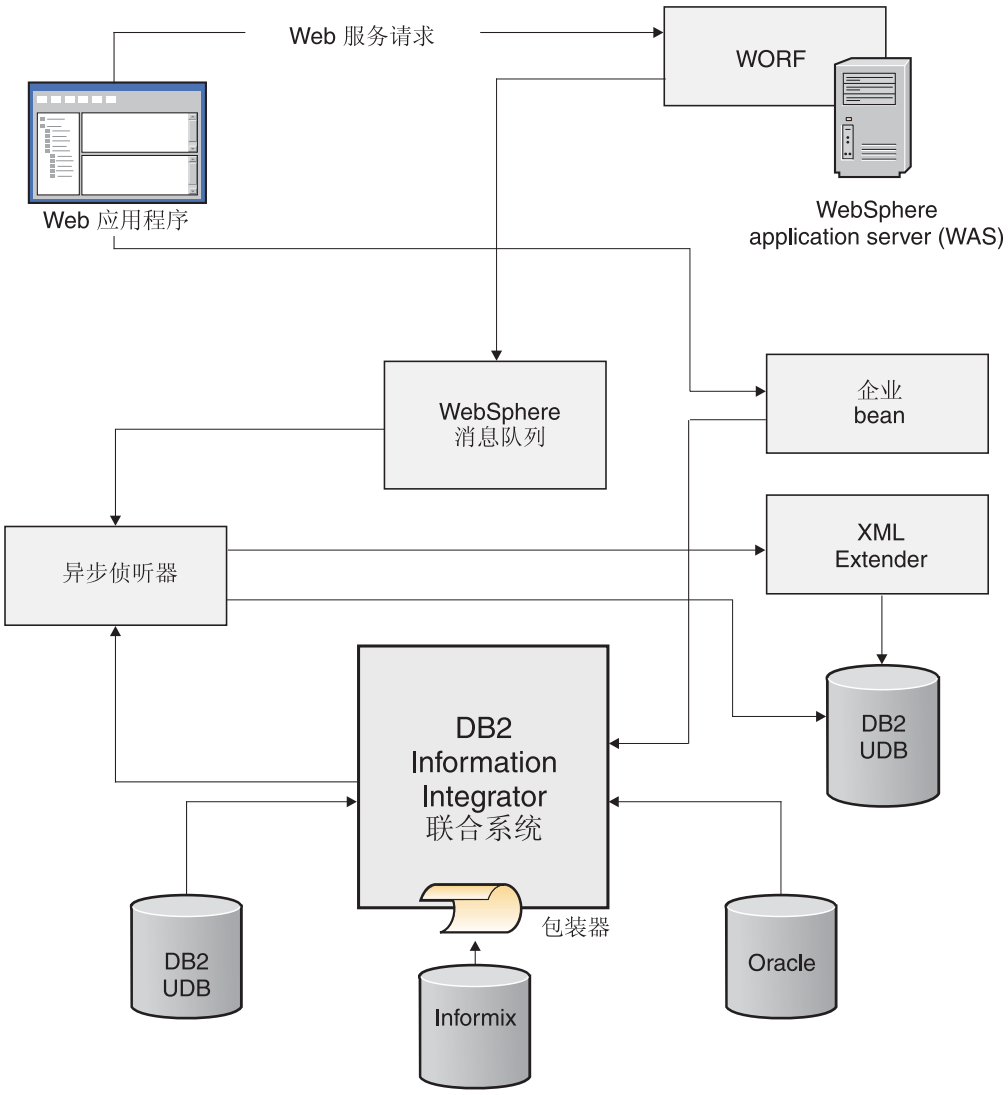


图 61. 使用 Web 应用程序的联合系统的组件

相关概念:

- 『Planning side tables』（*DB2 XML Extender Administration and Programming*）

- 『Using indexes for XML column data』（*DB2 XML Extender Administration and Programming*）
- 第 160 页的『职员数据库方案 – 解决方案设计』
- 第 10 页的『发现数据 – 职员技能方案』

职员数据库方案 – 解决方案设计

YBar, Incorporated 是一家致力于人力资源的公司，正在尝试解决两个问题:

- 他们必须找出适合某些内部工作机会的最佳候选人。他们根据 XML 格式的职员简历文件进行搜索。
- 他们向外部教育提供商提供职员列表。教育提供商根据职员需要定制类。

YBar, Inc. 使用 IBM® DB2® Information Integrator 的联合系统存储和检索数据，来解决其业务问题。YBar, Inc. 设计解决方案在第 160 页的图 62 中。此设计使用 DB2 XML Extenders 的一些函数，并且利用与数据类型定义文件中的模式相匹配的副表。副表是 DB2 UDB 表，用于抽取要经常搜索的 XML 文档的内容。XML 列与保存 XML 文档的内容的副表相关联。在应用程序表中更新 XML 文档时，副表中的值会自动更新。

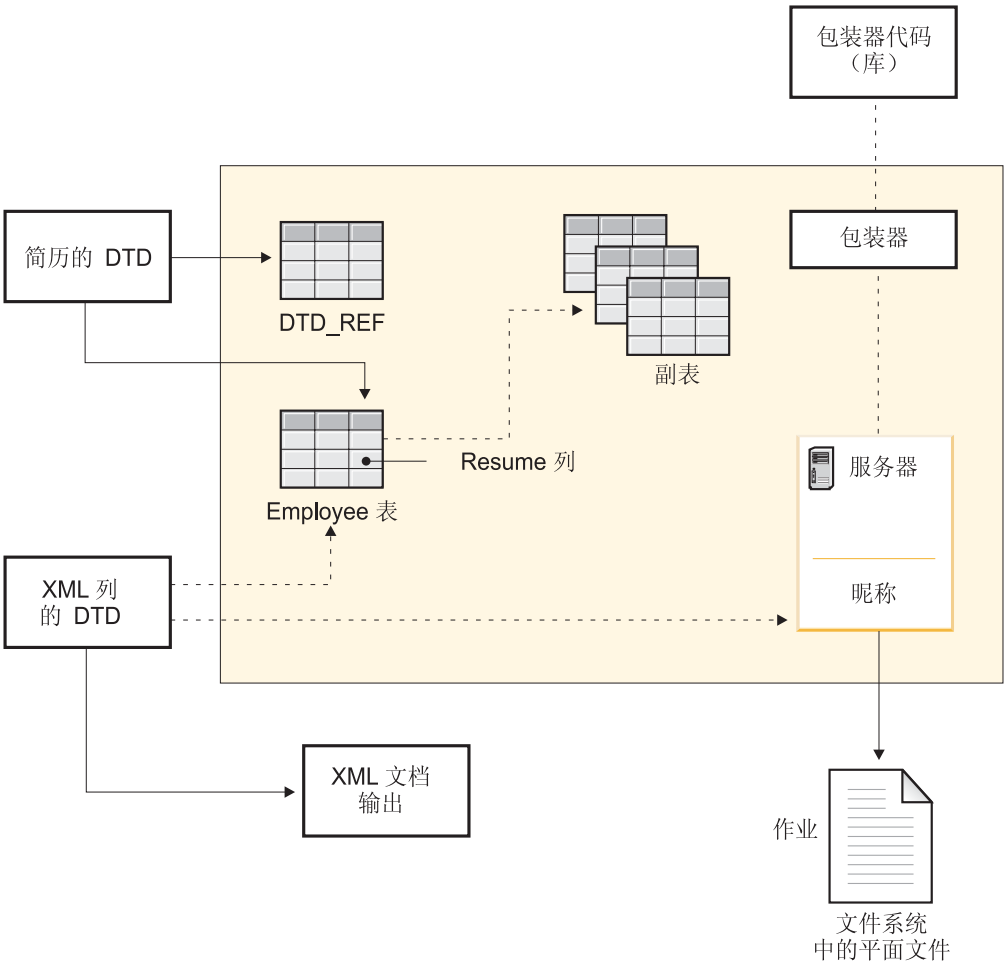


图 62. YBar, Inc. 设计流

YBar, Inc. 可以将 DB2 UDB 表中存储的 XML 数据视为可与其它数据列配合使用的关系数据。他们使用 DB2 XML Extender 的函数将 XML 格式的简历存储为完整的 XML 文档。

1. 必须为联合系统启用数据库，然后才能存取外部平面文件中存储的数据。

   ```
   DB2 UPDATE DBM CFG USING FEDERATED YES
   ```

2. 必须为 DB2 XML Extenders 启用数据库以使用 XML 函数。

   ```
   DXXADM ENABLE_DB emp_db
   ```

3. 数据库管理员向 Employee 表添加名为 “Resume” 的列。XMLVARCHAR 数据类型是 DB2 XML Extender 类型，此类型允许在 DB2 UDB 表中包含 XML 文档。

   ```
   ALTER TABLE employee ADD COLUMN Resume XMLVARCHAR
   ```

   新列包含职员数据库中的简历（格式为 XML 文档）。Resume 列提供 XML 搜索能力来查找哪些简历与特定项目所需的技能相匹配。

4. 必须创建联合数据源（例如包装器、服务器和昵称）。可以使用 DB2 UDB 控制中心来创建联合数据对象。

   以下示例在 AIX® 中注册表结构文件包装器:

   ```
   CREATE WRAPPER flat_files LIBRARY 'libdb2lsfile.a'
   ```

   必须为要使用的每个包装器注册服务器定义。

   ```
   CREATE SERVER local_flat_files WRAPPER flat_files
   ```

   注册昵称以便在 SQL 语句中使用非关系数据。FILE_PATH 选项定义指向简历数据的路径，此数据存储在另一个数据源中的平面文件内。

   ```
   CREATE NICKNAME job
   (Job_ID INTEGER,
   Job_description VARCHAR(255),
   Title VARCHAR(50),
   responsibilities VARCHAR(100))
   FOR SERVER local_flat_files OPTIONS
   (FILE_PATH 'C:\SPC\job_table.txt',
   COLUMN_DELIMITER ',',
   KEY_COLUMN 'job_id',
   VALIDATE_DATA_FILE 'y')"
   ```

5. 使用以下语句验证昵称选项是否有效:

   ```
   SELECT tabschema,
   tabname, option, setting
   FROM SYSCAT.TABOPTIONS
   ```

   SELECT 语句的结果显示在第 161 页的表 24 中。此表显示在 CREATE NICKNAME 语句中定义的模式名和表选项。

表 24. 从 TABOPTIONS 表选择的结果

TABSCHEMA	TABNAME	OPTION	SETTING
USERNAME	JOB	FILE_PATH	c:\SPC\job_table.txt
USERNAME	JOB	COLUMN_DELIMITER	,
USERNAME	JOB	KEY_COLUMN	JOB_ID
USERNAME	JOB	VALIDATE_DATA_FILE	Y
USERNAME	JOB	REMOTE_TABLE	JOB

表 24. 从 TABOPTIONS 表选择的结果 (续)

TABSCHEMA	TABNAME	OPTION	SETTING
USERNAME	JOB	REMOTE_SCHEMA	USERNAME
USERNAME	JOB	SERVER	LOCAL_FLAT_FILES

6. 必须注册文档类型定义（DTD）以将 XML 模式映射到 DB2 UDB 表，然后启用 XML 列 resume。

```
DXXADM enable_column Employee_db
  Employee resume resume.dad -v resume_view -r Emp_ID
```

DB2 XML Extender 命令 DXXADM 的enable_column 选项连接至数据库，并启用 XML 列，以便它可以包含 XML Extender 用户定义的类型。启用某个列时，XML Extender 完成几个任务，包括以下任务:

- 创建在 DAD 文件中指定的副表，此文件具有一个列，该列包含用于 XML 表中每一行的唯一标识。在此示例中，副表中的列为 Emp_ID。
- 在此示例中使用名称 resume_view，为 XML 表及其副表创建缺省视图。

以下定义用于 enable_column 语句中的各项:

- Employee_db: 数据库的名称。
- Employee: 表的名称。
- resume: XML 列的名称
- resume.dad: 包含模式定义的文档存取定义文件的名称。
- resume_view: 连接 XML 列和副表的缺省视图的名称。
- Emp_ID: XML 列表中的主键的名称，将用作副表的 root_id。

7. 可以使用 DTD 来验证 XML 列或 XML 集合中的 XML 数据。可以在 DTD 库表（一个名为 DTD_REF 的 DB2 UDB 表）中存储 DTD。DTD_REF 表的模式名为 DB2XML。 DTD_REF 表中的每个 DTD 都具有唯一标识，名为 DTDID。DTDID 可以是标识，也可以是指定 DTD 在本地系统上的位置的路径。DTDID 必须与 DAD 文件中为 <DTDID> 元素指定的值匹配。当您为 XML 启用数据库时，XML Extender 创建 DTD_REF 表。可以发出此语句从命令行插入 DTD。

```
INSERT INTO db2xml.dtd_ref VALUES
  ('resume.dtd',db2xml.XMLClobFromFile
      ('%PATH_DEMO%\resume.dtd'),0,
      'user1','user1','user1')
```

8. 可以发出此语句来验证是否正确地注册了 DTD:

```
SELECT dtdid, content FROM db2xml.dtd_ref;
```

在此语句中，content 是 DTD 的内容。

9. 使用副表和抽取用户定义的函数来查询数据:

```
SELECT e.firstname, e.lastname, e.resume
  FROM employee AS e,
 resume_skills_sidetable AS r
 TABLE(DB2XML.EXTRACTVARCHARS(e.resume,'/resume/skill')) AS t
  WHERE SUBSTR(t.returnedvarchar,1,8) LIKE '____Java'
```

XML 文档存储在列 resume 中。用户定义的函数 extractVarchars() 使用列简历作为输入，使用位置路径 /resume/skill 作为选择标识。用户定义的函数返回在其简历中列示的职员技能表。

相关概念:

- 『Planning side tables』（*DB2 XML Extender Administration and Programming*）
- 第 153 页的『联合系统的优点』
- 第 156 页的『职员技能方案 – 解决方案设计』

相关任务:

- 『使用联合视图存取不同种类的数据』（《*联合系统指南*》）
- 『将 DB2 系列数据源添加至联合服务器』（《*IBM DB2 Information Integrator 数据源配置指南*》）

创建和部署容器管理的持久性 bean

当定义容器管理的持久性实体 bean 时，使用部署描述符文件来控制包含持久数据和任何存取限制的数据源。在开发实体 bean 之后，必须设置用来管理该 bean 的特征的部署描述符，然后打包和部署 bean。

可以创建映射到联合系统昵称的容器管理的持久性实体 bean。昵称是联合对象，用来向数据库管理器提供远程数据。本节从头到尾使用的示例创建各个实体 bean，它们映射至与各 Oracle 表、DB2 通用数据库表和平面文件相关联的昵称。单个容器管理的持久性实体 bean 可以跨越多个数据源。实体 bean 提供了通过标准 Enterprise JavaBean 技术集成不同数据的方法。

先决条件:

安装 WebSphere Studio V5 或更新版本。

注册和创建 Oracle 表和平面文件的联合系统对象。

过程:

要使用 WebSphere Studio 创建和部署容器管理的持久性实体 bean:

1. 从 WebSphere Studio 中的 Java 2 Enterprise Edition（J2EE）透视图，为您创建的实体 bean 创建 EJB 项目。
2. 创建容器管理的持久性实体 bean。
   - 为容器管理的持久性实体 bean 指定与为数据源定义的昵称相同的名称。
   - 添加对应于昵称中每个列名的属性。为对应于昵称列中数据类型的每一列指定数据类型。将其中一个属性指定为关键字段，该字段将映射到昵称的主键列。
3. 打开 Enterprise JavaBean 数据建模窗口:
   a. 选择您创建的实体 bean 并右键单击以打开下拉菜单。选择**生成 —> EJB 至 RDB 映射**并选择**自顶向下建模。**
   b. 单击**下一步**。确保正确设置了数据库名称和模式名称。数据库名称必须映射至 DB2 UDB 客户机识别的联合数据库。模式名必须映射至授权的联合数据库用户。
   c. 清除**生成 DDL**。
   d. 单击**完成**。
4. 验证实体 bean 和数据库之间的映射是否成功完成。
   a. 选择实体 bean 模块并右键单击以打开下拉菜单。
   b. 选择**打开方式 —> 映射编辑器**。

c. 在**任务**窗口上纠正任何错误。

5. 将实体 bean 绑定至您为联合数据库创建的数据源。
   a. 选择实体 bean 并右键单击以打开下拉菜单。
   b. 选择**打开方式 —> 部署描述符编辑器**。
   c. 在“概述”页上，向下滚动至“WebSphere 绑定”。对于“JNDI-CMP 工厂绑定”，指定有效的 JNDI 名称和容器权限类型。
   d. 保存修改并关闭窗口。
6. 生成代码以部署实体 bean。
   a. 选择实体 bean 并右键单击以打开下拉菜单。
   b. 选择**生成部署代码**。

相关概念:

- 第 123 页的『Java 2 Enterprise Edition 应用程序』
- 第 155 页的『联合系统中的企业 bean』

为联合解决方案设计应用程序 - Cottonwood Distributors, Incorporated

Cottonwood Distributors, Incorporated（CDI）的数据库程序员创建了需要存取所有新获取的数据源的联合对象。程序员需要在访问远程系统的全部三个数据源上创建联合对象。程序员通过使用图 63 中显示的语句来创建访问。此示例显示其中一个数据源 DB2® 通用数据库的 SQL 数据定义语言（DDL）语句。

```
...
catalog tcpip node DB2_TPCH remote x.xx.xx.xx server 50000;
...
catalog database tpcd at node db2_tpch;
...
create wrapper drda;

create server db2_tpch type db2/udb version  8.1
  wrapper drda authorization "demo" password "xxxxx"
  options (dbname 'TPCD');
create user mapping
  for user SERVER db2_tpch
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname db2_part for db2_tpch.tpcd.part;
create nickname db2_supplier for db2_tpch.tpcd.supplier;
create nickname db2_partsupp for db2_tpch.tpcd.partsupp;
create nickname db2_nation for db2_tpch.tpcd.nation;
create nickname db2_region for db2_tpch.tpcd.region;
create nickname db2_customer for db2_tpch.tpcd.customer;
create nickname db2_orders for db2_tpch.tpcd.orders;
```

图 63. DB2 数据源的一些联合对象

Cottonwood 的程序员需要为三种不同的数据源创建主表（part、partsupp 和 supplier）的视图。他们通过对三种数据源使用 UNION ALL 语句来创建视图。第 165 页的图 64 中的示例显示其中一个视图。

```
CREATE  VIEW part_fed (
  p_partkey, p_mfgr, p_type, p_size, p_retailprice) AS
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM db2_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM inf_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM ora_part;
```

图 64. 联合三种数据源

Cottonwood 数据库程序员编写的程序需要访问来自外包供应商的 XML 文件。此 XML 文件包含人力资源信息，例如职员数据。要访问此文件，Cottonwood 的程序员会为 XML 文件创建包装器、服务器和昵称，如第 165 页的图 65 中所示。

```
create wrapper XML_files library 'db2lsxml.dll';
create server LOCAL_XML_FILES wrapper XML_FILES;
create nickname Employees_From_XML (
  doc Varchar(100) OPTIONS(DOCUMENT 'FILE'),
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (XPATH '//Employee');
```

图 65. 创建联合系统对象

由于 Web 应用程序将客户订单和供应商报价更新路由至队列，程序员会创建消息队列的表读取函数和视图（请参阅第 166 页的图 66）。

```
CREATE FUNCTION NEW_SUPPLIERS_READ()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                           SUPPLIER_PHONE VARCHAR(12),
                           PART_KEY DOUBLE,
                           PART_PRICE DOUBLE,
                           MAN_DAYS DOUBLE,
                           MAX_QUANTITY DOUBLE,
     CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQREADALL('DB2.DEFAULT.SERVICE',
      'DB2.DEFAULT.POLICY')) AS T;

CREATE VIEW READ_NEW_SUPPLIERS_FROM_QUEUE
  AS SELECT * FROM TABLE(NEW_SUPPLIERS_READ()) t
  WHERE CORRELID = 'CDI_NEW_SUPPLIER';
```

图 66. Cottonwood 的表读取函数

最后，Cottonwood 的程序员需要创建将在处理中使用的临时表:

- 用于运行 XML 排版:

  ```
  create table Ucustomers (x_doc DB2XML.XMLCLOB not logged);
  ```

- 用于存储 Web 请求:

  ```
  create table request_bid
    (reqkey integer not null,
     partkey integer not null,
     bid double not null);

  create table request_status
    (reqkey integer not null,
     partkey integer not null,
     suppkey integer not null,
     newquote double not null,
     currentquote double not null,
     status varchar(15) not null);
  ```

相关任务:

- 第 166 页的『为联合解决方案开发应用程序 – Cottonwood Distributors, Inc.』

相关参考:

- 第 201 页的附录 A, 『Cottonwood Distributors, Inc. 和 YBar, Inc. 方案的脚本示例』

为联合解决方案开发应用程序 – Cottonwood Distributors, Inc.

Cottonwood Distributors, Incorporated（CDI）可以通过调用具有 servlet 的 Web 服务来为任何新报价或竞标生成 XML 消息。可以依照以下步骤来创建类似的环境。

过程:

要开发应用程序以创建 Cottonwood Web 服务:

1. 配置 DB2 通用数据库服务器以用作联合数据库，并为 SVCENAME 服务器和 FEDERATED 选项设置属性:

   ```
   db2 update dbm cfg using SVCENAME myID authentication server
   db2 update dbm cfg using FEDERATED yes
   db2 connect reset
   db2stop
   db2start
   ```

2. 连接至本地 DB2 通用数据库:

   ```
   db2 connect to myLocalDB user user1 using myPW
   ```

3. 为每个数据源创建联合对象。
4. 配置 DB2 通用数据库客户机以标识 DB2 UDB 服务器所驻留的远程节点。
5. 为要存取的远程数据创建昵称:

   ```
   create nickname ora_part for oraserver.CDI.part;
   ```

6. 配置 WebSphere Application Server:
   - 确保与 Web 服务项目关联的 Java 构建路径包括 *db2java.zip* 或 *jcc.jar* 文件的位置。`group.properties` 文件中的 `dbDriver` 参数决定了要使用哪些数据库驱动程序包。
   - 在 WebSphere 环境中创建数据源对象，该环境映射到已为联合支持配置的 DB2 通用数据库。数据源对象显示池数据库连接，且可以使用 JNDI 服务建立以搜索数据源，并调用与数据源关联的方法。

所有连接就绪后，可以从联合数据对象选择、插入、更新或删除数据。因为创建了 ora_part 的昵称，您创建的 SQL 语句仅引用 ora_part 昵称。在 Cottonwood 示例中，语句包含多个源的连接数据，这些源具有引用多个昵称的单个 SQL 语句。

图 67 中显示的代码使用 Web 服务向队列写入消息。

```
public void doGet(HttpServletRequest req,
        HttpServletResponse res)
 throws javax.servlet.ServletException, java.io.IOException {
 try {
  PrintWriter pr = res.getWriter();
  pr.print(htmlHeader1);
  pr.print(message1);
  pr.print(message2);
  String method = "";

...
```

图 67. 新报价或竞标向队列写入消息（MessageFormatter.java） (1/5)

```
public class MessageFormatter extends javax.servlet.http.HttpServlet {
final static String htmlHeader1 = "<HTML><TITLE>MessageFormatter";
final static String message1 =
 "<p align=\"center\"><font color=\"navy\" face=\"verdana\"
 size=\"+2\"><b>Thank You<p><font color=\"black\"
 face=\"veranda\" size=\"+1\">";
 final static String message2 = "";

 final static String htmlHeader2 = "";
 static String quote = "";

 static Connection con = null;
 final static String url = "jdbc:db2:demo";

...
```

图 67. 新报价或竞标向队列写入消息（MessageFormatter.java） (2/5)

```
   WSProxy WSid = new WSProxy();
   boolean fCustomer = true;
   if (req.getParameter("method")!=null) {
    //Get the common parms to the servlet
    //from the REQ object
    key = req.getParameter("name");
    part = req.getParameter("part");
    method = req.getParameter("method");

    // If customer order
    if (method.equals("orderNewParts")) {
     fCustomer = true;
     // Get the quantity the customer is ordering
     quantity = req.getParameter("quantity");

      // set the table and column names for SELECT
     col_name1 = "c_name";
     col_name2 = "c_custkey";
     tab_name  = "db2_customer";

    ...
```

图 67. 新报价或竞标向队列写入消息（MessageFormatter.java） (3/5)

```
      // Get the real customer name from federated data source
    try {
   Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
     url = "jdbc:db2:demo";
     con = DriverManager.getConnection(url,"demo","xxxxx");
     stmt = con.createStatement();
     rs = stmt.executeQuery
        ("SELECT " + col_name1 + " from " + tab_name +
          " where " + col_name2 + " = " + key);
     while (rs.next()) {
      cust_name = rs.getString(1);
     }
...
```

图 67. 新报价或竞标向队列写入消息（MessageFormatter.java） (4/5)

```
     // Write request status back to user
   try{
    // If this is a supplier update request
    if (!fCustomer) {
     String message1id=  "2," + key + "," + part + ","
              + quantity + "," + price ;
           java.lang.String message1idTemp  = message1id;
     System.out.println
           ("message type 2 being written: " + message1idTemp);
     org.tempuri.worftestweb.
        demo.newdadx.dadx.xsd.Stmt1ResultElement mtemp =
     WSid.stmt1(message1idTemp);
```

图 67. 新报价或竞标向队列写入消息（MessageFormatter.java） (5/5)

CustomerRoutine 对消息进行语法分析。例程根据消息类型（报价或竞标）采取操作。例程将已语法分析的信息放入 REQUEST_BID 或 REQUEST_STATUS 表中。如果消息类型是对某个部件的报价，它会将新报价与现有的报价进行比较。如果这是对新部件的报价，状态会更改为 NEW。状态添加至本地表和状态表，显示为 NEW。如果报价是针对现有部件且价格较低，程序则会接受此价格。程序向本地表写入，且将状态更新为 ACCEPTED。如果报价是针对现有部件且价格较高，程序则会向状态表写入，状态为 REVIEW。

相关概念:

- 第 176 页的『部署应用程序 - Cottonwood Distributors, Inc. 解决方案』
- 第 173 页的『发现数据 - Cottonwood Distributors, Inc.』

部署联合应用程序

为了部署联合应用程序，Cottonwood Distributors, Incorporate 的数据库程序员为 Web 服务操作调用 *newdadx.dad*。以下是一些示例，可帮助您部署应用程序。

先决条件:

WebSphere Application Server 5.0 中的最小可部署单元是 Web 归档（WAR）文件。如果应用程序正在开发 Enterprise JavaBeans（EJB），则 Java 归档文件（JAR）和企业归档文件（EAR）是必需的。

必须为企业联合应用程序构建以下组件，然后才可以部署应用程序:

WAR 文件

Web 相关的组件（超文本标记语言（HTML）、JavaScript、JSP）

JAR 文件

构成业务逻辑组件的 Java 类

EAR 文件

组成企业解决方案的 JAR 文件和 WAR 文件

过程:

要部署联合应用程序:

1. 将 EAR 文件导入至 WebSphere Application Development 环境中。对于 Cottonwood Distributors, Incorporated, EAR 文件（CDI.ear）包含以下文件:
   - .project
   - META-INF/.modulemaps
   - META-INF/application.xml
   - META-INF/ibm-application-ext.xml
   - META-INF/MANIFEST.MF
   - WorfTestWeb.war
2. 通过选择 WebSphere Studio 窗口中的每个 DADX 文件然后单击**作为 Web 服务部署**按钮，部署 Web 服务（ “文档存取定义扩展”（DADX）文件）。

图 68. 选择 DADX 文件

3. 重新启动 WebSphere Application Development 服务器。

部署的应用程序提供 Cottonwood 的新合并企业所需要的客户竞标请求和竞标状态。客户竞标请求通过 Web 服务发送。实际的请求是被路由到 WebSphere 消息队列的消息。Cottonwood 在消息队列上有一个侦听活动的应用程序，因此程序可以在请求进入队列时从队列检索数据库请求。此侦听器应用程序调用一组操作以处理请求。 Cottonwood 使用的侦听器会平衡由进入队列的竞标请求数量所产生的负载，然后按以下步骤处理客户竞标请求:

1. 客户竞标请求向队列写入消息。
2. 侦听器应用程序调用 DB2 通用数据库函数以便从请求中抽取订单信息。
3. Cottonwood 的数据库程序员运行只读查询以获取对所请求部件的报价。
4. Cottonwood 的数据库程序员在本地表中插入一条记录以记录订单。

Cottonwood 通过与客户竞标请求类似的一系列步骤处理供应商报价更新。但是，供应商报价查询不是只读的，原因是程序需要更新报价。侦听器应用程序使 Cottonwood 可以先检查请求，然后才允许落实更新。以下是供应商报价请求的特定步骤:

1. 供应商报价请求向队列写入消息。
2. 根据消息的格式，侦听器应用程序调用 DB2 通用数据库函数来抽取请求以更新对某个部件的报价。
3. Cottonwood 的数据库程序员从该部件的供应商处查看先前的最低报价。如果新的报价较低，或者此报价是针对该供应商先前未提供的部件，应用程序会更新数据库。如果新的报价较高，Cottonwood 可能不进行更新。相反，Cottonwood 应用程序会对报价进行标记，供 Cottonwood 销售小组以后查看或与供应商进行谈判。

客户订单和供应商价格更新的整个流程显示在图 69 和第 172 页的图 70 中。

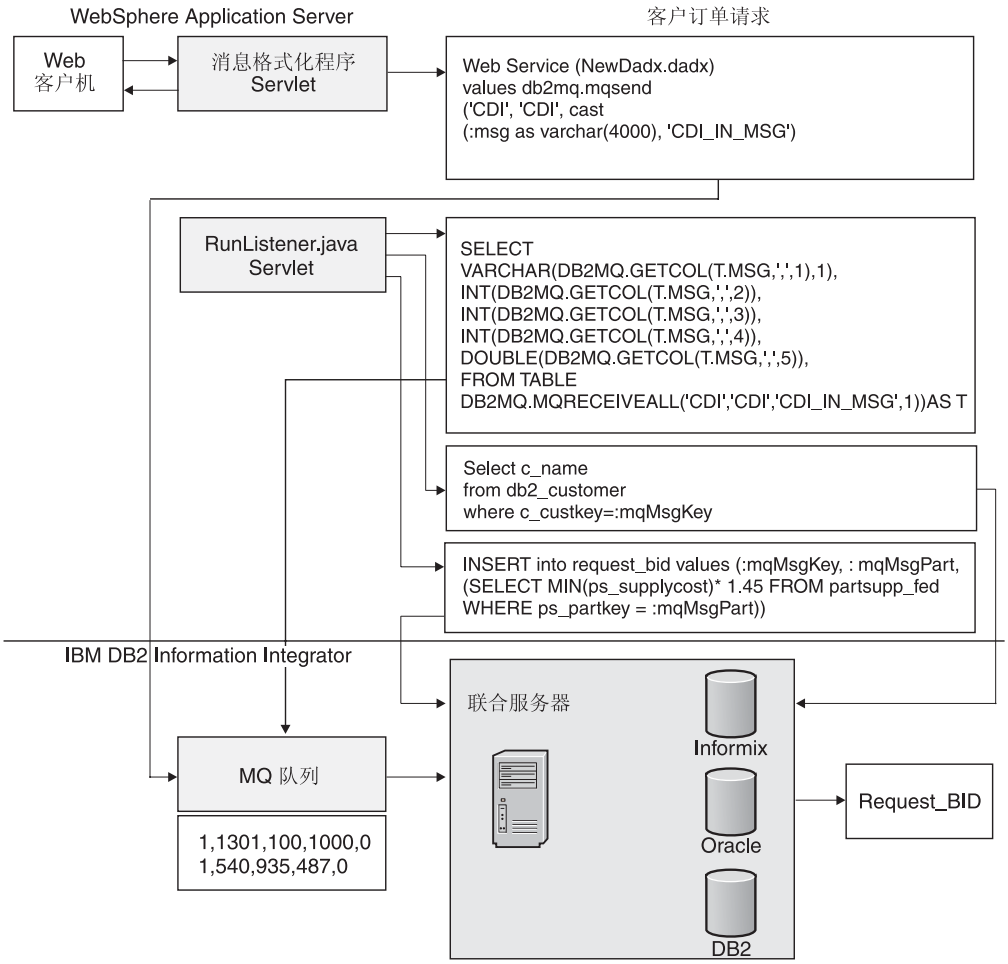


图 69. Cottonwood 客户订单请求

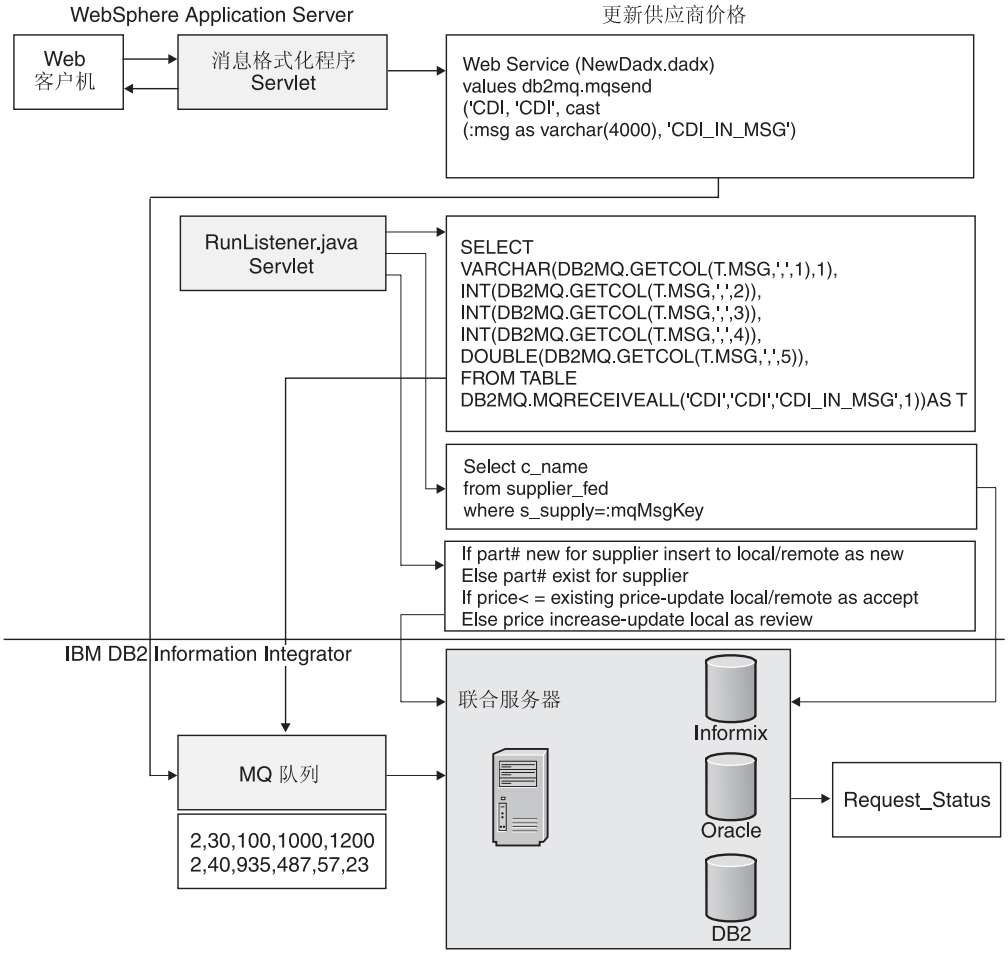


图 70. 可以更新供应商价格

相关任务:

- 第 127 页的『准备和创建 Web 归档文件』

扩展数据仓库

本节讨论了如何将 DB2 仓库管理器和数据仓库中心作为信息集成解决方案的一部分的特定示例。

业务解决方案: 扩展 DB2 仓库管理器

DB2® 仓库管理器改进了与 DB2 和 IBM® DB2 Information Integrator 实用程序函数的集成。可以使用 SQL UPDATE 语句，用来自源的数据更新目标数据。另外，可以定义一个进程，该进程等待多个步骤完成。仓库变换器（包括存储过程和用户定义的函数）提供了常用变换，用于构建数据仓库。

数据仓库中心是元数据驱动的系统。元数据或关于数据的信息向管理员和商业用户提供数据仓库中存储的数据的描述。可以创建信息目录，它描述业务项中的业务元数据。可以将元数据组织到主题区域，并根据您的工作组或企业需要对其进行定制。

通过使用“干净数据”变换器等技术，可以对源数据执行基本的擦除、替换和映射操作。还可以将应用程序定义至数据仓库中心，以便一个或多个步骤可以将程序用于处理。在将用户定义的程序定义至数据仓库中心之后，该程序定义可用作“进程技术模型”窗口中的一个步骤。仓库源识别将向仓库提供数据的表和文件。数据仓库中心使用仓库源中的规范来存取数据。源几乎可以是任何关系型或非关系型源（表、视图、文件或预定义的昵称）。

使用信息目录管理器来为仓库步骤提供数据关系和对象定义的图形表示法。信息目录管理器提供了功能强大的面向业务的解决方案，以帮助用户查找、了解和存取企业数据。信息目录管理器使商业用户可以查看聚集、历史、数据派生、数据源和数据描述。

相关概念:

- 『数据仓库中心配置』（*《数据仓库中心管理指南》*）
- 第 10 页的『Cottonwood Distributors, Inc. – 仓库示例』

发现数据 – Cottonwood Distributors, Inc.

当 Cottonwood Distributors, Incorporated 开始就收购公司的未来作出决策时，他们确定了一些特定的关注领域。

- Cottonwood 程序员处理的数据量比合并之前多。数据位于不同的物理位置，位于不同的数据库中，且具有不同的格式。
- Cottonwood 需要保持竞争性。公司决定用基于 Web 的经纪商系统代替基于电话的销售代表。Cottonwood 基于 Web 的经纪商系统的用户将对部件请求在线竞标，并对部件在线发出订单。Cottonwood 还有一组供应商需要提交部件的新报价或更新报商集合。

Cottonwood Distributors, Incorporated 必须确定他们新扩张的公司需要哪些数据。还必须确定数据将作何种用途。然后必须确定数据源的位置。

考虑到 Cottonwood Distributors, Inc. 所面临的挑战，IBM® DB2® Information Integrator 联合系统和数据仓储对他们是重要的技术。联合基础结构允许客户机应用程序查看不同集合的源中的数据，就好像数据是位于一个数据库中。应用程序员不必编写处理多个应用程序编程接口（API）的代码，单个 SQL 语句就可以将 Cottonwood 全部三个数据源的数据组合在一起。仓储提供了靠近 Cottonwood 服务器的数据高速缓存。此高速缓存使经常引用的数据易于被用户使用。

Cottonwood 数据管理小组要针对如何存取其数据以及如何交换所拥有的数据作出决定。当要做出决策且决策者（例如 Cottonwood）偶然发现与决策相关的数据时，公司可以从数据中获取有用的信息。Cottonwood 发现的关键是了解到特定的数据存在。Cottonwood 还知道可以收集数据且数据与决策有关。

Cottonwood 希望知道其各个供应商是否向每个合并的公司就部件提供不同的报价。Cottonwood 的数据库程序员需要知道包含供应商数据的数据源的位置。他们需要知道数据源的位置以及如何连接到数据源。如果没有联合系统，此任务需要使用三个 SQL 语句连接到三个数据库。其它发现涉及确定有关客户和供应商的信息和关系，以及客户向基于 Web 的经纪商系统发出的请求。关于未填写订单的客户的信息对于确定优化技术或界面改善很有用。

相关概念:

- 第 172 页的『业务解决方案: 扩展 DB2 仓库管理器』
- 第 164 页的『为联合解决方案设计应用程序 – Cottonwood Distributors, Incorporated』
- 第 174 页的『设计应用程序 – Cottonwood Distributors, Inc. 仓库方案』
- 第 10 页的『Cottonwood Distributors, Inc. – 仓库示例』
- 第 176 页的『部署应用程序 – Cottonwood Distributors, Inc. 解决方案』

相关任务:

- 第 166 页的『为联合解决方案开发应用程序 – Cottonwood Distributors, Inc.』

相关参考:

- 第 201 页的附录 A, 『Cottonwood Distributors, Inc. 和 YBar, Inc. 方案的脚本示例』

设计应用程序 – Cottonwood Distributors, Inc. 仓库方案

Cottonwood Distributors, Incorporated 将联合系统技术与仓储组合起来以解决数据可伸缩性、数据辅助功能和数据流通性的问题。

Cottonwood 具有一个数据库，该数据库包含表 PART、表 SUPPLIER 和表 PARTSUPP。

表 25. Cottonwood SQL 主表和列

PART	SUPPLIER	PARTSUPP
p_partkey（唯一值）	s_suppkey	ps_partkey
p_name	s_name	ps_suppkey
p_mfgr	s_address	ps_availqty
p_brand	s_nationkey	ps_supplycost
p_type	s_phone	ps_comment
p_size	s_acctbal	
p_container	s_comment	
p_retailprice		
p_comment		

Cottonwood 还具有其它表，这些表与主表形成关系以帮助为其数据管理问题创建解决方案。

表 26. CDI 辅助表和列

NATION	REGION	CUSTOMER	ORDERS
n_nationkey	r_regionkey	c_custkey	o_orderkey
n_name	r_name	c_name	o_custkey
n_regionkey	r_comment	c_address	o_orderstatus
n_comment		c_nationkey	o_totalprice
		c_phone	o_orderdate
		c_acctbal	o_orderpriority
		c_mktsegment	o_clerk
		c_comment	o_shippriority

表 26. CDI 辅助表和列 (续)

NATION	REGION	CUSTOMER	ORDERS
			o_comment

PART 表包含关于数据库中每个部件的标识关键字、生产商、类型、大小和零售价格的信息。部件标识在行业间是唯一的。部件号 1234 始终标识 widgetA。SUPPLIER 表存储关于供应商的名称和地址的信息，并包含标识关键字。PARTSUPP 表将部件与供应商关联。它包含部件和供应商表的标识以及供应商对部件的报价。

这些表包含数据的索引，以便于使用 Cottonwood 清单。 Cottonwood 的数据库程序员使用包含三种主表组合的视图（请参阅第 174 页的表 25）。尽管数据已存储在 DB2® 通用数据库、Informix® 和 Oracle 中，Cottonwood 的数据库程序员可以使用单个 SQL 语句来获取关于 Cottonwood 供应商及其针对部件向每个合并公司所作报价的最佳信息。

除了包含在数据库中的数据之外，他们还必须维护包含 XML 内容的有用报告系统。

Cottonwood 还具有包含 DB2 Universal Database™（DB2 通用数据库）服务器和 DB2 通用数据库客户机、消息队列客户机以及 Web 客户机的配置。在 DB2 服务器上，Cottonwood 具有数据仓库、联合系统、存储过程、XML 和消息队列集成。Web 应用程序服务器包含报价和竞标应用程序、某些 Web 服务以及 Java™ Server Pages（JSP）。WebSphere® MQ 也位于具有侦听器的服务器上。 Cottonwood 程序员需要存取远程 DB2 服务器、Oracle 以及 Informix 服务器上的数据。

Cottonwood 可以将联合系统与数据仓库组合以解决某些可伸缩性和系统装入问题。仓库包含所有数据管理系统中的数据。Cottonwood 在不同数据库中存储数据的事实对应用程序是透明的。仓库还提供了一种方法来保护精心调整的数据管理系统免受最终用户分析员所生成的流量的影响。例如，PART 表是 Informix、Oracle 和 DB2 通用数据库 PART 表的结合。而且，对表的引用与它们在底层数据库上的引用是相同的。先前仅在 DB2 通用数据库上运行的应用程序经过很少的应用程序更改就可迁移至数据仓库。

仓库方案设计（请参阅第 176 页的图 71）涉及设置仓库环境（包括仓库控制数据库）。Cottonwood 方案使用由数据仓库中心创建的缺省数据仓库安全组。Cottonwood 的数据库程序员根据某些预定义的计划从数据商店抽取数据。然后程序员应用特定的业务规则以擦除数据。随后，数据被装入数据仓库。Cottonwood 将数据仓库分为供应商数据商店、订单数据商店和部件数据商店。Cottonwood 的数据库程序员将定义的数据集从仓库抽取到数据集市。数据集市提供对特别收集以支持分析、报告和量度的数据的存取。从这些数据集市，用户可以运行数据挖掘功能，以发现数据中的趋势、关系和模式。

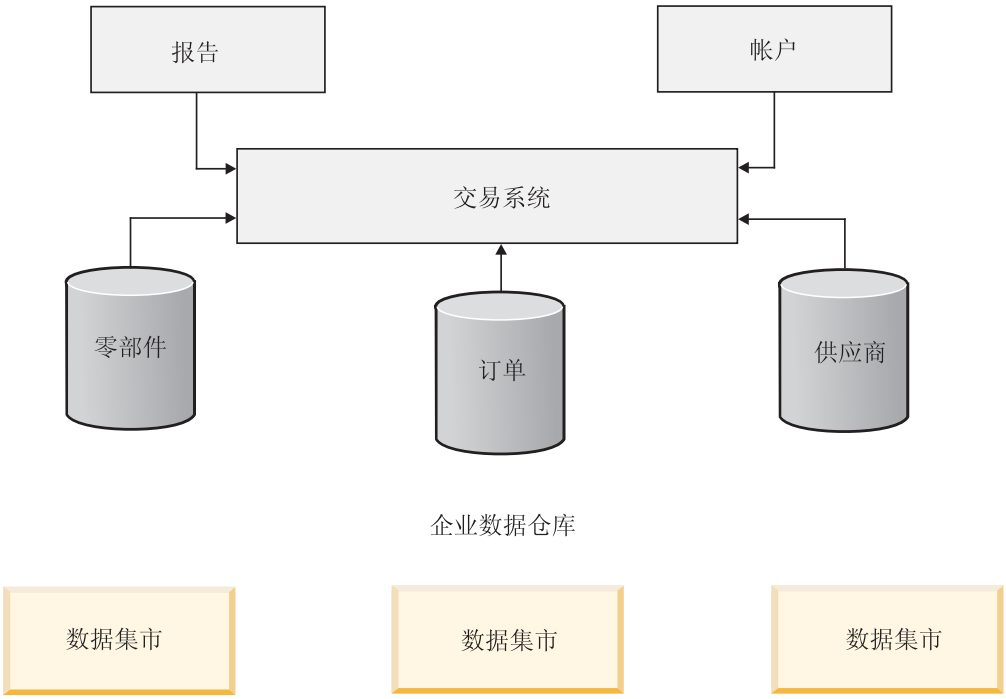


图 71. Cottonwood 的仓库配置

相关概念:

- 第 164 页的『为联合解决方案设计应用程序 - Cottonwood Distributors, Incorporated』
- 第 176 页的『部署应用程序 - Cottonwood Distributors, Inc. 解决方案』

部署应用程序 - Cottonwood Distributors, Inc. 解决方案

定义表且插入数据之后，Cottonwood Distributors, Incorporated 的数据库程序员可以生成关于可用部件和当前价格的信息。Cottonwood 的数据库程序员可以使用 DB2® 仓库管理器来帮助他们监视和管理其集成系统。DB2 仓库管理器允许这些程序员定义和运行用来存取运作联合系统数据源的变换进程，然后写入到仓库数据库。Cottonwood 的数据库系统可以通过使用 DB2 仓库管理器的外部触发器功能来与 DB2 仓库管理器进行交互作用。

“DB2 数据仓库中心”图形界面可帮助 Cottonwood 的数据库程序员管理数据从联合系统运作数据至仓库的移动。Cottonwood 的最终用户可以使用此仓库。通过使用“仓库中心”窗口，Cottonwood 的数据库程序员可以管理几个仓库进程。Cottonwood 程序员可以安排这些进程以指定的计划运行。

Cottonwood 程序员当前可以运行仓库进程以更新以下本地表:

- PART 表
- SUPPLIER 表
- PARTSUPP 表

为了 Cottonwood Distributors 解决方案，Cottonwood 程序员可以通过创建联合索引或索引规范来改进存取性能。有关在联合数据库中创建索引规范的更多信息，请参阅《*联合系统指南*》。

Cottonwood 数据库程序员可以使用他们的应用程序来存取联合仓库和本地仓库。例如，Cottonwood 的数据库程序员希望构建一个报告，使他们能够获取某个客户订单的当前最佳价格。仓库当前包含具有所有客户订单的表。Cottonwood 程序员希望向其最佳客户提供最佳价格。他们还希望跟踪供应商所作的价格更改。Cottonwood 的数据库程序员执行以下操作:

1. 创建报告，从仓库客户订单表中选择数据。
2. 从联合源中选择价格并对价格进行比较，以查看价格自从上次仓库更新以来是否有所降低。
3. 如果价格较低，Cottonwood 程序员则会选择性地通知其最佳客户: 成本降低了。

相关概念:

- 第 164 页的『为联合解决方案设计应用程序 - Cottonwood Distributors, Incorporated』
- 第 174 页的『设计应用程序 - Cottonwood Distributors, Inc. 仓库方案』
- 『联合开发方案 - Cottonwood Distributors, Inc.』（*DB2 Information Integrator Solutions Guide*）
- 第 173 页的『发现数据 - Cottonwood Distributors, Inc.』
- 第 19 页的『性能和调整规划 - 联合系统中的具体查询表』

相关任务:

- 第 166 页的『为联合解决方案开发应用程序 - Cottonwood Distributors, Inc.』

开发使用 WebSphere Message Queue 函数的数据库应用程序

IBM 认为 Web 服务不但提供集成应用程序的体系结构，而且还提供集成数据的体系结构。DB2 提供管理数据和提供对数据的智能优化存取的功能。

安装 DB2 WebSphere MQ 函数

在 DB2 通用数据库中，DB2 WebSphere MQ 函数是作为用户定义的函数提供的。这些函数允许用户从 DB2 UDB 对象访问 WebSphere MQ 队列。

先决条件:

1. 确保您的 DB2 UDB 安装添加了以下库:
   - UNIX 平台: *libdb2qgmq* 和 *libdb2mqsw*（在目录 sqllib/lib 中）
   - Windows 平台: *db2qgmq.dll* 和 *db2mqsw.dll*（在目录 sqllib\bin 中）
2. 使用随 DB2 通用数据库版本 8 或更新版本附带的 AMI 安装映像安装“应用程序消息传递接口”（AMI） V1.2.4 或更新版本。
3. 将 AMT_DATA_PATH 环境变量添加到 DB2 UDB 所使用的列表以确保消息排队用户定义的函数（MQ UDF）正确执行。可以编辑文件 *$INSTHOME/sqllib/profile.env*（UNIX）或 *%DB2PATH%\profile.env*（Windows），然后将 AMT_DATA_PATH 添加到 DB2ENVLIST。还可以使用 db2set 命令:

   ```
   db2set DB2ENVLIST="AMT_DATA_PATH"
   ```

   重新启动数据库实例以使环境变量更改生效。
4. 安装 DB2 通用数据库 XML Extender，它是作为 DB2 通用数据库版本 8 中的一个选项提供的。

5. 如果打算使用 publish 和 subscribe 函数，请使用 amtsamp.tst 文件创建系统缺省 AMI 对象。

配置和启用 DB2 WebSphere MQ 函数的基本步骤如下:

1. 安装 WebSphere MQ V5.3、“校正服务软盘”05（CSD05）或更新发行版。
2. 从位于 http://www.ibm.com/software/ts/mqseries/ 的 Support 页面安装下列 SupportPac:
   - WebSphere MQ 应用程序消息传递接口
   - 包含发布或预订能力的 WebSphere MQ SupportPac
3. 如果要使用事务性 MQ UDF，应确保为联合操作配置数据库。使用以下命令执行此操作:

   ```
   update dbm cfg using federated yes
   ```

4. 启用“DB2 UDB MQ 函数”（请参阅**过程**中的步骤第 178 页的 1）

限制:

- 存在于模式 **db2mq1c** 下的 DB2 UDB MQ 事务函数不支持 CLOB 类型消息。
- 事务性 MQ 用户定义的函数的启用实用程序只允许在用 -q 选项指定的队列管理器的 AMI 库文件中存在 40 个 AMI 策略。
- 事务 MQ 用户定义的函数在单个事务中只支持一个“队列管理器”。在“服务和策略”（通过 amthost.xml 中的连接）中指定的队列管理器必须匹配。如果您在服务点中将“队列管理器”保留为空白，则 WebSphere MQ 将“策略”指定的管理器作为缺省设置。存在一组缺省 MQ 队列和一个通常在 MQ 安装和 enable_MQFunctions 进程期间创建的缺省“队列管理器”。
- 必须创建队列和 WebSphere MQ 对象才能在 SQL 语句中使用它们。
- 如果使用发布和预订函数，则在将它们与 SQL 语句配合使用之前，需要创建特定的 WebSphere MQ 对象。可通过 WebSphere MQ 和 AMI 提供的 *.tst 文件（amtsamp.tst 和 amtsdfts.tst）发出 MQSC 命令来执行此操作。为此，使用下列步骤:
  1. 确保具有 amtsamp.tst 和 amtsdfts.tst 文件
  2. 如果需要的话，为队列管理器更新 *.tst 文件
  3. 启动 AMI 服务所使用的队列管理器
  4. 发出类似如下的命令:

     ```
     runmqsc QMName <amtsamp.tst
     ```

过程:

将命令 **enable_MQFunctions** 和 **disable_MQFunctions** 用于事务性和非事务性 MQ 用户定义的函数。MQ 用户定义的函数被定义为另一模式名下的组或集合。不支持事务的组具有模式 **db2mq**。支持事务的组具有模式 **db2mq1c**。具有支持事务的选项的 **enable_MQFunctions** 命令允许您为事务支持选择要安装或卸载的一组 MQ 用户定义的函数。要使用 **enable_MQFunctions**，请参阅、enable_MQFunctions 和 disable_MQFunctions 的完整命令语法。

1. 为 WebSphere MQ 函数配置和启用数据库。 enable_MQFunctions 实用程序是一个灵活的命令。它首先检查您是否正确设置了 WebSphere MQ 环境。接着它为 WebSphere MQ 函数安装和创建缺省配置。然后，它启用具有这些函数的特定数据库，并确认该配置能够工作。

   下列示例假定用户已连接至数据库 SAMPLE。

示例 1: 启用事务性和非事务性用户定义的函数:

```
enable_MQFunctions -n sample -u user1 -p password1
```

示例 2: 在模式 DB2MQ1C 下创建 DB2MQ1C 函数:

```
enable_MQFunctions -n sample -u user1 -p password1 -v 1pc
```

2. 通过使用“命令行处理器”（在 Windows 环境上）来测试 MQ 函数。在连接至当前已启用的数据库之后发出以下命令:

```
values DB2MQ1C.MQSEND('a test')
values DB2MQ1C.MQRECEIVE()
```

第一个语句将消息 a test 发送至 DB2MQ_DEFAULT_Q 队列。第二个语句接收回消息。此语句假定您已经使用了一些缺省配置。可以在可作为工作单元的一部分落实或回滚的 DB2 事务中使用这两个语句。

相关概念:

- 第 189 页的『DB2 Information Integrator 中的异步消息传递』
- 第 186 页的『如何在 DB2 中使用 WebSphere MQ 函数』

相关任务:

- 第 193 页的『配置 WebSphere MQ for MQListener』
- 第 195 页的『配置 MQListener』
- 第 192 页的『将 MQListener 配置为在 DB2 通用数据库环境中运行』
- 第 192 页的『配置和运行 MQListener』

相关参考:

- 『enable_MQFunctions』（*Command Reference*）
- 『disable_MQFunctions』（*Command Reference*）

WebSphere MQ 和 DB2 应用程序集成概述

使用 DB2® 通用数据库和 WebSphere® MQ 创建 SQL 请求、开发存储过程、扩展包含用户定义的函数的数据库，并将数据库请求输入 Web 服务。消息传递、排队、发布和预订都是数据库应用程序环境中的常见技术。这些技术帮助将完全不同的应用程序链接在一起、传播实时信息和集成企业中的数据和通信。

WebSphere MQ 是一个消息处理系统，它使应用程序能够在分布式环境中跨不同操作系统与网络进行通信。WebSphere MQ 通过使用应用程序编程接口（API）处理从一个程序至另一个程序的通信。

“应用程序消息传递接口”（AMI）是 WebSphere MQ 的常用 API，以多种高级语言提供。除了 AMI 之外，DB2 UDB 还通过一组外部用户定义的函数（称为 DB2 WebSphere MQ 函数）为 WebSphere MQ 消息传递系统提供了它自己的应用程序编程接口。在 SQL 语句中使用这些函数允许您组合 DB2 Universal Database™（DB2 通用数据库）存取与 WebSphere MQ 消息处理。

消息处理和 AMI 简介

WebSphere MQ 消息处理系统取用一段信息（消息）并将它发送至其目标。尽管可能会发生任何网络中断，但 WebSphere MQ 仍能保证传递成功。

应用程序员使用 AMI 来发送消息和接收消息。AMI 中的三个组件为:

- *消息*，它定义一个程序发送至另一个程序的**内容**
- *服务*，它定义消息来自或发送至**何处**
- *策略*，它定义**如何**处理消息

要发送使用 AMI 的消息，应用程序必须指定消息数据、服务和策略。系统管理员定义特定安装所需的 WebSphere MQ 配置，包括缺省服务和缺省策略。DB2 UDB 提供缺省服务、缺省策略、DB2.DEFAULT.SERVICE 和 DB2.DEFAULT.POLICY，应用程序员可以使用这些项目简化其程序。

有关 AMI 的详细信息，请参阅 *MQSeries Application Messaging Interface*。

WebSphere MQ 消息

WebSphere MQ 使用消息来在应用程序之间传递信息。消息由下列部件组成:

- 消息属性，它们标识消息及其属性。AMI 使用属性和策略来解释和构造 MQSeries® 头和消息描述符。
- 消息数据，它是消息中携带的应用程序数据。AMI 不对此数据执行操作。

属性是 AMI 消息的属性。借助 AMI，消息可包含属性，或者系统管理员可在缺省策略中定义属性。程序员不考虑消息属性的详细信息。

WebSphere MQ 服务

服务描述应用程序将消息发送至或应用程序从中接收消息的目标。WebSphere MQ 将目标称为消息队列，而队列驻留在队列管理器中。

应用程序可使用 AMI 来将消息放置在队列中或从队列中获取消息。系统管理员设置参数以管理队列，这些参数由服务定义。因此，AMI 对程序员隐藏了复杂性。应用程序通过将服务指定为 DB2 MQSeries 函数调用的参数来选择服务。

WebSphere MQ 策略

策略控制 AMI 函数如何处理消息。策略控制诸如以下各项:

- 消息的属性，例如，优先级
- 发送和接收操作的选项，例如，操作是不是工作单元的一部分

AMI 提供缺省策略。或者，系统管理员可定义定制策略并在资源库中存储这些策略。应用程序可将策略指定为参数以供 DB2 MQSeries 函数调用。

DB2 MQSeries 函数的功能

对 DB2 使用 MQSeries 函数有两种方法:

- 没有事务语义的用户定义的函数（模式名为 DB2MQ）
- 使用一阶段落实语义的用户定义的函数（模式名为 DB2MQ1C）

DB2 WebSphere MQ 函数支持下列操作类型:

- 发送和遗忘，其中消息不需要应答
- 读取，其中应用程序可阅读一条或全部消息而不从队列中除去它
- 接收，其中应用程序可从队列接收并除去一条或全部消息
- 请求和响应，其中发送应用程序需要对请求的响应

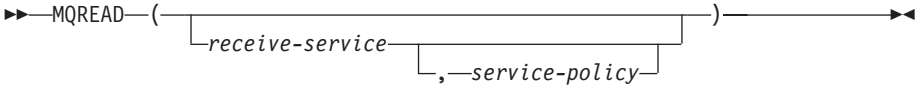
可以使用 DB2 WebSphere MQ 函数来将消息发送至消息队列或从消息队列接收消息。另外，可将请求发送至消息队列并接收响应。

应将 WebSphere MQ 服务器定位在 DB2 数据库服务器所在的系统上。通过使用 AMI 向 DB2 数据库服务器注册 DB2 WebSphere MQ 函数并提供对 WebSphere MQ 服务器的访问权。有关安装 DB2 MQSeries 函数的信息，请参阅安装 DB2 WebSphere MQ 函数。

DB2 WebSphere MQ 函数包括标量函数和表函数。请记住模式名指示用户定义的函数的类型。有关 MQ 函数的更多信息，请参阅设置 DB2 WebSphere MQ 函数。下列定义描述 DB2 WebSphere MQ 标量函数。

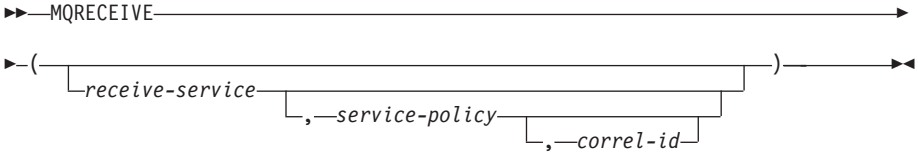
MQREAD

此操作使用在 *service-policy* 中定义的策略从 *receive-service* 指定的 MQSeries 位置在 VARCHAR 变量中返回消息。此操作不会从队列头中除去消息，只是返回消息。如果没有消息可供返回，则返回空值。



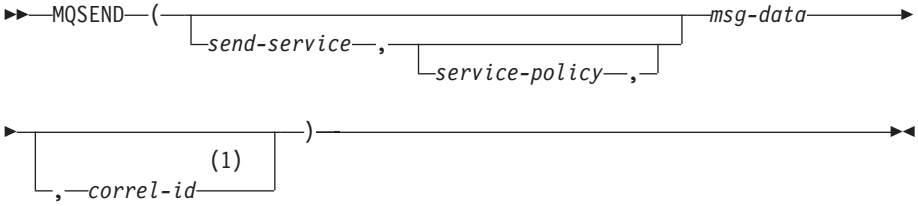
MQRECEIVE

此操作从 *receive-service* 指定的 MQSeries 位置使用在 *service-policy* 中定义的策略在 VARCHAR 变量中返回消息。此操作从队列中除去消息。如果指定了 *correlation-id*，则返回具有匹配相关标识的第一条消息；如果未指定 *correlation-id*，则返回队列头的消息。如果没有消息可供返回，则返回空值。



MQSEND

此操作使用在 *service-policy* 中定义的策略将 VARCHAR 变量 *msg-data* 中的数据发送至 *send-service* 指定的 MQSeries 位置。可以用 *correlation-id* 指定可选的用户定义的消息相关标识。如果成功，则返回值为 1，如果不成功，则返回值为 0。



注:

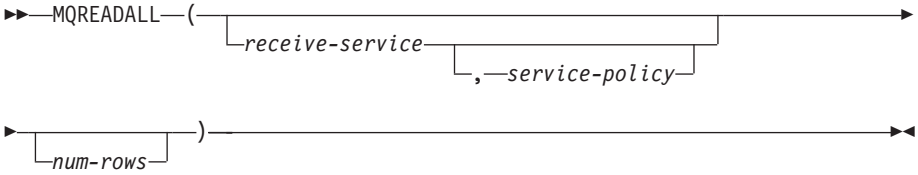
1 除非同时指定了 *service* 和 *policy*，否则不能指定 *correl-id*。

可以在模式 DB2MQ 和 DB2MQ1C 的 VARCHAR 变量中发送或接收消息。VARCHAR 变量中 DB2MQ 消息的最大长度是 4,000 个字节。DB2MQ1C 支持长达 32,000 个字节的 VARCHAR。

下列定义描述 DB2 MQSeries 表函数。

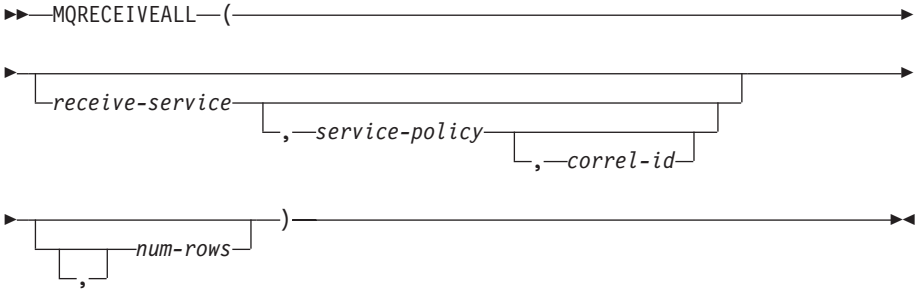
MQREADALL

此操作使用 *service-policy* 中定义的策略从 *receive-service* 指定的 MQSeries 位置在 VARCHAR 变量中返回包含消息和消息元数据的表。此操作不会从队列中除去消息。如果指定了 *num-rows*，则最多返回 *num-rows* 条消息；如果未指定 *num-rows*，则返回所有可用消息。



MQRECEIVEALL

此操作使用 *service-policy* 中定义的策略从 *receive-service* 指定的 MQSeries 位置在 VARCHAR 变量中返回包含消息和消息元数据的表。此操作从队列中除去消息。如果指定了 *correlation-id*，则只返回具有匹配的相关标识的那些消息；如果未指定 *correlation-id*，则返回所有可用消息。如果指定了 *num-rows*，则最多返回 *num-rows* 条消息；如果未指定 *num-rows*，则返回所有可用消息。



可以在 VARCHAR 变量中发送或接收消息。DB2MQ VARCHAR 变量中消息的最大长度是 4000 个字节。DB2MQ1C 变量中消息的最大长度是 VARCHAR 32,000 字节。DB2 MQSeries 表函数结果表的第一列包含消息。

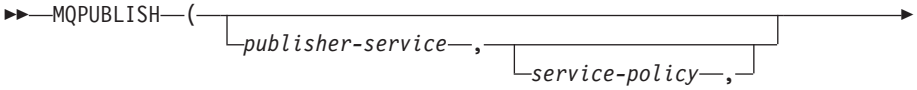
通过将 DB2 标量函数及表函数与具有 Information Integrator 的视图配合使用，可从任何环境中将消息处理操作合并到 SQL 查询中。如果您有 WebSphere MQ 客户机或服务器，则可在 SQL 语句中使用消息传递操作。例如:

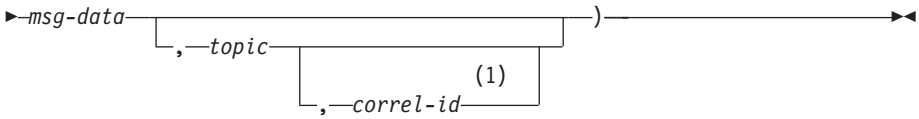
```
SELECT DB2MQ1C.MQSend ('MyAddress'|| firstname ||' '|| lastname)
  FROM employee
```

发布和预订消息使您对哪些服务应接收消息有更多的控制。发布和预订系统提供了可伸缩的安全环境，在该环境中，许多订户可进行注册以接收来自多个发布者的消息。可以在 DB2 通用数据库中使用触发器设施来自动将消息作为触发器调用的一部分发布。

MQPUBLISH

此函数将数据发布至 MQSeries。此函数要求安装 MQSeries Publish/Subscribe 或 MQSeries Integrator。



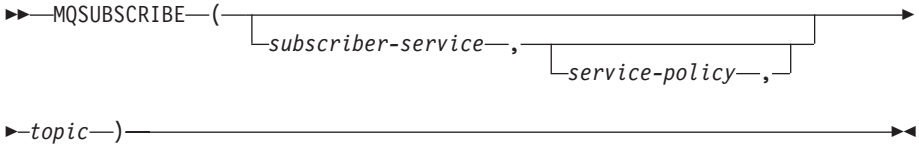


注:

1 除非同时指定了 *service* 和 *policy*，否则不能指定 *correl-id*。

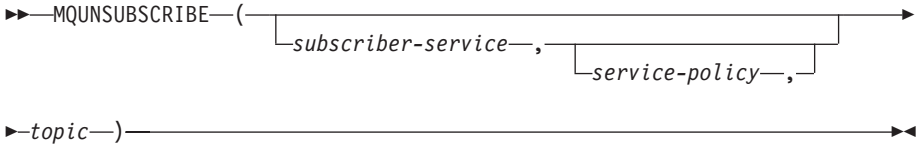
MQSUBSCRIBE

此函数注册对有关指定主题发布的 MQSeries 消息的兴趣。subscriber-service 指定与指定主题匹配的消息的逻辑目标。与主题匹配的消息将被放置在由 subscriber-service 定义的队列上，并且可通过对 MQREAD、MQRECEIVE、MQREADALL 或 MQRECEIVEALL 的后续调用来读取或接收。此函数需要安装和配置基于发布和预订系统（如 MQSeries Integrator 或 MQSeries Publish/Subscribe）的 MQSeries。



MQUNSUBSCRIBE

此函数用于注销现有消息预订。subscriber-service、service-policy 和 topic 用来标识取消哪个预订。此函数需要安装和配置基于发布和预订系统（如 MQSeries Integrator 或 MQSeries Publish/Subscribe）的 MQSeries。



简单数据发布的一个示例是当一个应用程序通知其它应用程序有关感兴趣的事件时。应用程序通过将消息发送至由另一个应用程序监视的队列来执行此操作。消息的内容可以是用户定义的字符串（组合自数据库列）、具有字符串值的函数调用或产生具有正确类型的字符串的任何有效表达式。

如果需要对哪些服务应接收任何特定消息有更多控制，则需要使用发布和预订函数。许多订户可进行注册以接收来自多个发布者的消息。可指定与您的消息相关联的主题。例如，DB2 应用程序可将消息发布至服务点 *Weather*。消息是 *Sleet*，主题是 *Austin*。

```
values DB2MQ1C.MQPublish ('Weather Bulletins','Sleet','Austin')
```

这通知有兴趣的订户 Austin 的天气是雨夹雪。订户使用以下语句注册对接收此类信息的兴趣:

```
values DB2MQ1C.MQSUBSCRIBE('aSubscriber', 'Austin')
```

当订户没有兴趣继续预订特定主题时，该订户必须使用诸如以下的语句显式取消预订:

```
values DB2MQ1C.MQUNSUBSCRIBE('aSubscriber','Austin')
```

DB2 WebSphere MQ 函数的落实环境

在 DB2 通用数据库中，通常将事务称为工作单元。工作单元是应用程序进程中可恢复的操作序列。它由数据库管理器用来确保数据库处于一致状态。从数据库的读取或对数据库的写入都是在工作单元中完成的。当对数据库发出第一条 SQL 语句时开始工作单元。应用程序必须通过发出 COMMIT 或 ROLLBACK 语句来结束工作单元。在您使用 DB2 WebSphere 用户定义的函数时，DB2 通用数据库提供了两个版本的落实:

- 模式名为 DB2MQC 的非事务 UDF
- 模式名为 DB2MQ1C 的单阶段落实

MQ 用户定义的函数的落实环境还依赖于应用程序包括的 CONNECT 的类型。CONNECT 语句在应用程序进程及其服务器之间建立连接。类型 1 CONNECT 支持每个工作单元（远程工作单元）语义一个数据库。类型 2 CONNECT 支持每个工作单元（指导应用程序的分布式工作单元）语义多个数据库。还可以指定类型为 ONEPHASE 的 SYNCPOINT。SYNCPOINT 定义 COMMIT 或 ROLLBACK 如何在多个数据库连接之间协作。借助类型为 ONEPHASE 的 SYNCPOINT，更新只能针对工作单元中的一个数据库进行，而所有其它数据库是只读的。

非事务函数 - 模式 DB2MQ: 如果应用程序使用非事务性用户定义的函数，则任何 DB2 COMMIT 或 ROLLBACK 操作都独立于 WebSphere MQ 操作。如果回滚事务，则 MQ 函数不会废弃在当前工作单元中发送至队列的消息。

在此环境中，WebSphere MQ 控制它自己的队列操作。DB2 COMMIT 或 ROLLBACK 不影响应用程序何时或者是否会添加消息至 WebSphere MQ 队列或从 WebSphere MQ 队列删除消息。

单阶段落实 - 模式 DB2MQ1C: 如果应用程序对数据源使用一阶段落实，并且回滚了事务，则应用程序可能会废弃消息，否则会产生错误。这导致不一致状态。使用 SYNCPOINT=ONEPHASE 进行一阶段落实的规则如下:

- 只允许对一个数据源进行更新
- 消息传递函数不能与其它更新组合

表 27. DB2 MQ 用户定义的函数语义

连接类型	一阶段落实（模式名 = DB2MQ1c）

类型 1（ONEPHASE）	`select db2mq1c.mqsend`
	`(e.LASTNAME || ' ' || d.DEPTNAME)`
	`from EMPLOYEE e,`
	`DEPT d`
	`where e.DEPARTMENT =`
	`d.DEPTNAME`
	
	应用程序可以选择和发送；它只能更新一个数据源

类型 2（TWOPHASE）	不允许消息函数

当 DB2 MQ 函数是 DB2 工作单元的一部分时: 可以使用 DB2 MQ UDF 作为 DB2 工作单元的一部分，或作为多种 DB2 操作中的事务。

多个连接

此项描述两个用户连接至同一个数据库的方案。他们执行 DB2 MQ UDF。一个连接发送消息。另一个连接接收消息。第二个连接在第一个连接落实之前看不到第一个连接的消息。在第一个连接落实后，第二个连接可看到第一个连接的

消息。如果第一个连接发出回滚，则第二个连接就看不到消息。

表 28. 两个用户连接至同一个数据库

连接 1	连接 2
db2 +c // Turn auto commit off	
values db2mq1c.mqsend (″test message″)	

	//The connection can not
	//see the
	//message yet:
	values db2mq1c.mqreceive();

commit;	

	//Now the connection
	//can see the message:
	values db2mq1c.mqreceive();

触发器 DB2 MQ UDF 可以是一个或多个语句 BEFORE 或 AFTER 触发器的一部分。

```
create table EMPLOYEE
  (NAME VARCHAR(30), LASTNAME VARCHAR(30) NOT NULL PRIMARY KEY);

create trigger AFTER_TEST
    after insert on EMPLOYEE
    referencing NEW as NEWEMP
    for each row mode DB2SQL
    VALUES db2mq.mqsend(newemp.lastname);

insert into EMPLOYEE values ('MORGAN', 'TONG');

create trigger BEFORE_TEST
    no cascade before update of NAME on EMPLOYEE
    referencing NEW as NEWNAME OLD as OLDNAME
    for each row mode db2sql
    values db2mq.mqsend (oldname.lastname);

update EMPLOYEE set NAME = 'RAY';

.
```

限制:

可将消息传递技术与大部分 DB2 SQL 语句中的数据库操作进行集成。如果使用 DB2 MQ 函数导致错误，则 DB2 通用数据库会自动回滚该事务。下面是不能使用 DB2 MQ 函数的语句的一些示例:

- 如果用户发出应用程序保存点
- 如果用户尝试从原子复合 SQL 语句中使用 DB2 MQ UDF

相关参考:

- 『CONNECT (Type 1) statement』（*SQL Reference, Volume 2*）
- 『CONNECT (Type 2) statement』（*SQL Reference, Volume 2*）
- 『SET CLIENT Command』（*Command Reference*）
- 『MQSEND scalar function』（*SQL Administrative Routines*）
- 『MQRECEIVE scalar function』（*SQL Administrative Routines*）
- 『MQREAD scalar function』（*SQL Administrative Routines*）

- 『MQPUBLISH scalar function』（*SQL Administrative Routines*）
- 『MQSUBSCRIBE scalar function』（*SQL Administrative Routines*）
- 『MQUNSUBSCRIBE scalar function』（*SQL Administrative Routines*）
- 『MQREADALL table function』（*SQL Administrative Routines*）
- 『MQRECEIVEALL table function』（*SQL Administrative Routines*）

如何在 DB2 中使用 WebSphere MQ 函数

WebSphere® MQ 和 DB2® 消息操作将单个工作单元中的数据库操作作为原子事务进行合并。当所有数据库应用程序连接至相同的 DB2 UDB 服务器，最基本的消息传递格式和 DB2 MQ 函数将出现。客户机对于数据库服务器可以是本地的，或者分布在网络环境中。

在简单的方案中，客户机 A 调用 MQSEND 函数来将用户定义的字符串发送至缺省服务定义的位置。DB2 UDB 执行 MQSeries® 函数，这些函数在数据库服务器上执行此操作。稍后的某个时间，客户机 B 调用 MQRECEIVE 函数。此函数除去队列头中缺省服务定义的消息。然后，函数将消息返回给客户机。DB2 UDB 执行 MQSeries 函数，这些函数在数据库服务器上执行此操作。

数据库客户机可通过许多方法来使用简单消息传递:

- 数据收集

  应用程序以消息的形式从一个或多个源中接收“信息”。信息源可以是任何应用程序。应用程序从队列接收数据并将数据存储在数据库表中以进行附加处理。

- 工作量分配

  应用程序将工作请求公布至同一应用程序的多个实例共享的队列。当应用程序实例准备好执行某些工作时，它就会接收到来自队列头的包含工作请求的消息。应用程序的多个实例可共享单个合用请求队列提供的工作量。

- 应用程序信号发送

  在若干进程协作的情况下，可使用消息来协调它们的各种工作量。这些消息可包含用来执行工作的命令或请求。有关此技术的更多信息，请参阅应用程序间连接。

以下方案将基本消息传递扩展为合并远程消息传递。假设机器 A 将一条消息发送至机器 B。

1. DB2 UDB 客户机执行 MQSEND 函数调用，并指定已定义为机器 B 上的远程队列的目录服务。
2. MQSeries 函数执行该工作以发送消息。机器 A 上的 MQSeries 服务器接受消息。服务器保证将把消息传递至目标。服务和机器 A 的当前配置定义该目标。服务器确定目标是机器 B 上的队列。然后，服务器尝试将消息传递至机器 B 上的 MQSeries 服务器，并根据需要进行重试。
3. 机器 B 上的 MQSeries 服务器接受来自机器 A 上的服务器的消息，并将该消息放置在机器 B 上的目标队列中。
4. 机器 B 上的 MQSeries 客户机请求队列头中的消息。

当使用 MQSEND 时，选择要发送什么数据、将数据发送至何处以及何时发送数据。此类消息传递称为*发送和遗忘*。发送方只发送消息，根据 MQSeries 来确保消息到达其目标。

下列示例将 DB2MQ1C 模式用于一阶段落实，并使用缺省服务 DB2.DEFAULT.SERVICE 和缺省策略 DB2.DEFAULT.POLICY。有关一阶段落实的更多信息，请参阅 WebSphere MQ 和 DB2 应用程序集成概述。所有示例都假设自动落实是关闭的。因此，需要 COMMIT。如果没有 COMMIT，您可能仍保持锁定状态，直到事务结束为止。

示例: 以下 CREATE TRIGGER 语句发送由插入到表 employees 中的名和姓组成的消息:

```
CREATE TRIGGER T1
AFTER INSERT ON employee REFERENCING new AS newemp
    FOR EACH ROW MODE DB2SQL
   VALUES DB2MQ.MQSEND(newemp.name)
```

示例: 假设您有一个 EMPLOYEE 表，该表有 VARCHAR 列 LASTNAME、FIRSTNAME 和 DEPARTMENT。同时假设关闭了自动落实。要发送包含 DEPARTMENT 5LGA 中的每个职员的此信息的消息，发出以下 SQL SELECT 语句:

```
SELECT DB2MQ1C.MQSEND (LASTNAME || ' ' || FIRSTNAME || ' ' || DEPARTMENT)
  FROM EMPLOYEE WHERE DEPARTMENT = '5lGA';
COMMIT;
```

消息内容可以是 SQL 语句、表达式、函数和用户指定数据的任何组合。因为此 MQSEND 函数使用具有一阶段落实语义的 DB2 MQ 事务性用户定义的函数，所以 COMMIT 语句确保将消息添加至 MQSeries 队列。

DB2 MQSeries 函数允许应用程序读取或接收消息。读取和接收之间的差别是读取返回队列头中的消息而不从队列中除去该消息。接收导致从队列中除去消息。使用接收操作检索到的消息只能被检索到一次。使用读取操作检索到的消息允许多次检索同一消息。

下列示例将 DB2MQ1C 模式用于一阶段落实，并使用缺省服务 DB2.DEFAULT.SERVICE 和缺省策略 DB2.DEFAULT.POLICY。有关一阶段落实的更多信息，请参阅 WebSphere MQ 和 DB2 应用程序集成概述。

示例: 以下 SQL SELECT 语句读取由缺省服务和策略指定的队列头中的消息。假设关闭了自动落实。

```
SELECT DB2MQ1C.MQREAD() FROM SYSIBM.SYSDUMMY1;
COMMIT;
```

调用 MQREAD 函数一次，原因是 SYSIBM.SYSDUMMY1 只有一行。SELECT 语句返回 VARCHAR(32000) 字符串。如果没有消息可供读取，则语句的结果是空值。

示例: 下列 SQL SELECT 语句将队列内容具体化为 DB2 UDB 表:

```
SELECT T.* FROM TABLE(DB2MQ1C.MQREADALL()) T;
```

表函数的结果表 T 由队列中的所有消息和有关这些消息的元数据组成。队列由缺省服务定义。具体化结果表的第一列是消息本身，其余列包含元数据。SELECT 语句同时返回消息和元数据。

要只返回消息，发出以下语句:

```
SELECT T.MSG FROM TABLE(DB2MQ1C.MQREADALL()) T;
```

表函数的结果表 T 由队列中的所有消息和有关这些消息的元数据组成。队列由缺省服务定义。此 SELECT 语句只返回消息。

示例: 以下 SQL SELECT 语句尝试从队列发送消息。假设关闭了自动落实。

```
SELECT DB2MQ1C.MQSEND(name) FROM employees e;
ROLLBACK;
```

ROLLBACK 语句意味着实际上未发送消息，原因是消息在 DB2 UDB 操作所在的工作单元中。

示例: 以下 SQL SELECT 语句从缺省服务队列获取所有消息。

```
SELECT t.msg FROM table(DB2MQ1C.MQRECEIVEALL()) t;
COMMIT;
```

表函数的结果表 T 由缺省服务队列中的所有消息和有关这些消息的元数据组成。该 SELECT 语句只返回消息。

相关概念:

- 第 179 页的『WebSphere MQ 和 DB2 应用程序集成概述』

相关任务:

- 第 177 页的『安装 DB2 WebSphere MQ 函数』

应用程序间的连接

通常使用应用程序间的连接来解决将一组不同的应用程序子系统放置在一起的问题。为了方便应用程序集成，MQSeries® 提供了应用程序相互连接的方法。本节描述一个称为*请求和应答*通信的常用方案。

请求和应答方法使一个应用程序能够请求另一个应用程序的服务。请求程序执行此操作的一种方法是发送消息至服务提供程序以请求执行某项工作。当提供程序完成工作后，可以决定是将结果还是只将完成确认发送回请求程序。除非请求程序在继续工作之前等待应答，否则 MQSeries 必须提供将应答与请求关联的方法。

MQSeries 提供相关标识以使请求程序与提供程序之间的交换中的消息相关。请求程序用已知的相关标识标记消息。提供程序用同一相关标识标记其应答。要检索相关联的应答，请求程序在接收到来自队列的消息时提供该相关标识。提供程序将具有匹配的相关标识的第一条消息返回给请求程序。

下列示例将 DB2MQ1C 模式用于单阶段落实。有关单阶段落实的更多信息，请参阅 第 184 页的『DB2 WebSphere MQ 函数的落实环境』。

示例: 以下 SQL SELECT 语句将由字符串 “Msg with corr id” 组成的消息发送至服务 MYSERVICE。它使用具有相关标识 CORRID1 的策略 MYPOLICY:

```
SELECT DB2MQ1C.MQSEND ('MYSERVICE', 'MYPOLICY', 'Msg with corr id', 'CORRID1')
  FROM SYSIBM.SYSDUMMY1;
COMMIT;
```

调用 MQSEND 函数一次，原因是 SYSIBM.SYSDUMMY1 只有一行。由于此 MQSEND 使用 DB2MQ1C 模式（它是一阶段落实 UDF），所以消息是 DB2® 事务的一部分。

示例: 以下 SQL SELECT 语句接收与标识 CORRID1 匹配的第一条消息。它接收服务 MYSERVICE 指定的队列中的消息。它使用策略 MYPOLICY:

```
SELECT DB2MQ1C.MQRECEIVE ('MYSERVICE', 'MYPOLICY', 'CORRID1')
  FROM SYSIBM.SYSDUMMY1;
```

SELECT 语句返回 VARCHAR(32000) 字符串。如果此相关标识没有可用消息，则语句的结果是空值，并且队列不改变。

可以使用 XML Extender 中提供的 WebSphere® MQSeries 用户定义的函数来在 DB2 与各种 WebSphere MQSeries 实现之间只传递 XML 消息。首先，为 XML Extender 启用数据库。接着，用以下方法启用 MQSeries XML Extender 函数:

```
enable_MQXML -n DATABASE -u USER -p PASSWORD
```

下表是某些 MQSeries XML 函数的简要描述。这些函数具有 DB2XML 数据库模式。它们不受 MQ UDF 事务的控制。

表 29. MQSeries XML 函数

MQSeries XML 函数	描述
DB2XML.MQSendXML	将 XML 消息发送至队列。
DB2XML.MQReadXML	非破坏性读取队列中匹配的 XML 消息。
DB2XML.MQReadAllXML	非破坏性读取队列中的所有 XML 消息。
DB2XML.MQReadXMLCLOB	非破坏性读取队列中匹配的 XML CLOB 消息。
DB2XML.MQReadAllXMLCLOB	非破坏性读取队列中所有 XML CLOB 消息。

相关概念:

- 『MQSeries Enablement』（*Application Development Guide: Programming Client Applications*）

相关参考:

- 『MQReadAllXML function』（*DB2 XML Extender Administration and Programming*）
- 『MQReadXMLCLOB function』（*DB2 XML Extender Administration and Programming*）
- 『MQReadAllXMLCLOB function』（*DB2 XML Extender Administration and Programming*）
- 『MQSENDXML function』（*DB2 XML Extender Administration and Programming*）

DB2 Information Integrator 中的异步消息传递

程序可通过在消息中发送数据而不是使用构造（如同步远程过程调用）来互相通信。对于异步消息传递，发送消息的程序在发送消息后将继续它的处理，而不必等待应答。如果程序需要来自应答的信息，则程序将暂挂处理并等待应答消息。如果消息传递程序使用包含消息的中间队列，则请求器程序和接收器程序无需同时运行。请求器程序在队列上放置一则请求消息然后退出。接收器程序从队列中检索请求并处理该请求。

异步操作要求服务提供程序能够从客户机接受请求而不另行通知。异步侦听器是监视消息传输程序（如 WebSphere® MQ）并根据消息类型执行操作的程序。异步侦听器可以使用 WebSphere MQ 来接收被发送至端点的所有消息。异步侦听器还可以向发布或预订基础结构注册预订，以限制只接收那些满足指定约束的消息。

下列示例显示异步消息传递的一些常见用法:

消息累加器

可以累积以异步方式发送的消息，以便侦听器检查消息并自动地将这些消息存储在数据库中。此数据库（它充当消息累加器）可以保存特定端点的所有消息（如审计跟踪）。异步侦听器可以预订消息的子集，如*只保存高额股票交易*。消息累加器存储整个消息，并且不支持对消息内容进行选择、变换或将其映射至数据库结构。消息累加器不对消息作出应答。

消息事件处理程序

异步事件处理程序侦听消息并为消息端点调用适当的处理程序（如存储过程）。可以调用任何存储过程。异步侦听器允许您选择、映射或重新格式化消息内容，以便插入到一个或多个数据库结构中。

下面列示了使用异步消息传递数据库交互作用的一些益处:

- 客户机和数据库不必同时可用。即使客户机只是间歇可用，或者在发出请求与发送响应之间的时间里客户机发生故障，客户机也仍然有可能接收到应答。或者，如果客户机位于移动式计算机上并且已经与数据库断开连接，并且已发送响应，则客户机仍然可以接收到应答。
- 数据库中的消息的内容包含关于何时处理特定请求的信息。数据库中的消息使用优先级和请求内容来确定如何调度请求。
- 异步消息侦听器可将请求委托给另一个节点。它可以将请求转发给第二台计算机以完成处理。当请求完成后，第二台计算机直接将响应返回至消息中指定的端点。
- 异步侦听器可以响应来自提供的客户机或用户定义的应用程序的消息。这大大增加了可作为数据库客户机的环境的数目。客户机（如工厂自动设备、普及型设备或嵌入式控制器）能够直接通过 WebSphere MQ 或通过一些支持 WebSphere MQ 的网关与 DB2® 通用数据库通信。

相关概念:

- 第 190 页的『DB2 Information Integrator 中的 MQListener』

相关任务:

- 第 192 页的『配置和运行 MQListener』

相关参考:

- 『db2mqlsn - MQ Listener Command』（*Command Reference*）
- 第 198 页的『在 MQListener 配置中使用的参数』

DB2 Information Integrator 中的 MQListener

IBM® DB2® Information Integrator 提供一个名为 MQListener 的异步侦听器。MQListener 是一个任务框架，它从 WebSphere® MQ 队列读取，并在消息到达时使用消息调用 DB2 Universal Database™（DB2 通用数据库）存储过程。

MQListener 将消息传递与数据库操作组合到一起。可以将 MQListener 守护进程配置为侦听您在配置数据库中指定的 WebSphere MQ 消息队列。MQListener 读取从队列抵达的消息，然后用消息作为输入参数来调用 DB2 UDB 存储过程。如果消息要求应答，则 MQListener 将根据存储过程生成的输出来创建应答。消息检索顺序固定为首先检索具有最高优先级的消息，当优先级相同时，首先处理最先抵达的消息。

MQListener 是作为单个多线程的进程运行的。每个线程或任务对于输入都与它的已配置的消息队列建立一个连接。每个任务还连接到运行存储过程的 DB2 UDB 数据库。关于队列和存储过程的信息存储在配置数据库中的一个表中。队列与存储过程组合起来是一个任务。

MQListener 任务都组合到命名配置中。缺省情况下，配置名为空的。如果不对任务指定配置的名称，MQListener 将使用具有空名称的配置。

MQListener 可以将消息队列的读和写操作与存储过程一起集成到单个事务中。当运行事务性任务时，即使在从队列中读取消息后计算机发生故障（但是需要在存储过程接收该消息之前），也不会丢失任何消息。缺省情况下，只有对存储过程的调用是事务性的。如果要将从队列中除去消息的操作和调用存储过程的操作组合到同一个事务中，可通过对 db2mqlsn 命令使用 *–mqcoordinated* 参数来将 WebSphere MQ 环境配置为协调程序。必须根据 WebSphere MQ 指导将有关的队列管理器配置为与适当的资源协调。如果不想指定事务性队列操作，则不应将队列管理器配置为事务管理器。不要对配置为事务协调程序的队列管理器运行非事务性任务。

指定配置用户（-configUser）和运行用户（-dbUser）作为 MQListener 配置的一部分。**配置用户配置用户**和**运行用户运行用户**可以是具有不同存取权限的单独用户。运行用户不会继承配置用户的特权。在常见的 MQListener 方案中，用户运行 MQListener 应用程序。运行 MQListener 的用户所需的唯一权限是访问 WebSphere MQ 功能的能力，在 Windows® 和 UNIX® 操作系统中，这通常表示成为 *mqm* 组的成员。执行 MQListener 的用户通常是配置用户。

MQListener 的存储过程接口将入局消息作为输入并返回应答（可能是 NULL）作为输出:

```
schema.proc(in inMsg inMsgType, out outMsg outMsgType)
```

inMsgType 和 *outMsgType* 的数据类型可以是任意长度的 VARCHAR、VARCHAR FOR BIT DATA、CLOB 或 BLOB。输入数据类型和输出数据类型可以是不同的数据类型。

相关概念:

- 第 189 页的『DB2 Information Integrator 中的异步消息传递』

相关任务:

- 第 195 页的『配置 MQListener』
- 第 192 页的『配置和运行 MQListener』

相关参考:

- 『db2mqlsn - MQ Listener Command』（*Command Reference*）
- 第 198 页的『在 MQListener 配置中使用的参数』

配置和运行 MQListener

使用此过程来为 MQListener 配置环境并开发一个简单的应用程序来接收消息、在表中插入消息和创建简单的响应消息。

过程:

要配置和运行 MQListener:

1. 配置 MQListener 以在 DB2 通用数据库环境中运行。
2. 为 MQListener 配置 WebSphere MQ。
3. 配置 MQListener。
4. 创建要与 MQListener 配合使用的存储过程。
5. 运行简单的 MQListener 应用程序。

相关概念:

- 第 190 页的『DB2 Information Integrator 中的 MQListener』

相关任务:

- 第 192 页的『将 MQListener 配置为在 DB2 通用数据库环境中运行』
- 第 193 页的『配置 WebSphere MQ for MQListener』
- 第 195 页的『配置 MQListener』
- 第 196 页的『创建要与 MQListener 配合使用的存储过程』

相关参考:

- 第 196 页的『MQListener 示例』
- 第 198 页的『在 MQListener 配置中使用的参数』
- 第 199 页的『在 MQListener 中使用的 WebSphere MQ 队列』

将 MQListener 配置为在 DB2 通用数据库环境中运行

配置数据库环境，以使应用程序可以将消息传递与数据库操作配合使用。

先决条件:

为 MQListener 配置创建数据库，并且为消息抵达时要调用的存储过程创建数据库（如果还没有有效的数据库可用的话）。可以将同一个数据库用于配置和存储过程。

配置用户必须具有下列特权和权限:

- 对 DB2 UDB 表 SYSMQL.LISTENERS 的读存取权和写存取权。MQListener 运行用户不需要存取 SYSMQL.LISTENERS
- 运行配置程序包 MQLConfi 的权限

运行用户必须具有运行 MQLRun 程序包的权限。

过程:

要将 MQListener 配置为与 DB2 通用数据库数据库配合运行:

1. 发出以下命令来建立连接。使用适合于数据库环境的值来进行替换:

```
db2 connect to ConfigDB user DBAdmin using DBAdminPwd
```

2. 运行 *MQLInstall.sql* 脚本，这将创建存储 MQListener 配置的表。脚本位于以下路径中:
   - 在 UNIX 环境中，是 .../sqllib/bin
   - 在 Windows 环境中，是 ...\sqllib\bin

   ```
   db2 -td; -f MQLInstall.sql
   ```
3. 发出下列命令以将存取权授予配置用户。使用适合于数据库环境的值来进行替换:

   ```
   db2 grant all privileges on table SYSMQL.LISTENERS to ConfigUser
   db2 connect reset
   ```
4. 绑定 MQListener 程序包并授予对程序包的访问权。必须在配置数据库中绑定 MQLConfig 程序包。发出下列命令。使用适合于数据库环境的值来进行替换:

   ```
   db2 connect to ConfigDB user DBAdmin using DBAdminPwd
   db2 bind MQLConfig.bnd
   db2 grant execute on package MQLConfi to ConfigUser
   db2 connect reset
   ```

   名称 MQLConfi 满足程序包名称长度不得超出 8 个字符的限制。
5. 在每个运行数据库中绑定 MQLRun 程序包。对每个运行数据库和该数据库中的每个运行用户发出下列命令:

   ```
   db2 connect to RunDB user DBAdmin using DBAdminPwd
   db2 bind MQLRun.bnd
   db2 grant execute on package MQLRun to RunUser
   db2 connect reset
   ```

相关概念:

- 第 190 页的『DB2 Information Integrator 中的 MQListener』

相关任务:

- 第 193 页的『配置 WebSphere MQ for MQListener』
- 第 195 页的『配置 MQListener』
- 第 196 页的『创建要与 MQListener 配合使用的存储过程』

相关参考:

- 第 198 页的『在 MQListener 配置中使用的参数』
- 第 199 页的『在 MQListener 中使用的 WebSphere MQ 队列』
- 第 196 页的『MQListener 示例』

配置 WebSphere MQ for MQListener

可以使用简单的 WebSphere MQ 配置来运行简单的 MQListener 应用程序。更复杂的应用程序可能需要更复杂的配置。配置至少两种类型的 WebSphere MQ 实体: 队列管理器和一些本地队列。配置这些实体的目的是为了用于事务管理、死信队列、回退重新排队和回退重试阈值之类的场合。

先决条件:

在 *mqm* 组中发出 WebSphere MQ 控制命令。*mqm* 组由 WebSphere MQ 管理员使用并用于内部 MQ 程序。这个组的所有成员都可以访问所有资源。

过程:

要为简单的 MQListener 应用程序配置 WebSphere MQ:

1. 创建队列管理器。

   ```
   crtmqm TransQM
   ```

2. 创建队列管理器。

   ```
   strmqm TransQM
   ```

3. 可选: 将队列管理器配置为与 DB2 通用数据库协调事务。

   如果将队列管理器配置为与 DB2 通用数据库协调事务，则 MQListener 应用程序可以除去消息并在单个事务中调用存储过程。为进行 DB2 UDB 协调而配置队列管理器:

   - 提供共享库（称为切换装入文件）的名称，WebSphere MQ 可使用该名称来查找 DB2 通用数据库 X/Open 资源管理器函数以及特定于 DB2 UDB 的扩展体系结构开放字符串（`xa_open` ）。
   - 通过编译返回 DB2 UDB 全局变量 `db2xa_switch` 的小 C 程序，在切换装入文件中创建 MQStart 例程。有关如何创建切换装入文件的特定信息，请参阅 *WebSphere MQ: System Administration Guide*。MQStart 返回一个指针结构，这些指针指向在 DB2 UDB 中实现 X/Open 资源管理器功能的函数。以下示例显示了扩展体系结构开放字符串的必需格式，并用适当的值替换了 MQListener 配置参数:

     ```
     DB=RunDB, UID=RunUser, PWD=RunUserPwd, TPM=MQ, TOC=P
     ```

     如果按照此示例那样在扩展体系结构开放字符串中使用 TPM=MQ，则不需要设置 DB2 TP_MON_NAME 实例变量。

   WebSphere MQ 根据操作系统环境获取它需要的参数。在 Windows 操作系统上，这些参数位于 Windows 注册表中。可以使用 WebSphere MQ MQServices 来指定参数。在 UNIX 操作系统上，这些参数位于队列管理器配置文件中。使用所在环境的有效文本编辑器来指定要使用的参数。

4. 使用 WebSphere MQ 脚本设施来创建本地队列。

   a. 创建包含下列命令的文件（对于本示例，文件是 mqconfig.mqs）:

      ```
      define qlocal('DLQ')
      alter qmgr deadq('DLQ')
      define qlocal('Backout')
      define qlocal('Admin')
      define qlocal('In') boqname('Backout') bothresh(3)
      define qlocal('SYSTEM.SAMPLE.REPLY')
      ```

   b. 通过发出以下命令将 mqconfig.mqs 文件重定向到脚本解释器:

      ```
      runmqsc TransQM < mqconfig.mqs
      ```

相关任务:

- 第 195 页的『配置 MQListener』

相关参考:

- 第 198 页的『在 MQListener 配置中使用的参数』
- 第 199 页的『在 MQListener 中使用的 WebSphere MQ 队列』

配置 MQListener

使用 MQListener 命令 **db2mqlsn** 来配置 MQListener。在任何目录中从命令行发出命令 **db2mqlsn**。在 Windows 系统上，在 DB2 UDB 命令行处理器窗口中发出命令以确保正确的消息显示。伴随 db2mqlsn 命令的 add 参数将更新 DB2 表 SYSMQL.LISTENERS 中的某一行。

限制:

- 将同一个队列管理器用于请求队列和应答队列。
- 在 Windows 系统上，每个线程都可以连接至一个队列管理器。
- 在 UNIX 系统上，每个过程都可以连接至一个队列管理器。在 UNIX 系统上，如果在同一个 MQListener 配置中指定了不同的队列管理器，则会从 WebSphere MQ 接收到运行时错误。
- MQListener 不支持由多个物理消息组成的逻辑消息。MQListener 独立地处理物理消息。

过程:

要指定 MQListener 配置:

- 要添加 MQListener 配置，发出以下命令:

```
db2mqlsn add
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -queueManager TransQM
  -inputQueue In
  -procSchema RunUser
  -procName aProc
  -dbName RunDB
  -dbUser RunUser
  -dbPwd RunUserPwd
  -mqCoordinated
```

- 要显示配置中的所有任务，发出以下命令:

```
db2mqlsn show
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
```

- 要除去消息传递任务，发出以下命令:

```
db2mqlsn remove
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -queueManager TransQM
  -inputQueue In
```

- 要获取有关命令和有效参数的帮助，发出以下命令:

```
db2mqlsn help
```

- 要获取有关特定参数的帮助，发出带有特定参数的命令，如以下示例所示:

```
db2mqlsn help <command>
```

相关参考:

- 『db2mqlsn - MQ Listener Command』（*Command Reference*）

- 第 198 页的『在 MQListener 配置中使用的参数』

创建要与 MQListener 配合使用的存储过程

运行数据库包含在消息抵达时要运行的存储过程。运行用户就是 MQListener 以其名义连接至运行数据库以运行存储过程的用户。将下列参数与 db2mqlsn add 命令配合使用，以定义运行数据库和运行用户:

- -dbName
- -dbUser

运行用户必须能够连接至运行数据库并运行存储过程。运行用户不需要是存储过程的所有者。运行用户不需要访问 MQListener 配置。

MQListener 使用存储过程 aProc 来在表中存储消息。如果成功地将消息插入到表中，则存储过程返回字符串 OK。

先决条件:

存储过程需要 C 编译器。

过程:

通过以下步骤可创建 DB2 通用数据库对象，这些对象可与 MQListener 应用程序配合使用。

1. 创建一个简单的表来作为运行用户（可以使用 DB2 UDB 命令行处理器）:

   ```
   CREATE TABLE aTable (val VARCHAR(25) CHECK (val NOT LIKE 'fail%'))
   ```

   该表包含检查约束，以便不能将以字符 fail 开头的消息插入表中。该检查约束用于演示当存储过程失败时 MQListener 的行为。

2. 创建以下存储过程:

   ```
   CREATE PROCEDURE aProc (IN pin VARCHAR(25), OUT pout VARCHAR(2))
   BEGIN
            INSERT INTO aTable VALUES(pin);
            SET pout = 'OK';
   END
   ```

相关任务:

- 第 192 页的『将 MQListener 配置为在 DB2 通用数据库环境中运行』

相关参考:

- 第 198 页的『在 MQListener 配置中使用的参数』

MQListener 示例

下列示例显示了一个简单的 MQListener 应用程序。该应用程序接收消息、在表中插入消息并生成简单的响应消息。为了模拟如何处理故障，该应用程序包含了对包含消息的表的检查约束。该约束可防止将任何以字符 fail 开头的字符串插入表中。如果尝试插入违反检查约束的消息，示例应用程序将返回错误消息将失败的消息重新排队到回退队列中。

要使用配置中指定的所有任务来运行 MQListener，可发出以下命令:

```
db2mqlsn run
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -adminQueue Admin
  -adminQMgr TransQM
```

下列示例显示如何使用 MQListener 来发送简单的消息并接着在 WebSphere MQ 队列管理器和数据库中检查消息的结果。这些示例中包括了查询，用于确定输入队列是否包含消息或者存储过程是否已将记录放入表中。有许多工具支持这些操作，包括 DB2 通用数据库命令行处理器、DB2 UDB 命令中心、某些 WebSphere MQ 命令行实用程序、样本程序、MQ 资源管理器和 MQ API 试验器。对于更复杂的应用程序，请考虑使用 DB2 通用数据库和 WebSphere MQ 工具。

MQListener 示例 1：运行简单的应用程序：

1. 从干净的数据库表入手：

   ```
   db2 delete from aTable
   ```

2. 向输入队列发送数据报：

   a. 将字符串 `a sample message` 放到名为 sampleMsg1.txt 的文件中

   b. 使用 WebSphere MQ 样本程序 **amqsput** 将消息放到队列中：

      ```
      amqsput In TransQM < sampleMsg1.txt
      ```

3. 查询表以验证是否已插入样本消息：

   ```
   db2 select * from aTable
   ```

4. 显示仍位于输入队列中的消息的数目以验证是否已除去了该消息：

   a. 将以下命令放到文件 checkIn.mqs 中：

      ```
      display queue('In') curdepth
      ```

   b. 将该命令重定向到脚本解释器：

      ```
      runmqsc TransQM < checkIn.mqs
      ```

MQListener 示例 2：将请求发送至输入队列并检查应答：

下列示例语句将请求发送至输入队列并检查应答：

1. 从干净的数据库表入手：

   ```
   db2 delete from aTable
   ```

2. 向输入队列发送请求：

   a. 将字符串 `another sample message` 放到名为 sampleMsg2.txt 的文件中

   b. 使用 WebSphere MQ 样本程序 **amqsreq** 将请求发送至输入队列：

      ```
      amqsreq In TransQM < sampleMsg2.txt
      ```

      amqsreq 程序将请求中的回复队列设置为 SYSTEM.SAMPLE.REPLY

3. 查询表以验证是否已插入样本消息：

   ```
   db2 select * from aTable
   ```

4. 显示仍位于输入队列中的消息的数目以验证是否已除去了该消息。

   ```
   display queue('In') curdepth
   ```

5. 使用 WebSphere MQ 样本程序 amqsget 在 SYSTEM.SAMPLE.REPLY 队列中查找应答。验证存储过程是否生成了 OK 字符串：

```
amqsget SYSTEM.SAMPLE.REPLY TransQM
```

MQListener 示例 3: 测试不成功的插入操作:

如果发送以字符串 `fail` 开头的消息，将违反表定义中的约束，存储过程将失败。

1. 从干净的数据库表入手:
   ```
   db2 delete from aTable
   ```
2. 向输入队列发送请求:
   a. 将字符串 `failing sample message` 放到名为 sampleMsg3.txt 的文件中
   b. 使用 WebSphere MQ 样本程序 **amqsreq** 将请求发送至输入队列:
      ```
      amqsreq In TransQM < sampleMsg3.txt
      ```
      amqsreq 程序将请求中的回复队列设置为 SYSTEM.SAMPLE.REPLY
3. 查询表以验证是否没有插入样本消息:
   ```
   db2 select * from aTable
   ```
4. 显示仍位于输入队列中的消息的数目以验证是否已除去了该消息:
   ```
   display queue('In') curdepth
   ```
5. 读取 SYSTEM.SAMPLE.REPLY 队列并查找异常报告而不是 OK 应答:
   ```
   amqsget SYSTEM.SAMPLE.REPLY  TransQM
   ```
6. 读取“回退”队列并查找原始消息:
   ```
   amqsget Backout TransQM
   ```

相关概念:

- 第 190 页的『DB2 Information Integrator 中的 MQListener』

相关任务:

- 第 193 页的『配置 WebSphere MQ for MQListener』
- 第 195 页的『配置 MQListener』
- 第 196 页的『创建要与 MQListener 配合使用的存储过程』
- 第 192 页的『将 MQListener 配置为在 DB2 通用数据库环境中运行』
- 第 192 页的『配置和运行 MQListener』

相关参考:

- 第 198 页的『在 MQListener 配置中使用的参数』
- 第 199 页的『在 MQListener 中使用的 WebSphere MQ 队列』

在 MQListener 配置中使用的参数

ConfigDB

配置数据库（可以是任何有效的 DB2 通用数据库）包含 MQListener 配置表。配置表包含关于 MQListener 应侦听的队列的信息以及 MQListener 应调用的存储过程。

ConfigUser

可以以其名义存取配置数据库的用户标识。配置用户无需是数据库管理员。可以在 MQListener 命令中指定配置用户和密码。如果不指定配置用户和密码，并且数据库安装支持隐式连接，则缺省情况下配置用户就是 MQListener 正在其帐户下运行的用户。

ConfigUserPwd
与配置用户标识配合使用的密码。

RunDB
运行数据库是包含当消息抵达时要运行的存储过程的数据库。存储过程可以位于与配置数据库不同的数据库中。

RunUser
以其名义存取运行数据库以运行存储过程的用户。除连接至运行数据库以及运行存储过程的能力之外，运行用户不需要任何特权。

RunUserPwd
与运行用户相关联的密码。

相关概念:

- 第 190 页的『DB2 Information Integrator 中的 MQListener』

相关参考:

- 『db2mqlsn - MQ Listener Command』（*Command Reference*）

在 MQListener 中使用的 WebSphere MQ 队列

在简单的 MQListener 应用程序中，通常使用下列 WebSphere MQ 队列:

死信队列
WebSphere MQ 中的死信队列（DLQ）存放不能处理的消息。MQListener 使用此队列来存放不能传递的应答，例如，由于应该向其发送应答的队列已满而不能传递的应答。死信队列在任何 MQ 安装中都非常有用，尤其对于恢复未发送的消息而言更是如此。

回退队列
对于 WebSphere MQ 在其中作为事务协调程序的 MQListener 任务，“回退”队列的用途与死信队列相似。在将请求回滚指定次数（称为回退阈值）之后，MQListener 将原始请求放到“回退”队列中。

管理队列
管理队列用于将控制消息（如 *shutdown* 和 *restart*）传递至 MQListener。如果没有提供管理队列，则关闭 MQListener 的唯一方法是发出 **kill** 命令。

应用程序输入和输出队列
应用程序使用输入队列和输出队列。应用程序从输入队列接收消息。应用程序向输出队列发送应答和异常，该队列为 SYSTEM.SAMPLE.REPLY，以便与 WebSphere MQ 样本程序 amqsreq 中的用法相符。

相关任务:

- 第 193 页的『配置 WebSphere MQ for MQListener』
- 第 192 页的『配置和运行 MQListener』

附录 A. Cottonwood Distributors, Inc. 和 YBar, Inc. 方案的脚本示例

```
-------------------------------------------------
-- Catalog remote DB2 UDB server machine and database
-------------------------------------------------
uncatalog node DB2_TPCH;
catalog tcpip node DB2_TPCH remote x.xx.xx.xx server 50000;

uncatalog database tpcd;
catalog database tpcd at node db2_tpch;

-------------------------------------------------
-- DB2 UDB wrapper, nicknames, MQTs, and indexes
-------------------------------------------------
drop wrapper drda;
create wrapper drda;

create server db2_tpch type db2/udb version  8.1
  wrapper drda authorization "demo" password "xxxxx"
  options (dbname 'TPCD');
create user mapping
  for user SERVER db2_tpch
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname db2_part for db2_tpch.tpcd.part;
create nickname db2_supplier for db2_tpch.tpcd.supplier;
create nickname db2_partsupp for db2_tpch.tpcd.partsupp;
create nickname db2_nation for db2_tpch.tpcd.nation;
create nickname db2_region for db2_tpch.tpcd.region;
create nickname db2_customer for db2_tpch.tpcd.customer;
create nickname db2_orders for db2_tpch.tpcd.orders;
```

图 72. federated.sql (1/9)

```
-------------------------------------------------
-- Oracle wrapper, nicknames, MQTs and indexes
-------------------------------------------------
drop wrapper net8;
create wrapper net8;

create server oraserver type oracle version 8
  wrapper net8
  options (node 'iidemo2');
create user mapping
  for user SERVER oraserver
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname ora_part for oraserver.demo.part;
create nickname ora_supplier for oraserver.demo.supplier;
create nickname ora_partsupp for oraserver.demo.partsupp;
create nickname ora_customer for oraserver.demo.customer;
create nickname ora_orders for oraserver.demo.orders;
create nickname ora_lineitem for oraserver.demo.lineitem;
```

图 72. federated.sql (2/9)

```
-------------------------------------------------
-- Informix wrapper, nicknames, MQTs, and indexes
-------------------------------------------------
drop wrapper informix;
create wrapper informix;

create server infserver type informix version 9
  wrapper informix
  options (node 'ol_informix',dbname 'tpcd2');
create user mapping
  for user SERVER infserver
  OPTIONS ( REMOTE_AUTHID 'informix', REMOTE_PASSWORD 'informix' );

create nickname inf_part for infserver."informix"."part";
create nickname inf_supplier for infserver."informix"."supplier";
create nickname inf_partsupp for infserver."informix"."partsupp";
create nickname inf_customer for infserver."informix"."customer";
create nickname inf_orders   for infserver."informix"."orders";
create nickname inf_lineitem for infserver."informix"."lineitem";
```

图 72. federated.sql (3/9)

```
-------------------------------------------------
-- Union views over federated nicknames
-------------------------------------------------
DROP VIEW part_fed;
CREATE  VIEW part_fed (
  p_partkey, p_mfgr, p_type, p_size, p_retailprice) AS
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM db2_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM inf_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM ora_part;

DROP VIEW partsupp_fed;
CREATE VIEW partsupp_fed (
   ps_partkey, ps_suppkey, ps_supplycost) AS
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  db2_partsupp
UNION ALL
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  inf_partsupp
UNION ALL
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  ora_partsupp;

DROP VIEW supplier_fed;
CREATE VIEW supplier_fed (
   s_suppkey, s_name, s_address ) AS
   SELECT s_suppkey, s_name, s_address
   FROM  db2_supplier
UNION ALL
   SELECT s_suppkey, s_name, s_address
   FROM inf_supplier
UNION ALL
   SELECT s_suppkey, s_name, s_address
   FROM ora_supplier;
```

图 72. federated.sql (4/9)

```
-------------------------------------------------
-- Create Wrapper, Server and nickname for XML
-------------------------------------------------

drop wrapper xml_files;

create wrapper XML_files library 'db2lsxml.dll';
create server LOCAL_XML_FILES wrapper XML_FILES;
create nickname Employees_From_XML (
  doc Varchar(100) OPTIONS(DOCUMENT 'FILE'),
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (XPATH '/Employee');
```

图 72. federated.sql (5/9)

```
create nickname Employees_From_XML_Included (
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (FILE_PATH 'c:\cdi_data_files\CDI_Employees.xml', XPATH '//Employee');
```

图 72. federated.sql (6/9)

```
-------------------------------------------------
-- Create read functions and views over MQ queues for CR
-------------------------------------------------

CREATE FUNCTION NEW_SUPPLIERS_READ()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                          SUPPLIER_PHONE VARCHAR(12),
                          PART_KEY DOUBLE,
                          PART_PRICE DOUBLE,
                          MAN_DAYS DOUBLE,
                          MAX_QUANTITY DOUBLE,
     CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQREADALL('DB2.DEFAULT.SERVICE',
      'DB2.DEFAULT.POLICY')) AS T;

create view READ_NEW_SUPPLIERS_FROM_QUEUE
  as select * from table(new_suppliers_read()) t
  where CORRELID = 'CDI_NEW_SUPPLIER';
```

图 72. federated.sql (7/9)

```
-------------------------------------------------
-- Create destroy functions and views over MQ queues for CR
-------------------------------------------------
CREATE FUNCTION NEW_SUPPLIERS_REC()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                           SUPPLIER_PHONE VARCHAR(12),
                           PART_KEY DOUBLE,
                           PART_PRICE DOUBLE,
                           MAN_DAYS DOUBLE,
                           MAX_QUANTITY DOUBLE,
     CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQRECEIVEALL('DB2.DEFAULT.SERVICE',
  'DB2.DEFAULT.POLICY', 'CDI_NEW_SUPPLIER', 1)) AS T;

create view RECEIVE_NEW_SUPPLIERS_FROM_QUEUE
  as select * from table(new_suppliers_rec()) t;
```

图 72. federated.sql (8/9)

```
-------------------------------------------------
-- Create temporary tables used in processing
-------------------------------------------------

-- For running custom java programs
drop table aux_table;
create table aux_table (part_key integer,
  supplier_key int, supply_cost double);

-- For running XML composition
drop table UCustomers;
create table Ucustomers (x_doc DB2XML.XMLCLOB not logged);

-- For storing the web request
drop table request_bid;
drop table request_status;
create table request_bid (reqkey integer not null,
  partkey integer not null, bid double not null);
create table request_status (reqkey integer not null,
  partkey integer not null, suppkey integer not null,
  newquote double not null,currentquote double not null,
  status varchar(15) not null);

-- For shredding XML documents to DB2
IMPORT FROM c:\CDI_Data_Files\Setup\CDI_Employees.ixf of
  IXF CREATE INTO EMPLOYEES_DB2;
CREATE TABLE EMPLOYEES_FROM_XML_FILE_SHRED LIKE EMPLOYEES_DB2;
```

图 72. federated.sql (9/9)

```
// Import all necessary classes
import java.lang.*;
import java.sql.*;
import java.util.*;

// USAGE: db, user, password, timeout

/**
 * Class Listener:
 * Class which will check a message queue for messages and
 *    call the "Director" stored procedure
 * to process the message.
 */
public class CDIListener{

 /**
  * Method checkQueue
  * Method to check the queue for messages.
    * If present calls the "Director" stored procedure
  *  otherwise it will wait for a user specified number of seconds
  */
```

图 73. CDIListener.java (1/12)

```
public void checkQueue(String db, String user,
          String pass, int timeout) {

  /* Local variables */
  String urlDB2 = null;
  Connection connDB2 = null;
  PreparedStatement stmtDB2 = null;
  PreparedStatement stmtDB2_2 = null;
  ResultSet rsDB2 = null;
  ResultSet rsDB2_2 = null;
  ResultSet rsTotal = null;
  String query = null;

  int mqMsgType  = 0;
  int mqMsgKey   = 0;
  int mqMsgPart  = 0;
  int mqMsgQuant = 0;
  double mqMsgPrice = 0;

  String comment = null;
  int intValue = 0;
  int numRecords = 0;

  System.out.println("\nStarting listener execution");
  System.out.println("\nConnecting to DB2");
```

图 73. CDIListener.java (2/12)

```
// Print out parms
System.out.println("\nUsing startup parms of: \n Database: " +
     db + "\n User: " + user + "\n Password: **********\n
     Sleep interval: " + timeout + " milliseconds");

try {
 //Load drivers' classes
 System.out.println("\nLoading DB2 driver");
 Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

 //URL for databases to be connected
 System.out.println("\nSetting URL to database");
 urlDB2 = "jdbc:db2:" + db;

 //Get database connections
 System.out.println("\nGetting DB2 connection");
 connDB2 = DriverManager.getConnection(urlDB2, user, pass);
} catch (Exception e) {
 e.printStackTrace();
}
```

图 73. CDIListener.java (3/12)

```
/*Issue a receive on the MQ queue*/
System.out.print("\nStart MQ queue processing...");
System.out.flush();

double test = 0;

/* Loop forever to read the queue */
while (test == 0) {
 // System.out.println("\nChecking queue...");
 try {
  stmtDB2 = connDB2.prepareStatement
            ("SELECT VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),1),
              INT(DB2MQ.GETCOL(T.MSG,',',2)),
              INT(DB2MQ.GETCOL(T.MSG,',',3)),
              INT(DB2MQ.GETCOL(T.MSG,',',4)),
              DOUBLE(DB2MQ.GETCOL(T.MSG,',',5))
              FROM TABLE (DB2MQ.MQRECEIVEALL('DB2.DEFAULT.SERVICE',
              'DB2.DEFAULT.POLICY','CDI_IN_MSG',1)) AS T");
      rsDB2 = stmtDB2.executeQuery();
 }
 catch (SQLException e) {
  if (e.getErrorCode() == 100) {
   test = 0;
  }
  else {
   System.out.println(e);
   test = 1;
  }
 }
 catch (Exception e) {
  e.printStackTrace();
  test = 1;
 }
```

图 73. CDIListener.java (4/12)

```
try {
  // Get a message off the queue
  if (rsDB2 != null)
  while(rsDB2. next()) {
  // Get the type from the MQ message and
           //   call appropriate function
  mqMsgType  = rsDB2.getInt(1);

  // if END message, exit
  if (mqMsgType == 0) {
    System.out.println("\nFound END message on queue");
    test = 1;
  }
  else {
   // Found a customer buy request message
   if (mqMsgType == 1) {
    // get other values from message
    mqMsgKey   = rsDB2.getInt(2);
    mqMsgPart  = rsDB2.getInt(3);
    System.out.println("\nFound message type="+ mqMsgType + ";
                        customer="+ mqMsgKey + "; part=" + mqMsgPart);

    // Update the REQUEST_BID table
    System.out.print("Executing update to
                   local REQUEST_BID table...");
    // put customer order into REQUEST_BID table
                //    using max part price + 45% markup
    stmtDB2_2 = connDB2.prepareStatement
                    ("INSERT into request_bid
                      values (" + mqMsgKey + "," + mqMsgPart + "," + "
                      (SELECT MIN(ps_supplycost)*1.45
                        FROM partsupp_fed
                        WHERE ps_partkey = " + mqMsgPart + "))");
         stmtDB2_2.executeUpdate();
    System.out.println("\nComplete");
    stmtDB2_2.close();
   }
```

图 73. CDIListener.java (5/12)

```
   else {
    // Else if it's a supplier price update
    if (mqMsgType == 2) {
     // Get the other values off the queue
     mqMsgKey   = rsDB2.getInt(2);
     mqMsgPart  = rsDB2.getInt(3);
     mqMsgQuant = rsDB2.getInt(4);
     mqMsgPrice = rsDB2.getDouble(5);
     System.out.println("\nFound message
                         type="+ mqMsgType + ";
                         supplier="+ mqMsgKey + ";
                         part=" + mqMsgPart + ";
                         price=" + mqMsgPrice);
```

图 73. CDIListener.java (6/12)

```
// Check if this supplier has supplier this part before
System.out.print("\nChecking if supplier
                        already supplies part...");
stmtDB2 = connDB2.prepareStatement
                       ("SELECT COUNT(*)
                         FROM partsupp_fed
                         WHERE ps_partkey = " + mqMsgPart + "
                         and ps_suppkey = " + mqMsgKey);
  rsDB2_2 = stmtDB2.executeQuery();
  rsDB2_2.next();
  intValue = rsDB2_2.getInt(1);
System.out.print(intValue + "...");
```

图 73. CDIListener.java (7/12)

```
  // If supplier has supplier before
  if (intValue > 0) {

   // Get current price by supplier
 System.out.print("Yes!\nGetting current
                         minimum price...");
   stmtDB2 = connDB2.prepareStatement
                    ("SELECT MIN(ps_supplycost)
                      FROM partsupp_fed
                      WHERE ps_partkey = " + mqMsgPart + "
                      and ps_suppkey = " + mqMsgKey);
   rsDB2_2 = stmtDB2.executeQuery();
   rsDB2_2.next();
   intValue = rsDB2_2.getInt(1);

   System.out.println("Current price = " + intValue + "\n");
```

图 73. CDIListener.java (8/12)

```
   //If new price less than or equal to
               //existing price, update database
               //and mark as accepted
   if (mqMsgPrice <= intValue) {
  comment = "'ACCEPT'";
   }
   // Else update the database but mark as review
   else {
  comment = "'REVIEW'";
   }
 System.out.print("\nExecuting " + comment + "
                      update to federated db2 table db2_partsupp...");
   stmtDB2 = connDB2.prepareStatement
                   ("UPDATE db2_partsupp
                     set ps_availqty = " + mqMsgQuant + ",
                     ps_supplycost = " + mqMsgPrice + "
                     where ps_partkey = " +  mqMsgPart + "
                     and ps_suppkey = " + mqMsgKey);
   stmtDB2.executeUpdate();
 System.out.println("Complete\n");
  }
```

图 73. CDIListener.java (9/12)

```
// Else this is a new supplier for this part
else {
 System.out.print("No!\nExecuting 'NEW' insert to
                    federated db2 table db2_partsupp...");
 comment = "'NEW'";

 // Add new record to database
 stmtDB2 = connDB2.prepareStatement
                   ("INSERT into db2_partsupp
                       values (" + mqMsgPart + "," +
                       mqMsgKey + ","+ mqMsgQuant + "," +
                       mqMsgPrice + ",'New supplier added at: " +
                       new java.util.Date()+ " ')");
 stmtDB2.executeUpdate();
 System.out.println("Complete\n");
}
// Update the local table
System.out.print("\nUpdating local REQUEST_STATUS table...");
  stmtDB2_2 = connDB2.prepareStatement
                 ("INSERT into request_status
                     values (" + mqMsgKey + "," +
                     mqMsgPart + "," +mqMsgKey + ","+
                     mqMsgPrice + "," +
                     intValue + "," + comment + ")");
  stmtDB2_2.executeUpdate();
System.out.println("Complete\n");
```

图 73. CDIListener.java (10/12)

```
     if (stmtDB2_2 != null)
      stmtDB2_2.close();
     if (rsDB2_2 != null)
      rsDB2_2.close();
    }
    else {
     System.out.println("\nError - unknown message type");
    }
   }
  }
 System.out.flush();
 } // End secondary while
 System.out.flush();
} // End try

catch (Exception e) {
 e.printStackTrace();
}

// Sleep for the necessary time
try {
 if (rsDB2 != null)
  rsDB2.close();
 if (stmtDB2 != null)
  stmtDB2.close();
 Thread.sleep(timeout);
 System.out.print(".");
 System.out.flush();
}
catch (Exception e) {
 e.printStackTrace();
}
} // end Main while
```

图 73. CDIListener.java (11/12)

```
 System.out.println("Listener stopped\n");
 System.out.println("\nListener stopped\n");

 /* Close all database connections */
 try {
  rsDB2.close();
  stmtDB2.close();
  connDB2.close();
 } catch (Exception e) {
  e.printStackTrace();
 }

 return;

}

/* Main program to invoke listener */
public static void main(String argv[]) {

 String DBName = argv[0];
 String DBUser = argv[1];
 String DBPass = argv[2];
 String TimeCk = argv[3];

 CDIListener dp = new CDIListener();
 int timeout = Integer.parseInt(TimeCk)*1000;
 dp.checkQueue(DBName, DBUser, DBPass, timeout);
}
}
```

图 73. CDIListener.java (12/12)

```
/*
 * @(#)MessageFormatter.java
 *
 * CopyrightVersion 1.0
 *
 */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.ServletException;
import proxy.soap.*;

public class MessageFormatter extends javax.servlet.http.HttpServlet {

 final static String htmlHeader1 = "<HTML><TITLE>MessageFormatter</TITLE>";
final static String message1 =
 "<p align=\"center\">
  <font color=\"navy\"
  face=\"verdana\" size=\"+2\">
<b>Thank You</b></font></p>
<p><font color=\"black\"
face=\"veranda\" size=\"+1\">";
```

图 74. MessageFormatter.java (1/7)

```
final static String message2 = "";

 final static String htmlHeader2 = "</HTML>";
 static String quote = "";

 static Connection con = null;
 final static String url = "jdbc:db2:demo";

  // method to generate a random REQUEST number
 public int randomint() {
  double first = java.lang.Math.random();
  String help = java.lang.Double.toString(first);
  help = help.substring(3,7);
  int number = Integer.parseInt(help);
  number = number / 5;
  return number;
 }
```

图 74. MessageFormatter.java (2/7)

```
// Overwriting doGet method to handle Http GET request

 public void doGet(HttpServletRequest req,
         HttpServletResponse res)
  throws javax.servlet.ServletException, java.io.IOException {
  try {


   PrintWriter pr = res.getWriter();
   pr.print(htmlHeader1);
   pr.print(message1);
   pr.print(message2);
   String method = "";
   String part = "";
   String price = "";
   String quantity = "";
   String key = "";
   String cust_name = "";
   String col_name1 = null;
   String col_name2 = null;
   String tab_name  = null;

   Connection con = null;
   String url = null;
   Statement stmt = null;
   ResultSet rs = null;

   WSProxy WSid = new WSProxy();
   boolean fCustomer = true;
```

图 74. MessageFormatter.java (3/7)

```
if (req.getParameter("method")!=null) {
 // Get the common parms to the servlet from the REQ object
 key = req.getParameter("name");
 part = req.getParameter("part");
 method = req.getParameter("method");

 // If customer order
 if (method.equals("orderNewParts")) {
  fCustomer = true;
  // Get the quantity the customer is ordering
  quantity = req.getParameter("quantity");

  // set the table and column names for SELECT
  col_name1 = "c_name";
  col_name2 = "c_custkey";
  tab_name  = "db2_customer";

 }
 else {
  // Else if supplier update
  if (method.equals("setSupplierQuotes")) {
   fCustomer = false;
   // Get the price the supplier is updating
   price = req.getParameter("price");

   // set the table and column names for SELECT
   col_name1 = "s_name";
   col_name2 = "s_suppkey";
   tab_name  = "supplier_fed";
  }
  else {
   pr.println
               ("method = "+req.getParameter("method")+
                    " Not supported! <br>");
   return;
  }
 }
```

图 74. MessageFormatter.java (4/7)

```
 // Get the real customer name from federated data source
 try {
  Class.forName
            ("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
  url = "jdbc:db2:demo";
  con = DriverManager.getConnection(url,"demo","xxxxx");
  stmt = con.createStatement();
  rs = stmt.executeQuery
            ("SELECT " + col_name1 + " from " +
              tab_name + " where " + col_name2 + " = " + key);
  while (rs.next()) {
   cust_name = rs.getString(1);
  }

  rs.close();
  stmt.close();
  con.close();
 }
 catch (Exception e) {
  pr.println(e);
  System.out.println(e);
  return;
 }


}
else {
 pr.println
          ("*** Severe error occurred-no method passed ***>");
 return;
}
```

图 74. MessageFormatter.java (5/7)

```
   // Write request status back to user
   try{
    // If this is a supplier update request
    if (!fCustomer) {
      String message1id=  "2,"
               + key + "," + part + ","
               + quantity + "," + price ;
            java.lang.String message1idTemp  = message1id;
     System.out.println
                ("message type 2 being written: "
                   + message1idTemp);
            org.tempuri.worftestweb
                .demo.newdadx.dadx.xsd
                .Stmt1ResultElement
                mtemp = WSid.stmt1(message1idTemp);
            if (mtemp == null)
      pr.println("*** Web Service error occurred ***>");
     else {
      pr.println
("<table border=\"1\"
bordercolor=\"navy\"
width=\"100%\">
<tr align+\"left\">
<td>Supplier</td>
<td>" + cust_name + "</td>
</tr>
<tr align+\"left\"><td>Part</td><td>" + part + "</td></tr>
<tr align+\"left\"><td>Price</td><td>" + price + "</td></tr>
</table>");
pr.println("<br><font color=\"red\" face=\"verdana\"
size=\"-1\">
Price update submitted for processing");
     }
    }
```

图 74. MessageFormatter.java (6/7)

```
    // else must be a customer buy request
    else {
     String message1id=  "1," + key +"," + part + ",0, 0";
           String message1idTemp  = message1id;
     System.out.println("message type 1 being written: " + message1idTemp);
           org.tempuri.worftestweb.demo
                    .newdadx.dadx.xsd
                    .Stmt1ResultElement
                    mtemp = WSid.stmt1(message1idTemp);
           if (mtemp == null)
      pr.println("*** Web Service error occurred ***>");
     else {
      pr.println
                 ("<table border=\"1" bordercolor=\"navy\"
wide=\"100%\"><tr align+\"left\"><td>Customer</td>
<td>" + customer_name + "</td></tr>
<tr align+\"left\"><td>Part</td>
<td>" + part + "</td></tr>
<tr align+\"left\"><td>Price</td>
<td>" + quantity + "</td></tr></table>");
       pr.println("<br><font color=\"red\"
face=\"verdana\"
size=\"-1\">
Order submitted for processing");
     }
    }
   }
   catch (Exception e) {
    System.out.println(e);
    pr.println(e);
    return;
   }

  }
  catch (Exception e) {
  }
 }
}
```

图 74. MessageFormatter.java (7/7)

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
   <dtdid>resume.dtd</dtdid>
   <validation>NO</validation>
   <Xcolumn>
     <table name="resume_skills_sidetable">
       <column name="skill"
         type="varchar(20)"
         path="/resume/skill"
         multi_occurrence="YES">
       </column>
     </table>
   </Xcolumn>
</DAD>
```

图 75. resume.dad

```
<?xml version="1.0"?>
<!DOCTYPE Employees SYSTEM "employees.dtd">
<Employees>
  <Employee SerialNum="12">
    <Firstname>Laurie</Firstname>
    <Lastname>Douglas</Lastname>
    <Job_Description>
       DB2 engine Development
    </Job_Description>
  </Employee>
  <Employee SerialNum="13">
    <Firstname>John</Firstname>
    <Lastname>Smith</Lastname>
    <Job_Description>
       Information Integration Technology Solutions
   </Job_Description>
  </Employee>
  <Employee SerialNum="14">
    <Firstname>George</Firstname>
    <Lastname>Jackson</Lastname>
    <Job_Description>
        Customer contact
    </Job_Description>
  </Employee>
</Employees>
```

图 76. All_Employees.xml

相关概念:

- 第 10 页的『Cottonwood Distributors, Inc. - 仓库示例』
- 第 10 页的『发现数据 - 职员技能方案』

附录 B. DADX 环境检查程序

DADX 环境检查程序对用于与 WORF 配合使用创建和运行 Web 服务的 NST、DAD 和 DADX 文件执行不同的语法和语义检查。使用 DADX 环境检查程序来帮助使用 WORF 部署 Web 服务时发生的错误数减至最少。

安装 DADX 环境检查程序

DADX 环境检查程序是从命令行调用的 Java 应用程序。当调用该程序时，它产生包含错误、警告和成功指示符的输出文件。输出文本文件的名称是用户定义的。如果没有指定名称，则使用标准输出。

在 WORF 安装中的 *tools\lib* 子目录中包括 DADX 环境检查程序。包含此工具的代码的 JAR 文件为 CheckersCommon.jar 和 DADXEnvChecker.jar。确保已在您的系统上安装了 JRE 或 JDK V1.3.1 或更新版本。更新 CLASSPATH 以包括所有下列归档:

- *CheckersCommon.jar*、*DADXEnvChecker.jar* 和 *worf.jar*，包括在 WORF 的安装目录 *tools\lib* 中。
- *xerces.jar*。对于 UNIX 和 Windows，在 Xerces-J 2.0.2 的二进制文件分发（可从 http://xml.apache.org/ 下载）中包括这些文件。对于 OS/390 和 z/OS，在带有 PTF UW95866 的 IBM XML Toolkit V1R4 中包括这些文件。
- *soap.jar*，包含在 SOAP 2.2 的二进制文件分发（可从 http://xml.apache.org 下载）中，或包含在 WebSphere Application Server 安装中。
- *j2ee.jar* V1.3 或更新版本。可以从 java.sun.com 下载此文件
- *qname.jar*。可以从 java.sun.com 下载此文件
- *wsdl4j.jar*。可以从 http://oss.software.ibm.com/developerworks/projects/wsdl4j 下载此文件。
- *activation.jar*，包括在 JavaBeans Activation Framework 1.0.1 的二进制文件分发（可从 http://java.sun.com 下载）中。
- *mail.jar*，包括在 JavaMail 1.2 的二进制文件分发（可从 http://java.sun.com 下载）中。
- *servlet.jar*，包括在 WebSphere Application Server 安装中或 Jakarta Tomcat V3.2.x 至 4.0.3 或更新版本的分发（可从 http://www.apache.org/ 下载）中。
- 对于 UNIX 和 Windows: 为 *db2java.zip*，该文件包括在 “DB2 通用数据库” 安装目录中的 /java 目录中。对于 OS/390 和 z/OS: 为 *db2j2classes.zip*，该文件包括在 HFS 中 “DB2 通用数据库” 安装目录的 classes/ 子目录中。还可以使用 *jcc.jar*。*group.properties* 文件中的 `dbDriver` 参数决定了要使用的驱动程序包。

例如，如果正在 Windows 环境下运行，则必须设置 CLASSPATH 以查找以下文件:

```
CheckersCommon.jar;
DADXEnvChecker.jar;
worf.jar;
xerces.jar;
j2ee.jar
```

```
qname.jar
wsdl4j.jar
soap.jar;
db2java.zip;
```

相关概念:

- 第 27 页的『DADX 文件的定义』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 222 页的『运行 DADX 环境检查程序』

运行 DADX 环境检查程序

DADX 环境检查程序是一个 Java 程序，可在 JDK V1.3.1 及更新版本上运行。要运行 DADX 环境检查程序，执行以下命令（在一行上）：

```
java com.ibm.etools.webservice.util.Check_install
   [-srv] [-schdir pathToSchemasDir]
   [-sch schemaLocations] [-out outputFile]
  fileToCheck
```

例如，如果您在 *c:\dxxworf* 目录中解压缩 *dxxworf.zip*，则将输入以下命令来对 c:\tomcat\webapps\services 目录包含的资源文件运行 DADX 检查程序，然后将输出发送至当前目录中的 *myOutputFile.txt*:

```
java com.ibm.etools.webservice.util.Check_install
-srv -schdir c:\dxxworf\schemas
-out myOutputFile.txt c:\tomcat\webapps\services
```

参数

以下是可用来运行 DADX 环境检查程序的参数:

-schdir *pathToSchemasDir*

指定存储模式（用于验证 NST 文件和 DADX 文件）的目录的绝对路径。

-sch *schemaLocations*

指定要由解析器用来验证文件的一系列模式。DADX 检查程序允许用户指定 Xerces 解析器的属性值。此属性可用来指定对要分析的文件执行验证所需的 XML 模式的位置。通过提供模式的目标名称空间的名称（例如，*http://myschema*）并后跟模式的实际位置来指定模式的位置。它可以是文件系统中的路径（例如，*c:\dir\schema1.xsd*）或有效的 URL。但是，XML 文档本身可包含模式位置的声明。schemaLocation 属性在 XML 文档中用来提供此信息。以下是 XML 文档开始部分的一个示例:

```
<purchaseReport
  xmlns="http://www.example.com/Report"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/Report
http://www.example.com/Report.xsd">
```

对于特定名称空间，解析器将使用用解析器的属性定义的模式位置，即使 schemaLocation 属性为同一名称空间定义了另一个模式位置。schemaLocations 的语法与实例文档中 schemaLocation 属性的语法相同: 例如，

http://www.example.com file_name.xsd。用户可以指定多个 XML 模式：例如，-sch *http://www.example_1.com file_name_1.xsd http://www.example_2.com file_name_2.xsd*。

-out *outputFile*
指定输出文本文件名；如果省略此项，则使用标准输出。

-srv 指示必须对 Web 服务模块目录（例如：c:\tomcat\webapps\services）中作为 *fileToCheck* 传递的所有 NST、DAD 和 DADX 文件执行检查。如果不使用此选项，则仅对作为要检查的文件传递的 DADX 文件和包含在其它资源文件中的相关数据执行检查。例如，将检查此 DADX 文件中引用的 DAD 文件，然后将检查 NST 文件中这些 DAD 文件引用的 DTDID。并且将只检查 NST 文件和 web.xml 文件中与 DADX 文件相关的数据。

fileToCheck
如果不使用参数 *-srv*，则 *fileToCheck* 的值是检查的 DADX 文件。如果使用参数 *-srv*，则 *fileToCheck* 值是 Web 服务模块的根目录；例如，已解压缩的 .war 文件的根目录（如 services.war 模块的 *services*）。

-help 显示命令行选项信息

-version
显示版本信息

样本文件

可在 dxxworf.zip 中的 *tools\samples* 目录中找到样本文件。*DADXEnvChecker_sample.txt* 是显示对 Web 服务模块执行检查的结果的输出文本文件。DADX 环境检查程序生成此文件。检查程序使用在 -out 参数中指定的文件名 *DADXEnvChecker_sample.txt*。

相关概念:

- 第 27 页的『DADX 文件的定义』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 221 页的『安装 DADX 环境检查程序』

在输出文本文件中指示错误和警告

当使用 **-srv** 参数时，错误、警告和成功指示符被组合到各段中。每个段与一个已检查的文件相关联。检查每个文件的结果显示在输出文件中（如果指示了输出文件名）或标准输出设备中（如果没有指示文件名）。

按路径或子目录将段组合到目录中 **groups** 中。以下是从一个输出文本文件中摘录的片段，显示在属于组 /groups/dxx_sales_db 的文件 sales_db.nst 和 getstart_xcollection.dad 中执行的检查所对应的错误消息:

```
## Checking group: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db
## Checking NST file: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db\sales_db.nst
INFO. Line 5: file "c:\dxx\samples\dtd\getstart.dtd" is accessible.
ERROR. Line 12: file "wrongDtd.dtd" CANNOT be found
either in the file system or in the database.
INFO. Line 8: file "getstart.dtd" is accessible.
```

```
## Checking DAD file: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db\getstart_xcollection.dad
WARNING. Line 4: DTDID "dtd_.dtd" CANNOT be found in the DTD_REF table.
INFO. Line 9: the DTDID "c:\dxx\samples\dtd\getstart.dtd"
has been declared in the NST file.
```

如果错误、警告或成功事件与特定行相关，则错误、警告和成功消息可以行号开始。输出文本中的行号指示与消息相关联的已检查的元素在文件中所在的行号。在段内没有与输出相关的次序。

相关参考:

- 第 227 页的『检查 DAD 文件中的错误』
- 第 228 页的『检查 DADX 文件中的错误』
- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 221 页的『安装 DADX 环境检查程序』
- 第 222 页的『运行 DADX 环境检查程序』
- 第 226 页的『检查 NST 文件中的错误』
- 第 225 页的『检查 web.xml 文件中的错误』

由 DADX 环境检查程序执行的错误检查

当调用带有 -srv 参数的 DADX 环境检查程序时，首先对 WEB-INF 目录内的 web.xml 文件执行检查。然后，DADX 环境检查程序对 WEB-INF\classes\groups 目录中的每个组目录中的下列类型的文件执行检查:

- NST 文件
- DAD 文件
- DADX 文件

当调用不带有 -srv 参数的 DADX 环境检查程序时，首先对作为要检查的文件传递的 DADX 文件执行检查。然后，DADX 环境检查程序检查在此 DADX 文件中引用的 DAD 文件。它还对 DADX 文件所属的组的 NST 文件执行检查。DADX 环境检查程序最后检查包含该 DADX 文件的 WEB-INF 目录中的 web.xml 文件。

数据库错误消息

对于某些对 NST 和 DADX 文件的检查，DADX 环境检查程序执行以下操作:

1. 尝试使用文件 group.properties 中包含的数据来建立与数据库的连接
2. 查询与组关联的数据库
3. 检查某个组的文件中的错误

如果连接到数据库失败，DADX 环境检查程序会发出一则错误消息。以下示例显示典型的错误消息:

```
Checking group: c:\test\jakarta-tomcat-3.2.2
##Checking group: c:\tomcat\webapps\services
\WEB-INF\classes\groups\dxx_travel
WARNING. Connection error [IBM][CLI Driver]
SQL1013N The database alias name or database name
"TRAVELLL" could not be found.
SQLSTATE=42705
```

相关任务:

- 第 59 页的『定制 group.properties 文件』
- 第 54 页的『定义 web.xml 和 group.properties 文件』

相关参考:

- 第 227 页的『检查 DAD 文件中的错误』
- 第 228 页的『检查 DADX 文件中的错误』
- 第 226 页的『检查 NST 文件中的错误』
- 第 225 页的『检查 web.xml 文件中的错误』

检查 web.xml 文件中的错误

DADX 环境检查程序检查“Web 服务”模块（在本示例中为 services）的根目录下的 **WEB-INF\web.xml** 文件。

以下是 web.xml 文件的片段:

```
<servlet>
<servlet-name>dxx_sales_db</servlet-name>
<servlet-class>com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker
</servlet-class>
        <init-param>
          <param-name>faultListener</param-name>
<param-value>org.apache.soap.server.DOMFaultListener
</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
<servlet-mapping>
<servlet-name>dxx_sales_db</servlet-name>
<url-pattern>/sales/*</url-pattern>
</servlet-mapping>
```

<servlet-class> 标记（它是 <servlet> 标记的直接子代）必须具有 com.ibm.etools.webservice.rt.isd.servlet.IsdInvoker 或 com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker 的值。当它们的值不同时，检查程序会提供错误消息。以下示例显示在 web.xml 文档中对 <servlet-class> 标记执行检查的结果:

```
INFO. Line 21: servlet class for
servlet "dxx_sales_db" is a correct servlet class.
ERROR. Line 31: servlet class
"com.ibm.etools.webservice.rt.dxx.servlet.OtherInvoker"
for servlet "dxx_sample"
is NOT a correct servlet class.
INFO. Line 41: servlet class
for servlet "dxx_travel"
is a correct servlet class.
```

每个 <servlet-mapping> 标记包含一个 <servlet-name> 标记，它的值必须与 <servlet> 标记的 <servlet-name> 标记的值相同。如果情况不是这样，检查程序将提供错误消息，如以下示例中所示:

```
ERROR. There is no <servlet>
tag declaring servlet
"isd_demos" mapped at line 50.
```

否则，每个 <servlet> 标记必须具有相应的 <servlet-mapping> 标记（具有相同的 servlet 名）。如果 <servlet> 标记没有相应的 <servlet-mapping> 标记，则检查程序提供以下类型的消息:

```
ERROR. There is no
<servlet-mapping> tag
for servlet "dxx_travel" declared
at line 40.
```

每个 <servlet-mapping> 标记还包含 <url-pattern> 标记（其值必须唯一）。如果两个 <url-pattern> 标记具有相同的值，则检查程序提供如以下示例所示的错误消息:

```
ERROR. Line 56: "/sales/*" is already
declared as the URL pattern for servlet "isd_demos"
(see line 50).
```

相关任务:

- 第 54 页的『定义 web.xml 和 group.properties 文件』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』

检查 NST 文件中的错误

在每个组目录中都可能有 NST 文件。NST 文件声明组的名称空间表。它们包含 DTD 标识与从 DTD 自动生成的 XML 模式的名称空间和位置之间的映射。

以下是 NST 文件的片段:

```
<namespaceTable
xmlns="http://schemas.ibm.com/db2/dxx/nst">
    <mapping dtdid="c:\dxx\samples\dtd\getstart.dtd"
      namespace="http://schemas.ibm.com/db2/dxx/samples/dtd/getstart.dtd"
    location="/dxx/samples/dtd/getstart.dtd/XSD"/>
    <mapping dtdid="getstart.dtd"
      namespace="http://schemas.myco.com/sales/getstart.dtd"
    location="/getstart.dtd/XSD"/>
```

DADX 环境检查程序首先在 **nst.xsd** 中验证 NST 文件的模式是否正确。以下是由检查程序报告的验证错误的示例:

```
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 8, column 35. cvc-complex-type.2.4.a:
Invalid content starting with element 'mappin'.
The content must match
'("http://schemas.ibm.com/db2/dxx/nst":mapping){0-UNBOUNDED}'.
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 17, column 32. cvc-complex-type.4: Attribute 'dtdid'
must appear on element 'mapping'.
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 17, column 32. Duplicate unique value
[ID Value: /order.dtd/XSD] declared for identity constraint
of element "namespaceTable".
```

最终，检查程序检查 <mapping> 元素的 dtdid 属性是:

- 文件系统中的正确路径，或者
- 存储在 db2xml.DTD_REF 表中的列 DTDID 中的值

以下示例显示对 NST 文件的 <mapping> 元素的检查结果:

```
INFO. Line 5: file
"c:\dxx\samples\dtd\getstart.dtd" is accessible.
ERROR. Line 14: file
"wrongDtd.dtd" CANNOT be found either in
the file system or in the database.
```

相关参考:

- 第 227 页的『检查 DAD 文件中的错误』
- 第 228 页的『检查 DADX 文件中的错误』
- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 221 页的『安装 DADX 环境检查程序』
- 第 222 页的『运行 DADX 环境检查程序』
- 第 225 页的『检查 web.xml 文件中的错误』

检查 DAD 文件中的错误

“文档存取定义”（DAD）文件是一个 XML 文件，在 DB2 XML Extender 中受支持。DAD 通过两种备用存取和存储方法（XML 列和 XML 集合）将 XML 文档与 DB2 通用数据库表相关联。

以下示例显示 DAD 文件的开始部分:

```
<?xml
version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>NO</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM
 "c:\dxx\samples\dtd\getstart.dtd"
</doctype>
<root_node>
<element_node name="Order">
...
```

DADX 环境检查程序首先根据 DAD 文件的 DTD **dad.dtd** 检查该 DAD 文件是否有效。必须确保在 DAD 的 DOCTYPE 声明中指定的 dad.dtd 路径是有效的。

然后，如果 <dtdid> 标记存在，则检查程序获取它的值。如果此标记的值与存储在 db2xml.DTD_REF 表中的列 DTDID 中的值不匹配，则检查程序发出警告。如果 DAD 中的 <validation> 标记包含值 *YES*，则检查程序发出错误消息:

```
## Checking DAD file:
c:\tomcat\webapps\services\WEB-INF
\classes\groups\dxx_sales_db\order.dad
ERROR. Line 4: DTDID "wrongDtd.dtd"
CANNOT be found in the DTD_REF table.
```

然后，检查程序确定 DAD 文件是声明 Xcollection 还是声明 Xcolumn。如果它声明 Xcollection，则将抽取在 <doctype> 元素中指定的 DTD。DADX 环境检查程序检查此 DTD 是否是在 NST 文件中声明的。

以下示例显示对属于同一个组的 Xcolumn 和 Xcollection DAD 的检查结果:

```
## Checking DAD file:
c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db\getstart_xcolumn.dad
INFO. Line 4: DTDID "getstart.dtd" was found in the DTD_REF table.

## Checking DAD file: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db\order-public.dad
INFO. Line 4: DTDID "order.dtd" was found in the DTD_REF table.
ERROR. Line 8: the DTDID "order.dtd" has NOT been
declared in the NST file.
```

还可以通过使用 DAD 检查程序对 DAD 文件执行其它检查。DAD 检查程序是一个独立的工具，还包含在 dxxworf.zip 中的 *tools\lib* 目录中。有关更多信息，请参阅 WebSphere Application Development Web 站点中的有关 dadchecker 工具的文档。

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』
- 第 119 页的『Web 服务样本 - PartOrders.dadx』

检查 DADX 文件中的错误

“文档存取定义扩展”（DADX）是用于快速创建访问数据库的 Web 服务的一项技术。DADX 让您使用标准 SQL 语句（SELECT、INSERT、UPDATE、DELETE 和 CALL）以及 DB2 XML Extender 存储过程来定义 Web 服务操作。

以下是 DADX 文件的片段:

```
<?xml version="1.0"?>
  <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<operation name="find">
    <documentation >
Returns the parts from order #1 with price > 20000.
      </documentation>
      <retrieveXML>
       <DAD_ref>getstart_xcollection.dad</DAD_ref>
       <no_override/>
      </retrieveXML>
</operation>
<operation name="findByMinPrice">
      <retrieveXML>
<collection_name>
           getstart_xcollection.dad
</collection_name>
       <no_override/>
<parameter name="minprice"
type="xsd:decrimal"/>
      </retrieveXML>
  </operation>
```

DADX 环境检查程序首先根据 DADX 文件的模式 dadx.xsd 验证该文件。然后，检查程序获取 <DAD_REF> 或 <collection_name> 标记的值并检查这些标记的值是否为以下情况:

- 对于 <DAD_REF> 标记，为文件系统中 DAD 文件的正确路径。
- 对于 <collection_name> 标记，为启用的集合的名称，它是在表 db2xml.xml_usage 中的列 COL_NAME 中存储的值。

以下示例显示对 DADX 文件执行的检查的结果:

```
## Checking DADX file: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sales_db\PartOrders.dadx
ERROR. Validation error, in "file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/PartOrders.dadx",
line 8, column 67. cvc-complex-type.2.4.c:
The matching wildcard is strict, but no declaration
can be found for element 'as'.
INFO. Line 16: for operation "find",
DAD "getstart_xcollection.dad" was found.
ERROR. Line 26: for operation "findAll",
DAD "non_existing_dad.dad" was NOT found.
INFO. Line 44: for operation "findByColor",
DAD "getstart_xcollection.dad" was found.
INFO. Line 65: for operation "findByMinPrice",
DAD "getstart_xcollection.dad" was found.
```

如果 <operation> 标记没有作为子代的 <DAD_REF> 或 <collection_name> 标记，则检查程序发出消息指示没有对此特定操作执行检查，如以下示例所示:

```
##
Checking DADX file: c:\tomcat\webapps\services\WEB-
INF\classes\groups\dxx_sample\HelloSample.dadx
INFO. Line 10: no <DAD_ref> or <collection_name>
elements to check for operation "listDepartments".
```

DADX 环境检查程序还检查 WORF 是否将能够查找在 DADX 文件中声明的参数的反序列化器。反序列化器将通过网络连接接收到的 XML 消息重构到指定的变量或对象中。对于每个 <parameter> 标记，其类型属性的值必须是可被反序列化的类型。如果找不到特定类型的反序列化器，则检查程序提供如以下示例所示的错误消息:

```
ERROR. Line 13: no deserializer was found
to deserialize a
 "http://www.w3.org/2001/XMLSchema:ssstring", using encoding
"http://schemas.xmlsoap.org/soap/encoding/".
```

相关概念:

- 第 27 页的『DADX 文件的定义』

相关参考:

- 第 224 页的『由 DADX 环境检查程序执行的错误检查』

附录 C. DADX 文件的 XML 模式

以下“可扩展标记语言”（XML）模式 *dadx.xsd* 描述 DADX。所有 WORF 模式文件都位于 dxxworf.zip 文件中，此文件是样本目录的一部分。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://schemas.ibm.com/db2/dxx/dadx"
  xmlns:dadx="http://schemas.ibm.com/db2/dxx/dadx"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" xml:lang="en">
 <annotation>
  <documentation>
      A Document Accession Definition Extension (DADX)
      document defines a Web Service
       that is implemented by operations that
       access a relational database and that optionally use
       stored procedures, types and functions provided
       by the DB2 XML Extender.
      </documentation>
 </annotation>
 <element name="DADX">
  <annotation>
   <documentation>
        Defines a Web Service.
        The Web Service is described by an optional
         WSDL documentation element.
        The Web Service may implement a set of WSDL
        bindings defined elsewhere.
        The Web Service consists of one or more
         uniquely named operations.
      </documentation>
  </annotation>
```

图 77. DADX 模式 (1/17)

```
        <complexType>
  <sequence>
   <element ref="dadx:documentation"
            minOccurs="0" maxOccurs="unbounded"/>
   <choice>
    <element ref="dadx:DQS"
                minOccurs="0"/>
    <sequence>
     <element ref="dadx:implements" minOccurs="0"/>
     <element ref="dadx:result_set_metadata"
                  minOccurs="0" maxOccurs="unbounded"/>
     <element ref="dadx:operation"
                  maxOccurs="unbounded"/>
    </sequence>
   </choice>
  </sequence>
 </complexType>
 <key name="result_set_metadataNames">
  <selector xpath="dadx:result_set_metadata"/>
  <field xpath="@name"/>
 </key>
 <keyref name="resultSetMetatdata"
        refer="dadx:result_set_metadataNames">
  <selector xpath="dadx:operation/dadx:call/dadx:result_set"/>
  <field xpath="@metadata"/>
 </keyref>
 <unique name="operationNames">
  <selector xpath="dadx:operation"/>
  <field xpath="@name"/>
 </unique>
</element>
```

图 77. DADX 模式 (2/17)

```
<element name="DQS">
  <annotation>
   <documentation>
        Defines the DQS tag.
      </documentation>
  </annotation>
  <complexType/>
 </element>
```

图 77. DADX 模式 (3/17)

```
<element name="documentation">
 <annotation>
  <documentation>
       Defines WSDL documentation for the Web service or an operation.
     </documentation>
 </annotation>
 <complexType mixed="true">
  <choice minOccurs="0" maxOccurs="unbounded">
   <any minOccurs="0" maxOccurs="unbounded"/>
  </choice>
  <anyAttribute/>
 </complexType>
</element>
```

图 77. DADX 模式 (4/17)

```
<element name="implements">
  <annotation>
   <documentation>
       Defines the namespace and location of a set of WSDL bindings
       defined elsewhere. This information is imported into the
       WSDL document generated for this Web Service.
     </documentation>
  </annotation>
  <complexType>
   <attribute name="namespace"
            type="anyURI" use="required"/>
   <attribute name="location"
            type="anyURI" use="required"/>
  </complexType>
 </element>
```

图 77. DADX 模式 (5/17)

```
<element name="result_set_metadata">
  <annotation>
   <documentation>
        Defines the metadata for a result set returned
        by a stored procedure.
        Each metadata element defines a global element
        in the WSDL for the Web Service.
        The metatdata name defines the name of its global element.
        The metadata rowName defines the element name of each row.
        The metadata contains one or more column definitions.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element ref="dadx:column"
             maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name"
              type="NCName"
              use="required"/>
   <attribute name="rowName"
              type="NCName"
              use="required"/>
  </complexType>
 </element>
```

图 77. DADX 模式 (6/17)

```
<element name="column">
  <annotation>
   <documentation>
        Defines the metadata for a column of a result set
        returned by a stored procedure.
        The column name, type, and nullability must match the values
           returned by the JDBC result set metadata at runtime.
        A column is considered to be nullable unless it is explicitly
           defined to not accept nulls.
        If the "nullable" attribute is absent then
        the column is considered to not be nullable.
        The element name associated with the column isdefined
           by the value of the "as" attribute if present,
          or the column name otherwise.
        The element may contain an XML document, in which case
           it must have an "element" attribute that
           defines the XML Schema name
        of its root element.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
              type="string"
              use="required"/>
   <attribute name="type"
              type="dadx:columnType"
              use="required"/>
   <attribute name="nullable"
              type="boolean"/>
   <attribute name="as"
              type="string"/>
   <attribute name="element"
              type="QName"/>
  </complexType>
 </element>
```

图 77. DADX 模式 (7/17)

```
 <simpleType name="columnType">
  <restriction base="string">
   <enumeration value="BIGINT"/>
   <enumeration value="CHAR"/>
   <enumeration value="CLOB"/>
   <enumeration value="DATE"/>
   <enumeration value="DECIMAL"/>
   <enumeration value="DOUBLE"/>
   <enumeration value="FLOAT"/>
   <enumeration value="INTEGER"/>
   <enumeration value="NUMERIC"/>
   <enumeration value="REAL"/>
   <enumeration value="SMALLINT"/>
   <enumeration value="TIME"/>
   <enumeration value="TIMESTAMP"/>
   <enumeration value="TINYINT"/>
   <enumeration value="VARCHAR"/>
  </restriction>
 </simpleType>
```

图 77. DADX 模式 (8/17)

```
<element name="operation">
  <annotation>
   <documentation>
        Defines an operation of the Web Service.
        Each operation has a unique name and is optionally described
        by WSDL documentation.
        An operation is defined using one of the supported operators.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element ref="dadx:documentation"
               minOccurs="0"/>
    <choice>
     <element ref="dadx:retrieveXML"/>
     <element ref="dadx:storeXML"/>
     <element ref="dadx:query"/>
     <element ref="dadx:update"/>
     <element ref="dadx:call"/>
    </choice>
   </sequence>
   <attribute name="name"
            type="NCName"
            use="required"/>
  </complexType>
 </element>
```

图 77. DADX 模式 (9/17)

```
<element name="retrieveXML">
  <annotation>
   <documentation>
        Retrieves a set of XML documents by composing
        them from relational data.
        This operator requires the DB2 XML Extender.
        The mapping from relational data to XML is defined by a
        Document Access Definition (DAD) which can be specified
        by refering to either a resource file or the name
        of an XML Collection
        that has been previously enabled in the database.
        The DAD must define an XML Collection and can use either SQL
        or RDB mapping. The DAD behavior may be modified by an override.
        If no override is desired, the no_override element must be used.
        Otherwise, the SQL_override element must be used
        for SQL mapping and the
        XML_override element must be used for RDB mapping. In either case, the
        override string may contain input parameters using
        the host variable syntax.
        The name and type of all parameters must be defined in a list of
        parameter elements that are uniquely named within this operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <choice>
     <element ref="dadx:DAD_ref"/>
     <element ref="dadx:collection_name"/>
    </choice>
    <choice>
     <element name="no_override">
      <complexType/>
     </element>
     <element name="SQL_override"
                 type="string"/>
     <element name="XML_override"
                 type="string"/>
    </choice>
    <element ref="dadx:parameter"
              minOccurs="0"
              maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="retrieveXmlParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

图 77. DADX 模式 (10/17)

```
<element name="storeXML">
  <annotation>
   <documentation>
        Stores an XML document by decomposing it into relational data.
        This operator requires the DB2 XML Extender.
        The mapping from relational data to XML is defined by a
        Document Access Definition (DAD) which can be specified
        by refering to either a resource file or the name of
        an XML Collection
        that has been previously enabled in the database.
        The DAD must define an XML Collection and must use RDB mapping.
      </documentation>
  </annotation>
  <complexType>
   <choice>
    <element ref="dadx:DAD_ref"/>
    <element ref="dadx:collection_name"/>
   </choice>
  </complexType>
 </element>
```

图 77. DADX 模式 (11/17)

```
<element name="query">
  <annotation>
   <documentation>
        Retrieves a set of relational data using an
        SQL SELECT statement.
        The result set must consist of uniquely named columns.
        If any result set column contains XML documents,
          the XML document type must be
        defined using an XML_result element.
        The statement may contain input parameters using
        the host variable syntax.
        The input parameters must be defined by a list of
        parameter elements that are
        uniquely named within this operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_query"
               type="string"/>
    <element ref="dadx:XML_result"
               minOccurs="0"
               maxOccurs="unbounded"/>
    <element ref="dadx:parameter"
               minOccurs="0"
               maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="XML_resultNames">
   <selector xpath="dadx:XML_result"/>
   <field xpath="@name"/>
  </unique>
  <unique name="queryParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

图 77. DADX 模式 (12/17)

```
<element name="update">
  <annotation>
   <documentation>
        Updates a relational table using an SQL INSERT,
        UPDATE, or DELETE statement and
        reports the number of rows affected.
        The statement may contain input parameters
            using the host variable syntax.
        The input parameters must be defined by a list of
        parameter elements that are
        uniquely named within this operation.
       </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_update"
                type="string"/>
    <element ref="dadx:parameter"
                minOccurs="0"
                maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="updateParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

图 77. DADX 模式 (13/17)

```
<element name="call">
  <annotation>
   <documentation>
        Calls a stored procedure.
        The call statement contains in, out, and
           in/out parameters using host variable syntax.
        The parameters are defined by a list of parameter
        elements that are uniquely named
        within the operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_call"
              type="string"/>
    <element ref="dadx:parameter"
              minOccurs="0"
              maxOccurs="unbounded"/>
    <element ref="dadx:result_set"
              minOccurs="0"
              maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="callParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
  <unique name="callResultSetNames">
   <selector xpath="dadx:result_set"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

图 77. DADX 模式 (14/17)

```
<element name="result_set">
  <annotation>
   <documentation>
        Defines a result set.
        The name defines the element name of the result
        set and becomes part of the output message.
        The metatdata name refers to a result set metadata
        element defined in the same document.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
            type="NCName" use="required"/>
   <attribute name="metadata"
            type="NCName" use="required"/>
  </complexType>
 </element>
 <element name="DAD_ref"
           type="string"/>
 <element name="collection_name"
           type="string"/>
```

图 77. DADX 模式 (15/17)

```
<element name="parameter">
  <annotation>
   <documentation>
        Defines a named parameter. A parameter
        must have it contents defined either by
        an XML Schema element or type, but not both.
        The parameter kind in one of in,
        out, or in/out, with in being the default.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
             type="NCName"
             use="required"/>
   <attribute name="element"
             type="QName"/>
   <attribute name="type"
             type="QName"/>
   <attribute name="kind"
             type="dadx:parameterKindType"
             default="in"/>
  </complexType>
 </element>
 <simpleType name="parameterKindType">
  <restriction base="string">
   <enumeration value="in"/>
   <enumeration value="out"/>
   <enumeration value="in/out"/>
  </restriction>
 </simpleType>
```

图 77. DADX 模式 (16/17)

```
<element name="XML_result">
  <annotation>
   <documentation>
        Defines a named column that contains XML documents.
        The document type
        must be defined by the XML Schema element of its root.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
            type="NCName"
            use="required"/>
   <attribute name="element"
            type="QName"
            use="required"/>
  </complexType>
 </element>
< /schema>
```

图 77. DADX 模式 (17/17)

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 27 页的『DADX 文件的定义』

相关任务:

- 第 80 页的『将文档类型定义转换为 XML 模式』

相关参考:

- 第 74 页的『DADX 操作示例』

附录 D. Web 服务编码算法

这是对 group.properties 文件中的密码进行编码和译码的算法。

1. 通过使用 UTF-8 字符编码将纯文本信息转换为数据字节序列。L 是数据字节序列的长度。
2. 将数据字节转换为进一步的数据字节序列 data8，长度是原来的 8 倍。如下所示计算 data8 的字节（k）。假设 *k = j * L + i*，其中 0 <= i < L 并且 0 <= j < 8。首先，隐去数据字节 *i* 的位 *j*。其次，将结果与 *k* 进行 *exclusive or*。此步骤将每个数据字节的位分布在 data8 序列的整个长度上。
3. 将标准 base64 编码算法应用于 data8。此步骤将字节转换为可打印字符，并且还将长度增加三分之四（4/3）。
4. 在已编码字符串前面加上前缀 “encoded:” 以表示已对它编码。

相关概念:

- 第 61 页的『使用文档存取定义扩展文件来定义 Web 服务』
- 第 27 页的『DADX 文件的定义』
- 第 102 页的『测试 Web 服务应用程序 - 方案』
- 第 29 页的『Web 服务过程概述』

相关参考:

- 第 247 页的附录 E, 『Web 服务命令参考』

附录 E. Web 服务命令参考

本节说明可以用来在 WORF 内执行特定功能的命令。

编码器 在 group.properties 文件中对密码进行编码或译码。

- 编码的示例（假定 worf.jar 已列示在 CLASSPATH 中）：

```
java com.ibm.etools.webservice.rt.util.Encoder
     -in group.properties -out group.properties
```

- 译码的示例（假定 worf.jar 已列示在 CLASSPATH 中）：

```
java com.ibm.etools.webservice.rt.util.Encoder
     -action decode -in group.properties -out group.properties
```

Dadx2Dd

从 DADX 文件生成部署描述符。

- 示例:

```
java com.ibm.etools.webservice.rt.dadx.Dadx2Dd
```

Check_install

验证 DADX 文件。

- 示例:

```
java com.ibm.etools.webservice.util.Check_install
   [-srv] [-schdir pathToSchemasDir]
   [-sch schemaLocations]  [-out outputFile] fileToCheck
```

dadchecker

验证 DAD 文件。有关用于此命令的参数的更多信息，请参阅 http://www.ibm.com/software/data/db2/extenders/xmlext/download/beta/dadcheck_rn.html

- 示例:

```
java dadchecker.Check_dad_xml  [-dad | -xml] [-all]
    [-dup dupName] [-enc encoding][-dtd dtdPath]
    [-xstruct xmlDocument] [-out outputFile] fileToCheck
```

相关概念:

- 第 23 页的『关于使用 DB2 作为 Web 服务提供程序的简介 - WORF』
- 第 27 页的『Web 服务提供程序功能部件』

相关任务:

- 第 124 页的『生成部署描述符』

相关参考:

- 第 245 页的附录 D, 『Web 服务编码算法』
- 第 119 页的『Web 服务样本 - PartOrders.dadx』

DB2 Information Integrator 文档

该主题提供了关于 DB2 Information Integrator 可用的文档的信息。该主题中的表提供了正式的文档标题、书号以及每本 PDF 书籍的位置。要订购印刷书籍，您必须知道正式的书名或文档书号。该主题还提供了 DB2 Information Integrator 发行说明和安装需求的标题、文件名以及位置。

该主题包含以下部分:

- 访问 DB2 Information Integrator 文档
- 关于 z/OS 上的复制功能的文档
- 关于 z/OS 版 DB2 通用数据库的事件发布功能的文档
- 关于 z/OS 上的 IMS 和 VSAM 的事件发布功能的文档
- 关于 Linux、UNIX 和 Windows 上的事件发布功能和复制功能的文档
- 关于 z/OS 上的联合功能的文档
- 关于 Linux、UNIX 和 Windows 上的联合功能的文档
- 关于 Linux、UNIX 和 Windows 上的企业搜索的文档
- 发行说明和安装需求

访问 DB2 Information Integrator 文档

所有 DB2 Information Integrator 书籍和发行说明都提供了 PDF 文件，在 DB2 Information Integrator Support Web 站点提供，网址为: www.ibm.com/software/data/integration/db2ii/support.html。

要访问最新的 DB2 Information Integrator 产品文档，可从 DB2 Information Integrator Support Web 站点单击 Product Information 链接，如第 250 页的图 78 所示。

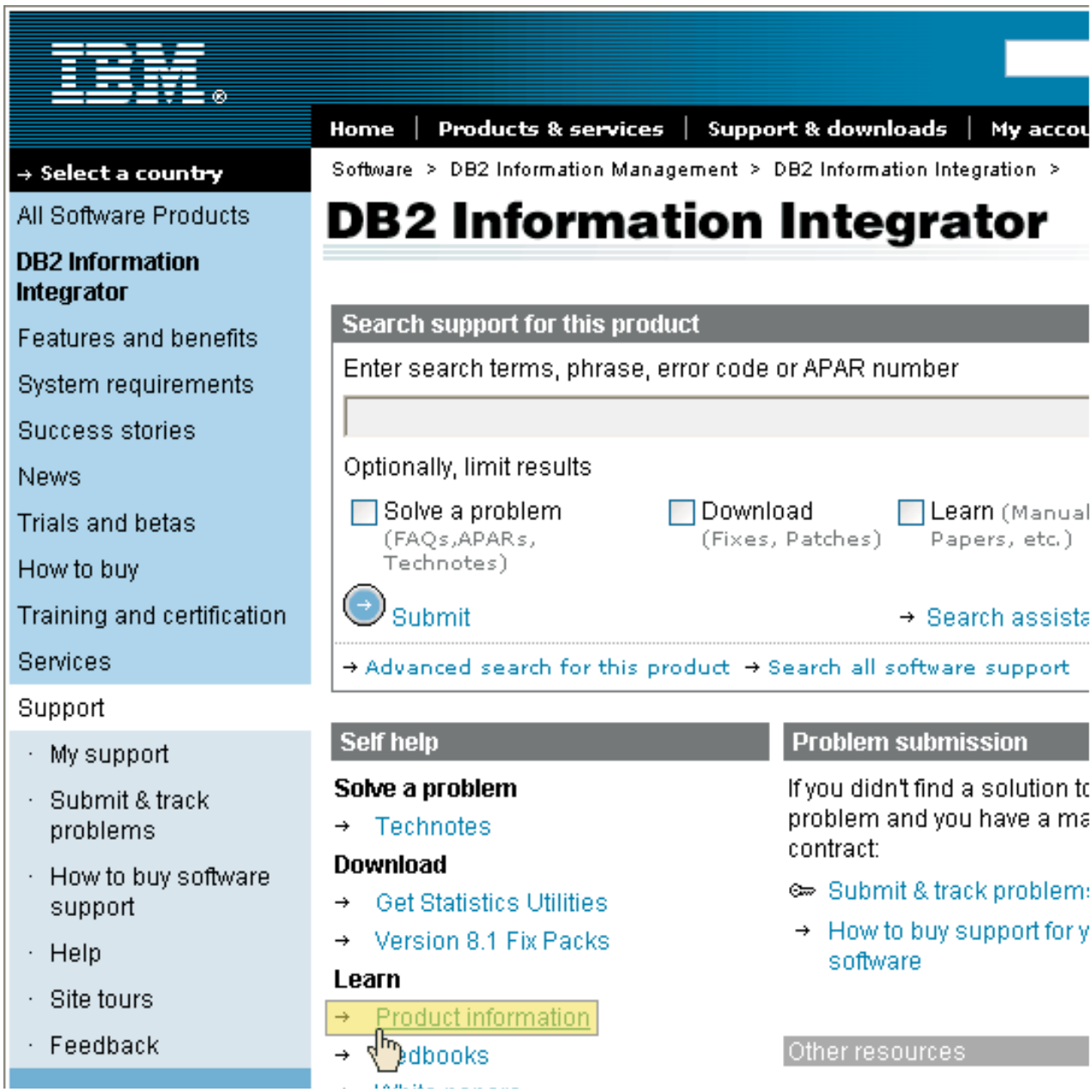


图 78. 从 DB2 Information Integrator Support Web 站点访问 Product Information 链接

从 Product Information 链接可访问所有受支持语言的最新 DB2 Information Integrator 文档:

- PDF 文件格式的 DB2 Information Integrator 产品文档
- 修订包产品文档，包括发行说明
- 下载和安装用于 Linux、UNIX 和 Windows 的 DB2 信息中心的指示信息
- 在线 DB2 信息中心的链接

在列表中滚动以查找您正在使用的 DB2 Information Integrator 版本的产品文档。

DB2 Information Integrator Support Web 站点也提供了支持文档、IBM 红皮书、白皮书、产品下载、对用户组的链接以及关于 DB2 Information Integrator 的新闻。

也可以从 *DB2 PDF 文档* CD 查看并打印 DB2 Information Integrator PDF 书籍。

要查看或打印 PDF 文档:

1. 从 *DB2 PDF 文档* CD 的根目录打开 *index.htm* 文件。
2. 单击要使用的语言。
3. 单击要查看的文档的链接。

关于 z/OS 上的复制功能的文档

表 30. 关于 z/OS 上的复制功能的 DB2 Information Integrator 文档

书名	书号	位置
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web 站点
Introduction to Replication and Event Publishing	GC18-7567	DB2 Information Integrator Support Web 站点
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web 站点
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Replication Installation and Customization Guide for z/OS	SC18-9127	DB2 Information Integrator Support Web 站点
《SQL 复制指南和参考》	S152-0734	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web 站点
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• 在 DB2 信息中心中，**产品概述 > 信息集成 > DB2 Information Integrator 概述 > 问题、解决办法和文档更新** • DB2 Information Integrator 安装启动板 • DB2 Information Integrator Support Web 站点 • The *DB2 Information Integrator* 产品 CD

关于 z/OS 版 DB2 通用数据库的事件发布功能的文档

表 31. 关于 z/OS 版 DB2 通用数据库的事件发布功能的 DB2 Information Integrator 文档

书名	书号	位置
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web 站点
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• 在 DB2 信息中心中，**产品概述 > 信息集成 > DB2 Information Integrator 概述 > 问题、解决办法和文档更新** • DB2 Information Integrator 安装启动板 • DB2 Information Integrator Support Web 站点 • The *DB2 Information Integrator* 产品 CD

关于 z/OS 上的 IMS 和 VSAM 的事件发布功能的文档

表 32. 关于 z/OS 上的 IMS 和 VSAM 的事件发布功能的 DB2 Information Integrator 文档

书名	书号	位置
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web 站点
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web 站点
Getting Started with Event Publisher for z/OS	GC18-9186	DB2 Information Integrator Support Web 站点
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web 站点
Operations Guide for Event Publisher for z/OS	SC18-9157	DB2 Information Integrator Support Web 站点
Planning Guide for Event Publisher for z/OS	SC18-9158	DB2 Information Integrator Support Web 站点
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web 站点

表 32. 关于 z/OS 上的 IMS 和 VSAM 的事件发布功能的 DB2 Information Integrator 文档 (续)

书名	书号	位置
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web 站点

关于 Linux、UNIX 和 Windows 上的事件发布功能和复制功能的文档

表 33. 关于 Linux、UNIX 和 Windows 上的事件发布功能和复制功能的 DB2 Information Integrator 文档

书名	书号	位置
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web 站点
《Linux、UNIX 和 Windows 上的安装指南》	G152-0550	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web 站点
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《SQL 复制指南和参考》	S152-0734	DB2 Information Integrator Support Web 站点
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web 站点
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• 在 DB2 信息中心中，**产品概述 > 信息集成 > DB2 Information Integrator 概述 > 问题、解决办法和文档更新** • DB2 Information Integrator 安装启动板 • DB2 Information Integrator Support Web 站点 • The *DB2 Information Integrator* 产品 CD

关于 z/OS 上的联合功能的文档

表 34. 关于 z/OS 上的联合功能的 DB2 Information Integrator 文档

书名	书号	位置
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web 站点
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web 站点
Getting Started with Classic Federation for z/OS	GC18-9155	DB2 Information Integrator Support Web 站点
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web 站点
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web 站点
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web 站点
Transaction Services Guide for Classic Federation for z/OS	SC18-9161	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web 站点

关于 Linux、UNIX 和 Windows 上的联合功能的文档

表 35. 关于 Linux、UNIX 和 Windows 上的联合功能的 DB2 Information Integrator 文档

书名	书号	位置
《应用程序开发者指南》	S152-0601	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《开发包装器的 C++ API 参考》	S152-0844	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《数据源配置指南》	N/A	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《联合系统指南》	S152-0600	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Guide to Configuring the Content Connector for VeniceBridge	N/A	DB2 Information Integrator Support Web 站点
《Linux、UNIX 和 Windows 上的安装指南》	G152-0550	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点

表 35. 关于 Linux、UNIX 和 Windows 上的联合功能的 DB2 Information Integrator 文档 (续)

书名	书号	位置
Java API Reference for Developing Wrappers	SC18-9173	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《迁移指南》	S152-0603	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
《包装器开发者指南》	S152-0845	• *DB2 PDF 文档* CD • DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• 在 DB2 信息中心中，**产品概述 > 信息集成 > DB2 Information Integrator 概述 > 问题、解决办法和文档更新** • DB2 Information Integrator 安装启动板 • DB2 Information Integrator Support Web 站点 • The *DB2 Information Integrator* 产品 CD

关于 Linux、UNIX 和 Windows 上的企业搜索功能的文档

表 36. 关于 Linux、UNIX 和 Windows 上的企业搜索功能的 DB2 Information Integrator 文档

书名	书号	位置
Administering Enterprise Search	SC18-9283	DB2 Information Integrator Support Web 站点
Installation Guide for Enterprise Search	GC18-9282	DB2 Information Integrator Support Web 站点
Programming Guide and API Reference for Enterprise Search	SC18-9284	DB2 Information Integrator Support Web 站点
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web 站点

发行说明和安装需求

发行说明提供了针对您所用产品的发行版和修订包级别的信息，还包括了对每个发行版的文档的最新修订。

安装需求提供了针对您所用产品的发行版的信息。

表 37. DB2 Information Integrator 发行说明和安装需求

书名	文件名	位置
Installation Requirements for IBM DB2 Information Integrator Event Publishing Edition, Replication Edition, Standard Edition, Advanced Edition, Advanced Edition Unlimited, Developer Edition, and Replication for z/OS	Prereqs	• The *DB2 Information Integrator* 产品 CD • DB2 Information Integrator 安装启动板
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	ReleaseNotes	• 在 DB2 信息中心中，**产品概述 > 信息集成 > DB2 Information Integrator 概述 > 问题、解决办法和文档更新** • DB2 Information Integrator 安装启动板 • DB2 Information Integrator Support Web 站点 • The *DB2 Information Integrator* 产品 CD
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web 站点
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web 站点
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web 站点

要查看产品 CD 上的安装需求和发行说明:

- 在 Windows 操作系统上输入:

  ```
  x:\doc\%L
  ```

  x 是 Windows CD 盘符，*%L* 是要使用的文档的语言环境，例如，*en_US*。
- 在 UNIX 操作系统上输入:

  ```
  /cdrom/doc/%L/
  ```

  cdrom 表示 UNIX 上的 CD 安装点，*%L* 是要使用的文档的语言环境，例如，*en_US*。

辅助功能

辅助功能部件可帮助那些身体有某些缺陷（如活动不方便或视力不太好）的用户成功地使用软件产品。以下列表指定 DB2® V8 产品中的主要辅助功能部件:

- 所有 DB2 功能可使用键盘（而不是鼠标）导航来实现。有关更多信息，请参阅『键盘输入和导航』。
- 可定制 DB2 用户界面上的字体大小和颜色。有关更多信息，请参阅『界面显示的辅助功能』。
- DB2 产品支持使用 Java™ Accessibility API 的辅助功能应用程序。有关更多信息，请参阅第 258 页的『与辅助技术的兼容性』。
- DB2 文档是以易使用格式提供的。有关更多信息，请参阅第 258 页的『文档的辅助功能』。

键盘输入和导航

键盘输入

只使用键盘就可以操作 DB2 工具。使用键或键组合就可以执行使用鼠标所能完成的操作。标准操作系统击键用于标准操作系统操作。

有关使用键或键组合执行操作的更多信息，请参阅 键盘快捷方式和加速键: 公共 GUI 帮助。

键盘导航

可使用键或键组合来导航 DB2 工具用户界面。

有关使用键或键组合来导航 DB2 工具的更多信息，请参阅 键盘快捷方式和加速键: 公共 GUI 帮助。

键盘焦点

在 UNIX® 操作系统中，击键操作起作用的活动窗口的区域将突出显示。

界面显示的辅助功能

DB2 工具所具有的功能部件使视力不太好的用户更易使用。这些辅助功能方面的增强包括了对可定制字体属性的支持。

字体设置

可使用“工具设置”笔记本来选择菜单和对话框窗口中文本的颜色、大小和字体。

有关指定字体设置的更多信息，请参阅 更改菜单和文本的字体: 公共 GUI 帮助。

不依赖于颜色

不需要分辨颜色就可以使用此产品中的任何功能。

与辅助技术的兼容性

DB2 工具界面支持 Java Accessibility API，它使您能够将屏幕阅读器和其它辅助技术与 DB2 产品配合使用。

文档的辅助功能

DB2 的相关文档是以 XHTML 1.0 格式提供的，它在大部分 Web 浏览器中是可查看的。XHTML 允许您根据浏览器中设置的显示首选项来查看文档。还允许您使用屏幕阅读器和其它辅助技术。

语法图是以点分十进制格式提供的。仅当使用屏幕阅读器访问联机文档时，此格式才可用。

相关概念:

- 『点分十进制语法图』（*基础结构主题（DB2 公共文件）*）

相关任务:

- 『键盘快捷方式和加速键: 公共 GUI 帮助』
- 『更改菜单和文本的字体: 公共 GUI 帮助』

文献目录

- *WebSphere: WebSphere Solution Bundles: Implementation and Integration Guide*, SG24-6550
- *MQSeries: Application Messaging Interface*, SC34-5604-07
- *Information Integrator: Planning, Installation and Configuration Guide*
- *《Information Integrator: 联合系统指南》*
- *《IBM DB2 通用数据库: Life Sciences Data Connect 规划、安装和配置指南》*, GC27-1235
- *IBM DB2 Universal Database: XML Extender Administration and Programming*, SC27-1234
- *《IBM DB2 通用数据库: 数据仓库中心管理指南》*, S152-0188
- *IBM DB2 Universal Database: Replication Guide and Reference*, SC27-1121
- *IBM Enterprise Information Portal for Multiplatforms: Planning and Installing Enterprise Information Portal*, GC27-0873
- *IBM Enterprise Information Portal for Multiplatforms: Managing Enterprise Information Portal*, SC27-0875
- *IMS: Administration Guide: Database Manager*, SC26-9419-02
- *IBM WebSphere: Application Server V4 for z/OS and OS/390: Installation and Customization*, GA22-7834-05
- *IBM WebSphere: Application Server V4.0.1 for z/OS and OS/390: System Management Scripting API*, SA22-7839
- *《DB2 XML Extender: 管理与编程》*
- *DB2 XML Extender: DB2 XML Extender Administration and Programming, Version 7.2 Release Notes*
- *Using WSDL in a UDDI Registry 1.07*
- *WebSphere Handbook*
- *Web Services Description Language (WSDL) 1.1*
- *Dynamic e-business: The next stage of e-business and Web services*
- *Web services zone*

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 代理咨询。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。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 106-0032, 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 可以随时对本资料中描述的产品和／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 可以它认为合适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:

IBM Corporation
J46A/G4
555 Bailey Avenue

San Jose, CA 95141-1003
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。

本资料中可能包含用于日常业务运作的数据和报告的示例。为了尽可能完整地说明问题，这些示例可能包含个人、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雷同，纯属巧合。

版权许可证:

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:

© (*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。
© Copyright IBM Corp. *(输入年份)*. All rights reserved.

商标

下列各项是国际商业机器公司在美国和／或其他国家或地区的商标:

IBM
AIX
DB2
DB2 Extenders
DB2 Universal Database

Domino
Informix
Intelligent Miner
Lotus
OS/390
UNIX
WebSphere
z/OS

下列各项是其他公司的商标或注册商标:

Java 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标或注册商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其他国家或地区的商标。

Intel、Intel Inside (logos)、MMX 和 Pentium 是 Intel Corporation 在美国和／或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

[Z]

A

C

D

E

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

Z

与 IBM 联系

在中国，请致电下列其中一个号码以与 IBM 联系:

- 800-810-1818 或 (010) 84981188 分机 5151，可获得售前客户服务;
- 800-810-1818 或 (010) 84981188 分机 5200，可获得售后客户服务;
- 800-810-1818 或 (010) 84981188 分机 5017，可获得市场营销与销售的信息;

要查找您所在国家或地区的 IBM 营业处，可在网上查看 IBM 全球联系人目录（Directory of Worldwide Contacts），网址为: www.ibm.com/planetwide。

产品信息

关于 DB2 Information Integrator 的信息可通过万维网获取，网址为: http://www-900.ibm.com/cn/software/db2/。

此站点包含有关 DB2 产品家族、DB2 解决方案、技术前沿与趋势、DB2 服务、成功案例、市场活动、培训与认证、DB2 开发者园地、合作伙伴、下载中心、资料库、第三方分析报告、殊荣与奖项、DB2 新闻以及如何购买 DB2 的最新信息。

要查找您所在国家或地区的 IBM 营业处，可在网上查看 IBM 全球联系人目录（Directory of Worldwide Contacts），网址为: www.ibm.com/planetwide。

对文档的意见

您的反馈有助于 IBM 提供高质量的信息。请发送您对本书或其它 DB2 Information Integrator 文档的任何意见。可以使用下列任何一种方法提出意见:

- 使用 www.ibm.com/software/data/rcf 上的联机读者意见表发送您的意见。
- 通过电子邮件（e-mail）将您的意见发送至 ctscrcf@cn.ibm.com。确保包括产品的名称、产品的版本号和书籍的名称及部件号（如果适用的话）。如果您对特定文本有意见，请包括此文本的位置（例如，标题、表号或页码）。

中国印刷

S152-0601-01

Spine information:

IBM DB2 Information Integrator

应用程序开发者指南

版本 8.2