Guía del desarrollador de servicios web

VERSIÓN 8

Borland®
JBuilder®

Borland Software CorporationN 8
Borland Software Corporation
100 Enterprise Way, Scotts Valley, CA 95066-3249

Impreso en EE.UU.

JBE0080WW21002websvcs 1E0R1002
0203040506-9 8 7 6 5 4 3 2 1
PDF

Índice de materias

Capítulo 1
Introducción 1-1
Convenciones de la documentación 1-4
Asistencia y recursos para desarrolladores . . . 1-6
El servicio de asistencia técnica de Borland . 1-6
Recursos en línea 1-7
World Wide Web 1-7
Grupos de noticias de Borland 1-7
Usenet, grupos de noticias 1-8
Información sobre errores 1-8

Capítulo 2
Introducción a los servicios web 2-1
Arquitectura de los servicios web 2-2
Estándares de servicios web 2-3
Simple Object Access Protocol (Protocolo de acceso a objetos sencillos, SOAP) 2-3
WSDL (Web Services Description Language, Lenguaje de descripción de servicios web) 2-4
UDDI (Universal Description, Discovery, and Integration, Descripción, detección e integración universales) 2-5
WSIL (Web Services Inspection Language, Lenguaje de inspección de servicios web) . 2-6
APIs de Java para llamadas a procedimientos remotos basadas en XML (JAX-RPC). . . . 2-6
JBuilder y los servicios web. 2-7
Consumo y creación de servicios web mediante asistentes de JBuilder. 2-8
Ejemplos de servicios web. 2-8
Servidores de aplicaciones empresariales admitidos 2-8

Capítulo 3
Configuración de proyectos para servicios web 3-1
Utilización del Asistente para configuración de servicios web 3-1
Asignación de un nombre para la WebApp 3-3
Selección de un EAR 3-3
Selección de un kit de herramientas de servicios web. 3-3
Kit de herramientas Axis de Apache . . . 3-4
Kit de herramientas WebLogic 3-4
Kit de herramientas SOAP 2 de Apache . 3-5
Definición de una configuración de ejecución para el servidor de servicios web 3-5
Elementos del nodo WebApp. 3-6
Cómo iniciar el servidor de servicios web 3-6
Configuración de las opciones de generación . . 3-7

Capítulo 4
Seguimiento de mensajes SOAP 4-1
Utilización del TCPMonitor. 4-2
Creación de monitores TCP/IP 4-3
Seguimiento de mensajes SOAP de un servicio 4-5
Utilización de la página de inicio de los servicios WebLogic 4-6

Capítulo 5
Utilización de WSDL 5-1
Términos WSDL. 5-2
Ejemplos de WSDL 5-3

Capítulo 6
Desarrollo de EJB como servicios web 6-1

Capítulo 7
El kit de herramientas Axis de Apache 7-1
Exportación de una clase como servicio web . . 7-2
Importación de archivos WSDL 7-6
Exportación de EJB como servicios web 7-10
Importación de servicios como aplicaciones EJB 7-12
Importación de servicios como aplicaciones EJB en el Explorador de servicios web 7-13
Distribución de servicios web. 7-14
Los archivos WSDD 7-15
deploy.wsdd 7-15
Edición de archivos WSDD para EJB . . 7-16
server-config.wsdd 7-18
Edición de server-config.wsdd 7-18

Capítulo 8
El kit de herramientas WebLogic 8-1
Exportación de una clase como servicio web . . 8-2
Exportación de varias clases como servicio web 8-6
Importación de un WSDL o un EAR como servicio web. 8-6
Importación de un EAR como servicio web . 8-6
Importación de un WSDL como servicio web 8-8
Comprobación de servicios distribuidos 8-9
Cómo escribir un cliente de prueba 8-11
Exportación de EJB como servicios web. 8-12
Distribución de servicios web 8-14
Los archivos WLDU 8-15
Edición de archivos WLDU para EJB 8-15
Modificación de las propiedades del nodo de distribución EJB. 8-16
Edición del WLDU y modificación del elemento <documentation> 8-17
web-services.xml 8-17
Distribución manual de servicios con web-services.xml 8-18
Configuración de los nombres de servicios por defecto 8-18

Capítulo 9
Utilización del kit de herramientas SOAP 2 de Apache 9-1
Exportación de una clase como servicio web . . 9-2
Importación de archivos WSDL 9-2

Capítulo 10
Modificar la configuración del servidor de aplicaciones empresariales para servicios web 10-1
Borland Enterprise Server 5.0.2-5.1.x 10-1
WebLogic Server 7.0 (con o sin SP1) 10-2
WebSphere Application Server 4.0 AES/AE. . .10-2

Capítulo 11
Búsqueda y publicación de servicios web 11-1
Descripción general del Explorador de servicios web 11-1
Aspectos generales de UDDI 11-3
Términos y definiciones UDDI 11-4
Aspectos generales de Axis 11-5
Aspectos generales de WSIL 11-6
Adición y eliminación de nodos en el árbol del Explorador. 11-6
Búsqueda en un registro UDDI 11-8
Búsqueda de negocios 11-8
Búsqueda por nombre. 11-8
Búsqueda por categoría 11-10
Búsqueda por identificador. 11-12
Búsqueda de servicios 11-13
Búsqueda de tModels 11-14
Búsqueda por nombre. 11-14
Examen de los resultados de las consultas UDDI 11-15
Fichas de detalles de UDDI. 11-15
Ficha Detalles. 11-16
Ficha Detalles del negocio 11-16
Ficha Detalles del servicio 11-16
Ficha Detalles de enlace 11-16
Ficha Detalles de la instancia de tModel. 11-16
Ficha Detalles de tModel 11-16
Búsqueda de servicios web en un servidor Axis 11-17
Presentación de los servicios 11-17
Importación de WSDL y difusión de servicios web Axis. 11-18
Acceso remoto a los servidores Axis . . . 11-19
Búsqueda de servicios web con documentos WSIL 11-20
Nodo Servicios 11-21
Nodo Enlaces 11-21
Ejecución de una búsqueda con un documento WSIL 11-23
Difusión de servicios web en un registro UDDI 11-24
Registro en el sitio UDDI mediante un navegador 11-24
Creación y distribución de un servicio web 11-24
Difusión de negocios y servicios 11-24
Publicación de tModels 11-26
Difusión de servicios web desde un servidor Axis 11-26
Creación y distribución de servicios web albergados en un servidor Axis 11-27
Apertura de servicios web albergados por Axis. 11-27
Publicación desde el servidor Axis 11-27
Seguimiento de mensajes UDDI. 11-28

Generación de clases Java a partir de documentos WSDL 11-29

Capítulo 12
Tutoriales de servicios web 12-1
Tutoriales de servicios web Axis 12-1
Tutoriales para servicios web de WebLogic . .12-2
Tutoriales generales para servicios web 12-2

Capítulo 13
Tutorial: Creación de un servicio web sencillo con Axis 13-1
Paso 1: Creación de un JavaBean de ejemplo . .13-2
Paso 2: Exportación del bean de ejemplo como servicio web y configuración del proyecto para servicios web 13-2
Paso 3: Distribución, ejecución y comprobación del servicio web 13-5

Capítulo 14
Tutorial: Creación de un servicio web desde un documento WSDL 14-1
Paso 1: Configuración del proyecto para servicios web 14-2
Paso 2: Importación del documento WSDL. . .14-3
Paso 3: Examen de los descriptores de distribución 14-4
Paso 4: Implementación del servicio. 14-5
Paso 5: Creación de la aplicación web pública .14-6
Paso 6: Creación de una JSP que llama al servicio web. 14-6
Paso 7: Implementación del bean 14-9
Paso 8: Llamada al servicio web y seguimiento de mensajes SOAP 14-11

Capítulo 15
Tutorial: Crear un servicio web desde una aplicación EJB con Borland Enterprise Server 15-1
Paso 1: Configuración del proyecto de ejemplo 15-2
Paso 2: Crear un servidor de servicios web y distribuir en el servidor de aplicaciones. . . .15-2
Paso 3: Generar de código para el cliente y el servidor a partir del documento WSDL .15-5
Paso 4: Comprobación del funcionamiento del servicio 15-6
Paso 5: Escritura del código del cliente y consumo del servicio 15-7

Capítulo 16
Tutorial: Importación de servicios web como aplicaciones EJB 16-1
Paso 1: Adición de un documento WSDL al proyecto 16-2
Paso 2: Creación de una aplicación EJB a partir del documento WSDL. 16-2
Paso 3: Comprobación del funcionamiento de la aplicación EJB 16-3

Capítulo 17
Tutorial: Creación de un servicio web sencillo con WebLogic 17-1
Paso 1: Creación de un JavaBean de ejemplo . 17-2
Paso 2: Exportación del bean de ejemplo como servicio web 17-3
Paso 3: Ejecución del servidor y distribución del servicio. 17-5
Paso 4: Comprobación del servicio distribuido 17-5
Paso 5: Escritura del código de un cliente para probar el funcionamiento del servicio 17-6

Capítulo 18
Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server 18-1
Paso 1: Configuración del proyecto. 18-2
Paso 2: Configuración del proyecto para servicios web 18-2
Paso 3: Distribución de un EJB como servicio web 18-3
Paso 4: Generación de código para el cliente a partir del documento WSDL. 18-4
Paso 5: Escritura del código del cliente y consumo del servicio de forma local 18-5

Capítulo 19
Tutorial: Búsqueda de servicios web UDDI 19-1
Paso 1: Examen de servicios web en el sitio Xmethods 19-2
Paso 2: Examen de tModels 19-5

Paso 3: Búsqueda de empresas de publicación de software en el UDDI de Microsoft 19-6
Paso 4: generación de clases Java. 19-8

Tutoriales

Creación de un servicio web sencillo con Axis . 13-1
Creación de un servicio web a partir de un documento WSDL, Axis 14-1
Creación de un servicio web a partir de una aplicación EJB, Axis 15-1
Importar servicios web como aplicaciones EJB, Axis. 16-1
Creación de un servicio web sencillo con WebLogic 17-1
Creación de un servicio web a partir de una aplicación EJB mediante el servidor WebLogic Server 18-1
Búsqueda de servicios web UDDI. 19-1

Capítulo

1

Introducción

Es una función de JBuilder Enterprise.

La *Guía del desarrollador de servicios web* explica cómo utilizar las funciones de servicios web de JBuilder para crear, buscar, consumir y publicar servicios web.

La *Guía del desarrollador de XML* contiene los siguientes capítulos:

- Capítulo 2, "Introducción a los servicios web"

  Proporciona una descripción general de los servicios web y de las funciones relacionadas con ellos disponibles en JBuilder.

- Capítulo 3, "Configuración de proyectos para servicios web"

  Explica cómo crear un servidor de servicios web para hospedar un servicio web mediante el Asistente para configuración de servicios web y cómo ejecutar el servidor de servicios web en el IDE de JBuilder.

- Capítulo 4, "Seguimiento de mensajes SOAP"

  Describe cómo monitorizar los mensajes SOAP a través de las herramientas proporcionadas por el kit de herramientas de servicios web.

- Capítulo 5, "Utilización de WSDL"

  Proporciona una introducción general al Lenguaje de descripción de servicios web (WSDL) y explica cómo se utiliza en los servicios web.

- Capítulo 6, "Desarrollo de EJB como servicios web"

  Proporciona una descripción general de cómo JBuilder desarrolla Enterprise JavaBeans como servicios web.

- Capítulo 7, "El kit de herramientas Axis de Apache"

Describe cómo utilizar el kit de herramientas Axis de Apache para desarrollar servicios web. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache.

- Capítulo 8, "El kit de herramientas WebLogic"

  Describe cómo utilizar el kit de herramientas WebLogic para desarrollar servicios web.

- Capítulo 9, "Utilización del kit de herramientas SOAP 2 de Apache"

  Describe cómo utilizar el kit de herramientas SOAP 2 de Apache para desarrollar servicios web.

- Capítulo 10, "Modificar la configuración del servidor de aplicaciones empresariales para servicios web"

  Describe los servidores de aplicaciones compatibles con JBuilder en cuanto a servicios web y proporciona información sobre configuraciones personalizadas para diversos servidores.

- Capítulo 11, "Búsqueda y publicación de servicios web"

  Explica cómo utilizar el Explorador de servicios web para buscar y publicar servicios web.

  - Ejemplos de servicios web

    En los siguientes directorios de JBuilder se pueden encontrar ejemplos de servicios web:

    - `samples/webservices/axis`
    - `samples/webservices/weblogic`
    - `thirdparty/apache-soap/samples`
    - `thirdparty/xml-axis/java/samples`

- Tutoriales de Axis:

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

  - Capítulo 13, "Tutorial: Creación de un servicio web sencillo con Axis"

    Explica cómo utilizar el asistente Exportar como servicio web con el fin de exportar un JavaBean como servicio web, mediante el kit de herramientas Axis, exponiendo determinados métodos seleccionados ante el consumidor del servicio web.

  - Capítulo 14, "Tutorial: Creación de un servicio web desde un documento WSDL"

    Explica cómo utilizar el asistente Importar un servicio web con el fin de generar clases Java para un servicio web mediante el kit de herramientas de Axis, y, a continuación, implementar las clases para proporcionar acceso al servicio de traducción BabelFish de AltaVista.

    - Capítulo 15, “Tutorial: Crear un servicio web desde una aplicación EJB con Borland Enterprise Server”

      Explica cómo crear un servicio web a partir de una aplicación Enterprise JavaBean mediante Borland Enterprise Server.

    - Capítulo 16, “Tutorial: Importación de servicios web como aplicaciones EJB”

      Describe cómo importar un documento WSDL como aplicación EJB mediante el kit de herramientas Axis y Borland Enterprise Server.

- Tutoriales de WebLogic:

    - Capítulo 17, “Tutorial: Creación de un servicio web sencillo con WebLogic”

      Explica cómo utilizar el asistente Exportar como servicio web con el fin de publicar un JavaBean como servicio web y exponer determinados métodos seleccionados ante el consumidor del servicio web.

    - Capítulo 18, “Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server”

      Explica cómo crear un servicio web a partir de una aplicación EJB mediante el servidor WebLogic.

- Tutoriales generales:

    - Capítulo 19, “Tutorial: Búsqueda de servicios web UDDI”

      Explica cómo utilizar el Explorador de servicios web para buscar servicios web y generar clases Java a partir de un documento WSDL.

Convenciones de la documentación

En la documentación de Borland para JBuilder, el texto con significado especial se identifica mediante la tipografía y los símbolos descritos en la siguiente tabla.

Tabla 1.1 Convenciones tipográficas y de símbolos

Tipo de letra	Significado
`Letra monoespaciada`	El tipo monoespaciado representa lo siguiente: • texto tal y como aparece en la pantalla • cualquier cosa que debe escribir, como "Escriba `Hola a todos` en el campo Título del Asistente para aplicaciones". • nombres de archivos • nombres de vías de acceso • nombres de directorios y carpetas • comandos, como `SET PATH`. • código Java • tipos de datos de Java, como `boolean`, `int` y `long`. • identificadores de Java, como nombres de variables, clases, nombres de paquetes, interfaces, componentes, propiedades, métodos y sucesos. • nombres de argumentos • nombres de campos • palabras clave de Java , como `void` y `static`.
Negrita	La negrita se utiliza para las herramientas java, bmj (Borland Make for Java), bcj (Borland Compiler for Java) y opciones del compilador. Por ejemplo: **javac**, **bmj**, **-vía de acceso a clases**.
Cursiva	Las palabras en cursiva indican los términos nuevos que se definen y los títulos de libros; ocasionalmente se usan para indicar énfasis.
Nombres de tecla	Este tipo de letra indica una tecla, como "Pulse *Esc* para salir de un menú".
[ ]	Los corchetes, en las listas de texto o sintaxis, encierran elementos optativos. En estos casos no se deben escribir los corchetes.

Tabla 1.1 Convenciones tipográficas y de símbolos (continuación)

Tipo de letra	Significado
< >	Los corchetes angulares se utilizan para indicar las variables en vías de acceso a directorios, opciones de comandos y ejemplos de código. Por ejemplo, `<nombredearchivo>` se puede utilizar para indicar que es necesario especificar un nombre de archivo (incluida su extensión); <nombredeusuario> indica que se debe escribir un nombre de usuario. Cuando reemplace las variables en vías de acceso a directorios, opciones de comandos y ejemplos de código, sustituya la variable entera, incluidos los corchetes angulares (< >). Por ejemplo, sustituya `<nombredearchivo>` por el nombre de un archivo, como por ejemplo `empleado.jds`, y quite los corchetes angulares. **Nota:** Los archivos HTML, XML, JSP y de otros formatos basados en etiquetas utilizan también corchetes angulares para delimitar elementos del documento, como por ejemplo: <font color=red> y <ejb-jar>. La siguiente convención describe la forma de especificar cadenas de variables dentro de los ejemplos de código en cuya sintaxis se utilizan corchetes angulares como delimitadores.
Cursiva, serif	Este formato de texto se utiliza para indicar cadenas de variables dentro de los ejemplos de código que ya utilizan corchetes angulares como delimitadores. Por ejemplo, `<url="jdbc:borland:`*jbuilder*`\\samples\\guestbook.jds">`
...	En los ejemplos de código, los puntos suspensivos (...) indican código que se ha omitido para ahorrar espacio y mejorar la claridad. Si están en un botón, los puntos suspensivos indican que éste conduce a un cuadro de diálogo de selección.

JBuilder se puede utilizar con diversas plataformas. En la tabla siguiente se proporciona una descripción de las convenciones de plataformas utilizadas en la documentación.

Tabla 1.2 Convenciones de las plataformas

Elementos	Significado
Vías de acceso	En las vías de acceso de la documentación se utiliza la barra normal (/). En la plataforma Windows se utiliza la barra invertida (\).

Tabla 1.2 Convenciones de las plataformas (continuación)

Elementos	Significado
Directorio de inicio	La ubicación del directorio inicial varía según la plataforma y se indica con la variable `<home>`. • En UNIX y Linux, el directorio inicial puede variar. Por ejemplo, puede ser `/user/<nombre de usuario>` o `/home/<nombre de usuario>` • En Windows NT, el directorio inicial es `C:\Winnt\Profiles\<nombre de usuario>` • En Windows 2000 y XP, el directorio inicial es `C:\Documents and Settings\<nombre de usuario>`
Imágenes de pantalla	Las imágenes o capturas de pantalla utilizan el aspecto Metal en diversas plataformas.

Asistencia y recursos para desarrolladores

Borland proporciona una serie de opciones de asistencia y recursos de información para ayudar a los desarrolladores a obtener el máximo rendimiento de los productos Borland. Entre estas opciones se incluye una gama de programas de asistencia técnica de Borland, así como servicios gratuitos en Internet, los cuales permiten consultar una amplia base de información y ponerse en contacto con otros usuarios de productos Borland.

El servicio de asistencia técnica de Borland

Borland ofrece varios programas de asistencia para clientes actuales y potenciales. Se puede elegir entre varios tipos de asistencia, que van desde la ayuda en la instalación de los productos Borland hasta el asesoramiento de expertos y la asistencia pormenorizada.

Si desea más información sobre el servicio al desarrollador de Borland, visite nuestra página Web, en `http://www.borland.com/devsupport`.

Cuando se ponga en contacto con el servicio técnico tenga a mano la información completa sobre el entorno, la versión del producto utilizada y una descripción detallada del problema.

Si necesita más información sobre las herramientas o la documentación de otros proveedores, póngase en contacto con ellos.

Recursos en línea

También puede obtener información de los siguientes recursos en línea:

World Wide Web	http://www.borland.com/
FTP	ftp://ftp.borland.com/ Documentación técnica disponible por ftp anónimo.
Listserv	Para suscribirse a las circulares electrónicas, rellene el formulario en línea que aparece en: http://info.borland.com/contact/listserv.html y para el servidor de listas internacional Borland: http://info.borland.com/contact/intlist.html

World Wide Web

Visite periódicamente www.borland.com/jbuilder. El equipo de desarrollo de productos Java publica en esta página documentación técnica, análisis de competitividad, respuestas a preguntas frecuentes, aplicaciones de ejemplo, software actualizado e información sobre productos nuevos y antiguos.

En particular, pueden resultar interesantes las siguientes direcciones:

- http://www.borland.com/jbuilder/ (actualizaciones de software y otros archivos)
- http://www.borland.com/techpubs/jbuilder/(actualizaciones de documentación y otros archivos)
- http://community.borland.com/ (contiene nuestra revista de noticias para desarrolladores en formato web)

Grupos de noticias de Borland

Puede registrar JBuilder y participar en los grupos de debate sobre JBuilder, estructurados en hilos. Los grupos de noticias de Borland ofrecen un medio para que la comunidad internacional de clientes de Borland pueda intercambiar sugerencias y técnicas sobre los productos de Borland, así como las herramientas y tecnologías relacionadas.

Puede encontrar grupos de noticias, moderados por los usuarios, sobre JBuilder y otros productos de Borland, en http://www.borland.com/newsgroups.

Usenet, grupos de noticias

En Usenet existen los siguientes grupos dedicados a Java y temas relacionados:

- news:comp.lang.java.advocacy
- news:comp.lang.java.announce
- news:comp.lang.java.beans
- news:comp.lang.java.databases
- news:comp.lang.java.gui
- news:comp.lang.java.help
- news:comp.lang.java.machine
- news:comp.lang.java.programmer
- news:comp.lang.java.security
- news:comp.lang.java.softwaretools

Nota Se trata de grupos moderados por usuarios; no son páginas oficiales de Borland.

Información sobre errores

Si encuentra algún error en el software, comuníquelo en la página Support Programs, en http://www.borland.com/devsupport/namerica/. Pulse el enlace "Reporting Defects" para llegar al formulario Entry.

Cuando informe sobre un fallo, incluya todos los pasos necesarios para llegar a él, así como toda la información posible sobre la configuración, el entorno y las aplicaciones que se estaban utilizando junto con JBuilder. Intente explicar con la mayor claridad posible las diferencias entre el comportamiento esperado y el obtenido.

Si desea enviar felicitaciones, sugerencias o quejas al equipo de documentación de JBuilder, envíe un mensaje a jpgpubs@borland.com. Envíe únicamente comentarios sobre la documentación. Tenga en cuenta que los asuntos relacionados con el servicio técnico se deben enviar al departamento de asistencia técnica para programadores.

JBuilder es una herramienta creada por desarrolladores y para desarrolladores. Valoramos sumamente sus aportaciones.

Capítulo

2

Introducción a los servicios web

Es una función de JBuilder Enterprise.

Los servicios web son módulos de software que realizan tareas o conjuntos de tareas discretas, a los que se accede y se llama a través de una red, sobre todo la World Wide Web. El desarrollador puede crear una aplicación cliente que llama a una serie de servicios web mediante llamadas a procedimientos remotos (RPC) o un servicio de mensajes en los que se proporciona un fragmento o la mayor parte de la lógica de la aplicación. Los servicios web publicados se describen de forma que los desarrolladores pueden localizarlos y determinar si se ajustan a sus necesidades.

Por ejemplo, una empresa puede proporcionar a sus clientes un servicio web por el cual se comprueba el inventario de productos antes de realizar un pedido. Otro ejemplo es el servicio de seguimiento de paquetes de la empresa de mensajería Federal Express, mediante el cual los clientes pueden examinar el estado de sus envíos.

Los servicios web utilizan SOAP (Simple Object Access Protocol) para la carga XML y utiliza un transporte del tipo HTTP para llevar los mensajes SOAP de un lado a otro. En realidad, los mensajes SOAP son los documentos XML que se envían entre el servicio web y la aplicación que efectúa la llamada.

Los servicios web se pueden escribir en cualquier lenguaje, y se ejecutan en todas las plataformas. Los clientes de servicios web también se pueden escribir en cualquier lenguaje, y se ejecutan en todas las plataformas. Así, por ejemplo, un cliente escrito en Delphi que se ejecuta en Windows puede llamar a un servicio web escrito en Java que se ejecuta en Linux.

Arquitectura de los servicios web

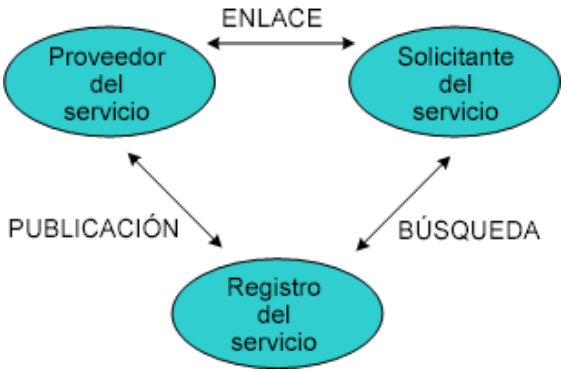
La arquitectura de los servicios web permite desarrollar servicios que encapsulan todos los niveles de funcionalidad empresarial. En otras palabras, un servicio web puede ser muy simple, como, por ejemplo, uno que informe sobre la temperatura actual, o una aplicación compleja. Esta arquitectura también permite la combinación de varios servicios web con el fin de crear nuevas funciones.

La arquitectura de servicios web tiene tres cometidos: proporciona datos, los solicita y hace de intermediario. Como proveedor, crea el servicio web y lo pone a disposición de los clientes que desean utilizarlo. Realizan la solicitud las aplicaciones clientes que consumen el servicio web. Este servicio solicitado también puede ser un cliente de otros servicios web. El intermediario, como un registro de servicio, permite interaccionar a la aplicación cliente y al proveedor.

Estos tres cometidos interaccionan por medio de las operaciones de publicación, búsqueda y enlace. El proveedor utiliza la interfaz de publicación del intermediario para comunicarle la existencia del servicio web, con el fin de ponerlo a disposición de los clientes. La información publicada describe el servicio e indica dónde se encuentra. La aplicación que hace la solicitud consulta al intermediario para que busque los servicios web publicados. Con la información obtenida del intermediario, la aplicación cliente puede enlazarse al servicio web (llamarlo).

En este diagrama se resume la forma en que interaccionan los tres cometidos.

Figura 2.1 Operaciones y cometidos de los servicios web

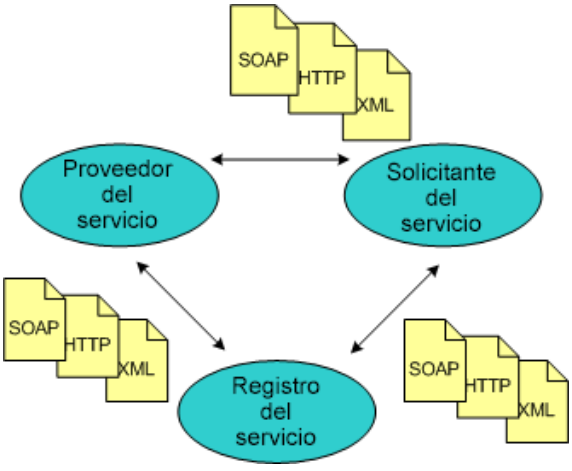


Estándares de servicios web

Las normas en las que se basa el desarrollo de servicios web son tecnologías en proceso de evolución. Las principales son SOAP (Simple Access Protocol, Protocolo de acceso a objetos sencillos), WSDL (Web Services Description Language, Lenguaje de descripción de servicios web), UDDI (Universal Description, Discovery, and Integration, Descripción, detección e integración universales) y WSIL (Web Services Inspection Language, Lenguaje de inspección de servicios web).

Simple Object Access Protocol (Protocolo de acceso a objetos sencillos, SOAP)

SOAP es un protocolo de mensajes independiente del transporte. Los mensajes SOAP son documentos XML. SOAP utiliza mensajes unidireccionales, aunque es posible combinar mensajes en secuencias de solicitud y respuesta. La especificación SOAP define el formato del mensaje XML, pero no su contenido ni la forma en que se envía. No obstante, SOAP especifica la forma en que los mensajes SOAP se encaminan por medio de HTTP.



Todos los documentos SOAP tienen un elemento raíz `<Envelope>`. El *elemento raíz* es el primero de un documento XML, y contiene los demás elementos del documento. Este "envelope" consta de dos partes: la cabecera (header) y el cuerpo (body). La cabecera contiene los datos de encaminado o contexto, y puede estar vacía. El cuerpo contiene el mensaje propiamente dicho. También puede estar vacío.

A continuación se proporciona un ejemplo de un mensaje SOAP sencillo enviado por HTTP que solicita el precio actual de las acciones de Borland:

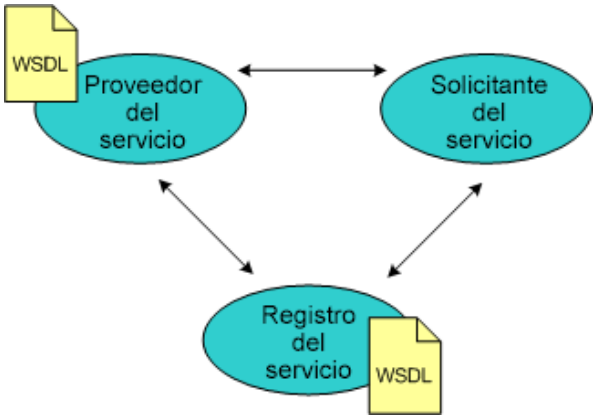
```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "urn:stock-quote-services"

<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>BORL</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Si desea más información sobre SOAP, empiece por consultar los documentos de SOAP de la página web del World Wide Consortium, en `http://www.w3.org/2002/ws/`. Visite también la página sobre SOAP de Apache, en `http://xml.apache.org/soap/`.

WSDL (Web Services Description Language, Lenguaje de descripción de servicios web)

Los servicios web sólo resultan útiles si otras aplicaciones pueden reconocer qué hacen y la forma de llamarlos. Los desarrolladores deben estar suficientemente informados sobre un servicio web para poder escribir un programa cliente que lo llame. WSDL es un lenguaje basado en XML que se utiliza para definir servicios web y describe la forma de acceder a ellos. Concretamente, describe los contratos de datos y mensajes que ofrece un servicio web. Los desarrolladores deben examinar el documento WSDL de los servicios web para averiguar qué métodos hay disponibles y cómo se realizan las llamadas con los parámetros adecuados.

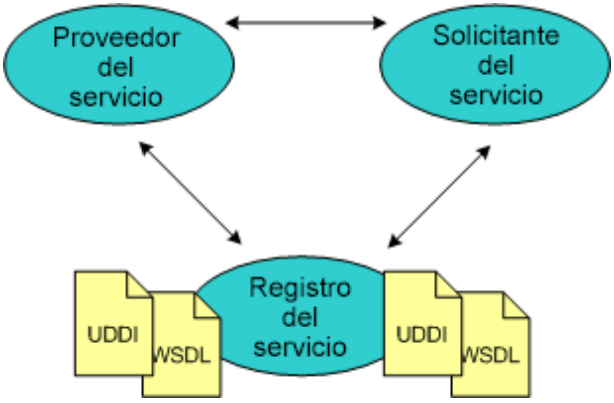


Si desea más información sobre WSDL, consulte la especificación Web Services Description Language 1.1, en `http://www.w3.org/TR/wsdl`, en la página web del World Wide Web Consortium.

UDDI (Universal Description, Discovery, and Integration, Descripción, detección e integración universales)

UDDI es una norma en proceso de evolución, que trata sobre la descripción, la publicación y la localización de los servicios web que ofrecen las empresas. Se trata de una especificación para el registro distribuido de información sobre servicios web. Cuando se ha desarrollado un servicio web y se ha creado un documento WSDL que lo describe, es necesario que exista una forma de proporcionar la información WSDL a los usuarios del servicio web. En cuando un servicio web se publica en un registro UDDI, los posibles usuarios pueden buscar e informarse de la existencia de los servicios web.

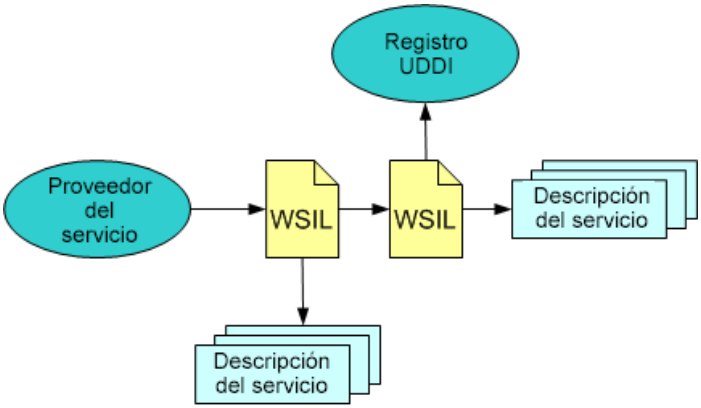
El contenido de un registro UDDI se puede comparar con las guías telefónicas. En las "páginas blancas" del registro figuran datos como el nombre, la dirección y el número de teléfono de la empresa que ofrece los servicios web. Las "páginas amarillas" identifican el tipo de negocio y lo clasifican por sectores de actividad. Las "páginas verdes" proporcionan datos sobre los servicios web que ofrece la empresa.



Si desea más información sobre UDDI, consulte la página web de UDDI.org, en `http://www.uddi.org`.

WSIL (Web Services Inspection Language, Lenguaje de inspección de servicios web)

WSIL, al igual que UDDI, proporciona un método de descubrimiento de servicios para servicios web. A diferencia de UDDI, WSIL utiliza un modelo descentralizado y distribuido, en lugar de un modelo centralizado. Los documentos WSIL son, básicamente, punteros de listas de servicios que permiten a los clientes de servicios web examinar los servicios disponibles. La norma WSIL establece la forma de utilizar documentos con formato XML estándar para inspeccionar sitios web en busca de servicios y series de reglas que determinan la forma en que se proporciona la información. Los documentos WSIL contienen varias referencias a documentos de descripción de servicios anteriores. El proveedor del servicio aloja el documento WSIL de forma que los consumidores pueden encontrar los servicios disponibles.



Si desea más información sobre la norma Web Services Inspection Language (WS-Inspection) 1.0, consulte `http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html`.

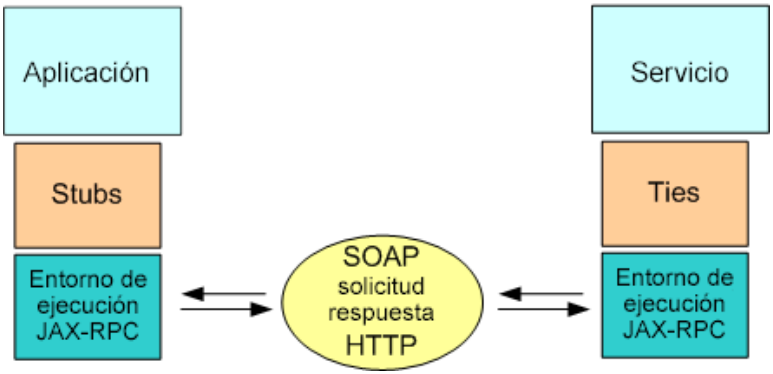
APIs de Java para llamadas a procedimientos remotos basadas en XML (JAX-RPC)

JAX-RPC define bibliotecas API de Java que los desarrolladores pueden utilizar en sus aplicaciones para desarrollar y consumir servicios web. Mediante JAX-RPC, un cliente Java puede consumir un servicio web que reside en un servidor remoto a través de Internet, incluso aunque el servicio esté escrito en otro idioma y se ejecute en una plataforma diferente. Los clientes que no sean de tipo-Java también pueden consumir servicios JAX-RPC.

JAX-RPC utiliza un protocolo de mensajería XML, tal como SOAP, para transmitir una llamada a procedimiento remoto a través de una red. Por

ejemplo, un servicio web que devuelve la cotización de unas acciones recibiría una solicitud HTTP SOAP con una llamada a un método realizada desde el cliente. Mediante JAX-RPC, el servicio extrae del mensaje SOAP la llamada al método, la traduce convenientemente y realiza esa llamada. A continuación, el servicio utiliza JAX-RPC para convertir la respuesta al método otra vez en un mensaje SOAP y enviar el mensaje de vuelta al cliente. El cliente recibe el mensaje SOAP y utiliza JAX-RPC para traducirlo en una respuesta.

El entorno de ejecución JAX-RPC genera stubs y ties, que son clases que permiten la comunicación entre el cliente y el servicio. Un stub, que reside en el cliente, es un objeto local que representa un servicio remoto y actúa como proxy para el servicio. Un objeto tie, que reside en el servidor, actúa como un proxy en el servidor.



Para obtener más información sobre JAX-RPC, consulte `http://java.sun.com/xml/jaxrpc/index.html` y el tutorial de JAX-RPC disponible en `http://java.sun.com/webservices/docs/1.0/tutorial/doc/JAXRPC.html`.

JBuilder y los servicios web

Aunque resulta útil informarse sobre las tecnologías que hacen posibles los servicios web, no es necesario estar familiarizado con ellas para crear los mensajes de SOAP y las descripciones de WSDL. JBuilder puede encargarse de esas tareas.

JBuilder utiliza los kits de herramientas Axis de Apache, WebLogic o SOAP 2 de Apache para ayudarle con SOAP. Asimismo, puede generar el documento WSDL para un servicio web ya creado. También puede tomar el WSDL ya creado de un servicio web y generar los archivos de clase Java o los EJB, con el fin de que el usuario pueda crear un cliente que llame al servicio web. Otra opción consiste en utilizar el código Java generado para implementar personalmente el servicio web. JBuilder también puede crear

rápidamente una aplicación web que hospede un servidor de servicios web.

JBuilder incluye el Explorador de servicios web, que permite buscar los servicios web deseados y publicar servicios web en un registro UDDI.

Si desea ver un ejemplo de cliente de página JavaServer que llama a un servicio web con el cual se traducen palabras de un idioma a otro, abra el proyecto `BasicWebService.jpx` del directorio `samples/webservices/axis` de JBuilder.

Consumo y creación de servicios web mediante asistentes de JBuilder

JBuilder proporciona asistentes para crear y consumir servicios web. El asistente Importar un servicio web genera clases Java a partir de un documento WSDL ya creado o un EAR. Después, se pueden implementar esas clases de modo que consuman el servicio especificado en el documento WSDL. También se pueden crear clases para un servicio web de servidor o implementar un servicio web como un Enterprise JavaBean (EJB) a partir del WSDL importado. El asistente Exportar como servicio web genera un documento WSDL a partir de una clase Java ya creada y expone los métodos seleccionados de la clase como un servicio web. Las funciones de servicios web disponibles varían dependiendo del kit de herramientas de servicios web.

Estos asistentes se pueden abrir desde la ficha Servicios web de la galería de objetos (Archivo | Nuevo). También se pueden utilizar desde los menús contextuales de los nodos apropiados del panel del proyecto.

Ejemplos de servicios web

En los siguientes directorios de JBuilder se pueden encontrar ejemplos de servicios web:

- `samples/webservices/axis`
- `samples/webservices/weblogic`
- `thirdparty/apache-soap/samples`
- `thirdparty/xml-axis/java/samples`

Servidores de aplicaciones empresariales admitidos

La funcionalidad de servicios web de JBuilder se puede utilizar con los siguientes servidores de aplicaciones empresariales:

- Borland Enterprise Server 5.0.2-5.1.x

- WebLogic Server 7.0 (con o sin SP1)
- WebSphere Application Server 4.0 AES/AE

Con JBuilder WebLogic Edition se suministran funciones de compatibilidad para Borland Enterprise Server y WebSphere Application Server.

Para obtener información sobre cómo utilizar estos servidores con la funcionalidad de servicios web de JBuilder, consulte el Capítulo 10, "Modificar la configuración del servidor de aplicaciones empresariales para servicios web".

Capítulo

3

Configuración de proyectos para servicios web

Es una función de JBuilder Enterprise.

Antes de que pueda trabajar con servicios web en JBuilder, tendrá que configurar el proyecto para servicios web. El Asistente para configuración de servicios web crea una configuración para hospedar un servicio web. Esto conlleva la creación de una webapp y su configuración mediante la biblioteca del kit de herramientas de servicio web seleccionada. En los servidores de aplicaciones, el asistente también crea un nodo eargrp y lo enlaza con la webapp. Además, el asistente crea una configuración de ejecución personalizada específica para el kit de herramientas de servidor, que refleja las selecciones del recopilatorio. La configuración es sensible a la configuración del servidor y permite seleccionar kits de herramientas que capacita a un servicio de servidor web.

Utilización del Asistente para configuración de servicios web

JBuilder incorpora el Asistente para configuración de servicios web para crear rápidamente una WebApp que hospede un servicio web con los archivos de distribución adecuados. Este asistente crea una implementación SOAP para el proyecto basándose en el kit de herramientas seleccionado, como Axis, SOAP de Apache o WebLogic. También crea una configuración de ejecución del servidor de servicios web que permite distribuir y ejecutar el servicio o una implementación del servicio. Cuando un servicio web se distribuye en un servidor de servicios web, el servicio web puede recibir y enviar mensajes SOAP a y desde aplicaciones cliente. Si desea obtener más información sobre SOAP, consulte el Capítulo 4, “Seguimiento de mensajes SOAP”.

El servicio web debe estar hospedado en una aplicación web. Si su proyecto no cuenta con una aplicación web ya creada, puede crearla desde el asistente Configuración de servicios web. Si desea obtener más información sobre las aplicaciones web, consulte "Las WebApps y los archivos WAR" en la *Guía del desarrollador de aplicaciones web* .

El Asistente para configuración de servicios web genera los siguientes nodos y archivos y los añade a su WebApp. El contenido de los nodos varía según el kit de herramientas seleccionado.

- Nodo Componentes del servicio web

  El nombre de este nodo depende del kit de herramientas seleccionado en el asistente. Este nodo cuenta con dos nodos dependientes:

  - El nodo Servicios basados en EJB, que contiene los archivos de distribución de los Enterprise JavaBean (EJB). Todos los beans sesión sin estado con métodos empresariales en la interfaz remota se distribuyen automáticamente. La información de distribución se genera en un archivo de distribución para cada módulo EJB. En el momento de la generación, las clases EJB de los archivos de distribución se distribuyen en el servidor.
  - El nodo Servicios basados en Java contiene los archivos de distribución de las clases Java.

- Nodo Descriptores de distribución

  El asistente añade la información necesaria SOAP al archivo `web.xml` que describe la distribución de la WebApp. Si se selecciona Axis como el kit de herramientas, el archivo de distribución final también se genera en este nodo.

- Nodo Directorio raíz

  El asistente añade el contenido a este nodo según el kit de herramientas seleccionado.

Para abrir el Asistente para configuración de servicios web:

1. Abra la galería de objetos seleccionando Archivo | Nuevo.
2. Haga clic en la pestaña Servicios web y doble clic en el icono Configuración de servicios web.
3. Seleccione un EAR en el proyecto, o bien, pulse Nuevo para crear un EAR. Esto sólo es necesario si se utiliza un servidor de aplicaciones empresariales que requiera un EAR.
4. Seleccione una WebApp en el proyecto para que sirva de host del servicio, o bien, pulse Nuevo para crear una nueva WebApp.
5. Seleccione un kit de herramientas para el proyecto.

6 Pulse Siguiente para ver la configuración de ejecución Servidor de servicios web creada para el proyecto.

7 Pulse Finalizar para configurar el proyecto para los servicios web.

Asignación de un nombre para la WebApp

El contexto web, que es el nombre de la WebApp y del directorio de la WebApp, es algo muy importante; constituye parte de la dirección para llamar al servicio web. Un ejemplo de dirección para llamar a un servicio web que se ejecuta en su servidor local sería `http://localhost:8080/<web context>/<serviceURI>`. El contexto web se utiliza en la dirección SOAP del documento WSDL. Además, el kit de herramientas Axis lo utiliza en la dirección del puerto del servicio en `<Nombre del servicio>Locator.java`.

Si desea obtener más información sobre WebApps, consulte "Las WebApps y los archivos WAR" en la *Guía del desarrollador de aplicaciones web* .

Selección de un EAR

Si se ha configurado el proyecto para que utilice un servidor de aplicaciones, el Asistente para configuración de servicios web le pedirá que seleccione o cree un EAR. El asistente configura el EAR automáticamente para que incluye el recopilatorio de la webapp y para que se actualice automáticamente durante la generación para que incluya cualquier JAR EJB al que se haya hecho referencia y que se utilize en el servicio we.

Selección de un kit de herramientas de servicios web

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache y SOAP 2 de Apache.

El campo Kit de herramientas del asistente Configuración de servicios web ofrece la posibilidad de utilizar todos los kits de herramientas de servicios web registrados. Existen varios kits de herramientas de servicios web que se incluyen con JBuilder: Axis de Apache, SOAP 2 de Apache y WebLogic. El campo Kit de herramientas puede contener plugins de kit de herramientas adicionales si están disponibles en su equipo.

La opción Copiar consola de administración a la webapp copia los archivos en la carpeta del directorio raíz, de forma que se puede utilizar la interfaz de usuario de los kits de herramientas para administrar el servidor de servicios web. Active esta opción si desea que estos archivos se añadan a la webapp. La opción Copiar consola de administración a la webapp se puede desactivar si se está utilizando un kit de herramientas que no cuente con una consola de administración. El kit de herramientas Axis de Apache y el SOAP 2 de Apache cuentan con una consola de administración, con lo que esta opción está activada para ambos.

Kit de herramientas Axis de Apache

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

El kit de herramientas Axis de Apache es una implementación de código abierto de SOAP, la última generación de Apache SOAP 2.0. Axis es una nueva versión de SOAP 2.0 de Apache que utiliza SAX en vez de DOM. Resulta más modular y flexible, y se trata de una implementación de SOAP de mayor rendimiento que SOAP 2.0 de Apache. El kit de herramientas Axis de Apache cumple con JAX-RPC (APIs de Java para llamadas a procedimientos remotos basadas en XML) y admite WSDL 1.1.

El kit de herramientas Axis de Apache genera archivos para administrar, enumerar los servicios distribuidos y validar. Si desea obtener información acerca de estos archivos, consulte la documentación del kit de herramientas.

Nota El kit de herramientas Axis de Apache combina cada archivo de distribución (`deploy.wsdd`) del proyecto en un `server-config.wsdd`. El kit de herramientas sobrescribe todas las modificaciones manuales realizadas en los archivos de distribución. Si modifica `server-config.wsdd`, desactive la opción Volver a generar distribución de la pestaña Servicios web de la ficha Generar de Propiedades de proyecto; si no lo hace, el kit de herramientas sobreescribe `server-config.wsdd` cuando se genera el proyecto. Consulte "Configuración de las opciones de generación" en la página 3-7 para obtener más información.

Consulte

- Capítulo 7, "El kit de herramientas Axis de Apache"
- Axis en `http://xml.apache.org/axis/`
- Documentación de Axis de Apache, en `<jbuilder>/thirdparty/xml-axis/java/docs/index.html`

Kit de herramientas WebLogic

WebLogic Server 7.x dispone de funciones integradas para servicios web. Esta opción se encuentra disponible cuando se establece WebLogic Server 7.x como servidor. WebLogic admite WSDL 1.1 y cumple con JAX-RPC.

Consulte

- Capítulo 8, "El kit de herramientas WebLogic"
- "Programming WebLogic Web Services" en `http://edocs.bea.com/wls/docs70/webserv/index.html`

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Soap 2 de Apache.

Kit de herramientas SOAP 2 de Apache

El kit de herramientas Apache SOAP 2 es una implementación de código abierto de SOAP 1.1, desarrollada por la comunidad Apache SOAP. Esta implementación de SOAP utiliza DOM.

Advertencia

Tenga en cuenta que SOAP 2 de Apache es una versión anterior de Axis por lo que **no** es compatible con JAX-RPC o WSDL. Debido a estas limitaciones, es recomendable que utilice Axis en su lugar. Si está utilizando el kit de herramientas SOAP 2 de Apache, es necesario que realice modificaciones en los asistentes de JBuilder y que lleve a cabo codificación adicional de forma manual. Para obtener más información, consulte el Capítulo 9, "Utilización del kit de herramientas SOAP 2 de Apache".

Consulte

- Apache SOAP en `http://xml.apache.org/soap/`
- Documentación de SOAP 2 de Apache en `<jbuilder>/thirdparty/apache-soap/docs/index.html`

Definición de una configuración de ejecución para el servidor de servicios web

En la última ficha del Asistente para configuración de servicios web, puede definir, si lo desea, una configuración de ejecución para el servidor de servicios web. El servidor de servicios web requiere un tipo de configuración de ejecución Servidor para que poder ejecutarse.

Si no existe un tipo de configuración de ejecución Servidor y prefiere no definir ninguno en esta ficha, se añade automáticamente una configuración de ejecución de nombre Servidor de servicios web. Esta configuración de ejecución utiliza la configuración por defecto Servidor del proyecto, definida en la ficha Servidor del cuadro de diálogo Propiedades de proyecto.

Para obtener más información sobre las configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.

Elementos del nodo WebApp

Después de crear el servidor de servicios web con la ayuda del asistente Configuración de servicios web, puede ampliar el nodo WebApp del panel del proyecto para ver los nodos dependientes que ha añadido el asistente. Puede observar los nodos de kit de herramientas, descriptores de distribución y directorio raíz. Si ha añadido al directorio raíz archivos para la administración de la aplicación web, se muestran cuando se amplía el nodo.

Si desea obtener más información sobre WebApps, consulte "Las WebApps y los archivos WAR" en la *Guía del desarrollador de aplicaciones web* .

Cómo iniciar el servidor de servicios web

Para ejecutar el servidor de servicios web, debe tener una configuración de ejecución Servidor que utilice el servidor web adecuado. El Asistente para configuración de servicios web añade la configuración de ejecución Servidor de servicios web al proyecto de forma automática. Para obtener más información sobre las configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.

Una vez que se ha completado el Asistente para configuración de servicios web, puede iniciar el servidor de servicios web mediante Ejecutar | Ejecutar el proyecto, cuando la configuración de ejecución Servidor de servicios web es la configuración por defecto, o pulsando sobre la pequeña flecha situada junto al icono Ejecutar de la barra de herramientas principal y seleccionando la configuración de ejecución Servidor de servicios web en la lista. Puede ver los progresos del servidor en el panel de mensajes. Si utiliza el kit de herramientas Axis, también puede hacer clic con el botón derecho del ratón sobre cualquier archivo servlet, JSP o HTML del panel del proyecto y seleccionar Ejecutar web utilizando Servidor de servicios web en el menú contextual.

Si antes ha activado la opción de copia de los archivos de consola de administración en el nodo webapp al crear el servidor de servicios web, ahora puede ampliar el nodo del directorio raíz, hacer clic con el botón derecho del ratón en el archivo `index.html` y elegir Ejecutar en web en el menú contextual para abrir la interfaz de administración del kit de herramientas Axis. El archivo de ejecución en web del kit de herramientas Soap 2 de Apache es `admin/index.html`. Si se ejecuta en web con la configuración por defecto del IDE cuando el servidor está ejecutándose, se accede al archivo desde el servidor web.

Configuración de las opciones de generación

La ficha Generar del cuadro de diálogo Propiedades de proyecto tiene una opción de generación que permite controlar la distribución durante la generación.

Para acceder a la opción de generación:

1. Elija Proyecto | Propiedades de proyecto y se abrirá el cuadro de diálogo homónimo.
2. Seleccione la pestaña Generar y, a continuación, la pestaña Servicios web.

La opción Volver a generar distribución está activada por defecto. Cuando esta opción está seleccionada, los archivos de distribución se generan a la vez que el proyecto. Si desea obtener más información acerca de esta opción, pulse el botón Ayuda de la ficha Servicios web de la ficha Generar.

Capítulo

4

Seguimiento de mensajes SOAP

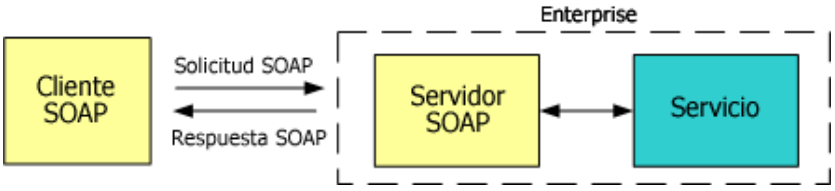
Es una función de JBuilder Enterprise.

Los servicios web utilizan SOAP (Simple Object Access Protocol), que proporciona un protolo de mensajes para que los clientes y servidores puedan intercambias mensajes. SOAP es un protocolo de mensajes independiente del transporte que permite el acceso a objetos remotos. SOAP utiliza un documento XML como protocolo de mensajes y una capa de transporte como HTTP. Los mensajes SOAP son unidireccionales, aunque es posible combinar mensajes en secuencias de solicitud y respuesta. La especificación SOAP define el formato del mensaje XML, pero no su contenido ni la forma en que se envía. No obstante, SOAP especifica la forma en que los mensajes SOAP se encaminan por medio de HTTP.

Todos los mensajes SOAP tienen un elemento raíz `<Envelope>`. Este "envelope" consta de dos partes: la cabecera (header) y el cuerpo (body). El elemento Envelope y el cuerpo son elementos necesarios en los mensajes SOAP, mientras que la cabecera es optativa. La cabecera contiene datos de encaminamiento o contexto, y puede estar vacía. El cuerpo, que de hecho contiene el mensaje, también puede estar vacío.

Los mensajes SOAP los envían y reciben clientes y servidores de servicios web SOAP. El cliente SOAP genera y envía solicitudes SOAP al servidor SOAP sobre HTTP. El servidor SOAP recibe estas solicitudes y genera las respuestas adecuadas para el cliente sobre HTTP.

Figura 4.1 Arquitectura SOAP



Algunos de los kit de herramientas de servicios web proporcionan herramientas para el seguimiento de mensajes SOAP entre el cliente y el servidor:

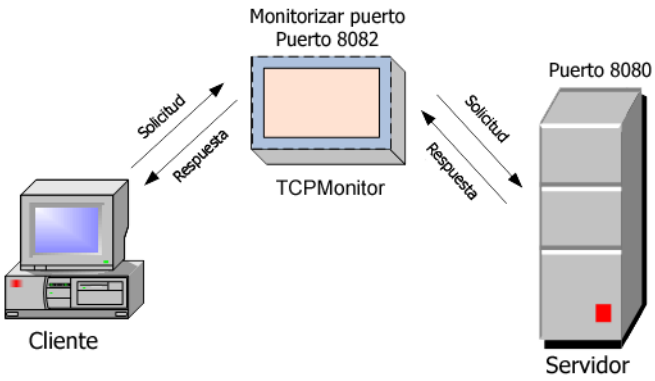
- TCPMonitor, una función del kit de herramientas Axis
- Página de inicio de los servicios WebLogic, una función del kit de herramientas WebLogic

Utilización del TCPMonitor

Esta es una función del kit de herramientas Axis. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

El TCPMonitor, al que se accede desde el kit de herramientas Axis, permite realizar un seguimiento de los sobres SOAP durante su transporte entre el cliente y el servidor. TCP (Transmission Control Protocol, Protocolo de control de transmisiones) es el protocolo de transporte por Internet más utilizado. El TCPMonitor se sitúa entre el cliente SOAP y el servidor. El cliente envía su solicitud al TCPMonitor; éste, a su vez, la reenvía al servidor. Las respuestas procedentes del servidor se envían al TCPMonitor y éste las reenvía al cliente. Estos mensajes se pueden seguir desde el equipo local, con el fin de comprobar un servicio o monitorizar una conexión.

Figura 4.2 Monitor JDBC



El TCPMonitor tiene las siguientes funciones:

- Adición de puertos de monitorización múltiples
- Modificación de mensajes SOAP
- Reenvío de mensajes SOAP
- Presentación de un historial de solicitudes y respuestas
- Guardado de solicitudes y respuestas en un archivo

El TCPMonitor se puede configurar de forma que monitorice los mensajes. Si se utiliza un cortafuegos es posible emplear un proxy. Cada kit de herramientas tiene distintos niveles de aceptación de servidores proxy y distintos usos. Si desea información sobre la configuración para el uso con un servidor proxy, consulte la documentación del kit de herramientas.

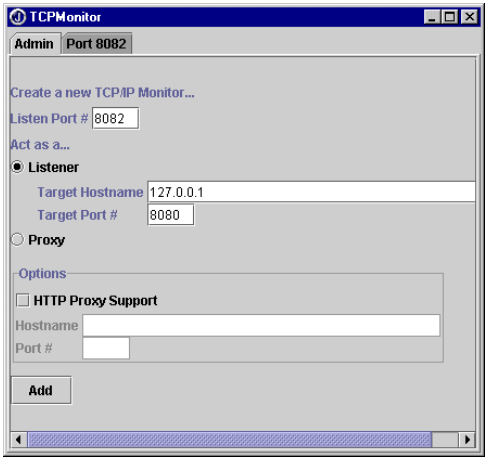
Creación de monitores TCP/IP

Por defecto, el TCPMonitor configura el puerto 8082 para el cliente y el puerto 8080 para el servidor.

Para crear un puerto de monitorización distinto:

1. Elija Herramientas | TCPMonitor para abrir el monitor.
2. Seleccione la pestaña Admin.
3. Introduzca un número de puerto de monitorización, como por ejemplo el 8082 para localhost. Debe ser un puerto que no esté utilizando el servicio.
4. Elija Listener.
5. Escriba el nombre del host de destino (Target Hostname) del que desee realizar el seguimiento, como 127.0.0.1 para localhost. Si escribe un nombre en lugar de una dirección, como services.xmethods.net, asegúrese de que elimina el protocolo de la dirección (`http://`).

6 Escriba el número de puerto de destino (Target Port) para el servidor, como 8080. Este es el puerto de destino del que desea realizar el seguimiento.



7 Pulse el botón Add (Añadir). Se añade un nuevo suceso.

8 Abra la nueva pestaña Port para realizar un seguimiento de los mensajes.

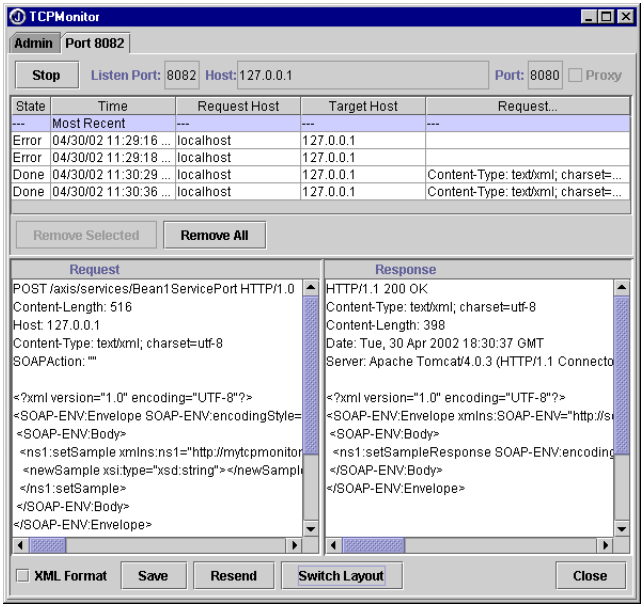
9 Modifique el cliente para enviar y recibir sus mensajes a y desde el TCPMonitor en el puerto de monitorización. Cambie la dirección y el número de puerto en el código, pero no elimine el contexto ni el destino del servicio de la dirección. Por ejemplo, el contexto y el destino del servicio de la dirección `"http://localhost:8082/soap/servlet/rpcrouter"` es `soap/servlet/rpcrouter` y no debe modificarse.

10 Vuelva a generar el cliente para guardar los cambios.

11 Ejecute la aplicación en el servidor web.

12 Interaccione con el servicio web y observe las solicitudes y respuestas SOAP del TCPMonitor. También puede realizar todas las modificaciones necesarias en las solicitudes del TCPMonitor y

reenviarlas, guardar las solicitudes y contestar a los mensajes del disco en formato XML o texto.



Seguimiento de mensajes SOAP de un servicio

En cualquier ejemplo de servicio web, se puede realizar un seguimiento de los mensajes SOAP con el TCPMonitor. En este ejemplo se crea un servicio web, se modifica la URL del cliente para que apunte al puerto de monitorización del TCPMonitor, se vuelve a generar el cliente, se ejecuta el test del servicio y se realiza un seguimiento de los mensajes reenviados entre el cliente y el servidor a través del TCPMonitor.

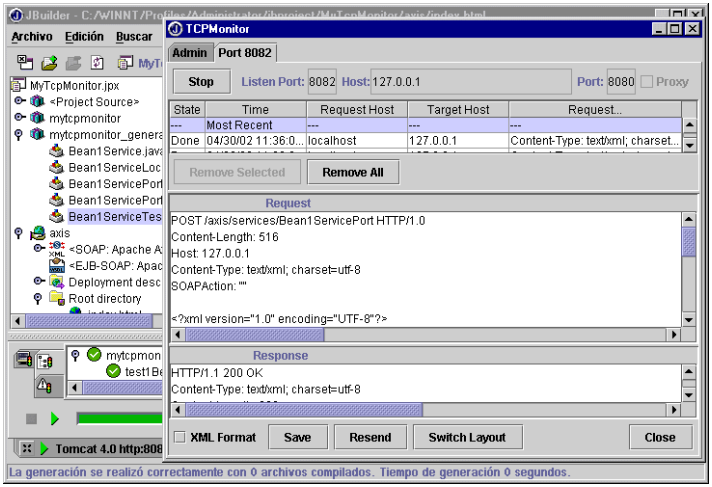
1. Cree un servicio web tal y como se describe en Capítulo 13, “Tutorial: Creación de un servicio web sencillo con Axis”.
2. Abra `Bean1ServiceLocator.java` en el paquete `<nombre del proyecto>.generated` y cambie la URL para la dirección del puerto de servicio de 8080 a 8082. El TCPMonitor está configurado por defecto para que monitorice los mensajes del puerto 8082. Es necesario que redirija al cliente para que envíe los mensajes al puerto de monitorización y el Monitor pueda así recibir los mensajes y enviarlos al servicio.
3. Elija Herramientas | TCPMonitor para abrir el monitor. Observe que el puerto de monitorización está configurado en 8082 y el puerto de destino en 8080. Esta es la configuración por defecto para el TCPMonitor.

Nota La configuración por defecto se puede modificar en cualquier momento. Pulse el botón Stop y modifique los campos Listen Port, Host y Port. También puede crear un nuevo TCPMonitor en la ficha Admin.

4 Amplíe el nodo Directorio raíz de Axis, pulse con el botón derecho del ratón sobre `index.html` y seleccione Ejecutar utilizando "Servidor de servicios web" para ejecutar el servidor de los servicios web.

5 Pulse con el botón derecho del ratón sobre el test, `Bean1ServiceTestCase.java`, y seleccione Ejecutar test utilizando valores por defecto.

6 Vuelva al TCPMonitor para ver la solicitud y respuesta que han enviado al Monitor el cliente y el servidor.



Consulte

- Capítulo 14, “Tutorial: Creación de un servicio web desde un documento WSDL”

Utilización de la página de inicio de los servicios WebLogic

Esta es una función de JBuilder Enterprise y del kit de herramientas WebLogic

Todas los servicios web que se distribuyen en el servidor WebLogic dispone de una página de inicio. Desde la página de inicio se pueden visualizar los métodos expuestos y las solicitudes y repuestas, comprobar cada operación, ver el documento WSDL que describe el servicio y copiar el código de ejemplo que llama al servicio.

Para abrir la página de inicio del servicio web, utilice la URL del servicio web:

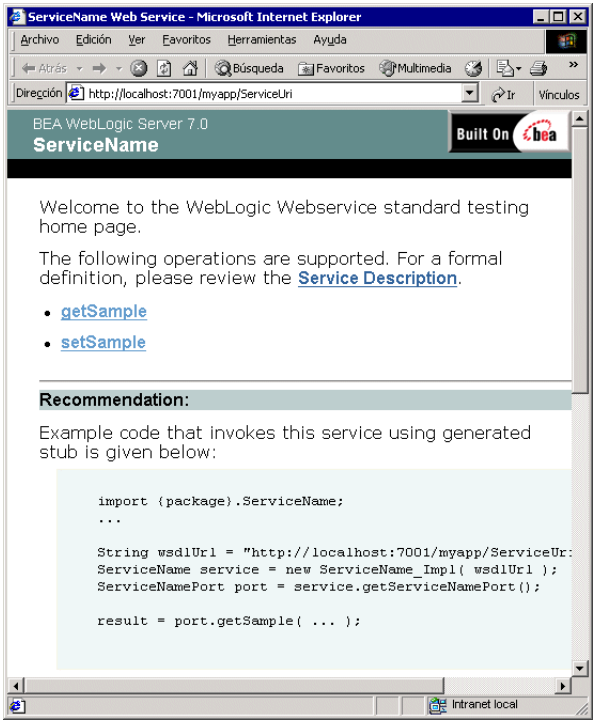
```
<protocol>://<host:port>/<contextURI>/<serviceURI>
```

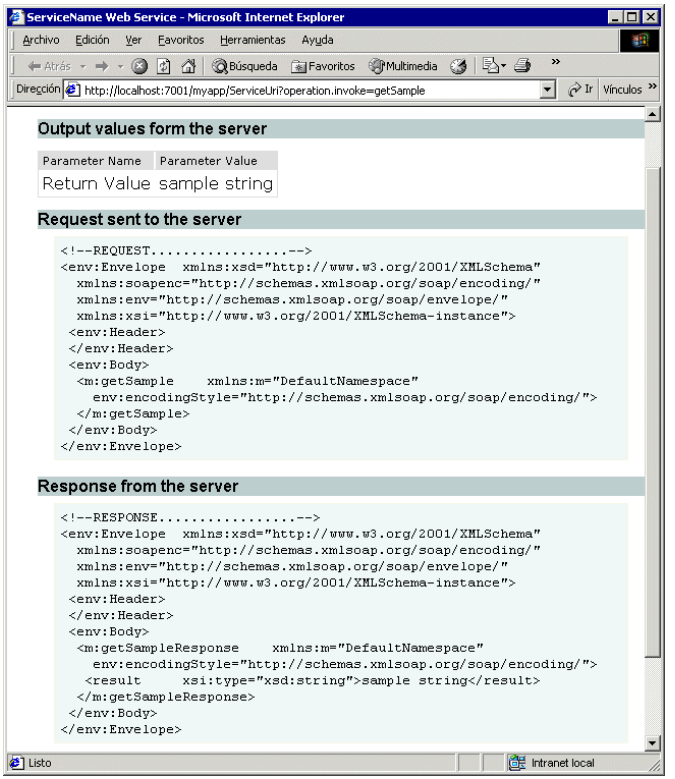
donde:

`<protocol>`	El protocolo del servicio, por ejemplo http.
`<host>`	El ordenador en el que se está ejecutando el servidor WebLogic, por ejemplo `localhost`.
`<port>`	El número de puerto en el que se encuentra el servidor WebLogic, por ejemplo `localhost:7001`.
`<contextURI>`	La raíz de contexto de la webapp o el nombre del archivo recopilatorio de la webapp, por ejemplo `web-services`.
`<serviceURI>`	El URI del servicio.

Por ejemplo, si se ejecuta WebLogic de forma local en el puerto por defecto 7001, la webapp es `web-services`, el ServiceUri es Bean1 y la dirección del servicio web será: `http://localhost:7001/web-services/Bean1`.

Si desconoce el ServiceURI, puede encontrarla en el archivo `servicegen.wldu` o haciendo clic con el botón derecho del ratón sobre el nodo Componentes de servicio web del nodo WebApp del panel del proyecto y seleccionado Propiedades.





Para abrir el documento WSDL del servicio web en la página de inicio, seleccione el enlace Descripción del servicio o escriba esta URL:

```
<protocol>://<host:port>/<contextURI>/<serviceURI>?WSDL
```

Consulte

- "Comprobación de servicios distribuidos" en la página 8-9
- "The WebLogic Web Services Home Page and WSDL URLs" en la documentación WebLogic en `http://edocs.bea.com/wls/docs70/webserv/client.html#1051033`

Capítulo

5

Utilización de WSDL

Es una función de JBuilder Enterprise.

Para crear una aplicación cliente que llame a un servicio web, el desarrollador necesita saber dónde se encuentra ubicado el servicio y cómo acceder a él. El Lenguaje de descripción de servicios web, WSDL, ofrece esta información. Un documento WSDL, escrito en XML, describe un servicio web y el proceso de llamarlo.

Los documentos WSDL indican la firma de método del servicio web, el protocolo que debe usar, su dirección en la red y su formato de datos. Con la información disponible en el documento WSDL, un programador puede escribir una aplicación que interaccione con un servicio web. Los desarrolladores deben examinar el documento WSDL de los servicios web para averiguar qué métodos hay disponibles y cómo se realizan las llamadas con los parámetros adecuados.

Los programadores que deseen detectar otros servicios web y sus correspondientes documentos WSDL para publicar servicios web pueden utilizar en el registro empresarial UDDI, donde las empresas registran sus servicios. JBuilder proporciona el Explorador de servicios web, que está disponible en el menú Herramientas, para descubrir y publicar los servicios web. Para obtener más información, consulte el Capítulo 11, "Búsqueda y publicación de servicios web".

Términos WSDL

Para poder comprender mejor los términos utilizados en los documentos WSDL, consulte la siguiente tabla. Si desea más información sobre WSDL, consulte la especificación Web Services Description Language 1.1, en `http://www.w3.org/TR/wsdl`, en la página web del World Wide Web Consortium.

Tabla 5.1 Términos WSDL

Término	Elemento WSDL	Definición
Endpoint		Un puerto. Consulte puerto.
Message	`<message name="GetQuoteRequest">`	Una definición abstracta y escrita de los datos (parámetro).
Escriba	`<part name="symbol" type="xsd:string"/>`	Un contenedor para las definiciones de los tipos de datos, como XSD, que es un atributo del elemento `<part>` de un mensaje.
Port Type	`<portType name="GetQuote">`	Un conjunto de operaciones abstractas admitidas por uno o más endpoints.
Binding	`<binding name="GetQuoteBinding" type="tns:GetQuote">`	Un protocolo y formato de datos para un tipo de puerto en concreto. Esta es la asociación entre la definición abstracta de WSDL y la implementación. SOAP es un ejemplo de tipo de asociación.
Operation	`<operation name="getQuote">`	Una definición abstracta de una acción del servicio (método).
Port	`<port name="GetQuote" binding="tns:GetQuoteBinding"> <soap:address location="http://localhost:8080/axis/servlet/AxisServlet"/>`	Una dirección para un enlace y un endpoint de comunicación definido como una combinación de un enlace y una dirección de red.
Service	`<service name="GetQuoteService">`	Un conjunto de endpoints relacionados (puertos).

```
<?xml version="1.0"?>
<definitions name="urn:GetQuote"
             targetNamespace="urn:xmltoday-delayed-quotes"
             xmlns:tns="urn:xmltoday-delayed-quotes"
             xmlns:xsd="http://www.w3.org/1999/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/" >
<!-- message declns -->
  <message name="GetQuoteRequest">
    <part name="symbol" type="xsd:string"/>
  </message>

  <message name="GetQuoteResponse">
```

```
    <part name="result" type="xsd:float"/>
  </message>
<!-- port type declns -->
  <portType name="GetQuote">
    <operation name="getQuote">
      <input  message="tns:GetQuoteRequest"/>
      <output message="tns:GetQuoteResponse"/>
    </operation>
  </portType>

  <!-- binding declns -->
  <binding name="GetQuoteBinding" type="tns:GetQuote">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getQuote">
      <soap:operation soapAction="getQuote"/>
      <input>
        <soap:body use="encoded"
                   namespace="urn:xmltoday-delayed-quotes"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
                   namespace="urn:xmltoday-delayed-quotes"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <!-- service decln -->
  <service name="GetQuoteService">
    <port name="GetQuote" binding="tns:GetQuoteBinding">
      <soap:address location="http://localhost:8080/axis/servlet/AxisServlet"/>
    </port>
  </service>
</definitions>
```

Ejemplos de WSDL

Si desea obtener ejemplos de archivos WSDL, consulte `<jbuilder>/samples/webservices/axis/wsdl`. Existen más ejemplos WSDL de SOAP de Apache disponibles en `<jbuilder>/thirdparty/apache-soap/samples`.

Consulte

- Capítulo 14, “Tutorial: Creación de un servicio web desde un documento WSDL”
- Capítulo 11, “Búsqueda y publicación de servicios web”

Capítulo

6

Desarrollo de EJB como servicios web

Es una función de JBuilder Enterprise.

Las funciones de servicios web de JBuilder están diseñadas para ayudarle en la creación de aplicaciones de servicios web de clase empresarial en la plataforma J2EE. Las soluciones son estándar, y se pueden trasladar a varios servidores de aplicaciones. La plataforma J2EE ha evolucionado con el tiempo y, actualmente, es la plataforma utilizada para el desarrollo Java empresarial. Los contenedores Enterprise JavaBean (EJB), que proporcionan una sólida plataforma para datos empresariales y de lógica empresarial, junto con los contenedores web, como los servlets y las páginas JSP, han ampliado las funciones de las aplicaciones, mejorando la penetración de los exploradores y de HTML.

Los servicios web amplían las funciones de la plataforma J2EE, ofreciendo soluciones multiplataforma e independientes del lenguaje. Además, la plataforma J2EE puede mejorar por sí misma los servicios web creados fuera de su dominio. Los servicios web no están limitados a un entorno de explorador, y se pueden integrar fácilmente en otras aplicaciones y servidores de aplicaciones.

Normalmente, como ocurre con los contenedores web, los métodos empresariales con mayor nivel de detalle son los candidatos adecuados con el diseño apropiado para exponer las funciones. JBuilder lo facilita exponiendo automáticamente los métodos adecuados en los beans sesión sin estado del proyecto. También se puede modificar este funcionamiento por defecto y seleccionar sólo los módulos EJB, beans y métodos que desee exponer como servicios web. Además de crear servicios web basados en EJB, JBuilder ofrece funciones para implementar rápidamente los servicios web ya creados como EJB.

Tal y como se describe en los apartados que tratan sobre los kits de herramientas, JBuilder exporta automáticamente los EJB como servicios web. Simplemente tiene que desarrollar su aplicación EJB como de costumbre, configurar el proyecto para los servicios web mediante el Asistente para la configuración de servicios web y ejecutarlo con la configuración de ejecución Servidor de servicios web que se ha creado con el asistente. Todos los beans sesión sin estado con métodos empresariales en la interfaz remota se distribuyen automáticamente al servidor como servicios web sin necesidad de realizar ningún paso adicional. Algunos servidores de aplicaciones puede que necesiten algún paso adicional para distribuir el EAR al servidor.

El desarrollo de EJB como servicios web varía según el kit de herramientas seleccionado. Si desea obtener más información sobre los EJB, consulte el Capítulo 7, "El kit de herramientas Axis de Apache", y Capítulo 8, "El kit de herramientas WebLogic".

Importante Si desea obtener información acerca de la configuración con servidores de aplicaciones, consulte el Capítulo 10, "Modificar la configuración del servidor de aplicaciones empresariales para servicios web", y "las Notas de esta versión" (Ayuda | Notas de esta versión).

Si desea conocer qué servidores de aplicaciones empresariales admite JBuilder, consulte la lista que aparece en "Servidores de aplicaciones empresariales admitidos" en la página 2-8.

Capítulo

7

El kit de herramientas Axis de Apache

Es una característica de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Axis de Apache es una implementación abierta del Protocolo de acceso a objetos sencillos, (Simple Object Access Protocol, SOAP), un protocolo basado en XML para el intercambio de información. Si selecciona Axis de Apache como el kit de herramientas de un asistente para servicios web de JBuilder, ese asistente utiliza Axis para generar WSDL, clases Java, archivos de distribución, archivos de administración, etc., dependiendo del asistente que se utilice.

Por ejemplo, si se selecciona Axis como el kit de herramientas en el asistente Importar un servicios web, el asistente genera clases Java a partir de un documento WSDL ya creado, al igual que los archivos de distribución. Después, se pueden implementar esas clases de modo que consuman el servicio especificado en el documento WSDL. También puede crear clases en el servidor para un servicio web. Si el kit de herramientas Axis se utiliza en el asistente Exportar como servicio web, el asistente genera un documento WSDL a partir de una clase Java ya creada y expone los métodos seleccionados de la clase como un servicio web.

Si desea obtener más información acerca del kit de herramientas Axis de Apache, consulte la documentación disponible en `<jbuilder>/thirdparty/xml-axis/java/docs`.

Exportación de una clase como servicio web

El asistente Exportar como servicio web se utiliza para exportar clases como servicios web. Dispone de opciones que permiten seleccionar qué métodos deben de ponerse a disposición del servicio. Este asistente se puede utilizar para exportar cualquier clase Java como un servicio web; por ejemplo, un JavaBean. Sin embargo, si la clase utiliza tipos ajenos al bean como parámetros o valores devueltos, tendrá que proporcionar serializadores y deserializadores. El asistente dispone de opciones para generar implementaciones de servidor y cliente del servicio. También genera un WSDL, que describe el servicio, e información de distribución para el kit de herramientas Axis.

Nota El proceso de exportación de clases difiere del de exportación de EJB. Las clases Java se deben exportar explícitamente mediante el asistente Exportar un servicio web. Sin embargo, los EJB se exportan automáticamente si el proyecto se ha configurado para servicios web mediante el Asistente para configuración de servicios web.

Si no se ha configurado el proyecto para que hospede el servicio web, aparece el Asistente para configuración de servicios web antes que el asistente Exportar un servicio web.

Según la configuración que seleccione, el asistente Exportar como servicio web genera algunos o todos los archivos siguientes de la clase o interfaz Java. Si no marca la opción Generar stub cliente, no se generan archivos Java, sólo los archivos WSDL y WSDD. Los ejemplos de nombres de archivos de la tabla siguiente proceden del ejemplo `AddressBook.wsdl` de `<jbuilder>/samples/webservices/axis/wsdl/AddressBook.wsdl`.

Tabla 7.1 Archivos generados por el asistente Exportar como servicio web

Nombre del archivo	Descripción	Elemento WSDL	Ejemplo
class name[PortType]`.wsdl`	El documento WSDL que describe el servicio web.		
service name`.java`	Una interfaz de servicio que define un método de obtención para cada uno de los puertos que figuran en el elemento `service` del WSDL. Esta interfaz de servicio define una clase de factoría para obtener una instancia stub.	`<service name= "AddressBookService">`	`AddressBookService.java`
service name`Locator.java`	Una clase de localizador que constituye la implementación del servidor cliente de la interfaz de servicio.	`<service name= "AddressBookService">`	`AddressBookServiceLocator.java`
service name`TestCase.java`	Un test JUnit optativo para comprobar el servicio web.	`<service name= "AddressBookService">`	`AddressBookServiceTestCase.java`

Tabla 7.1 Archivos generados por el asistente Exportar como servicio web (continuación)

Nombre del archivo	Descripción	Elemento WSDL	Ejemplo
portType name.java	Una interfaz para cada portType del WSDL. La implementación de esta interfaz se utiliza para llamar a los métodos remotos.	<portType name= "AddressBookPortType">	AddressBookPortType.java
*binding name*Impl.java	Una clase de implementación para cada interfaz portType. Este archivo se modifica para añadir su implementación.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingImpl.java
*binding name*Skeleton.java	Una clase de esqueleto optativa, que encapsula una implementación del servidor.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingSkeleton.java
*binding name*Stub.java	Una clase stub cliente (optativa) que actúa como proxy para un servicio web remoto. Esto permite llamar al servicio Web como si fuera un objeto local. Esta clase implementa la interfaz *class name[PortType]*.java.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingStub.java
tipos de datos	Archivos Java para todos los demás tipos y marcadores necesarios para el servicio web.		
deploy.wsdd	Un archivo XML que proporciona información de distribución en el servidor de los servicios web.		
server-config.wsdd	Un archivo XML que proporciona información de distribución en el servidor.		

El asistente genera un nombre de paquete basado en el nombre del paquete de la clase, añadiéndole .generated.

El asistente Exportar como servicio web puede generar de forma optativa clases del stub cliente. Si se selecciona la opción Generar stub cliente en la primera ficha del asistente, éste cuenta con pasos adicionales. Los pasos adicionales proporcionan las mismas funciones que el asistente Importar un servicio web y generan un stub cliente a partir de un documento WSDL.

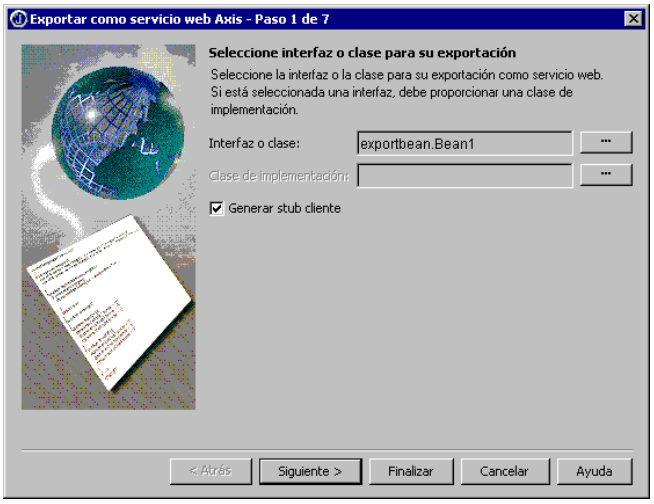
El asistente Exportar como servicio web está disponible en la pestaña Servicios web de la galería de objetos. También puede pulsar con el botón derecho del ratón en cualquier archivo .java del panel del proyecto y seleccionar Exportar como servicio web en el menú contextual.

Para exportar una clase como un servicio web y generar clases Java y un archivo WSDL para el servicio, siga estos pasos básicos:

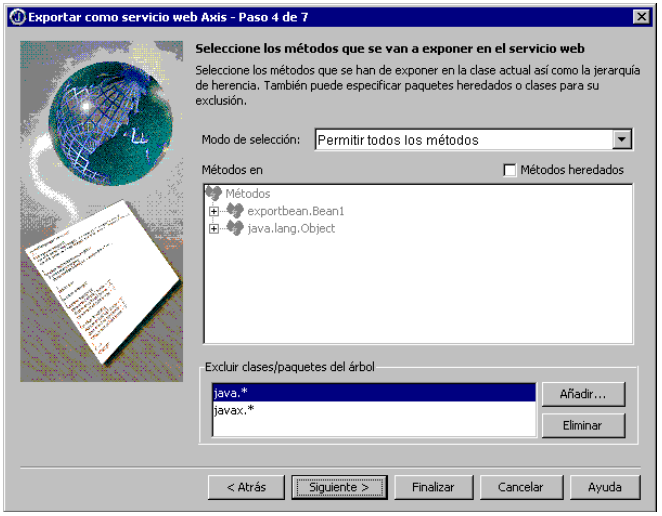
1. Cree un proyecto mediante Archivo | Proyecto nuevo.
2. Cree un JavaBean (Archivo | Nuevo | General | JavaBean), seleccione `java.lang.Object` como clase base y active la opción Generar propiedad de ejemplo.
3. Compile el proyecto (Proyecto | Ejecutar Make del proyecto).
4. Pulse el bean con el botón derecho del ratón y seleccione Exportar como servicio web para que se abra el asistente Exportar como servicio web.

   Debido a que todavía no se ha configurado el proyecto para sercivios web, aparece el Asistente para configuración de servicios web antes que el asistente Exportar un servicio web.
5. Cree una implementación SOAP con el kit de herramientas Axis para que hospede el servicio web tal y como se describe en "Utilización del Asistente para configuración de servicios web" en la página 3-1.

   Después de haber configurado el proyecto para servicios web, aparecerá el asistente Exportar como servicio web.

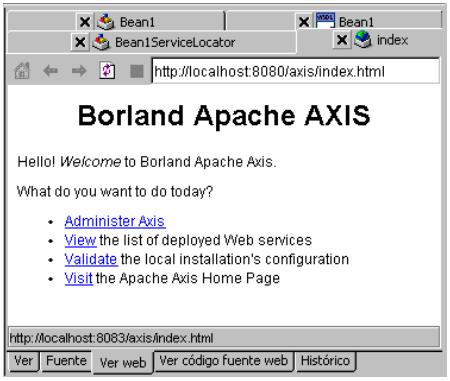


6 Continúe con el resto de los pasos del asistente y seleccione las opciones que desee y los métodos que vaya a exponer como servicio en el asistente. Si desea obtener más información acerca de las opciones del asistente Exportar como servicio web, pulse el botón de ayuda del asistente.



7 Amplíe el nodo del servidor de servicios web, pulse con el botón derecho del ratón `index.html` y seleccione Ejecutar utilizando "Servidor de servicios web" para distribuir el servicio en el servidor de servicios web. También puede seleccionar Ejecutar | Ejecutar proyecto para que el Asistente para configuración de servicios web ejecute la configuración Servidor de servicios web.

El servicio se distribuye al servidor web y en la ficha de administración Axis se muestra dónde se puede administrar, visualizar y validar el servicio.



8 Amplíe el paquete generado por el asistente, pulse con el botón derecho del ratón sobre el test JUnit, `<class name>ServiceTestCase.java` y, por último, seleccione Ejecutar test utilizando valores por defecto para probar el servicio generado.

Importante Si realiza cambios importantes en una clase después de haberla exportado, es necesario que vuelva a compilar, exporte de nuevo como servicio web y vuelva a distribuir en el servidor de servicios web para que los cambios surtan efecto.

Si desea obtener un tutorial que utilice el asistente Exportar como servicio web en la exportación de un JavaBean como un servicio web, consulte el Capítulo 13, "Tutorial: Creación de un servicio web sencillo con Axis". Para obtener más información sobre SOAP, consulte el Capítulo 3, "Configuración de proyectos para servicios web".

Importación de archivos WSDL

El asistente Importar un servicio web genera clases Java que implementan el servicio web definido en el WSDL. Puede generar tanto las clases en el servidor como las del cliente para crear y consumir los servicios. El asistente también puede generar sólo las clases del cliente para consumir un servicio. Puede generar, de forma optativa, un test JUnit para comprobar la interacción con el servicio web generado. El código se añade a las clases generadas para implementar los métodos como desee. El asistente Importar un servicio web también genera un archivo `deploy.wsdd` que proporciona información de distribución al servidor de los servicios web.

Según la configuración que seleccione, el asistente Importar un servicio web genera algunos o todos los archivos a partir del WSDL. Los ejemplos de los nombres de archivos proceden del ejemplo `AddressBook.wsdl` de `<jbuilder>/samples/webservices/axis/wsdl/AddressBook.wsdl`.

Tabla 7.2 Archivos generados por el asistente Importar un servicio web

Nombre del archivo	Descripción	Elemento WSDL	Ejemplo
complexType name`.java`	Una clase para cada `complexType` en la sección de tipos del WSDL. Una clase de marcador de este tipo se utiliza como un parámetro inout/out.	`<xsd:complexType name="phone">`	`Address.java` `Phone.java`

Tabla 7.2 Archivos generados por el asistente Importar un servicio web (continuación)

Nombre del archivo	Descripción	Elemento WSDL	Ejemplo
service name.java	Una interfaz de servicio para cada servicio que define un método de obtención para cada uno de los puertos que figuran en el elemento service del WSDL. Esta interfaz de servicio define una clase de factoría para obtener una instancia stub.	<service name= "AddressBookService">	AddressBookService. java
*service name*Locator.java	Una clase de localizador de cada servicio que constituye la implementación de la interfaz de servicio.	<service name= "AddressBookService">	AddressBookServiceLocator. java
*service name*TestCase.java	Un test JUnit optativo para comprobar el servicio web.	<service name= "AddressBookService">	AddressBookServiceTestCase. java
portType name.java	Una interfaz para cada portType del WSDL. La implementación de esta interfaz se utiliza para llamar a los métodos remotos.	<portType name= "AddressBookPortType">	AddressBookPortType. java
*binding name*Impl.java	Una clase de implementación para cada interfaz portType.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingImpl. java
*binding name*Skeleton.java	Una clase de esqueleto optativa, que encapsula una implementación del servidor.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingSkeleton .java
*binding name*Stub.java	Una clase stub de cada asociación que actúa como proxy para un servicio web remoto. Esto permite llamar al servicio Web como si fuera un objeto local.	<binding name= "AddressBookSOAPBinding">	AddressBookSOAPBindingStub. java
deploy.wsdd	Un archivo XML para cada servicio que proporciona información de distribución en el servidor de servicios web y que se puede utilizar con AdminClient.		
server-config.wsdd	Un archivo XML que proporciona información de distribución en el servidor.		

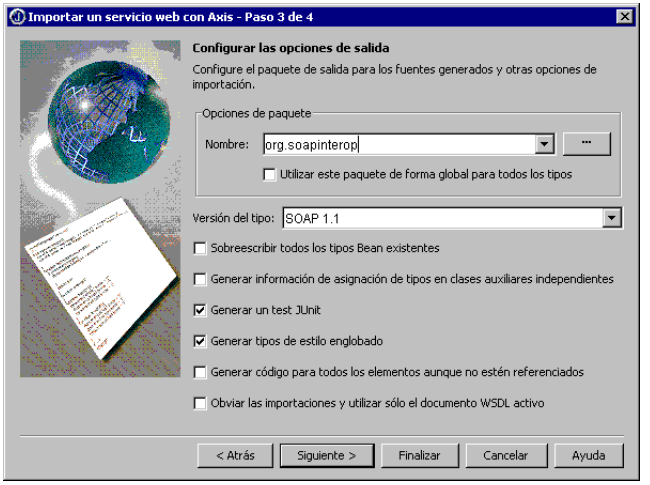
El asistente genera un nombre de paquete basado en el espacio de nombre de destino del WSDL o utiliza el especificado por el desarrollador, y además añade todas las bibliotecas de proyecto necesarias.

El asistente Importar un servicio web está disponible en la pestaña Servicios web de la galería de objetos. También puede pulsar con el botón derecho del ratón sobre cualquier documento WSDL del panel del proyecto y seleccionar Importar un servicio web en el menú contextual. Este asistente también se encuentra disponible en el Explorador de servicios web cuando se selecciona un nodo que especifica un WSDL.

Para importar un WSDL y generar las clases Java a partir de él, siga estos pasos básicos:

1. Cree un proyecto mediante Archivo | Proyecto nuevo.
2. Cree una implementación SOAP para albergar al servicio web tal y como se describe en "Utilización del Asistente para configuración de servicios web" en la página 3-1 (Achivo | Nuevo | Servicios web | Configuración de servicios web).
3. Seleccione uno de estos métodos para importar un WSDL y abrir el asistente Importar un servicio web:
   - Añada un WSDL a su proyecto, púlselo con el botón derecho del ratón en el panel del proyecto y seleccione Importar un servicio web.
   - Añada un WSDL a su proyecto, selecciónelo en el panel del proyecto, seleccione Archivo | Nuevo | Servicios web y, por último, pulse dos veces el icono Importar un servicio web.
   - Seleccione Herramientas | Explorador de servicios web, busque un WSDL y elija Archivo | Importar un servicio web.
4. Deje Axis de Apache como kit de herramientas y pulse Aceptar.
5. Acepte la URL del WSDL o pulse el botón de puntos suspensivos para buscar un WSDL. Escriba el nombre de usuario y la contraseña si el WSDL lo necesita. Pulse Siguiente para continuar.

6 Seleccione las opciones de salida y del servidor que desee en los pasos 2 y 3. Tenga en cuenta el nombre del paquete en Opciones de paquete. Este es el paquete en el que el asistente coloca los archivos de clase Java generados. Si desea obtener más información acerca de las opciones del asistente Importar un servicio web, pulse el botón de ayuda en cualquier paso del asistente.



7 Personalice los nombres de los paquetes en el paso 4 y pulse Finalizar para cerrar el asistente.

8 Amplíe el nuevo paquete creado por el asistente para ver los archivos de clase Java generados.

9 Amplíe el nodo del servidor de servicios web, pulse con el botón derecho del ratón `index.html` y seleccione Ejecutar utilizando "Servidor de servicios web" para distribuir el servicio en el servidor de servicios web.

10 Ejecute el cliente para consumir el servicio o modificar las clases del servidor para implementar el servicio de forma local.

11 Pulse con el botón derecho del ratón sobre el test JUnit si lo ha creado y seleccione Ejecutar test utilizando valores por defecto para probar el servicio generado.

Si desea obtener un tutorial que utilice un documento WSDL para consumir un servicio web, consulte el Capítulo 14, “Tutorial: Creación de un servicio web desde un documento WSDL”.

Exportación de EJB como servicios web

La exportación de Enterprise JavaBeans (EJBs) como servicios web a través de JBuilder no requiere mayor esfuerzo que el necesario para configurar el proyecto para los servicios web. Simplemente tiene que desarrollar su aplicación EJB como de costumbre, configurar el proyecto para los servicios web mediante el Asistente para la configuración de servicios web y ejecutarlo con la configuración de ejecución Servidor de servicios web que se ha creado con el asistente. Sin necesidad de llevar a cabo pasos adicionales, todos los EJB del proyecto se distribuyen automáticamente al servidor web Axis como servicios web. Por defecto, JBuilder expone automáticamente todos los beans sesión sin estado con métodos empresariales en la interfaz remota. También se puede modificar este funcionamiento por defecto y seleccionar sólo los módulos EJB, beans y métodos que desee exponer como servicios web.

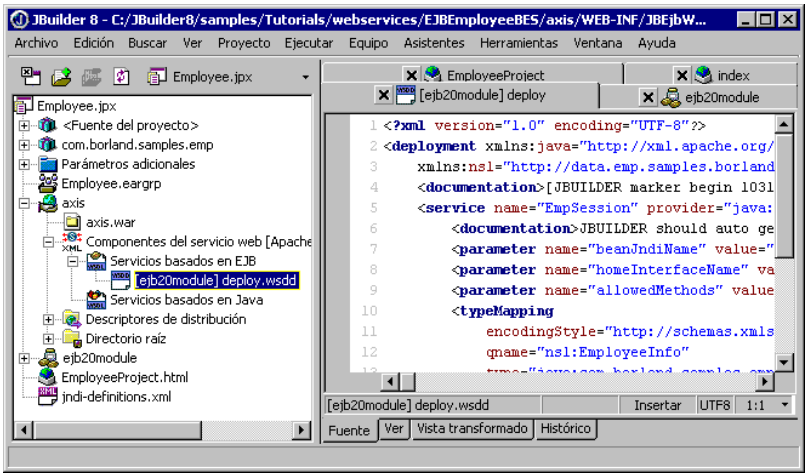
Nota El proceso de exportación de EJB difiere del de exportación de clases Java. Las clases Java se deben exportar explícitamente mediante el asistente Exportar un servicio web. Sin embargo, los EJB se exportan automáticamente si el proyecto se ha configurado para servicios web mediante el Asistente para configuración de servicios web.

Importante Si desea obtener información acerca de la configuración con servidores de aplicaciones, consulte el Capítulo 10, “Modificar la configuración del servidor de aplicaciones empresariales para servicios web”, y "las Notas de esta versión" (Ayuda | Notas de esta versión).

Para crear un servicio web basado en EJB, siga los pasos siguientes:

1. Utilice el asistente Configuración de servicios web y cree la WebApp de Axis como la crearía si se tratase de una aplicación de servicios web autónoma. Esto crea una configuración de ejecución basada en el servidor de aplicaciones del proyecto. Para obtener más información sobre el asistente Configuración de servicios web, consulte “Utilización del Asistente para configuración de servicios web” en la página 3-1.
2. Cree uno o varios grupos EJB vacíos y rellénelos con beans sesión y entidad. Impleméntelos como lo haría con cualquier aplicación EJB. Puede que tenga beans sesión de alto nivel de detalle sin estado. Si no los tiene, es necesario que cree uno, ya que sólo los beans sesión sin estado se exponen como servicios web. Los beans sesión sin estado deben contar como mínimo con un método válido en la interfaz remota. Este modelo estable un paralelismo en el acceso de los beans desde un contenedor web, como ocurre con un Servlet y las páginas JSP. Para obtener más información sobre los EJB, consulte la *Guía del desarrollador de Enterprise JavaBeans*.
3. Genere el proyecto.

Nota El kit de herramientas Axis de Apache combina cada archivo de distribución (`deploy.wsdd`) del proyecto en un `server-config.wsdd`. El kit de herramientas sobrescribe todas las modificaciones manuales realizadas en los archivos de distribución. Si modifica `server-config.wsdd`, desactive la opción Volver a generar distribución de la pestaña Servicios web de la ficha Generar de Propiedades de proyecto; si no lo hace, el kit de herramientas sobrescribe `server-config.wsdd` cuando se genera el proyecto. Para obtener más información, consulte "Configuración de las opciones de generación" en la página 3-7.



4 Amplíe el nodo de la WebApp Axis, pulse con el botón derecho del ratón `index.html` y seleccione Ejecutar utilizando "Servidor de servicios web" para ejecutar la implementación SOAP y el servidor de aplicaciones EJB.

Tenga en cuenta que si modifica la aplicación EJB después de exportarla como un servicio, tiene que exportarla y distribuirla de nuevo para que los cambios surtan efecto.

Si desea obtener un tutorial que muestre el ciclo completo de servicios web utilizando EJB, consulte el Capítulo 15, "Tutorial: Crear un servicio web desde una aplicación EJB con Borland Enterprise Server".

Importación de servicios como aplicaciones EJB

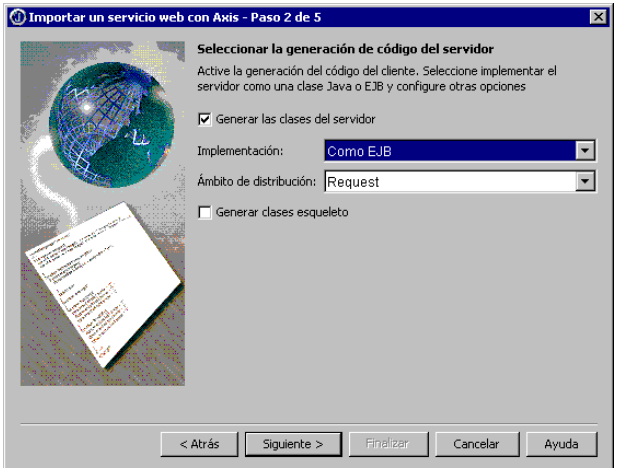
El asistente Importar un servicio web proporciona varias opciones para importar un servicio desde un archivo WSDL. Si desea implementar el servicio como un EJB, seleccione la opción de implementación, Como EJB, en el Paso 2 del asistente. Debe especificar un servidor de aplicaciones para el proyecto en la ficha Servidor (Proyecto | Propiedades de proyecto) y un módulo EJB. Puede crear un grupo EJB vacío antes de utilizar el asistente, o crearlo mientras lo utiliza.

Para implementar un servicio web como un EJB:

1. Añada un documento WSDL a su proyecto, o desplácese hasta un WSDL en el Explorador de servicios web.

Importante Debe especificar un servidor de aplicaciones como servidor para el proyecto.

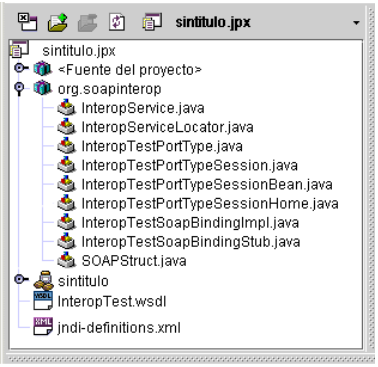
2. Pulse el WSDL con el botón derecho del ratón en el panel del proyecto y seleccione Importar un servicio web para que se abra el asistente Importar un servicio web. Si está utilizando el Explorador de servicios web, seleccione Archivo | Importar un servicio web.
3. Deje Axis de Apache como kit de herramientas y pulse Aceptar. Si selecciona el kit de herramientas WebLogic, los pasos a seguir serán diferentes. Consulte el Capítulo 8, “El kit de herramientas WebLogic”.
4. Acepte el nombre del WSDL en el campo WSDL y pulse Siguiente. No necesita especificar un nombre de usuario ni una contraseña.
5. Seleccione la opción de implementación, Como EJB, en el Paso 2 del asistente y, a continuación, pulse Siguiente.



6 Configure el paquete de salida, seleccione las opciones de importación, desactive la opción Generar test JUnit y pulse Siguiente.

7 Cree un nuevo grupo EJB vacío con el Asistente para grupos EJB vacíos o seleccione uno ya creado de la lista desplegable del Paso 4 y, a continuación, pulse Siguiente.

8 Personalice la asignación de nombres de paquetes y pulse Finalizar para cerrar el asistente. Observe que el asistente genera un paquete según el targetNamespace del WSDL. Amplíe el paquete para ver los archivos `.java` que ha generado el asistente.



9 Genere el proyecto.

10 Cree un cliente de prueba EJB (Archivo | Nuevo | Enterprise), modifíquelo, ejecute el servidor y compruebe la aplicación EJB.

Junto con Borland Enterprise Server, dispone de un tutorial sobre importación de WSDL como aplicaciones EJB en Capítulo 16, "Tutorial: Importación de servicios web como aplicaciones EJB".

Importación de servicios como aplicaciones EJB en el Explorador de servicios web

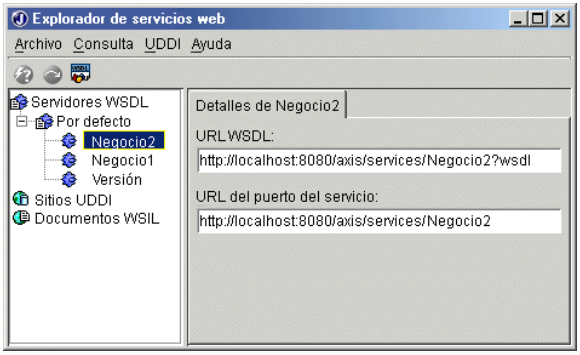
Una vez que ha creado y exportado su aplicación como un servicio web en un servidor Axis, puede generar una aplicación cliente para utilizar el servicio que haya creado en el Explorador de servicios web. En los pasos siguientes, se utiliza el Explorador de servicios web para buscar todos los servicios web disponibles del servidor e importar el WSDL para generar una aplicación cliente.

1 Seleccione Herramientas | Explorador de servicios web para abrir el Explorador. Para obtener más información, consulte el Capítulo 11, "Búsqueda y publicación de servicios web".

2 Expanda el nodo Servidores WSDL y seleccione el nodo Por defecto en el árbol.

3 Modifique el campo URL de la ficha Detalles con el contexto de la WebApp y la ubicación del servidor del servicio web. Por ejemplo, `http://localhost:8080/axis/services` o `http://localhost:8080/axis/servlet/AxisServlet`, donde `http://localhost:8080` es la ubicación del servidor y `axis` es el contexto web (nombre de la WebApp).

4 Pulse el botón Mostrar servicios de la ficha Detalles para que aparezcan los servicios web EJB publicados. Observe que los beans se encuentran disponibles en la lista. Tenga en cuenta que el servidor de aplicaciones debe estar ejecutándose antes de que pueda ver sus servicios.



5 Seleccione el EJB en el árbol y pulse el botón Importar un servicio web de la barra de herramientas. El asistente Importar un servicio web ofrece varias opciones de implementación: implementar el servicio como una clase, un EJB o sólo como un código cliente. Para obtener información sobre los archivos generados por el asistente Importar un servicio web, consulte "Importación de archivos WSDL" en la página 7-6. Para ver un tutorial, consulte el Capítulo 14, "Tutorial: Creación de un servicio web desde un documento WSDL".

6 A continuación, puede publicar un servicio web en un registro UDDI. Para obtener más información, consulte "Difusión de servicios web desde un servidor Axis" en la página 11-26.

Distribución de servicios web

Los asistentes Importar un servicio web y Exportar como servicio web crean un archivo descriptor de distribución de servicios web XML llamado `deploy.wsdd` que proporciona información de distribución al servidor de servicios web. Cuando se exportan EJB como servicios web (JBuilder los exporta automáticamente sin necesidad de utilizar un asistente) se genera un archivo `deploy.wsdd` por cada módulo EJB del proyecto.

Los archivos WSDD

Los archivos WSDD son descriptores de distribución de servicios web que representan a los servicios expuestos como servicios web. Puede personalizar la información generada y añadir elementos adicionales, tales como manejadores personalizado, cadenas, etc.

deploy.wsdd

El archivo `deploy.wsdd` aparece en el nodo Webapp que hospeda el servicio web. Si se exporta una clase Java, aparece en el nodo Servicios basados en Java de la webapp (WebApp | Componentes del servicio web | Servicios basados en Java). Los WSDD EJB aparecen en el nodo Servicios basados en EJB de la webapp. El archivo viene precedido por `[<nombre del paquete>]` para las clases exportadas y `[<nombre del módulo EJB>]` para los EJB con el fin de diferenciarlo de otros archivos `deploy.wsdd` que se puedan haber generado para otros paquetes del proyecto. Todos los archivos WSDD del árbol `src` se recogen en el nodo Componentes del servicio web que sea conveniente. Tenga en cuenta, sin embargo, que este nodo no corresponde a una ubicación física del disco. El archivo `deploy.wsdd` se guarda en el mismo directorio que los stubs y esqueletos generados.

En el caso de las clases exportadas, los valores del archivo `deploy.wsdd` se derivan de la información del documento WSDL y de los métodos expuestos elegidos en el asistente Importar como servicio web. La configuración del asistente Importar como servicio web no cambia los valores del descriptor de distribución. En cuanto a los EJB, que JBuilder distribuye automáticamente sin necesidad de utilizar el asistente, los valores de `deploy.wsdd` se derivan de los beans sesión sin estado con métodos empresariales de la interfaz remota. Tenga en cuenta que el formato WSDD es un formato patentado específico del kit de herramientas Axis. Si desea obtener más información, consulte la

documentación Axis disponible en <jbuilder>/thirdparty/xml-axis/java/docs.

En cualquier momento puede modificar los valores de deploy.wsdd para las clases Java classes y los EJB, así como añadir elementos adicionales, tales como manejadores personalizador, cadenas, etc. Si desea obtener más información sobre la personalización de la distribución de EJB, consulte "Edición de archivos WSDD para EJB" en la página 7-16.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:ns="http://untitled33"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Bean1" provider="java:RPC">
      <parameter name="className" value="untitled33.Bean1"/>
      <parameter name="allowedMethods" value="babelFish"/>
      <parameter name="scope" value="Request"/>
  </service>
</deployment>
```

Consulte

- "Custom Deployment - Introducing WSDD" de la documentación de Axis en <jbuilder>/thirdparty/xml-axis/java/docs/user-guide.html
- "Axis Web Services Deployment Descriptor (WSDD)" de la documentación de Axis en <jbuilder>/thirdparty/xml-axis/java/wsdd/docs/index.html
- "Deployment (WSDD) Reference" en la *Axis Reference Guide* de <jbuilder>/thirdparty/xml-axis/java/docs/reference.html#Deployment

Edición de archivos WSDD para EJB

La distribucion EJB de los servicios web se puede personalizar de dos modos:

- "Modificación de las propiedades del nodo de distribución" en la página 7-16
- "Edición de deploy.wsdd y modificación del elemento <documentation>" en la página 7-17

Modificación de las propiedades del nodo de distribución

Por defecto, JBuilder expone automáticamente todos los beans sesión sin estado con métodos empresariales en la interfaz remota. También se puede modificar este funcionamiento por defecto y seleccionar sólo los módulos de Enterprise JavaBean (EJB), beans y métodos que desee exponer como servicios web.

El comportamiento de distribución automático se puede modificar en la ficha Módulos del cuadro de diálogo Propiedades de servicios basados en EJB:

1. Amplíe el nodo WebApp que hospeda el servicio web en el panel del proyecto, amplíe el nodo Componentes del servicio web, pulse con el botón derecho del ratón sobre el nodo de servicios basados en EJB y seleccione Propiedades. En la ficha Módulos, seleccione los grupos EJB vacíos y los beans sesión sin estado que desee exponer como servicios web. Los beans sesión sin estado deben contar como mínimo con un método válido en la interfaz remota. Por defecto se exponen todos los beans, que se seleccionan en la ficha Propiedades.

2. Quite la marca de todos los grupos EJB vacíos y/o beans que no desee exponer como servicios web.
3. Pulse Aceptar para cerrar el cuadro de diálogo.
4. Vuelva a generar el proyecto.

Edición de deploy.wsdd y modificación del elemento <documentation>

Si se modifica el archivo `deploy.wsdd` para un EJB exportado, tendrá que modificar el elemento `<documentation>` para evitar que el kit de herramientas sobrescriba los cambios. En `deploy.wsdd` para EJB, existe un elemento `<documentation>` para cada elemento `<service>`. Cambie la entrada de YES a NO en el elemento `<documentation>` de cada elemento `<service>` que se haya modificado:

```
<documentation>JBUILDER debe autogenerar esta entrada:
    NO(YES/NO)</documentation>
```

Tras haber modificado esta entrada, el kit de herramientas **no** sobrescribirá las modificaciones en el archivo `deploy.wsdd`. Debe generar de nuevo el proyecto para que se acepten los cambios.

server-config.wsdd

Durante la generación, JBuilder genera servicios web para todo el proyecto utilizando el archivo individual `deploy.wsdd` como entrada. El resultado de esta operación es un archivo `server-config.wsdd`, que se encuentra en el nodo Descriptores de distribución de la webapp. Se puede modificar este archivo, pero debe desactivar la opción de generación para evitar que el kit de herramientas lo sobreescriba en la próxima generación. Si desea más información, consulte la pestaña Servicios web de la ficha Generar (Proyecto | Propiedades de proyecto | Generar).

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns="http://xml.apache.org/axis/wsdd/" xmlns:java=
    "http://xml.apache.org/axis/wsdd/providers/java">
<globalConfiguration>
  ...
  <handler name="MsgDispatcher" type=
    "java:org.apache.axis.providers.java.MsgProvider"/>
  <service name="Bean1" provider="java:RPC"/>
  <parameter name="allowedMethods" value="*"//>
  <parameter name="scope" value="Session"//>
  <parameter name="className" value="untitled33.Bean1"//>
  </service>
  ...
</deployment>
```

Edición de server-config.wsdd

Para modificar `server-config.wsdd`:

1. Genere el proyecto para crear el archivo `server-config.wsdd` a partir de los archivos `deploy.wsdd` de los servicios web del proyecto.
2. Abra el archivo `server-config.wsdd` en el nodo Descriptores de distribución de la webapp que hospeda el servicio web.
3. Busque y modifique lo que le interese. Por ejemplo, puede restringir los métodos expuestos desde `<parameter name="allowedMethods" value="getSample, setSample"/>` y permitir sólo el método get y ningún set: `<parameter name="allowedMethods" value="getSample"/>`.
4. Utilice el mismo mecanismo para añadir requestFlow, asignaciones de tipos o cualquier otro elemento de distribución al archivo WSDD, para que JBuilder los distribuya.
5. Desactive la opción Volver a generar distribución de la pestaña Servicios web de la ficha Generar (Proyecto | Propiedades de proyecto | Generar). Esto evita que el kit de herramientas sobreescriba el archivo en la próxima generación.
6. Vuelva a generar el proyecto.

Capítulo

8

El kit de herramientas WebLogic

Es una función de JBuilder Enterprise.

JBuilder proporciona asistentes para utilizar el kit de herramientas WebLogic para poder consumir y crear servicios web con facilidad. El asistente Exportar como servicio web exporta una o varias clases Java como servicio web. Sin embargo, los EJB se exportan automáticamente si el proyecto se ha configurado para servicios web mediante el Asistente para configuración de servicios web. El asistente Importar un servicio web permite importar un servicio web a partir de un documento WSDL o un EAR.

Ambos asistentes pueden crear una JAR cliente que se genera en una carpeta GeneratedWebServiceClients del panel del proyecto. Para poder ver las clases Java generadas por el asistente y por el kit de herramientas WebLogic, abra el archivo JAR cliente. El archivo, `<ServiceName>Port.java`, contiene la definición de la interfaz del servicio web, en la que `<ServiceName>` hace referencia al nombre del servicio web. El stub `<ServiceName>_Impl` implementa la interfaz JAX-RPC `Service`. El kit de herramientas WebLogic utiliza una implementación de JAX-RPC (API de Java para llamadas a procedimientos remotos basadas en XML) compatible con stubs cliente.

Tabla 8.1 Interfaces y clases JAX-RPC

Interfaz o clase	Descripción
`Service`	Interfaces cliente principales que se utilizan para realizar llamadas estáticas y dinámicas.
`ServiceFactory`	Clase de factoría para crear instacias `Service`.
`Stub`	El proxy cliente para llamar a las operaciones (métodos) que se utilizan para las llamada estáticas de un servicio.
`Call`	Se utiliza para llamar de forma dinámica a un servicio web.

Si desea obtener más información sobre el servidor WebLogic y los servicios web, consulte la documentación de BEA en `http://edocs.bea.com/wls/docs70/webservices.html`.

Exportación de una clase como servicio web

El asistente Exportar como servicio web se utiliza para exportar clases como servicios web. Dispone de opciones que permiten seleccionar qué métodos deben de ponerse a disposición del servicio. Este asistente se puede utilizar para exportar cualquier clase o clases Java como un servicio web; por ejemplo, un JavaBean o Enterprise JavaBean (EJB). Sin embargo, si la clase utiliza tipos ajenos al bean como parámetros o valores devueltos, tendrá que proporcionar serializadores y deserializadores. El asistente dispone de opciones para generar implementaciones de cliente del servicio. También genera un WSDL, que describe el servicio, e información de distribución para el kit de herramientas WebLogic.

El asistente proporciona la opción que permite generar un stub cliente o sólo los archivos de distribución WLDU. Si no lo genera, el asistente tiene dos pasos, y sólo se genera un archivo `servicegen.wldu`, que proporciona la información de distribución al servidor WebLogic. Si desea obtener más información sobre el archivo de distribución, consulte "Los archivos WLDU" en la página 8-15. Si marca la opción Generar stub cliente, el asistente tiene siete pasos y genera un `<client>.jar`, que contiene las clases y el WSDL. El archivo `<client>.jar` aparece como un nodo en la carpeta GeneratedWebServiceClients del panel del proyecto. El asistente también añade la biblioteca GeneratedWebServiceClients al proyecto (Proyecto | Propiedades de proyecto | Vías de acceso | Bibliotecas necesarias), para que JBuilder pueda encontrar las clases en el archivo JAR.

Si no se ha configurado el proyecto para que hospede el cursivo web, aparece el Asistente para configuración de servicios web antes que el asistente Exportar un servicio web.

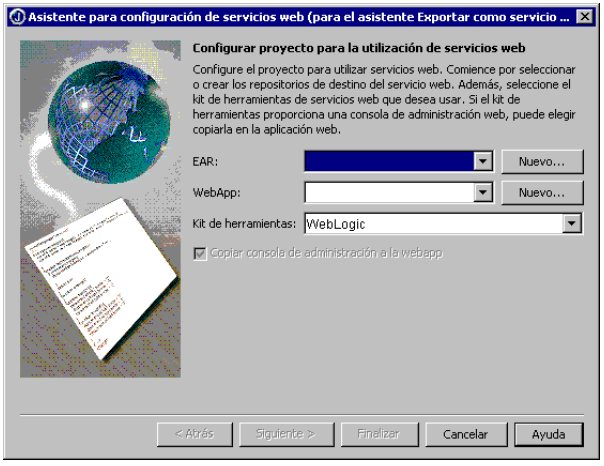
Nota El proceso de exportación de clases difiere del de exportación de EJB. Las clases Java se deben exportar explícitamente mediante el asistente Exportar un servicio web. Sin embargo, los EJB se exportan automáticamente si el proyecto se ha configurado para servicios web mediante el Asistente para configuración de servicios web.

Para crear un servicio web a partir de una clase Java con el kit de herramientas WebLogic, siga estos pasos:

1. Seleccione Archivo | Nuevo proyecto para crear un proyecto mediante el Asistente para proyectos.

Importante Asegúrese de que crea el proyecto en un directorio **sin** espacios en el nombre o, de lo contrario, el servidor WebLogic generará errores.

2. Seleccione Proyecto | Propiedades de proyecto y, a continuación, abra la pestaña Servidor.
3. Especifique WebLogic Application Server 7.x como el servidor único del proyecto.
4. Seleccione Archivo | Nuevo | General y haga doble clic sobre el icono JavaBean para crear un JavaBean y exportarlo como un servicio.
5. Seleccione `java.lang.Object` como la clase base.
6. Marque la opción Generar propiedad de ejemplo en el Asistente para JavaBean y pulse Finalizar para crear el JavaBean.
7. Pulse con el botón derecho del ratón sobre el JavaBean en el panel del proyecto y seleccione Exportar como servicio web para que se abra el asistente Exportar como servicio web. Ya que el proyecto todavía no está configurado para los servicios web, se abre el Asistente para configuración de servicios web, para que pueda crear un EAR, una WebApp para hospedar el servicio y seleccionar el kit de herramientas de los servicios.



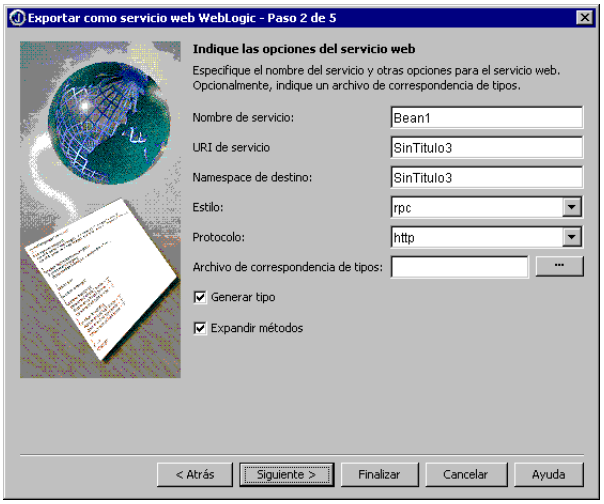
8. Configure el proyecto para los servicios web del siguiente modo:
   a. Pulse el botón Nuevo junto al campo EAR y cree un EAR con el Asistente para EAR. Pulse Finalizar para volver al Asistente para configuración de servicios web.
   b. Pulse el botón Nuevo junto al campo WebApp para crear una WebApp SOAP con el Asistente para aplicaciones web. Es necesario que haya una WebApp para hospedar el servicio. Escriba el nombre y el directorio de la WebApp. Pulse Aceptar para volver al Asistente para configuración de servicios web.
   c. Seleccione WebLogic como el kit de herramientas de servicios web y pulse Siguiente para ir al último paso.

d Acepte la configuración de ejecución por defecto Servidor de servicios web. La utilizará más adelante para generar y ejecutar el proyecto.

e Pulse Finalizar para cerrar el Asistente para configuración de servicios web. Ahora se abre el asistente Exportar como servicio web.

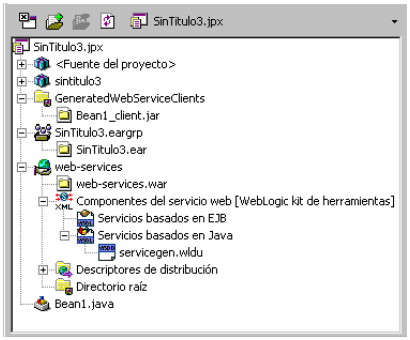
9 Acepte la clase que se va a exportar, el nombre del EAR y el nombre de la WebApp en el Paso 1 del asistente Exportar como servicio web. Observe que la opción Generar stub cliente está marcada. Si no desea generar el stub cliente, quite la marca de esta opción. Si no se marca esta opción, el asistente sólo tiene dos pasos. Si está marcada, el asistente cuenta con pasos adicionales y genera un JAR cliente en el nodo GeneratedWebServiceClients del panel del proyecto.

10 Continúe al paso 2 y tome nota del URI del servicio. El URI del servicio se utiliza en la URL para llamar al servicio web. La URL del servicio se utilizará posteriormente para comprobar el servicio en la página de inicio de servicios web WebLogic. Los valores de esta ficha del asistente se guardan en el archivo de distribución WLDU, `servicegen.wldu`. Puede modificar cualquiera de estos valores en la ficha Propiedades WebLogic y, a continuación, exportar de nuevo la clase Java. Para obtener más información o modificar estos valores, consulte "Configuración de los nombres de servicios por defecto" en la página 8-18.



11 Pulse Siguiente y complete los pasos restantes del asistente Exportar como servicio web. Observe los nombres por defecto del paquete y el JAR cliente en el Paso 4: `<nombre-del-proyecto>.generated` y `<nombre-de-clase>_client.jar`. Pulse el botón de ayuda para obtener más información acerca de las opciones.

12 Pulse Finalizar para cerrar el asistente. Amplíe el nodo WebApp y, a continuación, el nodo Componentes de servicio web y el nodo Servicios basados en Java para ver el archivo de distribución de WebLogic, `servicegen.wldu`, que ofrece información de distribución para el servidor. El archivo WLDU contiene información como: nombre del servicio, URI del servicio, contexto web (contextURI), espacio de nombre y las clases que se han exportado. Si desea obtener más información sobre este archivo de distribución, consulte "Los archivos WLDU" en la página 8-15. Amplíe el nodo GeneratedWebServiceClients y haga doble clic sobre el archivo JAR para ver los archivos de clase y el WSDL que ha creado el kit de herramientas WebLogic.



13 Seleccione Proyecto | Ejecutar Make del proyecto para generarlo.

14 Seleccione Ejecutar | Ejecutar el proyecto para ejecutar el proyecto con la configuración de ejecución Servidor de servicios web creada por el asistente Configuración de servicios web. Esta configuración de ejecución ejecuta el servidor WebLogic.

15 Pulse con el botón derecho del ratón sobre el nodo eargrp y seleccione Opciones de distribución | Distribuir para distribuir el EAR en el servidor.

16 Pruebe el servicio web distribuido en la página de inicio de un navegador web. Para obtener más información, consulte "Comprobación de servicios distribuidos" en la página 8-9.

17 Copie el código de ejemplo de la página de inicio para llamar al servicio y modificarlo para crear un cliente para comprobar el servicio. Consulte "Cómo escribir un cliente de prueba" en la página 8-11.

Si desea obtener más información sobre cómo escribir clientes para llamar a un servicio web, consulte "Writing the Java Client Application Code" en la documentación de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/client.html#1024463` y "Escritura del código de un cliente para probar el funcionamiento del servicio" en el Capítulo 17, "Tutorial: Creación de un servicio web sencillo con WebLogic".

Nota Si modifica las clases de su servicio web, es necesario que compile, exporte como servicio de nuevo y vuelva a distribuir para que los cambios tengan efecto.

Si desea un tutorial acerca de la exportación de una clase como servicio web con WebLogic, consulte el Capítulo 17, "Tutorial: Creación de un servicio web sencillo con WebLogic".

Exportación de varias clases como servicio web

Puede exportar de forma simultánea varias clases como un servicio web, tal y como se describe en los pasos siguientes:

1. Seleccione una o varias clases y haga clic sobre ellas con el botón derecho del ratón en el panel del proyecto y, a continuación, elija Exportar como servicio web.
2. Pulse el botón Añadir del asistente si desea añadir clases adicionales para exponerlas como servicios.
3. Elija las opciones que le interesen en el asistente Exportar como servicio web y pulse Finalizar.
4. Compile y distribuya en el servidor.
5. En la página de inicio, utilice un navegador para realizar un test del servicio que se acaba de distribuir.

Importación de un WSDL o un EAR como servicio web

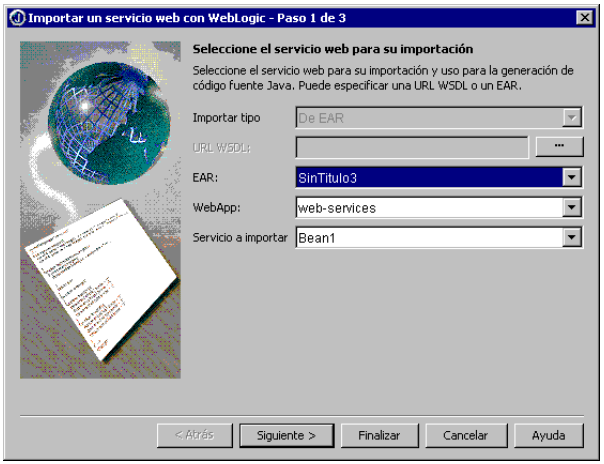
El asistente Importar un servicio web genera un cliente de servicio web. Si utiliza el asistente Importar un servicio web con el kit de herramientas WebLogic, puede importar un WSDL o un EAR que contenga un servicio web específico de WebLogic. El asistente genera una JAX-RPC (API de Java para llamadas a procedimientos remotos basadas en XML) compatible con stubs cliente. El código generado se almacena en un archivo JAR de la carpeta GeneratedWebServiceClients del proyecto. De forma automática se crea una biblioteca con el mismo nombre y se añade al proyecto.

Importación de un EAR como servicio web

Para importar un EAR como un servicio web, siga estos pasos básicos:

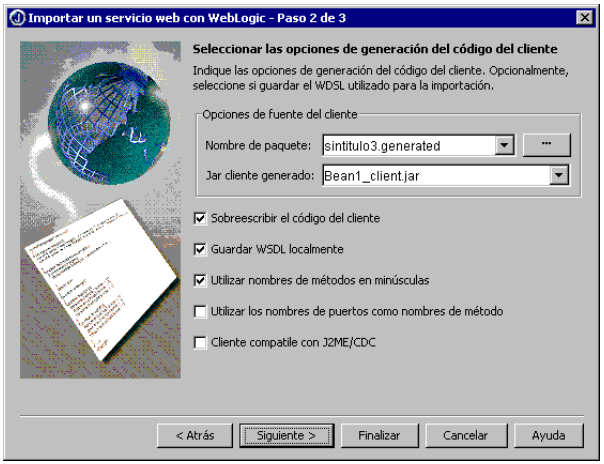
1. Lleve a cabo los pasos para la creación de un servicio web distribuido tal y como se describen en "Exportación de una clase como servicio web" en la página 8-2.

2 Pulse con el botón derecho del ratón sobre el nodo eargrp del panel del proyecto y seleccione Importar un servicio web para que se abra el asistente Importar un servicio web.



3 Acepte el nombre del archivo EAR que desee importar y seleccione el servicio para se ha de importar en la lista desplegable, si está disponible y, a continuación, pulse Siguiente.

4 Elija las opciones de generación de código en el cliente y pulse Siguiente.



5 Especifique el nombre del paquete y pulse Finalizar.

6 Amplíe el nodo GeneratedWebServiceClients para ver el JAR cliente generado por el servidor WebLogic.

7 Pruebe el servicio distribuido en la página de inicio de WebLogic tal y como se describe en "Comprobación de servicios distribuidos" en la página 8-9.

8 Copie el código de ejemplo de la página de inicio para llamar al servicio y crear un cliente para comprobar el servicio. Consulte "Cómo escribir un cliente de prueba" en la página 8-11.

Si desea obtener más información sobre cómo escribir clientes para llamar a un servicio web, consulte "Writing the Java Client Application Code" en la documentación de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/client.html#1024463` y "Escritura del código de un cliente para probar el funcionamiento del servicio" en el Capítulo 17, "Tutorial: Creación de un servicio web sencillo con WebLogic".

Importación de un WSDL como servicio web

La importación de un WSDL es muy parecida a la de un EAR. Siga estos pasos básicos:

1 Seleccione Archivo | Nuevo proyecto para crear un proyecto mediante el Asistente para proyectos.

Importante Asegúrese de que crea el proyecto en un directorio **sin** espacios en el nombre o, de lo contrario, el servidor WebLogic generará errores.

2 Seleccione Proyecto | Propiedades de proyecto y, a continuación, abra la pestaña Servidor. Elija WebLogic como el servidor único del proyecto.

3 Añada un documento WSDL al proyecto con el botón Añadir archivos/paquetes de la barra de herramientas del panel del proyecto, o bien, utilice el Explorador de servicios web para importarlo. Para obtener más información, consulte "Generación de clases Java a partir de documentos WSDL" en la página 11-29.

4 Pulse el WSDL con el botón derecho del ratón en el panel del proyecto y seleccione Importar un servicio web para que se abra el asistente Importar un servicio web.

5 Seleccione WebLogic como kit de herramientas y pulse Aceptar.

6 Acepte la URL de WSDL y seleccione el servicio para importar si el WSDL cuenta con varios servicios.

7 Pulse Siguiente y elija las opciones de codificación del cliente que desee.

8 Pulse Siguiente y acepte el nombre del paquete o escriba uno nuevo.

9 Pulse Finalizar para cerrar el asistente y generar el archivo JAR. El asistente crea el archivo JAR de las clases Java a partir del WSDL y lo muestra en el panel del proyecto en un nodo GeneratedWebServiceClients.

10 Escriba un cliente para probar el servicio y ejecutar el test. Consulte "Cómo escribir un cliente de prueba" en la página 8-11.

Si desea obtener más información sobre cómo escribir clientes para llamar a un servicio web, consulte "Writing the Java Client Application Code" en la documentación de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/client.html#1024463` y "Escritura del código de un cliente para probar el funcionamiento del servicio" en el Capítulo 17, "Tutorial: Creación de un servicio web sencillo con WebLogic".

Comprobación de servicios distribuidos

Después de distribuir un servicio web, utilice la página de inicio de servicios web de WebLogic para ver el WSDL, comprobar el servicio y ver los mensajes SOAP entre el cliente y el servidor.

1 Escriba la dirección del servicio distribuido en un navegador.

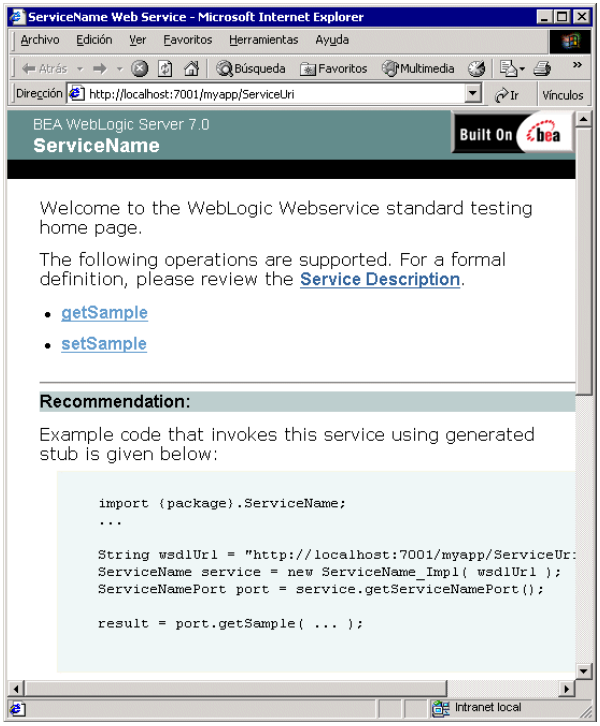
La dirección para llamar al servicio web tiene la siguiente forma:

```
<protocol>://<host:port>/<contextURI>/<serviceURI>
```

donde: `<protocol>` es el protocolo del servicio; `<host>` es el ordenador en el que se ejecuta el servidor WebLogic; `<port>` es el número de puerto en el que se encuentra el servidor WebLogic; `<contextURI>` es la raíz de contexto de la webapp o el nombre del archivo recopilatorio de la webapp y `<serviceURI>` es el URI del servicio.

Por ejemplo, si se ejecuta WebLogic de forma local en el puerto por defecto 7001, la WebApp es `web-services`, el ServiceUri es `Bean1` y la dirección del servicio web será: `http://localhost:7001/web-services/Bean1`

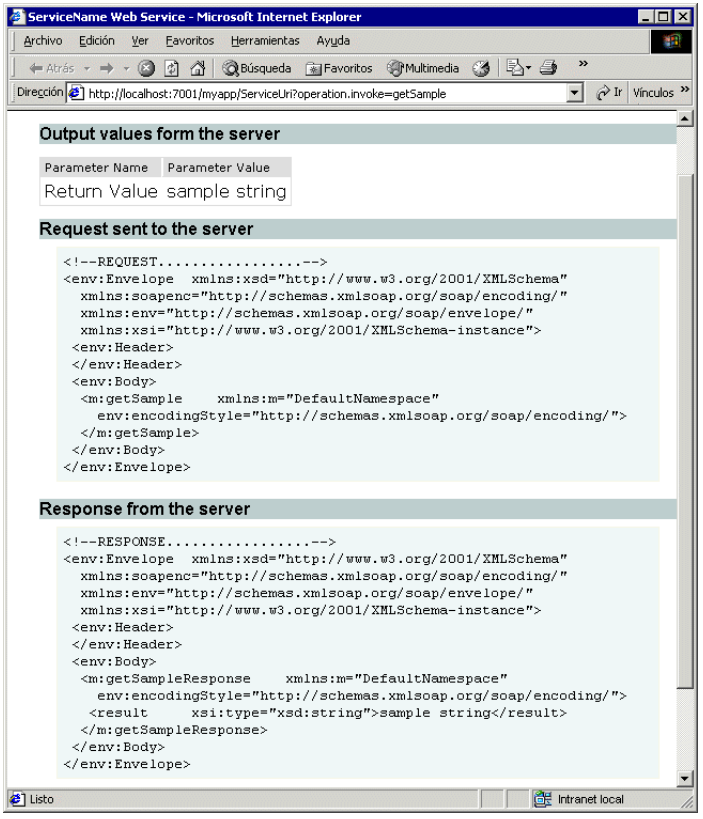
Si no conoce el ServiceURI, puede encontrarlo en el archivo `servicegen.wldu`.



La página de inicio muestra el nombre del servicio, un enlace a la descripción del servicio (WSDL), los métodos expuestos en el servicio y un código de ejemplo para llamar al servicio.

2 Haga clic sobre el enlace setSample y pulse el botón Llamar para ver la solicitud y respuesta SOAP que entra y sale del servidor. Observe que el valor para setSample es “Sample”.

3 Vuelva a la primera ficha, haga clic sobre el enlace getSample y pulse el botón Llamar para obtener el valor del ejemplo. La respuesta del servidor devuelve un valor de “Sample”.



4 Pulse el enlace Descripción de servicios para ver el WSDL generado para el servicio.

La página de inicio también cuenta con código de ejemplo para llamar al servidor que se puede copiar, modificar y utilizar para probar el servicio que haya distribuido.

Cómo escribir un cliente de prueba

Tras haber distribuido un servicio web, puede utilizar el código fuente de la página de inicio de servicios web de WebLogic para llamar al servicio web.

Escriba un cliente de prueba del siguiente modo:

1 Copie el código de ejemplo en la página de inicio.

2 En JBuilder seleccione Archivo | Nueva clase para crear un nuevo archivo de clase y pulse Aceptar.

3 Pegue el código de ejemplo en la nueva clase.

4 Modifique el código para llamar y probar el servicio.

Por ejemplo:

```
package untitled1;

import untitled1.generated.*; //<client>.jar
import java.io.*;

public class Untitled1 {
  public static void main(String[] args) {
   try {
          String wsdlUrl =
              "http://localhost:7001/web-services/untitled1?WSDL";
          Bean1 service = new Bean1_Impl(wsdlUrl);
          Bean1Port port = service.getBean1Port();
          System.out.println(port.getSample());
  }
  catch (IOException ex) {
       ex.printStackTrace();
    }
  }
}
```

5 Pulse con el botón derecho sobre la nueva clase de test y elija Ejecutar para ejecutar el test y llamar al servicio web.

Consulte

- "Utilización de la página de inicio de los servicios WebLogic" en la página 4-6
- "The WebLogic Web Services Home Page and WSDL URLs" en la documentación de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/client.html#1051033`

Exportación de EJB como servicios web

La exportación de Enterprise JavaBeans (EJBs) como servicios web a través de JBuilder no requiere mayor esfuerzo que el necesario para configurar el proyecto para los servicios web. Simplemente tiene que desarrollar su aplicación EJB como de costumbre, configurar el proyecto para los servicios web mediante el Asistente para configuración de servicios web y ejecutarlo con la configuración de ejecución Servidor de servicios web que se ha creado con el asistente. Sin necesidad de llevar a cabo pasos adicionales, todos los EJB del proyecto se distribuyen automáticamente al servidor WebLogic como servicios web. Por defecto,

JBuilder expone automáticamente todos los beans sesión sin estado con métodos empresariales en la interfaz remota. También se puede modificar este funcionamiento por defecto y seleccionar sólo los módulos de Enterprise JavaBean (EJB), beans y métodos que desee exponer como servicios web.

Nota El proceso de exportación de EJB difiere del de exportación de clases Java. Las clases Java se deben exportar explícitamente mediante el asistente Exportar un servicio web. Sin embargo, los EJB se exportan automáticamente si el proyecto se ha configurado para servicios web mediante el Asistente para configuración de servicios web.

Importante Si desea obtener información acerca de la configuración con servidores de aplicaciones, consulte el Capítulo 10, "Modificar la configuración del servidor de aplicaciones empresariales para servicios web", y "las Notas de esta versión" (Ayuda | Notas de esta versión).

Para crear un servicio web basado en EJB, siga los pasos siguientes:

1. Seleccione Archivo | Proyecto nuevo para crear un proyecto nuevo.
2. Configure WebLogic Application Server 7.x como el servidor del proyecto en la pestaña Servidor de Propiedades de proyecto (Proyecto | Propiedades de proyecto).
3. Utilice el Asistente para configuración de servicios web para configurar el proyecto para los servicios web. Cree una WebApp de servicios web y un EAR. Este asistente crea una configuración de ejecución basada en el servidor de aplicaciones del proyecto. Para obtener más información sobre el Asistente para configuración de servicios web, consulte "Utilización del Asistente para configuración de servicios web" en la página 3-1.
4. Cree uno o varios grupos EJB 2.0 vacíos y rellénelos con beans sesión y entidad. Impleméntelos como lo haría con cualquier aplicación EJB. Puede que tenga beans sesión de alto nivel de detalle sin estado. Si no los tiene, es necesario que cree uno, ya que sólo los beans sesión sin estado se exponen como servicios web. Los beans sesión sin estado deben contar como mínimo con un método válido en la interfaz remota. Este modelo estable un paralelismo en el acceso de los beans desde un contenedor web, como ocurre con un Servlet y las páginas JSP. Para obtener más información sobre los EJB, consulte *Guía del desarrollador de Enterprise JavaBeans*.
5. Seleccione Proyecto | Ejecutar Make del proyecto para generarlo.

Nota El asistente crea un archivo de distribución, `servicegen.wldu`. El kit de herramientas sobrescribe todas las modificaciones manuales realizadas en los archivos de distribución. Si modifica `servercegen.wldu`, desactive la opción Volver a generar distribución de la pestaña Servicios web de la ficha Generar de Propiedades de proyecto; si no lo hace, el kit de herramientas sobrescribe `servercegen.wldu` cuando se genera el proyecto.

Para obtener más información, consulte "Configuración de las opciones de generación" en la página 3-7.

6 Seleccione Ejecutar | Ejecutar el proyecto para ejecutar el proyecto con la configuración de ejecución Servidor de servicios web. Esta configuración de ejecución genera el proyecto, distribuye los EJB y ejecuta el servidor. JBuilder distribuye automáticamente todos los grupos EJB vacíos compatibles con 2.0 y los beans sesión del proyecto. Si desea cambiar este funcionamiento en la distribución, pulse con el botón derecho del ratón sobre el nodo de servicios basados en EJB de la WebApp de los servicios web y, a continuación, seleccione Propiedades. Quite la marca de todos los grupos o beans que no desee exponer.

7 Pulse con el botón derecho del ratón sobre el nodo eargrp, que contiene el EAR, y seleccione Opciones de distribución | Distribuir para distribuir el EAR en el servidor WebLogic.

8 Pruebe la distribución EJB en la página de inicio de WebLogic tal y como se describe en "Comprobación de servicios distribuidos" en la página 8-9.

9 Copie el código de ejemplo de la página de inicio y modifíquelo para crea una cliente que llame a y compruebe el servicio web exportado. Si necesita un ejemplo, consulte "Escritura del código del cliente y consumo del servicio de forma local" en el Capítulo 18, "Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server".

Tenga en cuenta que si modifica la aplicación EJB después de exportarla como un servicio, tiene que exportarla y distribuirla de nuevo para que los cambios surtan efecto.

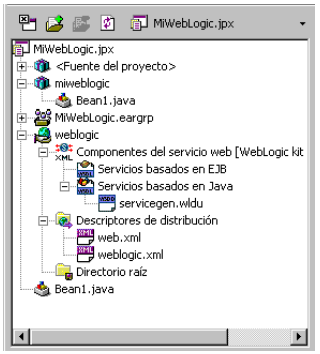
Si desea obtener un tutorial que muestre el ciclo completo de servicios web utilizando EJB, consulte el Capítulo 18, "Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server".

Distribución de servicios web

El asistente Exportar como servicio web crea un archivo descriptor de distribución de servicios web XML llamado `servicegen.wldu` que proporciona información de distribución al servidor WebLogic. Cuando los EJB se exportan como servicios web (JBuilder los exporta automáticamente sin necesidad de utilizar un asistente), se genera un archivo `.wldu` para cada módulo EJB del proyecto.

Los archivos WLDU

El archivo `servicegen.wldu` aparece en el panel del proyecto como nodo dependiente del nodo WebApp que hospeda el servicio web. Si se exporta una clase Java, aparece en el nodo Servicios basados en Java de la webapp (WebApp | Componentes del servicio web | Servicios basados en Java). Los archivos WLDU EJB aparecen en el nodo Servicios basados en EJB de la webapp. Si el proyecto cuenta con varios módulos EJB, se genera un archivo `servicegen.wldu` para cada módulo EJB, como `[móduloejb1]servicegen.wldu`, `[móduloejb2]servicegen.wldu`, etc.



El archivo WLDU, que contiene todo lo que hay dentro del elemento `<servicegen>` de `servicegen.xml`, proporciona la información que el servidor necesita sobre el servicio para poder crear las clases como, por ejemplo, el nombre EAR, el nombre WAR, el nombre de la clase, el nombre del servicio, la URI del servicio, etc. Puede modificar este archivo en cualquier momento y cambiar cualquier valor. Si desea más información acerca de `servicegen.xml`, consulte la información de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/anttasks.html#1063540`.

```
<?xml version="1.0"?>
<servicegen destEar="bean.ear" overwrite="true" warName="weblogic.war">
    <service javaClassComponents="bean.Bean1" serviceName="Bean1"
        serviceURI="bean" targetNamespace="bean" protocol="http"
        style="rpc" expandMethods="true" generateTypes="true"/>
</servicegen>
```

Edición de archivos WLDU para EJB

La distribución EJB de los servicios web se puede personalizar de dos modos:

- Modificación de las propiedades del nodo de distribución
- Edición del WLDU y modificación del elemento <documentation>

Modificación de las propiedades del nodo de distribución EJB

Por defecto, JBuilder expone automáticamente todos los beans sesión sin estado con métodos empresariales en la interfaz remota. También se puede modificar este funcionamiento por defecto y seleccionar sólo los módulos de Enterprise JavaBean (EJB), beans y métodos que desee exponer como servicios web.

El comportamiento de distribución automático se puede modificar en la ficha Módulos del cuadro de diálogo Propiedades de servicios basados en EJB:

1. Amplíe el nodo WebApp de WebLogic en el panel del proyecto, amplíe el nodo Componentes del servicio web, pulse con el botón derecho del ratón sobre el nodo de servicios basados en EJB y seleccione Propiedades. En la ficha Módulos, seleccione los grupos EJB vacíos y los beans sesión sin estado que desee exponer como servicios web. Los beans sesión sin estado deben contar como mínimo con un método válido en la interfaz remota. Por defecto se exponen todos los beans, que se seleccionan en la ficha Propiedades.

2. Quite la marca de todos los grupos EJB vacíos y/o beans que no desee exponer como servicios web.
3. Pulse Aceptar para cerrar la ficha Propiedades de servicios basados en EJB.
4. Vuelva a generar y a distribuir el proyecto.

Edición del WLDU y modificación del elemento <documentation>

En el caso de los EJB, una vez que se haya modificado el archivo WLDU, debe dar instrucciones al kit de herramientas WebLogic para que no genere automáticamente el archivo WLDU y para que JBuilder no lo sobreescriba al generar el proyecto. El `servicegen.wldu` para un EJB exportado dispone un elemento `<documentation>` para cada elemento `<service>` que se pueda modificar, con el fin de evitar que el kit de herramientas sobreescriba los cambios. Cambie la entrada de YES a NO en el elemento `<documentation>` de cada elemento `<service>` que se haya modificado:

```
<documentation>JBUILDER debe autogenerar esta entrada:
    NO(YES/NO)</documentation>
```

Tras haber modificado esta entrada, el kit de herramientas **no** sobreescribirá las modificaciones en el archivo `servicegen.wldu`. Debe generar de nuevo el proyecto para que se acepten los cambios.

web-services.xml

Cuando se genera el proyecto y se distribuye al servidor WebLogic, se transfieren los archivos de distribución WLDU al archivo `servicegen.xml` específico del kit de herramientas WebLogic, que se utiliza para ejecutar la tarea `servicegen` Ant. La tarea Ant servicegen coge el JAR EJB o una lista de clases Java y genera los componentes del servicio web y un archivo de distribución de servicios web, `web-services.xml`. El archivo descriptor de distribución `web-services.xml`, que se encuentra en el WAR del EAR, contiene información que describe un servicio web WebLogic como, por ejemplo, componentes en segundo plano (backend) que implementan el servicio web, los tipos de datos no integrados utilizados como parámetros y valores devueltos, el manejador de mensajes SOAP que interceptan los mensajes SOAP, etc. Si desea más información sobre la tarea Ant servicegen y `web-services.xml`, consulte la información de WebLogic, Programming WebLogic Web Services en `http://edocs.bea.com/wls/docs70/webserv/index.html` .

Nota El kit de herramientas WebLogic trabaja directamente con el EAR. Por lo tanto, el archivo `web-services.xml` y otros componentes de servicios web sólo se encuentran presentes en el EAR y no en la WebApp.

```
<web-services>
          <web-service name="stockquotes" targetNamespace="http://example.com"
                       uri="/myStockQuoteService">
               <components>
                    <stateless-ejb name="simpleStockQuoteBean">
                         <ejb-link path="stockquoteapp.jar#StockQuoteBean" />
                    </stateless-ejb>
               </components>
               <operations>
                    <operation method="getLastTradePrice"
```

```
                                component="simpleStockQuoteBean" />
                    </operations>
                </web-service>
            </web-services>
```

Distribución manual de servicios con web-services.xml

Si decide crear componentes de servicio web de forma manual y añadir otros elementos, tales como cadenas de mapeo de tipos y de manejadores, debe desactivar la opción Volver a generar distribución en la ficha Generar (Proyecto | Propiedades de proyecto | Generar | Servicios web). Al desactivar esta opción, JBuilder utiliza los componentes que se han creado manualmente y no genera los servicios web automáticamente.

Para crear componentes manualmente y añadir otros elementos, siga los siguientes pasos básicos:

1. Cree el archivo `web-services.xml` y guárdelo en el directorio `<contexto web >/WEB-INF` del proyecto.
2. Cree las clases del ayudante del servicio web en el proyecto, como los serializadores.
3. Desactive la opción Volver a generar distribución de la pestaña Servicios web de la ficha Generar (Proyecto | Propiedades de proyecto | Generar | Servicios web).
4. Genere el proyecto.

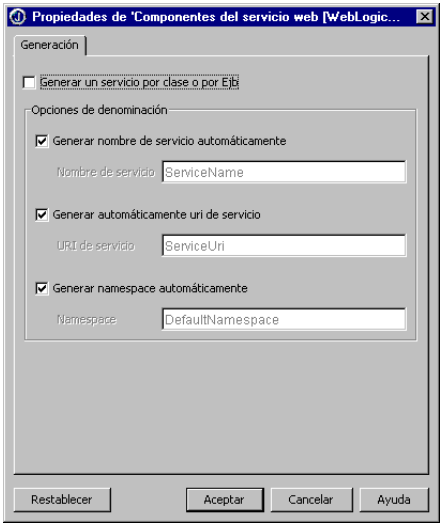
Si desea más información acerca de la creación manual de `web-services.xml`, consulte la documentación de WebLogic en `http://edocs.bea.com/wls/docs70/webserv/dd.html#1052144`.

Configuración de los nombres de servicios por defecto

Puede modificar las nombres de los servicios por defecto en el cuadro de diálogo Componentes de servicio web [kit de herramientas WebLogic] Propiedades. Haga clic con el botón derecho del ratón sobre el nodo Componentes de servicio web en el panel del proyecto y selecciones Propiedades. Aquí puede elegir cómo generar el servicio y seleccionar las opciones de asignación de nombres. Para obtener más información sobre estas opciones, pulse el botón Ayuda.

Si se modifican estas propiedades, se reflejan inmediatamente en los archivos WLDU para los EJB. Sin embargo, en lo que respecta a las clases

Java, los cambios no se reflejan a menos que se vuelvan a exportar como servicio web, ya que los asistentes las crean estáticamente.



Capítulo

9

Utilización del kit de herramientas SOAP 2 de Apache

Es una característica de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas SOAP 2 de Apache.

Si utiliza el conjunto de herramientas SOAP 2 de Apache, necesitará realizar algunas modificaciones al utilizar los asistentes de JBuilder y añadir código manualmente. Debido a que SOAP 2 es una versión anterior de Axis de Apache y es anterior a la especificación WSDL, no existe compatibilidad WSDL integrada para SOAP. Sin embargo, puede utilizar los asistentes Importar un servicios web y Exportar como servicio web y el kit de herramientas Axis, ya que JBuilder cuenta con un Axis para el distribuidor múltiple SOAP de Apache que convierte la información de distribución de Axis en SOAP de Apache.

Advertencia Tenga en cuenta que SOAP 2 de Apache es una versión anterior de Axis por lo que **no** es compatible con JAX-RPC o WSDL. Debido a estas limitaciones, es recomendable que utilice Axis en su lugar.

Para obtener más información sobre el kit de herramientas SOAP 2 de Apache, consulte la documentación que reside en `<jbuilder>/thirdparty/apache-soap/docs` o visite el sitio web de Apache en `http://xml.apache.org/soap/`

Cree este ejemplo sencillo y, a continuación, realice los cambios apropiados en los asistentes y en el documento WSDL.

1. Cree un proyecto denominado `SinTitulo` mediante Archivo | Proyecto nuevo.
2. Cree un método JavaBean denominado `Bean1` con el Asistente para JavaBean, seleccione `java.lang.Object` como clase base y active la opción Generar propiedad de ejemplo (Archivo | Nuevo | General).

3 Haga clic con el botón derecho del ratón sobre `Bean1.java` en el panel del proyecto y seleccione Exportar clase como un servicio web. Debido a que el proyecto no está configurado para servicios web, el Asistente para la configuración de servicios web se abre en primer lugar para que pueda configurar el proyecto.

4 Cree una WebApp denominada `soapsample` para que hospede el servicio, seleccione SOAP 2 de Apache como kit de herramientas en el Asistente para la configuración de servicios web y pulse Finalizar. Ahora se abre el asistente Exportar como servicio web.

5 Seleccione Axis como kit de herramientas en el asistente Exportar como servicio web y realice los cambios que se especifican en "Exportación de una clase como servicio web" en la página 9-2. Tenga en cuenta que no hay ningún kit de herramientas SOAP disponible en el asistente ya que SOAP no admite WSDL.

Exportación de una clase como servicio web

Para poder exportar una clase como servicio web, es necesario realizar los siguientes cambios en el asistente Exportar como servicio web:

1 Asimismo, desactive la opción Generar stub cliente en el paso 1.

2 En el paso 2 del asistente, cambie la dirección URL de `"http://localhost:8080/axis/services/Bean1"` a `"http://localhost:8080/<soapsample>/servlet/rpcrouter/"`, donde `<soapsample>` es el nombre de la webapp.

3 En el paso 4, seleccione los métodos que desee exponer.

   a En la lista desplegable Modo de selección, seleccione Permitir todos los métodos.

   b Seleccione los métodos que desee exponer en el árbol.

4 Pulse Finalizar para cerrar el asistente.

Importación de archivos WSDL

Antes de importar un archivo WSDL para crear un cliente, debe modificar el WSDL. A continuación, importe el archivo WSDL mediante el asistente Importar un servicio web.

1 En la sección de asociaciones (binding) del archivo WSDL, modifique los atributos namespace (espacio de nombres) comprendidos en las etiquetas input (entrada) y output (salida) de la definición de operación (operation) de modo que se correspondan con el puerto del servicio. Por ejemplo, cambie `namespace="http://SinTítulo"` al nombre del puerto,

"Bean1". Este cambio es necesario debido a que el servlet repartidor de SOAP necesita saber en qué servicio debe repartir y el espacio de nombre es el mecanismo que utiliza. Tenga en cuenta que en la próxima generación de kits de herramientas, como Axis, no es necesario realizar este cambio, ya que se utiliza el mapeo URL.

WSDL antes de su modificación

```
<?xml version="1.0" encoding="UTF-8"?>
      <wsdl:definitions targetNamespace="http://untitled1"
       xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:apachesoap="http://xml.apache.org/xml-soap"
       xmlns:impl="http://untitled1-impl" xmlns:intf="http://untitled1"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     [...  rest of WSDL snipped for brevity  ...]
   <wsdl:binding name="Bean1SoapBinding" type="intf:Bean1">
      <wsdlsoap:binding style="rpc"
          transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getSample">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSampleRequest">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://SinTitulo" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getSampleResponse">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://SinTitulo" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="setSample">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="setSampleRequest">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://SinTitulo" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="setSampleResponse">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://SinTitulo" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="Bean1Service">
      <wsdl:port binding="intf:Bean1SoapBinding" name="Bean1">
         <wsdlsoap:address location=
             "http://localhost:8080/soapsample/servlet/rpcrouter"/>
      </wsdl:port>
```

```
   </wsdl:service>
</wsdl:definitions>
```

WSDL modificado

```
<?xml version="1.0" encoding="UTF-8"?>
   <wsdl:definitions targetNamespace="http://untitled1"
       xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://untitled1-impl"
    xmlns:intf="http://untitled1"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     [...  rest of WSDL snipped for brevity  ...]
   <wsdl:binding name="Bean1SoapBinding" type="intf:Bean1">
      <wsdlsoap:binding style="rpc" transport=
          "http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getSample">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSampleRequest">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="Bean1" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getSampleResponse">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="Bean1" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="setSample">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="setSampleRequest">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="Bean1" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="setSampleResponse">
            <wsdlsoap:body encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="Bean1" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="Bean1Service">
      <wsdl:port binding="intf:Bean1SoapBinding" name="Bean1">
         <wsdlsoap:address location=
             "http://localhost:8080/soapsample/servlet/rpcrouter"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

2. Haga clic sobre el archivo WSDL modificado con el botón derecho del ratón y seleccione Importar un servicio web para que se abra el asistente Importar un servicio web.
3. Escoja Axis de Apache como kit de herramientas.
4. Desactive Generar las clases del servidor en el Paso 2 del asistente Importar un servicio web.
5. Cambie el nombre del paquete a `SinTitulo.generated` en el paso 3 del asistente. Si no desea cambiar el nombre del paquete, el asistente sobreescribirá la clase `Bean1` que se está exportando.
6. Pulse el botón Finalizar.
7. Genere y ejecute el proyecto con la configuración Servidor de servicios web.
8. Haga clic con el botón derecho del ratón sobre el test y ejecútelo para comprobar el servicio.

Si prefiere codificar el cliente manualmente, siga este ejemplo.

```
public class Test
{
  public static void main(String[] args) throws Exception
  {

    URL url = new URL("http://localhost:8082/soapsample/servlet/rpcrouter");
//  String nameToLookup = "Bean1";

    Call call = new Call();

    call.setTargetObjectURI("Bean1");
    call.setMethodName("getSample");
    call.setEncodingStyleURI(encodingStyleURI);

//  Vector params = new Vector();
//
//  params.addElement(new Parameter("sample", String.class, nameToLookup, null));
//  call.setParams(params);

    // Realizar la llamada.
    Response resp;

try
    {
      resp = call.invoke(url, "");
    }
    catch (SOAPException e)
    {
      System.err.println("Caught SOAPException (" +
                         e.getFaultCode() + "): " +
                         e.getMessage());
    return;
```

```
        }

        // Comprobar la respuesta.
        if (!resp.generatedFault())
        {
          Parameter ret = resp.getReturnValue();
          Object value = ret.getValue();

          System.out.println(value != null ? "\n" + value : "Desconocido.");
        }
        else
        {
          Fault fault = resp.getFault();

          System.err.println("Generated fault: ");
          System.out.println ("  Código erróneo   = " + fault.getFaultCode());
          System.out.println ("  Cadena errónea = " + fault.getFaultString());

        }
      }
    }
```

Capítulo

10

Modificar la configuración del servidor de aplicaciones empresariales para servicios web

Es una función de JBuilder Enterprise.

La funcionalidad de servicios web de JBuilder se puede utilizar con los siguientes servidores de aplicaciones empresariales:

- Borland Enterprise Server 5.0.2-5.1.x
- WebLogic Server 7.0 (con o sin SP1)
- WebSphere Application Server 4.0 AES/AE

Ya que los servidores de aplicaciones empresariales requieren configuración diferente cuando se utilizan las funciones de servicios web, puede que necesite llevar a cabo modificaciones al utilizar los asistentes para servicios web y al distribuir su aplicación.

Si desea obtener más información para configurar estos servidores en JBuilder, consulte "Configuración del servidor de aplicaciones de destino" en la *Guía del desarrollador de Enterprise JavaBeans* y las Notas de esta versión (Ayuda | Notas de esta versión).

Borland Enterprise Server 5.0.2-5.1.x

Es una característica de JBuilder Enterprise. La Edición WebLogic de JBuilder no es compatible con este servidor de aplicaciones

Cuando se ejecutan múltiples particiones en JBuilder con archivos distribuidos a través de estas particiones, asegúrese de que el servicio de denominación sólo está activado para una de ellas. Para ello:

1. Haga clic en Proyecto | Propiedades de proyecto.
2. Pulse sobre la pestaña Ejecutar.

3. Modifique la configuración de ejecución del servidor. **Nota:** Si no hay una configuración de ejecución del servidor, cree una nueva y pulse la pestaña Servidor.
4. Desactive el servicio Denominación/Directorio en la lista Servicios. Utilice esta configuración para iniciar particiones sin el servicio de denominación.
5. Cree una nueva configuración de ejecución con el servicio de denominación activado para iniciar una de las particiones con este servicio.
6. Asegúrese de que el EJB y los servicios de denominación/directorio se encuentran activados en la misma partición y de que se inicia esta participación en primer lugar.

WebLogic Server 7.0 (con o sin SP1)

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache y SOAP 2 de Apache

Si está utilizando Axis con el servidor WebLogic 7.0, es necesario que modifique la configuración de ejecución para que incluya las bibliotecas del proyecto, y así encontrar Axis al ejecutar el servidor. Si elige la opción Hacer que las bibliotecas del proyecto estén disponibles al ejecutar, se añade Axis a la vía de acceso a clases. Si no selecciona esta opción, no puede tener acceso a los servicios web EJB. Modifique la configuración de ejecución del siguiente modo:

1. Seleccione Herramientas | Configurar servidores y escoja WebLogic Application Server 7.x.
2. Haga clic en la pestaña General.
3. Abra la pestaña Bibliotecas necesarias y añada cualquier biblioteca que necesite al iniciar el servidor.

WebSphere Application Server 4.0 AES/AE

Es una característica de JBuilder Enterprise. La Edición WebLogic de JBuilder no es compatible con este servidor de aplicaciones

Después de crear el nodo del servidor de servicios web con el Asistente para configuración de servicios web, mueva Axis a la parte superior de la lista de bibliotecas necesarias del cuadro de diálogo Propiedades de proyecto:

1. Seleccione Proyecto | Propiedades de proyecto.
2. Seleccione la pestaña Bibliotecas necesarias de la ficha Vías de acceso.
3. Seleccione Axis y pulse el botón Desplazar hacia arriba para moverlo al principio de la lista.

Compruebe que no ha marcado la opción Hacer que las bibliotecas del proyecto estén disponibles al ejecutar cuando inicie el servidor; en caso

contrario, el servidor no se inicia debido a los conflictos con las versiones del analizador XML.

Si desea añadir bibliotecas a la vía de acceso a clases del servidor en el inicio, hágalo en la ficha Bibliotecas necesarias de la configuración del servidor para WebSphere 4.0 AE/AES (Herramientas|Configurar servidores).

Capítulo

11

Búsqueda y publicación de servicios web

Es una característica de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Ya que cualquier persona puede crear y consumir servicios web, es importante contar con un mecanismo para registrar, difundir y encontrar un servicio. En la actualidad, el marco de trabajo UDDI (Universal Description, Discovery and Integration, Descripción, detección e integración universales) se puede utilizar para registrar y detectar un servicio. Además de buscar registros UDDI para los servicios web, también se pueden buscar servicios web disponibles en sitios web mediante documentos WSIL, al igual que buscar servicios en servidores Axis. JBuilder proporciona el Explorador de servicios web para que le ayude en la búsqueda y difusión de servicios web.

Descripción general del Explorador de servicios web

JBuilder proporciona el Explorador de servicios web, disponible en el menú Herramientas, para buscar en los sitios los servicios disponibles y para publicarlos en los registros UDDI. Mediante el Explorador de servicios web, puede buscar negocios, sitios web o tModels, y también añadir negocios y entradas tModel. También se puede realizar un seguimiento del tráfico de mensajes desde y hacia los sitios UDDI en el Monitor de mensajes UDDI. Para obtener más información sobre UDDI, consulte "Aspectos generales de UDDI" en la página 11-3.

Importante La función UDDI del Explorador de servicios web requiere el uso de la extensión Java Secure Sockect Extension (JSSE) de Sun. Si está utilizando el JDK 1.3 o una versión anterior, debe descargar e instalar la Extensión JSSE de Sun en `http://java.sun.com/products/jsse/index-103.html`.

También permite examinar páginas en las que se ejecutan servicios web hospedados por Apache Axis. En estas páginas se proporciona información general que describe los servicios web disponibles. Un documento WSDL describe cada servicio web. Si desea obtener más información acerca de las funciones de Axis en el Explorador de servicios web, consulte "Aspectos generales de Axis" en la página 11-5.

Además, el Explorador de servicios web admite la búsqueda de documentos WSIL (Lenguaje de inspección de servicios web). Los documentos WSIL son un conjunto de referencias a servicios web disponibles en un sitio web que están albergados por el proveedor de servicios web. Estas referencias pueden apuntar a otros documentos WSIL, entradas UDDI o documentos WSDL.

El Explorador de servicios web también proporciona un enlace al asistente Importar un servicio web, que genera clases Java a partir de un documento WSDL seleccionado. Busque cualquier servicio web disponible en un sitio UDDI, un sitio Axis u otro sitio web que recoja los servicios, y seleccione el nodo que especifique el documento. A continuación, pulse el botón Importar un servicio web en el Explorador para generar clases Java rápidamente. Si no tiene abierto ningún proyecto, se abre primero el Asistente para proyectos y, a continuación, el asistente Importar un servicios web. Después de crear un servicio web, puede utilizar las funciones de publicación del Explorador de servicios web para difundir su servicio web en los registros UDDI.

Consulte

- "Adición y eliminación de nodos en el árbol del Explorador" en la página 11-6.
- "Menús del Explorador de servicios web" en la ayuda en línea.
- Capítulo 5, "Utilización de WSDL".
- "Difusión de negocios y servicios" en la página 11-24.
- Capítulo 19, "Tutorial: Búsqueda de servicios web UDDI".

Aspectos generales de UDDI

El marco de trabajo *Universal Description, Discovery and Integration* (Descripción, detección e integración universales, UDDI) proporciona un mecanismo para buscar, describir y registrar servicios basados en web en un registro de negocios central en Internet. A través de este registro, las empresas pueden encontrar socios de negocios y conectarse de forma dinámica con sus servicios web, además de publicar sus servicios.

Este marco basado en XML utiliza el Protocolo de acceso a objetos sencillos (SOAP) y HTTP para detectar un servicio en concreto en el registro utilizando mensajes XML para realizar llamadas de procedimiento remotas y enviar información desde y hacia el registro. XML, http y SOAP ofrecen la ventaja añadida de la programación multiplataforma e independiente del lenguaje.

Un registro UDDI no contiene las especificaciones reales de cómo los negocios comparten sus productos y servicios de forma electrónica. En su lugar, contiene apuntes o referencias que enlazan con esa información. Algunas de estas referencias pueden ser archivos WSDL (Lenguaje de descripción de servicios web), que describen un servicio web.

Los registros UDDI pueden ser privados, públicos e incluso locales, en su propio equipo. Algunos registros, como el Registro de negocios UDDI, se actualizan y replican con la misma información, a cargo de varios sitios de operadores UDDI. Otros registros son públicos, pero no duplican su información con otros sitios.

El Registro de negocios UDDI es un registro UDDI universal que se mantiene y opera como un servicio distribuido por varios sitios de operadores. Estos sitios de operadores, como Microsoft e IBM, replican los datos los unos en los otros para sincronizar las instancias del registro. También cuentan con sitios UDDI de prueba, en los que se pueden probar exhaustivamente los servicios web antes de publicarlos.

UDDI acepta y organiza tres tipos de información de negocios:

- Publicar – “páginas blancas”, con la información del negocio, como los datos de contacto o la dirección.
- Buscar – “páginas amarillas”, con la información de los servicios del negocio, organizada por categorías industriales y situación geográfica.
- Asociar – “páginas verdes”, con la información técnica y las especificaciones sobre los servicios, necesaria para conectarse mediante programa.

Consulte

- "Búsqueda en un registro UDDI" en la página 11-8.
- "Difusión de servicios web en un registro UDDI" en la página 11-24.
- El proyecto UDDI en `http://www.uddi.org`
- El sitio UDDI de Microsoft en `http://uddi.microsoft.com`
- El sitio UDDI de IBM en `https://www-3.ibm.com/services/uddi/protect/registry.html`

Términos y definiciones UDDI

El Explorador de servicios web utiliza numerosos tipos de datos y términos UDDI que se pueden encontrar en la especificación UDDI. Aunque esta documentación supone que el usuario está familiarizado con la especificación UDDI, a continuación se definen los términos principales de UDDI. Si desea obtener información más completa, consulte la especificación UDDI en `http://uddi.org/specification.html`.

Tabla 11.1 Términos UDDI

Término UDDI	Definición
accessPoint	Una dirección de punto de entrada para obtener acceso a un servicio web. Puede ser una URL, una dirección de correo electrónico o un número de teléfono.
businessKey	Un identificador único para una entidad de negocios dada.
category	Información de categoría que es parecida a un identificador pero que utiliza una taxonomía predeterminada, como códigos industriales, de productos o geográficos. También utiliza pares nombre/valor. Entre los ejemplos de estos códigos se incluyen: NAICS (North American Industry Classification System), UNSPSC 3.1 (United Nations Standard Products and Services Code System) y SIC (Standard Industrial Classification).
identifier	Un par nombre/valor que actúa como un identificador para un negocio. Entre los ejemplos se incluyen el D-U-N-S ID (Dun & Bradstreet's Data Universal Numbering System) y el Thomas Register ID.
keyName	Un comentario que ayuda en la legibilidad, pero que normalmente no es necesario. Por ejemplo, el D-U-N-S ID para IBM tiene un keyName de D-U-N-S, mientras que keyValue es el número real D-U-N-S de nueve dígitos, 00-136-8083. Sin embargo, keyName no tiene un significado importante en la categoría uddi-org:misc-taxonomy.
keyValue	Especifica una clasificación de negocios, como el número D-U-N-S de nueve dígitos utilizado en los negocios de todo el mundo.

Tabla 11.1 Términos UDDI (continuación)

Término UDDI	Definición
sitio de operadores	Es cada una de las instancias de un registro de negocios UDDI. Por ejemplo, Microsoft e IBM mantienen instancias del Registro de negocios UDDI.
URL del documento explicativo	Es un puntero a la ubicación de la descripción del documento explicativo.
serviceKey	Una clave única para un servicio de negocios que se genera al registrar el servicio.
tModel	Es una referencia a una especificación técnica de un servicio. Los tModels son descripciones de servicios Web que definen los tipos de servicio. Cada tModel tiene un identificador y apunta a una especificación que describe el servicio web. Los tModels proporcionan un punto de referencia común que permite identificar con facilidad los servicios compatibles.
detalles de tModelInstance	Una lista de estructuras de información de tModel que actúa como una huella dactilar para el servicio. Incluye detalles como la descripción, el nombre tModel, la clave tModel y la documentación.
tModelKey	Un identificador único para un tModel que asigna el UDDI cuando se registra el servicio.

Aspectos generales de Axis

Esta es una función del kit de herramientas Axis. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

El Explorador de servicios web le permite examinar sitios web en los que se ejecutan servicios web hospedados por Axis de Apache. La búsqueda en estos sitios devuelve información descriptiva de los servicios web disponibles en forma de documento WSDL. JBuilder genera servidores Axis con el acceso remoto desactivado. Para buscar servicios en los servidores Axis, debe activar primero el acceso remoto. Consulte "Acceso remoto a los servidores Axis" en la página 11-19.

Consulte

- "Búsqueda de servicios web en un servidor Axis" en la página 11-17
- "Difusión de servicios web desde un servidor Axis" en la página 11-26
- Capítulo 7, "El kit de herramientas Axis de Apache"

Aspectos generales de WSIL

WSIL, al igual que UDDI, proporciona un método de detección de servicios web. A diferencia de UDDI, WSIL utiliza un modelo descentralizado y distribuido, en lugar de un modelo centralizado. Los documentos WSIL son, básicamente, punteros de listas de servicios que permiten a los clientes de servicios Web examinar los servicios disponibles que no se encuentran en los registros UDDI. La norma WSIL establece la forma de utilizar documentos con formato XML estándar para inspeccionar sitios web en busca de servicios y series de reglas que determinan la forma en que se proporciona la información. Los documentos WSIL contienen varias referencias a documentos de descripción de servicios anteriores. El proveedor del servicio aloja el documento WSIL de forma que los consumidores pueden encontrar los servicios disponibles.

Consulte

- "Búsqueda de servicios web con documentos WSIL" en la página 11-20

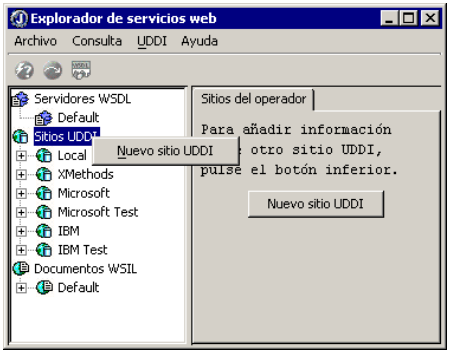
Adición y eliminación de nodos en el árbol del Explorador

El Explorador de servicios web tiene disponibles varios nodos en le vista de árbol que le pueden ayudar a buscar los servicios web disponibles en diferentes ubicaciones:

- Servidores WSDL, una función del kit de herramientas Axis
- Sitios UDDI
- Documentos WSIL

Amplíe el nodo Servidores WSDL, una función del kit de herramientas Axis, y consulte si existe un nodo Por defecto, que apunta a una ubicación común para un entorno de pruebas. El nodo Sitios UDDI tiene varios nodos UDDI disponibles, como Local, XMethods, Microsoft, IBM y Sitios UDDI de prueba. El nodo Documentos WSIL también contiene un nodo

Por defecto. Además, puede añadir nodos a los ya creados y eliminar nodos en el árbol del Explorador.



El contenido de los nodos Servidores WSDL y Sitios UDDI en el Explorador se lee a partir de archivos XML del directorio `defaults` de JBuilder. Si realiza algún cambio en uno de estos nodos, los cambios se guardan en el directorio `<inicio>/<.jbuilder>`. El archivo Documentos WSIL se genera después de añadir un nodo, y también se guarda en el directorio `<inicio>/<.jbuilder>`. Los nombres de los archivos son los siguientes:

- Nodo Servidores WSDL (servidores Axis): `AxisServers.xml`
- Nodo Sitios UDDI: `UDDIOps.xml`
- Nodo Documentos WSIL: `WSILDocs.xml`

Para añadir un nodo nuevo a los nodos Servidores WSDL, Sitios UDDI y Documentos WSIL:

1. Seleccione el nodo en la vista de árbol. Aparece una ficha a la derecha: ficha Servidores WSDL, ficha Sitios de operadores o ficha Documentos WSIL.
2. Cree un nodo mediante uno de los métodos siguientes:
   - Pulse el botón Nuevo en la ficha de la derecha.
   - Seleccione Archivo | Nuevo.
   - Haga clic con el botón derecho sobre el nodo y seleccione Nodo.
3. Escriba el nombre que desee para el nuevo nodo en el cuadro de diálogo de entrada y, a continuación, pulse Aceptar.
4. Seleccione el nuevo nodo en el árbol para que aparezca la información de detalle a la derecha.
5. Escriba la información adecuada, como URL de difusión y búsqueda para los sitios UDDI o URL de un servidor Axis o un documento WSIL.
6. Pulse el botón Guardar.

Para eliminar un nodo del árbol del Explorador:

1. Elija el nodo que desee eliminar para que aparezca a la derecha la ficha Detalles.
2. Pulse el botón Borrar de la ficha Detalles, seleccione Archivo | Borrar, o bien, pulse con el botón derecho del ratón sobre el nodo del árbol y seleccione Borrar.

Nota Los nodos de los niveles superiores no se pueden borrar.

Búsqueda en un registro UDDI

Existen varios modos de buscar servicios web en un registro UDDI con el Explorador de servicios web:

- Buscar negocios.
- Buscar servicios.
- Buscar tModels.

Importante Es posible aumentar el tiempo de espera del Explorador de servicios web. Elija UDDI | Configuración de conexión y escriba un nuevo valor en el campo Tiempo de espera. Para aplicar este valor a las conexiones, cierre el Explorador y vuelva a abrirlo. Es posible que el servidor detenga la búsqueda si determina que el resultado es demasiado voluminoso.

Búsqueda de negocios

Puede buscar un negocio determinado con el que desee trabajar y ver los tipos de servicios que ofrece. También puede buscar negocios de una industria determinada, como la publicación de software. Existen varios modos de buscar un negocio mediante el Explorador de servicios web:

- Búsqueda por nombre de negocio.
- Búsqueda por categoría de negocio.
- Búsqueda por identificador de negocio.
- Búsqueda por tModel.

Búsqueda por nombre

Para buscar un negocio en concreto por el nombre en el Explorador de servicios web, siga los pasos siguientes:

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Amplíe el nodo Sitios UDDI en el árbol de la izquierda y haga doble clic sobre el nodo de sitios de operadores UDDI para ampliarlo.

3 Seleccione el nodo Lista de negocios para abrir la ficha Consulta por negocio a la derecha.

4 Escriba el nombre del negocio o las primeras letras del nombre en el campo Buscar por nombre de la pestaña Nombre. Si deja este campo en blanco, se buscan todos los negocios del sitio.

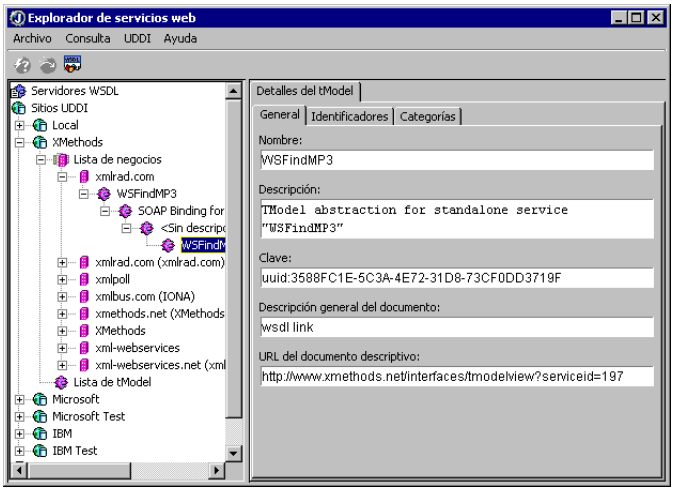
Nota El Explorador de servicios web admite el símbolo % como carácter comodín.

5 Ejecute la consulta mediante uno de estos métodos:

- Seleccione Consulta | Ejecutar.

- Haga clic en el botón Ejecutar consulta de la barra de herramientas.
- Pulse el botón Ejecutar de la ficha Consulta por negocio a la derecha del árbol.
- Pulse `Intro` en el campo Buscar por nombre después de escribir el nombre para buscar.

6 Amplíe el nodo Lista de negocios para ver los resultados de la consulta y poder desplegarlos para averiguar más acerca de un negocio y sus servicios.



Búsqueda por categoría

Otro modo de buscar un negocio es buscar por la categoría de negocio. Hay dos clasificaciones de negocios por las que puede buscar:

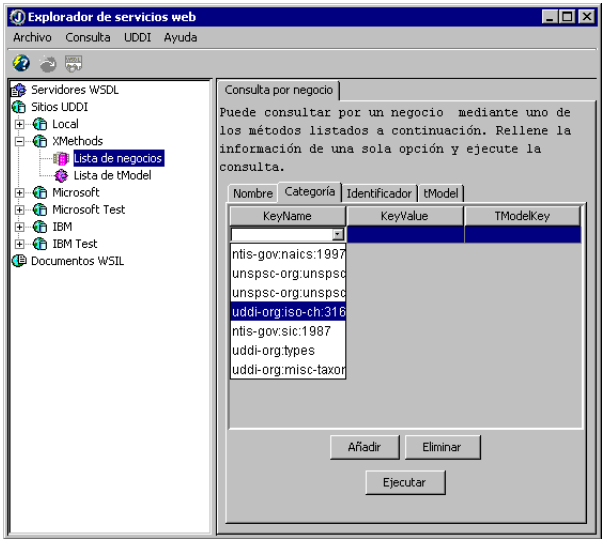
- ntis-gov:naics:1997 (North American Industry Classification System)
- unspsc-org:unspsc:3-1 (United Nations Standard Products and Services Code System)
- unspsc-org:unspsc (Universal Standard Products and Services Classification)
- uddi-org:iso-ch:3166-1999 (códigos para la ubicación geográfica)
- ntis-gov:sic:1987 (Standard Industrial Classification)
- uddi-org:types
- uddi-org:misc-taxonomy

Nota También se pueden utilizar otros sistemas de clasificación. Si un esquema no está disponible en la lista desplegable, puede pegar la clave del esquema en el campo tModelKey. También puede añadir o eliminar criterios de búsqueda mediante los botones Añadir y Eliminar.

Cada uno de estos sistemas de clasificación de negocios tiene sus propios códigos para las distintas categorías. Las grandes empresas que realizan varios tipos de negocios se pueden clasificar en varios sistemas y bajo diversas clasificaciones en cada sistema. Por ejemplo, una empresa puede vender software y hardware para ordenadores. En NAICS, este negocio se podría recoger bajo varias clasificaciones, como formación en informática,

servicios de procesado de datos y difusión de software. Este mismo negocio se podría clasificar también en UNSPSC como instrucciones de programación informática, software de bases de datos y ordenadores centrales.

Estas clasificaciones de negocios se recogen en la lista desplegable KeyName de la ficha Categoría del Explorador de servicios web. Si se desea realizar una clasificación más amplia, se utiliza un valor KeyValue para ofrecer una descripción más específica del negocio. Cada clasificación de negocios cuenta con su correspondiente tModelKey.



Para buscar por categoría de negocios:

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Amplíe el nodo Sitios UDDI en el árbol de la izquierda y haga doble clic sobre el nodo de sitios de operadores UDDI para ampliarlo.
3. Seleccione el nodo Lista de negocios para abrir la ficha Consulta por negocio a la derecha.
4. Seleccione la pestaña Categoría de la ficha Consulta por negocio.
5. Pulse el campo KeyName para activar la lista desplegable y seleccione una de las categorías de negocios.
6. Escriba un código apropiado para la categoría de negocios seleccionada en la columna KeyValue.

   Por ejemplo, si desea buscar todas las empresas editoras de software de un sitio UDDI, puede seleccionar `Ntis-gov:naics:1997` en la lista desplegable KeyName, y escribir el código de los editores de software

en NAICS en el campo KeyValue: `51121`. Cada tipo de categoría de negocios cuenta con su propio conjunto de códigos de clasificación.

7 Pulse *Intro* para confirmar el valor de KeyValue.

8 Pulse el botón Ejecutar de la ficha Consulta por negocio para realizar la consulta.

9 Amplíe el nodo Lista de negocios para ver la lista de negocios registrados en la categoría seleccionada.

Búsqueda por identificador

También es posible buscar un negocio a través del identificador de negocios. Existen varios esquemas de identificación integrados por los que se puede buscar:

- Thomasregister-com:supplierID
- Dnb-com:D-U-N-S (Dun & Bradstreet Number Identifier System)

Estos identificadores se encuentran recogidos en la lista desplegable KeyName de la ficha Identificador del Explorador de servicios web. Cada una de estas clasificaciones cuenta con su tModelKey correspondiente.

Para buscar por identificador de negocios:

1 Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.

2 Amplíe el nodo Sitios UDDI en el árbol.

3 Amplíe un nodo de un sitio de un operador UDDI y elija el nodo Lista de negocios, para mostrar a la derecha la ficha Consulta por negocio.

4 Seleccione la pestaña Identificador de la ficha Consulta por negocio.

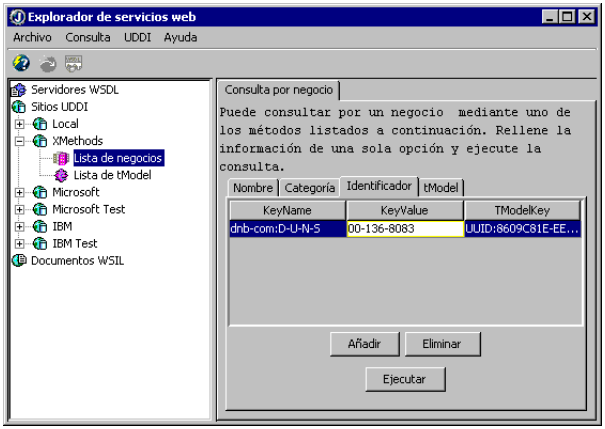
5 Elija una de los siguientes identificadores de negocios en la lista desplegable KeyName:

   - Thomasregister-com:supplierID
   - Dnb-com:D-U-N-S

6 Escriba el código apropiado del identificador de negocios seleccionado en la columna KeyValue.

   Por ejemplo, si desea buscar un negocio con un identificador D-U-N-S de 00-136-8083 en un sitio UDDI, seleccione `Dnb-com:D-U-N-S` de la lista

desplegable KeyName y escriba el número de identificador en el campo KeyValue: `00-136-8083`. Este es el identificador D-U-N-S para IBM.



7 Pulse *Intro* para confirmar el valor de KeyValue.

8 Pulse el botón Ejecutar para ejecutar la consulta.

9 Amplíe el nodo Lista de negocios para ver los negocios registrados con el identificador seleccionado.

Búsqueda de servicios

Sólo es posible buscar por servicio cuando se ha elegido una empresa. Antes de buscar un servicio, debe buscar el nombre de un negocio y seleccionar el nodo del negocio en el árbol de la izquierda.

Para buscar un servicio, siga los pasos siguientes:

1 Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.

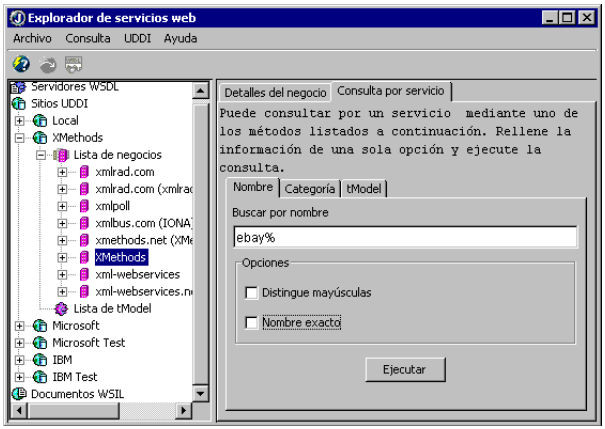
2 Amplíe el nodo Registros UDDI del árbol de la izquierda.

3 Amplíe un nodo de un sitio de un operador UDDI y ejecute Consulta por negocio tal y como se explica en "Búsqueda por nombre" en la página 11-8.

4 Amplíe el nodo Lista de negocios para ver los resultados de la consulta.

5 Seleccione un árbol de la lista y elija la pestaña Consulta por servicio de la derecha.

6 Escriba el nombre del servicio web en el campo Buscar por nombre.



7 Pulse Ejecutar para realizar la consulta.

8 Amplíe el nodo de negocios para ver los servicios que devuelve la consulta.

Nota También puede buscar un servicio por la categoría o tModel.

Búsqueda de tModels

En el Explorador de servicios web también puede buscar tModels. Los tModels representan una especificación técnica de un servicio web. La búsqueda de tModels se puede hacer por nombre de tModel, categoría e identificador.

Búsqueda por nombre

Para buscar tModels por nombre:

1 Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.

2 Amplíe el nodo Sitios UDDI en el árbol de la izquierda y haga doble clic sobre el nodo de sitios de operadores UDDI para ampliarlo.

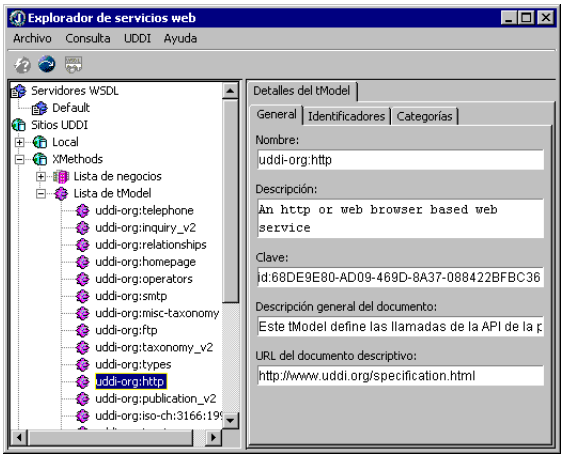
3 Elija el nodo Lista de tModel en el árbol para que se abra la ficha Consulta por TModel a la derecha.

4 Escriba el nombre del tModel en el campo Buscar por nombre.

Por ejemplo, si desea buscar todos los tModels que empiecen por el nombre `uddi-org`, debe escribir

5 Pulse el botón Ejecutar para ejecutar la consulta.

6 Amplíe el nodo Lista de TModel para ver el resultado de la consulta.



Examen de los resultados de las consultas UDDI

Una vez que ha realizado una consulta UDDI, puede buscar información más detallada acerca de un servicio web en el Explorador de servicios web. Cada uno de los nodos en el árbol del explorador muestra información diferente acerca del servicio seleccionado. Si selecciona un nodo en el árbol, la información correspondiente aparece a la derecha del árbol. Por ejemplo, al seleccionar el nodo de negocios en el árbol, aparece la ficha Detalles del negocio a la derecha con una descripción general del negocio, junto con la información de contacto, identificadores y categorías. Al ampliar el nodo de negocios y seleccionar un servicio de negocios, aparece la ficha Detalles del servicio, y así sucesivamente.

Fichas de detalles de UDDI

Las búsquedas de negocios y servicios devuelven varios nodos en el árbol, como el nombre del negocio, el nombre del servicio, la información de enlace, la instancia de tModel y los detalles de tModel. Según el servicio web y cómo se ha publicado, puede que no cuente con todas estas fichas de detalles. Las búsquedas de tModels sólo devuelven nodos tModel en el árbol, y sólo muestra la ficha Detalles de TModel. Algunos detalles de estas fichas son obligatorios y otros optativos, por lo que algunos pueden aparecer en blanco.

Existen varias fichas de detalles UDDI en el Explorador de servicios web:

- Detalles.
- Detalles del negocio.
- Detalles del servicio.

- Detalles de enlace.
- Detalles de la instancia de tModel.
- Detalles de tModel.

Ficha Detalles

La ficha Detalles aparece al seleccionar un nodo de sitios de operadores UDDI. En esta ficha se muestra la información de sitios de operadores, como el nombre, URL de búsqueda y URL de difusión.

Ficha Detalles del negocio

La ficha Detalles del negocio aparece cuando se selecciona el nodo del negocio. Esta ficha muestra información del negocio, como el nombre y descripción del tipo de negocio, información de contacto, identificadores y categorías de negocios.

Ficha Detalles del servicio

La ficha Detalles del servicio aparece al seleccionar en el árbol el nodo de servicios web. Esta ficha muestra información del servicio, como el nombre, descripción y la clave del servicio.

Ficha Detalles de enlace

La ficha Detalles de enlace aparece al seleccionar en el árbol el nodo del enlace. Esta ficha muestra una descripción, el punto de acceso al servicio y el tipo de URL.

Ficha Detalles de la instancia de tModel

La ficha Detalles de la instancia de tModel aparece al seleccionar en el árbol el nodo Detalles de la instancia de tModel, que normalmente aparece por defecto como <Sin descripción>. Esta ficha muestra una descripción, el nombre del tModel, una descripción de los aspectos generales y una URL para acceder a documentación explicativa.

Ficha Detalles de tModel

La ficha Detalles de Tmodel aparece al seleccionar en el árbol el nodo tModel, el último nodo de la Lista de negocios. También aparece cuando se selecciona un nodo tModel en la Lista de tModel. Esta ficha muestra los aspectos generales, como el nombre y la descripción del tModel, la clave del tModel, una descripción del documento de aspectos generales, una URL de documentación explicativa y todos los identificadores y categorías relevantes.

Si en esta ficha se encuentra disponible un documento WSDL puede utilizar el asistente Importar un servicio web para generar clases Java a partir del WSDL. Seleccione Archivo | Importar un servicio web, o bien, pulse el botón Importar un servicio web. Si no tiene abierto ningún proyecto, se abre primero el Asistente para proyectos y, a continuación, el asistente Importar un servicios web. Si desea obtener más información acerca del asistente Importar un servicio web, consulte el Capítulo 5, "Utilización de WSDL".

Búsqueda de servicios web en un servidor Axis

Esta es una función del kit de herramientas Axis. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Algunos servicios web se distribuyen y están disponibles en sitios de internet que ejecutan servicios web albergados por Axis de Apache. Puede buscarlos con el Explorador de servicios web. Si está buscando en un servidor Axis remoto, el acceso remoto debe estar activado en ese servidor. Consulte "Acceso remoto a los servidores Axis" en la página 11-19. Además, puede publicar un servicio web de Axis en un registro UDDI. Consulte "Difusión de servicios web desde un servidor Axis" en la página 11-26.

También puede crear servicios web de Axis de forma local y moverse por ellos del siguiente modo:

1. Cree un servicio web tal y como se describe en Capítulo 13, "Tutorial: Creación de un servicio web sencillo con Axis".
2. Amplíe el nodo de Axis en el panel del proyecto y amplíe el Directorio raíz de la WebApp de Axis.
3. Pulse con el botón derecho del ratón sobre `index.html` en el panel del proyecto, y seleccione Ejecutar utilizando "Servidor de servicios web" para distribuir el servicio y ejecutar el servidor web.
4. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
5. Cree un servicio web tal y como se describe en "Apertura de servicios web albergados por Axis" en la página 11-27.

Presentación de los servicios

Cuando se distribuyen los servicios web en un servidor Axis, puede buscar esos servicios en el Explorador de servicios web.

Para mostrar los servicios disponibles en un servidor Axis:

1. Amplíe el nodo Servidores WSDL y cree un nodo dependiente Servidores WSDL para el servicio Axis del siguiente modo:

a Pulse sobre el nodo Servidores WSDL con el botón derecho del ratón y seleccione Nuevo sitio de servidor WSDL o, bien, pulse el botón Nuevo servidor WSDL de la ficha Servidores WSDL.

b Escriba el nombre del nuevo nodo, escoja Servidor Axis como tipo de servidor y pulse Aceptar.

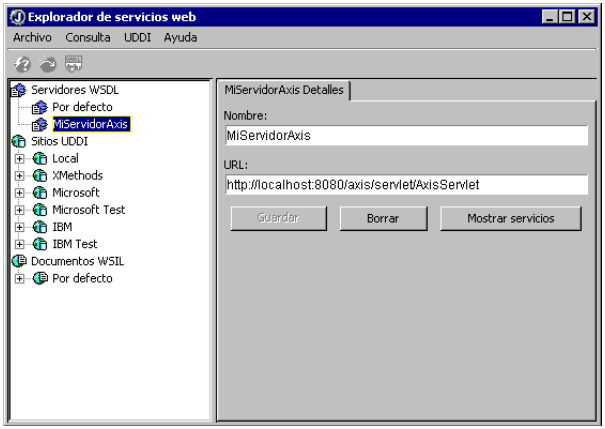
2 Seleccione el nuevo nodo Servidores WSDL y escriba la URL del servidor en el campo URL, incluido el contexto web del servidor. La URL de un servicio en el servidor Axis coincidirá con una de estas formas:

```
<protocolo:>//<puerto:número de puerto>/<contexto web>/servicios/
<protocolo:>//<puerto:número de puerto>/<contexto web >/servlet/AxisServlet/
```

En la que el protocolo se corresponde con el protocolo del servicio; el puerto es el ordenador en el que se está ejecutando el servidor Axis; número de puerto es el número de puerto en el que se encuentra Axis; contexto web es la raíz de contexto o el nombre de la webapp que hospeda el servicio web. Por ejemplo, `http://localhost:8080/<axis>/servlet/AxisServlet`.

3 Pulse el botón Mostrar servicios de la ficha Detalles.

4 Amplíe el nodo para ver los servicios disponibles. Observe que Versión es uno de los servicios de la lista. Este es un servicio integrado de Axis que presenta la versión de Axis.



Importación de WSDL y difusión de servicios web Axis

Una vez que aparezcan los servicios albergados por Axis, puede importar un WSDL para el servicio y generar clases Java para implementar y/o crear el servicio. Para importar un WSDL para el servicio y generar clases Java, seleccione uno de los servicios albergados por Axis y elija Archivo | Importar un servicio web, o bien, pulse el botón Importar un servicio web de la barra de herramientas. Siga los pasos del asistente Importar un

servicio web, pulse Finalizar y consulte el documento WSDL y las clases Java generadas. Si desea obtener más información, consulte Capítulo 5, "Utilización de WSDL", o pulse el botón Ayuda del asistente.

También puede publicar servicios web Axis en un sitio UDDI en el Explorador de servicios web. Consulte "Difusión de servicios web desde un servidor Axis" en la página 11-26.

Nota Para poder buscar los servicios, el acceso remoto debe estar activado en los servidores remotos Axis. Consulte "Acceso remoto a los servidores Axis" en la página 11-19.

Acceso remoto a los servidores Axis

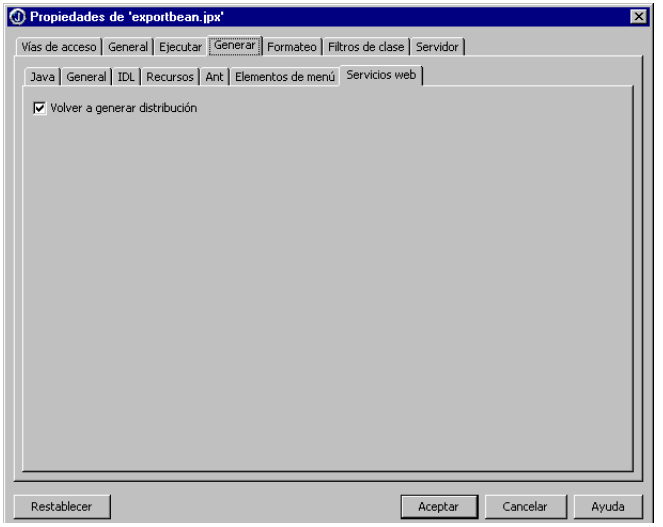
JBuilder genera servidores Axis con el acceso remoto desactivado. Si desea cambiar el funcionamiento por defecto, es necesario que modifique el archivo `server-config.wsdd` del siguiente modo.

Advertencia Si activa la administración remota, personas no autorizadas podrían tener acceso a su equipo. Asegúrese de incrementar la seguridad de su configuración.

Para activar el acceso remoto:

1. Genere el proyecto.
2. Abra el archivo `server-config.wsdd` que se encuentra en el nodo del kit de herramientas de la webapp que hospeda el servicio web.
3. Cambie el valor del elemento `AdminService` de `false` a `true`: `<parameter name="enableRemoteAdmin" value="true"/>`.
4. Desactive la regeneración de `server-config.wsdd` durante la distribución quitando la marca de la opción SOAP del proyecto del siguiente modo:
   a. Elija Proyecto | Propiedades de proyecto y se abrirá el cuadro de diálogo homónimo.
   b. Seleccione la pestaña Generar.

c Seleccione la pestaña Servicios web de la ficha Generar y desactive la opción Volver a generar distribución.



d Pulse Aceptar para cerrar el cuadro de diálogo Propiedades de proyecto.

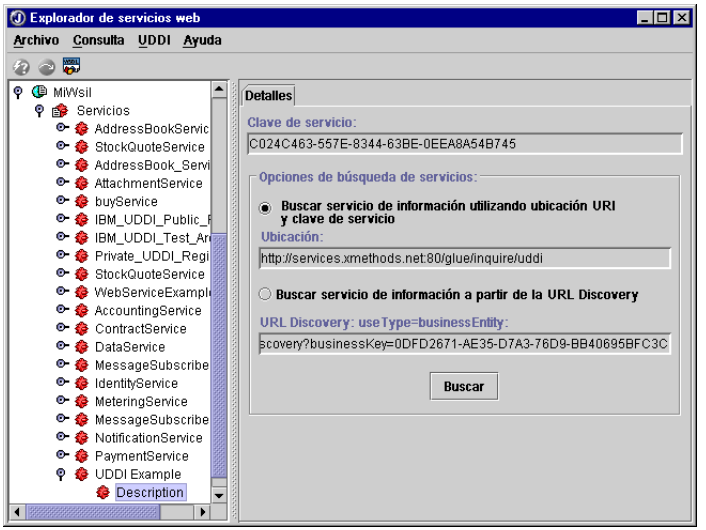
Búsqueda de servicios web con documentos WSIL

Otro modo de buscar servicios web disponibles es hacerlo mediante los documentos WSIL. Los documentos WSIL son conjuntos de punteros hacia otros documentos que recogen los servicios web disponibles en un sitio web. Los documentos WSIL pueden apuntar a otros documentos WSIL, a un negocio o entrada de servicio UDDI y a un documento WSDL. Normalmente, los sitios web con servicios web disponibles reúnen sus enlaces en un sólo documento WSIL en una ubicación por defecto, como `http://www.xmethods.net/inspection.wsil`. Una vez encontrado en el sitio el servicio que desea, puede importar el documento WSDL con el asistente Importar un servicio web.

Según la especificación del Lenguaje de inspección de servicios web, los documentos WSIL deben contar, como mínimo, con un elemento `<service>` o un elemento `<link>`, o pueden tener ambos. Los servicios y enlaces aparecen en el Explorador de servicios web como nodos dependientes de un nodo WSIL.

Nodo Servicios

El nodo Servicios muestra los elementos <service> en el documento WSIL. El elemento <service> especifica el documento WSDL para el servicio o puede hacer referencia a una entrada UDDI que especifica el WSDL. Amplíe el nodo Servicios para que aparezcan los servicios disponibles en el sitio. Amplíe un servicio web y seleccione el nodo Descripción junto a él para abrir la ficha Detalles. Esta ficha ofrece información como la ubicación del documento WSDL, si el servicio es un endpoint, y la información de enlace.



Si el documento hace referencia a una entrada UDDI, puede buscar los servicios de dos modos diferentes:

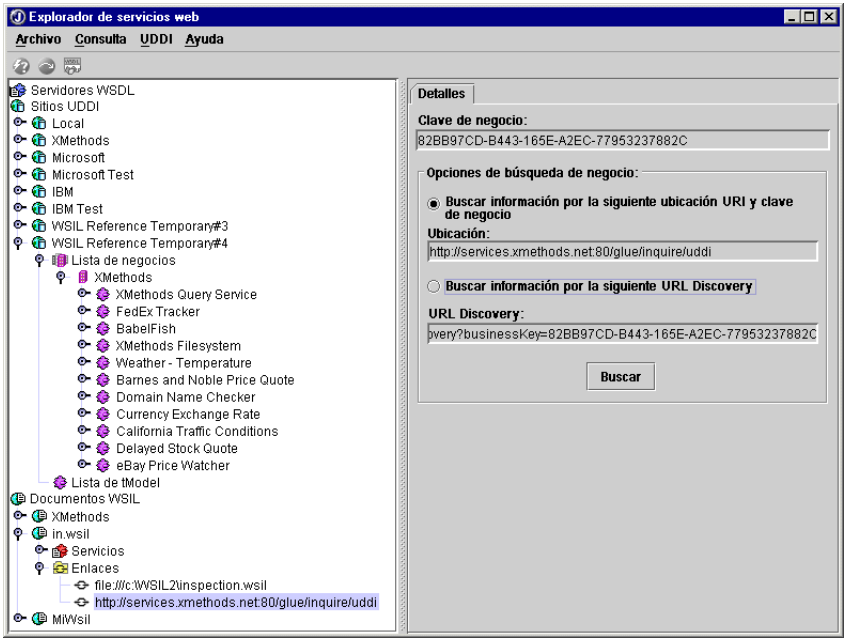
- Clave de servicio y Ubicación URI
- Clave de servicio y URL Discovery

Elija una opción y pulse el botón Buscar para obtener la información. Los resultados de la búsqueda aparecen en un nodo temporal, Referencia temporal a WSIL, en el árbol UDDI.

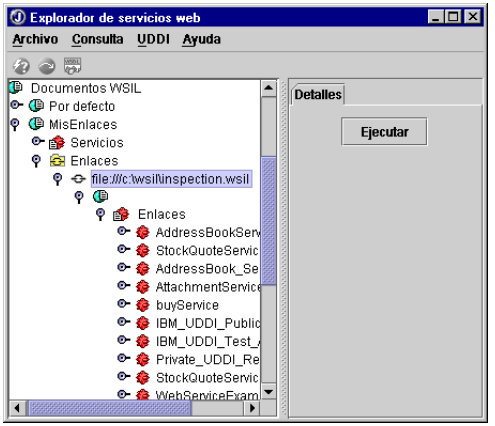
Nodo Enlaces

Si hay algún elemento <link> contenido en el documento WSIL, aparece como un nodo bajo el nodo Enlaces. Esto enlaces pueden llevar a otro documento WSIL o a una entrada de negocios UDDI. Amplíe el nodo Enlaces para ver los elementos <link> en el documento WSIL y seleccione un enlace para que se abra la ficha Detalles.

Si el enlace es una referencia a una entrada de negocios UDDI, la ficha Detalles muestra información UDDI que se puede extraer del registro UDDI. Los resultados de la búsqueda aparecen en un nodo Referencia temporal a WSIL bajo el nodo UDDI.



Si el enlace lleva a otro documento WSIL, puede pulsar el botón Ejecutar de la ficha Detalles para que aparezca la lista de servicios del WSIL.



Ejecución de una búsqueda con un documento WSIL

Para buscar servicios disponibles en un sitio web:

1. Seleccione el nodo Documentos WSIL en el Explorador.
2. Pulse Nueva ubicación WSIL en la ficha Documentos WSIL a la derecha para crear un nuevo nodo WSIL.

Sugerencia También puede pulsar con el botón derecho del ratón sobre el nodo Documentos WSIL y seleccionar Nuevo o Archivo | Nuevo.

3. Escriba el nombre del nodo y pulse Aceptar.
4. Introduzca la URL del sitio web o la URL local en la que desee buscar. Por ejemplo, `http://www.xmethods.net/inspection.wsil` o `file:///c:\wsil\inspection.wsil`.
5. Pulse Guardar para guardar los cambios.
6. Amplíe el nuevo nodo WSIL para ver los nodos Servicios y Enlaces que hay disponibles.
7. Realice una de las operaciones siguientes:
   - Amplíe el nodo Servicios y seleccione un nodo Descripción bajo uno de los servicios para ver la ubicación del documento WSDL. A continuación, puede utilizar el asistente Importar un servicio web para importarlo y generar clases Java para consumir el servicio.
   - Amplíe el nodo Enlaces y seleccione un enlace para ver los elementos disponibles en la ficha Detalles.

     Si hay alguna referencia a una entrada de negocios UDDI, seleccione Buscar para obtener la información. Los resultados aparecen en un nodo Referencia temporal a WSIL bajo el nodo Sitios UDDI. Puede desplazarse hasta un nodo tModel del servicio donde se encuentre el WSDL y utilizar el asistente Importar un servicio web para importarlo y generar clases Java para consumir el servicio.

     Si la referencia es a otro documento WSIL, también puede buscar esos servicios y enlaces. Seleccione el enlace con la referencia WSIL y pulse el botón Ejecutar de la ficha Detalles. Amplíe el nodo de enlaces para que aparezcan los servicios recogidos en el documento WSIL.

Difusión de servicios web en un registro UDDI

El Explorador de servicios web admite la publicación de nuevas negocios, servicios web y sus tModels correspondientes. En algunos casos, los sitios UDDI requieren un registro antes de poder publicar en ellos. Algunos sitios UDDI cuentan con sitios de prueba en los que se puede publicar y probar nuevos servicios antes de distribuirlos en el sitio UDDI oficial. La mayoría de los sitios UDDI también requieren que especifique una URL de difusión antes de publicar el servicio. La URL de difusión está disponible en el sitio web del sitio. Por ejemplo, la URL de difusión del sitio UDDI de Microsoft es `https://uddi.microsoft.com/publish`.

La difusión de servicios web en un sitio UDDI con JBuilder incluye los pasos siguientes:

1. Registro en el sitio UDDI mediante un navegador.
2. Creación y distribución de un servicio web.
3. Publicación de un negocio y servicio en el sitio UDDI.
4. Publicación de un tModel para el servicio.

Registro en el sitio UDDI mediante un navegador

Antes de publicar negocios y servicios web, es necesario que se registre en el sitio UDDI en el que se desea publicar el servicio. En la mayoría de los casos, puede que desee publicar primero en un sitio de prueba para probar el servicio. IBM y Microsoft ofrecen sitios de prueba para este fin. Visite el sitio de operadores UDDI con su navegador para registrarse.

Creación y distribución de un servicio web

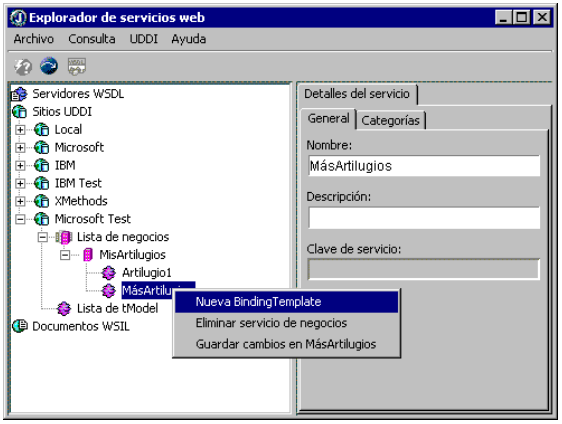
Utilice los asistentes de servicios web de JBuilder para crear su servicio web y distribuirlo según el kit de herramientas seleccionado. Para obtener más información, consulte el Capítulo 7, “El kit de herramientas Axis de Apache”, y el Capítulo 8, “El kit de herramientas WebLogic”.

Difusión de negocios y servicios

Para publicar un servicio web, debe antes publicar un negocio, si todavía no existe, y, a continuación, añadirle información del negocio, servicio, plantilla de enlace y tModel. Si ya tiene un negocio publicado en un sitio UDDI, puede saltarse este paso.

En primer lugar, añada el nuevo negocio del siguiente modo:

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Seleccione un nodo de operadores UDDI, como Microsoft Test, bajo el nodo Sitios UDDI en el árbol. El sitio del operador debe contar con una URL de difusión asociada. Si el campo URL de difusión está vacío, diríjase al sitio web del operador y busque la URL de difusión de ese sitio en concreto. Para introducir la URL de difusión de un sitio de un operador, seleccione un nodo de un operador y escriba la URL en el campo URL de difusión de la ficha Detalles. Seleccione Guardar para guardar los cambios.
3. Pulse con el botón derecho del ratón sobre el nodo Lista de negocios en el árbol y seleccione Nuevo negocio o Archivo | Nuevo negocio.
4. Escriba el nombre del negocio en el cuadro de diálogo y pulse Aceptar.
5. Seleccione el nodo del nuevo negocio e introduzca la información adecuada en la pestaña General de la ficha Detalles del negocio, a la derecha. El sitio UDDI rellena algunos de estos campos, pero el Explorador permite que se modifiquen.
6. Seleccione la pestaña Contacto y pulse Añadir. Introduzca la información de contacto apropiada.
7. Puede seguir añadiendo información a este negocio si pulsa con el botón derecho del ratón sobre el nodo del negocio y selecciona Nuevo para crear el nodo del servicio. A continuación, pulse con el botón derecho del ratón sobre el nodo del servicio y elija Nuevo para crear el nodo de asociaciones, y así sucesivamente.



8 Seleccione el nodo del operador UDDI, el nodo Lista de negocios o el nodo del negocio y, a continuación, seleccione Archivo | Guardar cambios. Esto guarda todos los cambios y los envía al sitio UDDI. También puede pulsar con el botón derecho del ratón sobre un nodo y seleccionar Guardar cambios o utilizar el botón de la barra de herramientas. Si el sitio UDDI requiere un login, se le solicitará un nombre de usuario y una contraseña.

Importante Los cambios guardados dependen del nodo seleccionado. Puede seleccionar un nodo dependiente y guardar los cambios sólo en ese nodo y en los que dependan de él. También puede seleccionar el nodo de operadores UDDI o el nodo Lista de negocios y guardar todos los cambios.

Publicación de tModels

Para añadir y publicar un tModel:

1 Complete los pasos 1 y 2 de "Difusión de negocios y servicios" en la página 11-24.

2 Pulse con el botón derecho del ratón el nodo Lista de tModel del árbol y seleccione Nuevo tModel. Se añade al árbol un nuevo nodo tModel.

3 Seleccione el nodo Nuevo tModel e introduzca la información adecuada en la ficha Detalles de tModel. Si el sitio UDDI requiere un login, se le solicitará un nombre de usuario y una contraseña.

4 Seleccione el nodo tModel y elija Archivo | Guardar cambios para guardar el nuevo tModel y enviarlo al sitio UDDI.

Difusión de servicios web desde un servidor Axis

Esta es una función del kit de herramientas Axis. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

La publicación de servicios web en un sitio UDDI desde un servidor Axis con JBuilder incluye los siguientes pasos:

1 Registro en el sitio UDDI mediante un navegador.

2 Creación y distribución de servicios web albergados en un servidor Axis.

3 Apertura de servicios web albergados por Axis.

4 Publicación desde el servidor Axis.

Creación y distribución de servicios web albergados en un servidor Axis

A continuación, cree un servicio web que se albergue en un servidor Axis local.

1. Cree un servicio web Axis local tal y como se describe en el Capítulo 13, "Tutorial: Creación de un servicio web sencillo con Axis".
2. Distribuya el servicio y ejecute el servidor web.

Apertura de servicios web albergados por Axis

Una vez que el servicio web está distribuido en el servidor Axis y el servidor se está ejecutando, puede abrir el servicio en el Explorador de servicios web.

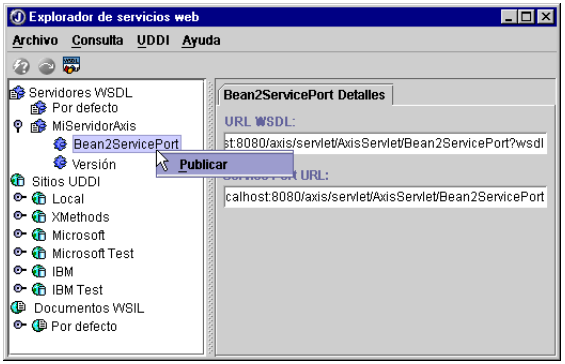
1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Amplíe el nodo Axis y seleccione el nodo Axis por defecto.
3. Modifique la URL en la ficha Detalles con el contexto WebApp, que en este ejemplo es `foo`, y la ubicación del servidor del servicio web. Por ejemplo, `http://localhost:8080/foo/servlet/AxisServlet`.
4. Pulse el botón Mostrar servicios de la ficha Detalles para que aparezca el servicio como nodo dependiente del nodo Axis por defecto.

Publicación desde el servidor Axis

Una vez que se está ejecutando el servidor Axis, puede publicar el servicio en el sitio UDDI. En primer lugar, es necesario que establezca la configuración por defecto para poder publicar. A continuación, es necesario que ejecute de forma local el servicio Axis en el servidor web y que lo publique.

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Amplíe el nodo Sitios UDDI y amplíe el sitio del operador UDDI en el que desee publicar. Seleccione el nodo Lista de negocios y busque el negocio en la que desea publicar el servicio. Si todavía no ha creado un negocio, créelo tal y como se describe en "Difusión de negocios y servicios" en la página 11-24.
3. Pulse el nodo de negocios con el botón derecho del ratón y seleccione Configurar como publicación por defecto.

4 Haga clic con el botón derecho del ratón sobre el servicio que aparece como nodo dependiente del nodo Servidores WSDL por defecto y seleccione Publicar.



5 Pulse Aceptar en el cuadro de diálogo Publicar un servicio Axis en UDDI para cerrarlo.

6 Inicie una sesión en el registro UDDI si es necesario. Aparece un cuadro de diálogo que indica que se ha publicado el servicio y que se han creado nuevos nodos para ese servicio bajo el nodo de negocios configurado para la publicación por defecto.

Seguimiento de mensajes UDDI

El Explorador de servicios web cuenta con un Monitor de mensajes UDDI que permite ver los mensajes SOAP que entran y salen del registro UDDI seleccionado.

Para ver estos mensajes:

1 Amplíe el nodo Sitios UDDI del árbol del Explorador y elija el nodo UDDI.

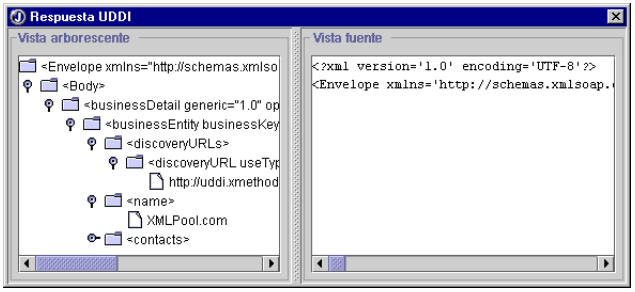
2 Elija UDDI | Mensajes UDDI en caché. De este modo se activa el Monitor de mensajes UDDI, que guarda las solicitudes enviadas al registro UDDI y las respuestas recibidas.

3 Realice una búsqueda en el sitio.

4 Para abrir el Monitor de mensajes UDDI, seleccione el nodo del operador UDDI y elija UDDI | Ver mensajes.

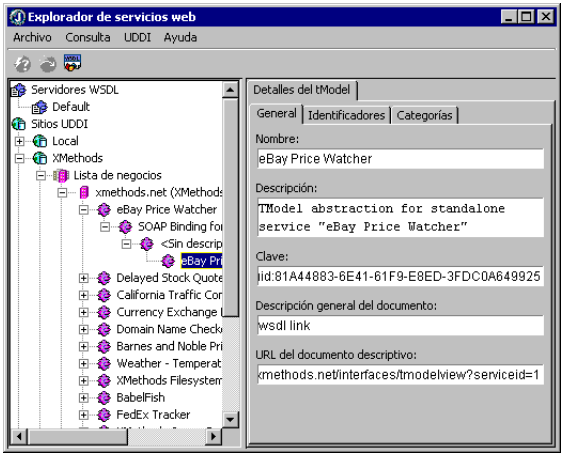
5 Seleccione una respuesta o una solicitud de la lista y pulse Ver para presentar el mensaje. En la vista en árbol de la izquierda se muestran el

elemento raíz de SOAP y los mensajes XML que contiene. A la derecha se muestra el código fuente del mensaje enviado o recibido.



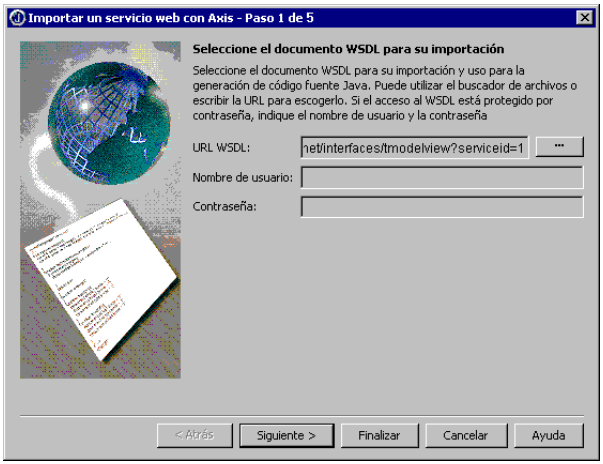
Generación de clases Java a partir de documentos WSDL

El Explorador de servicios web ofrece acceso al asistente Importar un servicio web para generar clases Java basadas en el documento WSDL. El asistente Importar un servicio web se encuentra disponible como un comando de menú, Archivo | Importar un servicio web, y como un botón de la barra de herramientas. El comando de menú y el botón de la barra de herramientas sólo están activos cuando hay un documento WSDL disponible. Por ejemplo, si se selecciona el nodo Detalles del tModel en el árbol y se especifica un documento WSDL en el campo Descripción general del documento, se activan el botón y el comando de menú Importar un servicio web.



Utilice el comando o el botón para abrir el asistente Importar un servicios web. Si no ha abierto ningún proyecto, el Asistente para proyectos se abrirá en primer lugar. Tras crear un proyecto, se abrirá el asistente Importar un servicio web. Seleccione un kit de herramientas de servicios web y pulse Aceptar. Observe que el nombre del documento WSDL se

introduce automáticamente en el campo URL de WSDL del asistente Importar un servicio web. Si un sitio web requiere un nombre de usuario y una contraseña, el primer paso del asistente cuenta con campos Nombre de usuario y Contraseña que debe rellenar. Por ejemplo, en algunos sitios UDDI es necesario que se registre como usuario. Continúe llevando a cabo los pasos del asistente para generar las clases adecuadas. Pulse el botón de ayuda en cualquier paso del asistente si lo necesita.



Capítulo

12

Tutoriales de servicios web

Es una función de JBuilder Enterprise.

Los tutoriales de servicios web se encuentran clasificados en categorías dependiendo del kit de herramientas utilizado. Existen tutoriales generales para todos los kits de herramientas.

Tutoriales de servicios web Axis

Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache ni para Borland Enterprise Server.

- Capítulo 13, "Tutorial: Creación de un servicio web sencillo con Axis"

  Explica cómo utilizar el asistente Exportar como servicio web con el fin de publicar un JavaBean como servicio web y exponer determinados métodos seleccionados ante el consumidor del servicio web.
- Capítulo 14, "Tutorial: Creación de un servicio web desde un documento WSDL"

  Explica cómo utilizar el asistente Importar un servicio web con el fin de generar clases Java para un servicio web que proporciona un servicio de traducción y el modo de implementar en servicio.
- Capítulo 15, "Tutorial: Crear un servicio web desde una aplicación EJB con Borland Enterprise Server"

  Explica cómo crear un servicio web a partir de una aplicación Enterprise JavaBean mediante Borland Enterprise Server.
- Capítulo 16, "Tutorial: Importación de servicios web como aplicaciones EJB"

  Describe cómo importar un documento WSDL como aplicación EJB mediante el kit de herramientas Axis y Borland Enterprise Server.

Tutoriales para servicios web de WebLogic

- Capítulo 17, "Tutorial: Creación de un servicio web sencillo con WebLogic"

  Explica cómo utilizar el asistente Exportar como servicio web con el fin de publicar un JavaBean como servicio web y exponer determinados métodos seleccionados ante el consumidor del servicio web.

- Capítulo 18, "Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server"

  Explica cómo crear un servicio web a partir de una aplicación EJB mediante el servidor WebLogic.

Tutoriales generales para servicios web

- Capítulo 19, "Tutorial: Búsqueda de servicios web UDDI"

  Explica cómo utilizar el Explorador de servicios web para buscar servicios web y generar clases Java a partir de un documento WSDL.

Capítulo

13

Tutorial: Creación de un servicio web sencillo con Axis

En este tutorial se utilizan funciones de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Este tutorial muestra cómo utilizar el asistente Exportar como servicio web con el fin de exportar un JavaBean como servicio web, generar un documento WSDL que describa el servicio y crear un servidor de servicios web, mediante el kit de herramientas Axis de Apache, para albergar el servicio.

En este tutorial, se completan las siguientes tareas:

- Creación de un JavaBean de ejemplo.
- Exportación del bean de ejemplo como servicio web y configuración del proyecto para servicios web.
- Distribución, ejecución y prueba de funcionamiento del servicio.

Este tutorial asume que usted está familiarizado con Java y con el IDE (entorno integrado de desarrollo) de JBuilder. Para obtener más información sobre las JSP, consulte *Procedimientos iniciales con Java*. Si desea obtener más información sobre el IDE de JBuilder, consulte "El entorno de JBuilder" en *Introducción a JBuilder*.

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Creación de un JavaBean de ejemplo

En este paso se crea un proyecto mediante el Asistente para proyectos. A continuación, creará un JavaBean que exportará como servicio web.

1. Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
2. Introduzca `exportbean` en el campo Nombre.
3. Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.
4. En la ficha Servidor de Propiedades de proyecto (Proyecto | Propiedades de proyecto), compruebe que el servidor establecido para el proyecto es Tomcat 4.0.
5. Elija Archivo | Nuevo para abrir la galería de objetos y pulse la pestaña General.
6. Seleccione JavaBean y pulse Aceptar para iniciar el Asistente para JavaBean.
7. Seleccione `java.lang.Object` como la clase base.
8. Active Generar propiedad de ejemplo y acepte el resto de los valores por defecto.
9. Pulse Aceptar para cerrar el asistente. En el paquete `exportbean` se genera un archivo `Bean1.java`.

Paso 2: Exportación del bean de ejemplo como servicio web y configuración del proyecto para servicios web

En este paso, configurará el proyecto para servicios web y utilizará el asistente Exportar como servicio web con el fin de exportar el bean como un servicio web. El asistente expondrá todos los métodos del bean en el servicio, creará un documento WSDL que describa el servicio, creará también un archivo de distribución de servicios web y generará las clases Java para el servicio.

1. En el panel del proyecto, pulse con el botón derecho del ratón sobre `Bean1.java` y seleccione Exportar como servicio web. Como el proyecto no está configurado aún para servicios web, se abrirá el Asistente para configuración de servicios web de modo que pueda crear una WebApp para hospedar el servicio web. Después de haber configurado el proyecto, aparecerá el asistente Exportar como servicio web.

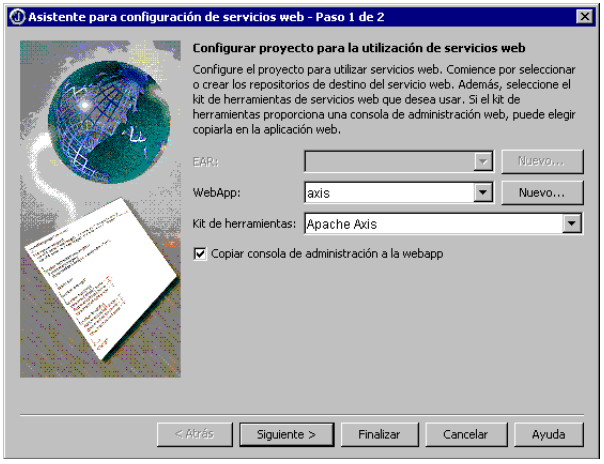
2 Configure el proyecto para servicios web mediante el asistente Configuración de servicios web como se indica a continuación:

a Pulse el botón Nuevo, situado junto al campo WebApp, y cree una WebApp mediante el Asistente para aplicaciones web. Es necesario que haya una WebApp para hospedar el servicio.

b Escriba `axis` como nombre y directorio de la WebApp y pulse Aceptar para volver al asistente Configuración de servicios web.

c Seleccione Axis de Apache en la lista desplegable Kit de herramientas. El contexto web, formado por el nombre y directorio de la WebApp, se utiliza en el documento WSDL y en el archivo `Bean1ServiceLocator.java`. En este ejemplo, la dirección SOAP por defecto que aparece en el documento WSDL generado por el asistente es `<wsdlsoap:address location="http://localhost:8080/`**axis**`/services/Bean1"/>`. La dirección para el puerto del servicio que aparece en `Bean1ServiceLocator.java` es `http://localhost:8080/`**axis**`/services/Bean1`.

El Asistente para configuración de servicios web presentará el siguiente aspecto:

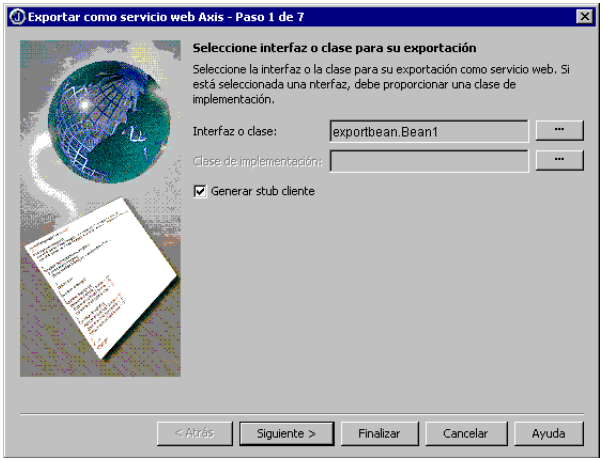


d Pulse Siguiente y podrá ver la configuración de ejecución del servidor de servicios web creada por el asistente. Utilizará esta configuración de ejecución más adelante para ejecutar el proyecto.

e Pulse Finalizar para cerrar el Asistente para configuración de servicios web y continuar con el asistente Exportar como servicio web.

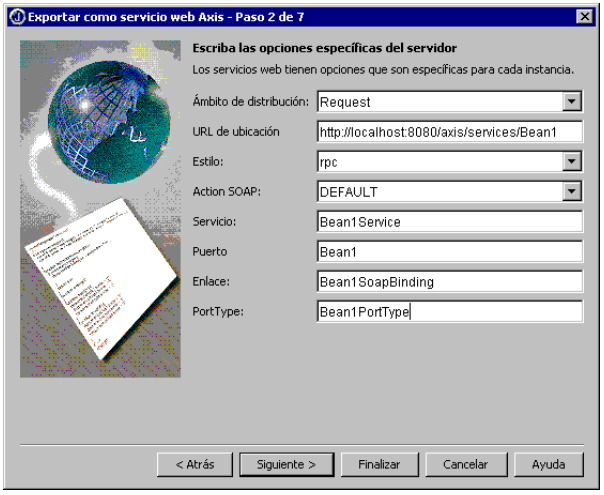
Ahora que ya ha configurado el proyecto para servicios web, utilizará el asistente Exportar como servicio web con el fin de

exportar la clase como un servicio web. El asistente Exportar como servicio web presenta el siguiente aspecto:



3 Acepte la clase del paso 1 del asistente y pulse Siguiente.

4 Escriba `Bean1PortType` en el campo PortType.



5 Continúe con el resto de los pasos del asistente y acepte los valores por defecto. Observe que, en la ficha Métodos, el modo de selección es el de permitir que todos los métodos aparezcan expuestos en el servicio. Si desea información sobre las opciones, pulse el botón Ayuda de cualquier ficha del asistente.

6 Pulse Finalizar para cerrar el asistente y generar el documento WSDL y las clases Java para el servicio.

El documento WSDL generado se denomina `Bean1.wsdl`. Este archivo contiene la información de WSDL que define la forma de establecer la

conexión con el servicio web. Los archivos Java que generan el servicio web también se generan y se sitúan en un paquete llamado `exportbean_generated`. Se trata de los siguientes archivos:

- `Bean1PortType.java` es la interfaz del servicio.
- `Bean1Service.java` es una interfaz abstracta que define una clase de fábrica para obtener una instancia del stub.
- `Bean1ServiceLocator.java` es la implementación de la interfaz abstracta.
- `Bean1ServiceTestCase.java` es el test JUnit del servicio. Si desea más información sobre los tests JUnit, consulte "Tests de módulos" en *Creación de aplicaciones con JBuilder*.
- `Bean1SoapBindingStub.java` es el stub cliente que serializa en SOAP la llamada y los parámetros de Java.

Paso 3: Distribución, ejecución y comprobación del servicio web

En este paso, distribuirá el servicio y ejecutará el servidor de servicios web. A continuación se ejecuta el test JUnit generado por el asistente Exportar un servicio web, con el fin de verificar la ejecución del JavaBean como servicio web.

1. Seleccione Ejecutar | Ejecutar proyecto para ejecutar la configuración de ejecución creada por el asistente, Servidor de servicios web. Esta configuración de ejecución distribuye el servicio y ejecuta el servidor de servicios web. Para obtener más información sobre las configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.

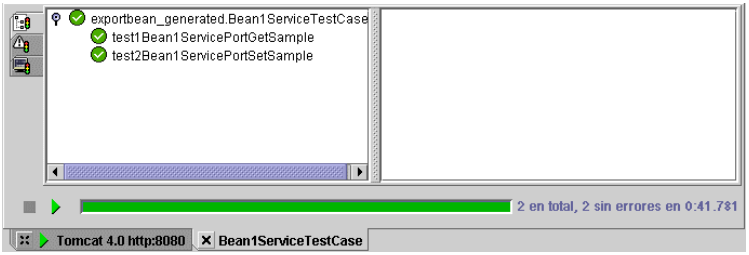
   El servidor de servicios web se inicia y el servicio se distribuye en el servidor. Una vez distribuido el servicio, la ficha Administración de Axis aparece en el panel de contenido de JBuilder.

2. Compruebe que el servicio se ha distribuido. Pulse el enlace Ver para mostrar los servicios distribuidos. Podrá ver que `Bean1` y sus dos métodos, `getSample()` y `setSample()`, han sido distribuidos.

3 Pulse sobre el enlace Bean1 (wsdl), en la ficha de administración, o haga doble clic en `Bean1.wsdl`, en el panel del proyecto, para ver el documento WSDL generado por el asistente para el servicio. Observe que tanto el método `getSample()` como el método `setSample()` se exponen como operaciones en la descripción del servicio.

4 Amplíe el nodo del paquete `exportbean.generated` en el panel del proyecto.

5 Haga clic con el botón derecho en `Bean1ServiceTestCase.java` y elija Ejecutar comprobación utilizando valores por defecto en el menú contextual. De esta forma se ejecuta el test JUnit generado, en el comprobador por defecto. Éste es el aspecto que presenta JBTestRunner después de ejecutar el test:



La comprobación por defecto accede a los métodos públicos del JavaBean que se ejecuta como servicio web en el servidor de servicios web. Si desea obtener información más detallada sobre tests de módulos, consulte “Tests de módulos” en *Creación de aplicaciones con JBuilder*.

Importante Si realiza cambios en `Bean1.java` después de haberlo exportado, debería volver a compilarlo, exportarlo como servicio web y distribuirlo en el servidor de servicios web para que los cambios surtan efecto.

¡Enhorabuena! Ha finalizado este tutorial. En él, ha aprendido a exportar un JavaBean como servicio web, a configurar el proyecto para servicios web y a probar el servicio. Para obtener más información acerca de WSDL, consulte el Capítulo 5, “Utilización de WSDL”. Para obtener más información sobre el Asistente para configuración de servicios web, consulte el Capítulo 3, “Configuración de proyectos para servicios web”. Para obtener más información sobre los tests de módulos, consulte “Tests de módulos” en *Creación de aplicaciones con JBuilder*.

Capítulo

14

Tutorial: Creación de un servicio web desde un documento WSDL

En este tutorial se utilizan funciones de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Este tutorial ilustra cómo generar una aplicación de servicio web a partir de un documento WSDL existente, cómo llamar a un servicio web externo y cómo implementar ese mismo servicio web de forma local. Para generar clases Java a partir del documento WSDL, se utilizará el asistente Importar un servicio web y el kit de herramientas Axis. A continuación se completan los pasos restantes para acceder a un servicio web que traduce de unos idiomas a otros por medio de BabelFish de AltaVista. Para obtener más información sobre los archivos generados por el asistente Importar un servicio web, consulte "Importación de archivos WSDL" en la página 7-6.

En este tutorial, se completan las siguientes tareas:

- Configuración del proyecto para servicios web.
- Importación de un WSDL y generación de clases Java.
- Examen de los descriptores de distribución.
- Implementación del servicio.
- Creación de una WebApp pública para albergar una página JavaServer (JSP).
- Creación de una JSP que llama al servicio.
- Implementación del bean.
- Consumo del servicio hospedado públicamente en XMethods.

- Utilización del TCPMonitor para realizar un seguimiento de los mensajes SOAP entre el cliente y el servicio.
- Consumo del servicio hospedado localmente.

Este tutorial también se encuentra disponible como ejemplo en el directorio de JBuilder: `<jbuilder>/samples/webservices/axis/BasicWebService`.

Este tutorial asume que usted está familiarizado con Java y con el IDE (entorno integrado de desarrollo) de JBuilder. Para obtener más información sobre las JSP, consulte *Procedimientos iniciales con Java*. Para obtener más información sobre las Páginas JavaServer, consulte "Desarrollo de Páginas JavaServer (JSP)" en la *Guía del desarrollador de aplicaciones web*. Si desea obtener más información sobre el IDE de JBuilder, consulte "El entorno de JBuilder" en *Introducción a JBuilder*.

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Configuración del proyecto para servicios web

En este paso, creará un proyecto y, a continuación, lo configurará mediante el Asistente para configuración de servicios web. El Asistente para configuración de servicios web crea una implementación SOAP para el proyecto basada en el kit de herramientas seleccionado (en este caso, Axis). También crea una configuración de ejecución del servidor de servicios web que permite distribuir y ejecutar el servicio o una implementación del servicio. Cuando un servicio web se distribuye en un servidor de servicios web, el servicio puede recibir y enviar mensajes SOAP XML desde y hacia aplicaciones cliente.

1. Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
2. Introduzca `wsdltutorial` en el campo Nombre.
3. Pulse Finalizar para cerrar el Asistente para proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente. Se crea un proyecto.
4. En la ficha Servidor de Propiedades de proyecto (Proyecto | Propiedades de proyecto), compruebe que el servidor seleccionado es Tomcat 4.0.

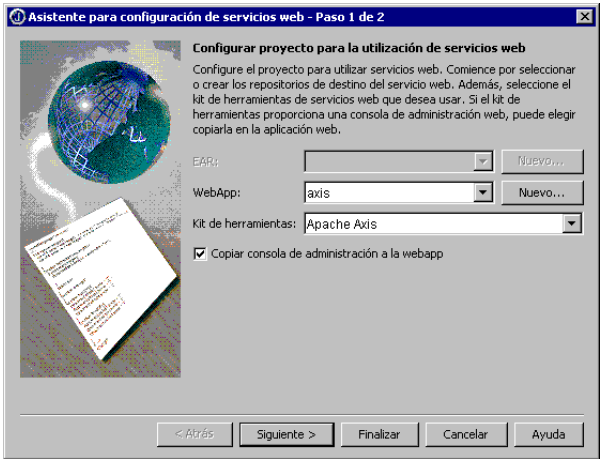
5 Elija Archivo | Nuevo y seleccione la pestaña Servicios web de la galería de objetos.

6 Haga doble clic sobre el icono Configuración de servicios web. Puesto que el nuevo proyecto no contiene ninguna aplicación web, deberá crear una. El servidor de servicios web debe tener una aplicación web como host.

7 Pulse el botón Nuevo, situado junto al campo WebApp, para abrir el Asistente para aplicaciones web.

8 Escriba `axis` como nombre de la aplicación y del directorio.

9 Pulse Aceptar para cerrar el Asistente para aplicaciones web, crear la WebApp de Axis y volver al Asistente para configuración de servicios web, que presentará el siguiente aspecto:



10 Acepte Axis como WebApp y Axis de Apache como kit de herramientas; a continuación, pulse Siguiente y podrá ver la configuración creada por el asistente (Servidor de servicios web).

11 Pulse Finalizar para cerrar el asistente y crear el servidor de servicios web.

Paso 2: Importación del documento WSDL

En este paso se añade al proyecto el documento WSDL y se utiliza el asistente Importar un servicio web para generar las clases Java a partir del dicho documento. El documento WSDL describe el servicio web.

1 Pulse el botón Añadir Archivos/Paquetes en la barra de herramientas del proyecto.

2 Desplácese a `BabelFishService.wsdl`, que se encuentra en `<jbuilder>/samples/webservices/axis/BasicWebService`. Selecciónelo y pulse Aceptar

para añadir el archivo al proyecto. Este documento WSDL se utiliza para generar las clases Java que implementan el servicio web.

3. En el panel del proyecto, pulse con el botón derecho del ratón sobre `BabelFishService.wsdl` y, en el menú contextual, seleccione Importar un servicio web para abrir el asistente.
4. Deje Axis de Apache como kit de herramientas y pulse Aceptar.
5. En el paso 1, acepte la dirección URL del WSDL y pulse Siguiente. No necesita especificar un nombre de usuario ni una contraseña.
6. Acepte los valores por defecto del resto de las fichas y pulse Finalizar. En el panel del proyecto, aparece un nuevo nodo de paquete, `net.xmethods.www`. El nombre del paquete se basa, por defecto, en el espacio de nombres destino del WSDL.
7. Expanda el nodo del paquete `net.xmethods.www` en el panel del proyecto y podrá ver los archivos generados.
   - `BabelFishBindingImpl.java` es la implementación del servicio. El código se escribe en esta clase. Esta clase implementa `BabelFishPortType.java`.
   - `BabelFishBindingStub.java` es el stub cliente que serializa en SOAP la llamada y los parámetros de Java.
   - `BabelFishPortType.java` es la interfaz abstracta del servicio.
   - `BabelFishService.java` es una interfaz abstracta que define una clase de fábrica para obtener una instancia del stub.
   - `BabelFishServiceLocator.java` es la implementación de la interfaz abstracta del servicio.
   - `BabelFishServiceTestCase.java` es el test JUnit del servicio. Si desea más información sobre los tests JUnit, consulte "Tests de módulos" en *Creación de aplicaciones con JBuilder*.

   Para obtener más información sobre los archivos creados por el asistente, consulte "Importación de archivos WSDL" en la página 7-6.
8. Seleccione Proyecto | Ejecutar Make del proyecto para compilar las clases generadas.

Paso 3: Examen de los descriptores de distribución

El asistente Importar un servicio web crea varios archivos de distribución: `deploy.wsdd` y `server-config.wsdd`. El archivo `deploy.wsdd` es un archivo XML descriptor de distribución de servicios web. Los valores del archivo `deploy.wsdd` se derivan de la información contenida en el documento WSDL. El kit de herramientas Axis de Apache crea un archivo de distribución final, `server-config.wsdd`, que consta de varias distribuciones

independientes. Este archivo contiene una relación de los servicios distribuidos, así como gestores de especificación, transportes, administración remota, etc. En este paso se examina los descriptor de distribución del editor.

1. En el panel del proyecto, expanda el nodo Axis y su nodo subordinado, Componentes de servicio web [kit de herramientas Axis de Apache].
2. Expanda el nodo Servicios basados en Java y haga doble clic en `[net.methods.www]deploy.wsdd` para abrir el archivo en el editor.

   Observe los siguientes elementos:

`<service name="BabelFishPort" provider="java:RPC">`	Nombre del servicio publicado.
`<parameter name="className" value="net.xmethods.www. BabelFishBindingImpl"/>`	Nombre de la clase de implementación para la interfaz `portType`.
`<parameter name="allowedMethods" value="babelFish"/>`	El método publicado.

3. Expanda el nodo Descriptores de distribución y haga doble clic en `server-config.wsdd` para abrir el archivo en el editor.

   Observe los siguientes elementos en este archivo de distribución final de Axis:

`<handler name="" type=""/>`	Gestor utilizado para procesar mensajes SOAP.
`<parameter name= "enableRemoteAdmin" value=""/>`	Parámetro que especifica si se puede acceder remotamente al servicio Axis. Para obtener más información sobre el acceso remoto, consulte "Acceso remoto a los servidores Axis" en la página 11-19.
`<transport name="">`	Se utiliza para enviar y monitorizar mensajes SOAP.

Paso 4: Implementación del servicio

En este paso se sustituye el código generado de `BabelFishBindingImpl.java` por otro que implementa el servicio web. Escriba el código en esta clase, que implementa `BabelFishPortType.java`. `BabelFishBindingImpl.java` es la implementación del servicio (implementa la interfaz PortType, `BabelFishPortType.java`). Aquí es donde se introduce el código que implementa el servicio web. Cuando se envía una solicitud de traducción a esta implementación local del servicio, se recibe una respuesta estándar. Si elige traducción de inglés a francés, la respuesta estándar es "Lo siento, no hablo inglés". Si elige cualquier otro idioma, la respuesta es "Lo siento, no hablo su idioma".

1. En el panel del proyecto, expanda el nodo de paquete `net.xmethods.www` y haga doble clic en `BabelFishBindingImpl` para abrirlo en el editor.
2. Elimine el cuerpo generado de la clase `BabelFishBindingImpl` y reemplácelo por el código indicado en negrita que implementa el servicio y devuelve la traducción solicitada:

```
public class BabelFishBindingImpl implements
net.xmethods.www.BabelFishPortType {
private String translatedOutput = "Lo siento, no hablo";
  public java.lang.String babelFish(java.lang.String translationmode,
   java.lang.String sourcedata) throws java.rmi.RemoteException {
    if (translationmode.equals("en_fr"))
      translatedOutput = translatedOutput + " inglés";
    else
      translatedOutput = translatedOutput + " su idioma";
    return translatedOutput;
  }
}
```

Más adelante, en este mismo tutorial, utilizará la JSP para acceder al servicio local, `IndexBean.java`, mediante esta clase.

Paso 5: Creación de la aplicación web pública

En este paso se crea una WebApp que hospeda la página JSP que se creará en el paso siguiente.

1. Seleccione Archivo | Nuevo.
2. Pulse la pestaña Web de la galería de objetos y seleccione Aplicación web. Pulse Aceptar. Aparece el Asistente para aplicaciones web.
3. Escriba `Public` como nombre de la aplicación web y del directorio.
4. Pulse Aceptar para cerrar el asistente.

Paso 6: Creación de una JSP que llama al servicio web

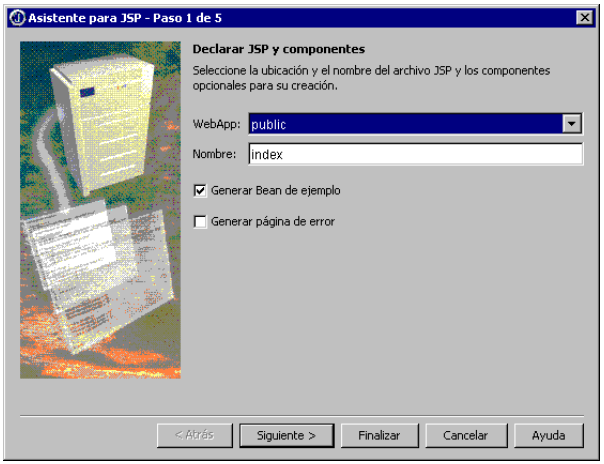
En este paso se crea el esqueleto de una JSP mediante el Asistente para JSP. A continuación, añadirá código JSP que generará un formulario para interactuar con el servicio web. El formulario contiene un campo de texto para introducir el texto que se desea traducir, una lista desplegable de idiomas, una lista desplegable que permite elegir si se desea llamar al servicio web de XMethods o al servicio local, y un botón Submit y otro Reset.

1. Seleccione Archivo | Nuevo.
2. Abra la pestaña Web de la galería de objetos y haga doble clic sobre el icono Página JavaServer para abrir el Asistente para JSP.

3 Cerciórese de que la WebApp seleccionada es `Public`.

4 Escriba `index` como nombre de la página JSP.

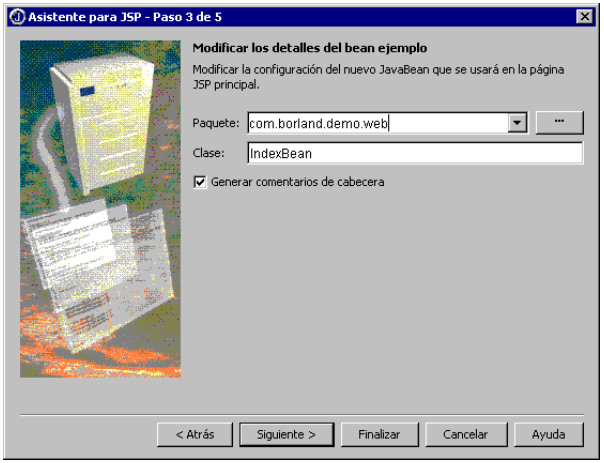
5 Asegúrese de que está seleccionada la opción Generar Bean de ejemplo. Éste es el aspecto del Asistente para JSP:



6 Pulse Siguiente y desactive Generar formulario de envío en el paso 2.

7 Pulse Siguiente y escriba `com.borland.demo.web` como nombre del paquete del bean de ejemplo.

8 Compruebe que el nombre del bean es `IndexBean`. Éste es el aspecto del Asistente para JSP:



9 Pulse el botón Finalizar.

10 Cerciórese de que el archivo `index.jsp` está abierto en el editor.

11 Cambie el contenido de las etiquetas <body></body> por el siguiente código de JSP:

```
<h1>Una JSP sencilla que utiliza un servicio web</h1>
<form method="post">
<table>
  <tr>
    <td>Escriba el texto para traducir</td>
    <td><input name="inputParam" value="<jsp:getProperty name="indexBeanId"
        property="inputParam" />"></td>
  </tr>
  <tr>
    <td>Elija el idioma al que se traduce el texto</td>
    <td><select name="inputOption">
        <option value="en_fr">Inglés a francés
        <option value="en_de">Inglés a alemán
        <option value="en_it">Inglés a italiano
        <option value="en_pt">Inglés a portugués
        <option value="en_es">Inglés a español
        <option value="fr_en">Francés a inglés
        <option value="de_en">Alemán a inglés
        <option value="it_en">Italiano a inglés
        <option value="pt_en">Portugués a inglés
        <option value="ru_en">Ruso a inglés
        <option value="es_en">Español a inglés
        </select>
    </td>
  </tr>
  <tr>
    <td>Seleccione el servidor</td>
    <td><select name="serverOption">
        <option value="public">Servicio hospedado públicamente en XMethods
        <option value="local">Servicio propio hospedado localmente
        </select>
    </td>
  </tr>
</table>
<input type="enviar" name="Enviar" value="Enviar"><input type="borrar"
value="Borrar"></br>
</form>

<br>El texto se ha traducido</br>
<br><b>"<jsp:getProperty name="indexBeanId" property="inputParam" />"</b> </
br>
<br>al idioma elegido como</br>
<br><b>"<jsp:getProperty name="indexBeanId" property="outputParam"/>"</b></
br>
```

Paso 7: Implementación del bean

En este paso se escribe el código que implementa la clase IndexBean, creada durante el paso anterior por el Asistente para JSP. IndexBean.java asigna valores por defecto a los campos de texto del formulario, llama al servicio por medio del stub cliente (BabelFishBindingStub.java) y envía al servicio la información introducida por el usuario.

1. Amplíe el nodo del paquete com.borland.demo.web en el panel del proyecto.
2. Haga doble clic en IndexBean.java para abrirlo en el editor.
3. Elimine el cuerpo de la clase IndexBean y añada el código marcado en **negrita**:

```
package com.borland.demo.web;

import net.xmethods.www.*;
import java.net.URL;

public class IndexBean {
  public static void main(String[] args) {
      BabelFishPortType ws = null;
   try {

     ws = new BabelFishServiceLocator().getBabelFishPort
        (new URL("http://localhost:8080/axis/services/BabelFishPort"));
     System.out.println(ws.babelFish("en_fr", "Hello"));
    }
  catch (Exception ex) {
       ex.printStackTrace();
    }
  }

  private BabelFishPortType ws;
  private String outputParam;
  private String inputParam = "Hello";
  private String inputOption = "en_fr";
  private String serverOption = "public";


  public String getInputParam() {
    return inputParam;
  }
  public String getInputOption() {
    return inputOption;
  }
```

```
public String getOutputParam() {
 try {
    if (serverOption.equals("public"))
       ws = new BabelFishServiceLocator().getBabelFishPort();
  else
   ws = new BabelFishServiceLocator().getBabelFishPort
      (new URL("http://localhost:8080/axis/services/BabelFishPort"));
    outputParam = ws.babelFish(inputOption, inputParam);
  }
catch (Exception ex) {
     ex.printStackTrace();
  }
  return outputParam;
}
public void setInputParam(String newInput) {
  if (newInput!=null) {
    inputParam = newInput;
  }
}
public void setInputOption(String newOption) {
  if (newOption!=null) {
    inputOption = newOption;
  }
}

public String getServerOption() {
  return serverOption;
}

public void setServerOption(String newOption) {
  if (newOption!=null) {
    serverOption = newOption;
  }
}

}
```

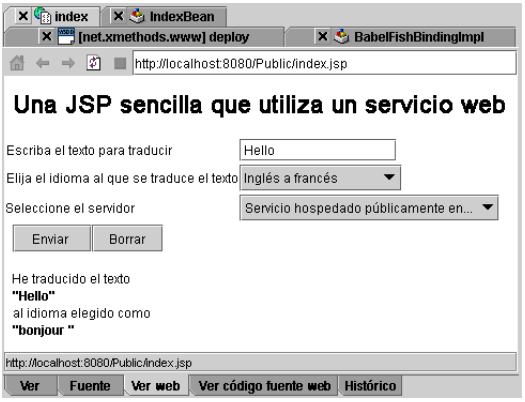
Observe que el método `getOutputParam()` recibe la opción de servidor especificada por el usuario consumidor del servicio en el formulario JSP y, si es igual a “public”, realiza una llamada al método `getBabelFishPort()` con la dirección del servicio público que se encuentra en XMethods, “http://services.xmethods.net:80/perl/soaplite.cgi”. Si la opción de servidor especificada por el usuario no es “public”, se utiliza el servicio local, que se encuentra en `"http://localhost:8080/axis/services/BabelFishPort"`.

Paso 8: Llamada al servicio web y seguimiento de mensajes SOAP

Ahora, compilará el proyecto y llamará al servicio web. En primer lugar, consumirá el servicio público que se encuentra en el sitio web de XMethods. A continuación, consumirá el servicio creado localmente y utilizará el TCPMonitor para realizar un seguimiento de los mensajes SOAP que se transmiten entre el cliente y el servicio.

1. En el editor, pulse con el botón secundario del ratón sobre la pestaña del archivo `index.jsp` y, a continuación, seleccione en el menú contextual Ejecutar web utilizando "Servidor de servicios web". Esta configuración de ejecución genera el proyecto y, a continuación, inicia ambas aplicaciones web: `axis`, que hospeda el servicio web, y `Public`, que contiene la página JSP que se utiliza para acceder a él. Tenga en cuenta que la opción "Servidor de servicios web" del comando Ejecutar en web del menú contextual hace referencia a una configuración de ejecución creada automáticamente por el Asistente para configuración de servicios web. Para obtener más información sobre las configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.

   El formulario JSP presenta el siguiente aspecto:



2. Escriba `bon voyage` en el campo de texto y seleccione Francés a inglés en la lista desplegable. Observe que el servidor seleccionado es un servicio público albergado en XMethods.

3. Pulse Enviar para enviar al servicio el texto que se desea traducir. La solicitud se envía a XMethods y la respuesta traducida se muestra en la parte inferior del formulario: `He traducido el texto "bon voyage" al idioma elegido como "happy voyage"`.

A continuación, consumirá el servicio local creado a partir del WSDL y utilizará el TCPMonitor para realizar un seguimiento de los mensajes SOAP. En primer lugar, establecerá el TCPMonitor, modificará `IndexBean.java` de modo que pueda enviar solicitudes y respuestas al

TCPMonitor por el puerto 8082, volverá a compilar el proyecto y, por último, consumirá el servicio albergado localmente. El TCPMonitor se sitúa entre el cliente SOAP y el servidor. El cliente envía su solicitud al TCPMonitor; éste, a su vez, la reenvía al servidor. Las respuestas procedentes del servidor se envían al TCPMonitor y éste las reenvía al cliente. Para obtener más información sobre el TCPMonitor, consulte el Capítulo 4, "Seguimiento de mensajes SOAP". Cuando se utiliza el TCPMonitor, es necesario modificar el número de puerto y la dirección en el código del cliente de modo que se puedan enviar y recibir mensajes hacia y desde el TCPMonitor a través del puerto.

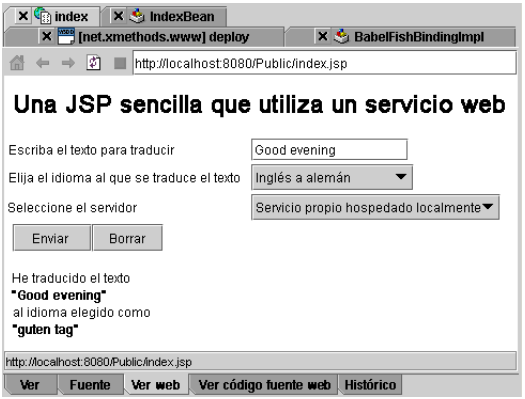
1. Elija Herramientas | TCPMonitor para abrir el monitor. Observe que el puerto por defecto para que el TCPMonitor realice el seguimiento es el 8082. Acepte los valores por defecto y vuelva a JBuilder (deje el TCPMonitor abierto).
2. Abra `IndexBean.java` en el nodo de paquete `com.borland.demo.web` y cambie la dirección del puerto de `http://localhost:`**`8080`**`/axis/services/BabelFishPort` a la del puerto en el que el TCPMonitor está realizando el seguimiento: `http://localhost:`**`8082`**`/axis/services/BabelFishPort`. `IndexBean.java`, que es el servicio local, debe recibir solicitudes y enviar respuestas por el mismo puerto que utilice el TCPMonitor (puerto 8082).

3. Pulse el botón Terminar el programa.
4. Seleccione Proyecto | Generar de nuevo el proyecto para volver a compilar e incluir el cambio de puerto en `IndexBean.java`.
5. Ejecute `index.jsp` de nuevo para volver a cargar el formulario JSP.

Ahora, consumirá el servicio local que ha creado a partir del documento WSDL y observará los mensajes SOAP transmitidos entre el cliente y el servicio.

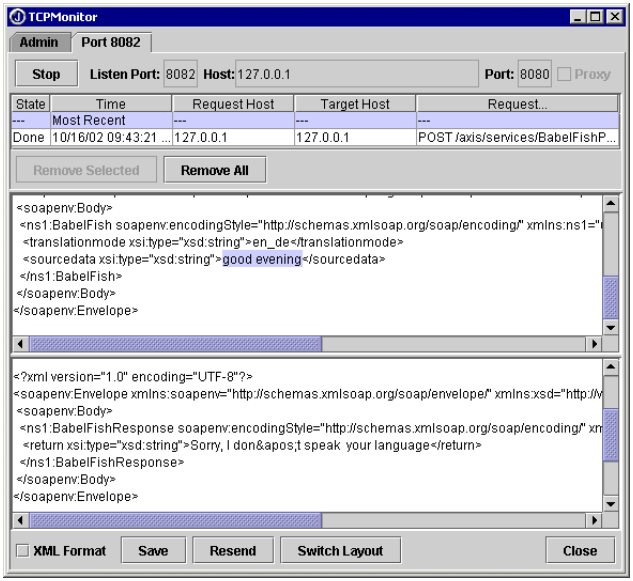
1. Vuelva al formulario `index.jsp` donde escribió el texto que deseaba traducir.
2. Escriba `good evening` en el campo texto.
3. Seleccione Inglés a alemán en la lista desplegable de idiomas.

4 En la lista de servidores, seleccione Servicio propio hospedado localmente.



5 Pulse el botón Enviar para enviar el texto que desea traducir al servicio albergado localmente. La traducción se muestra bajo este botón: `"Lo siento, no hablo su idioma."`

6 Vuelva al TCPMonitor y examine los mensajes enviados entre el cliente y el servicio a través del TCPMonitor. Puede utilizar el TCPMonitor para modificar y volver a enviar un mensaje SOAP al servidor.

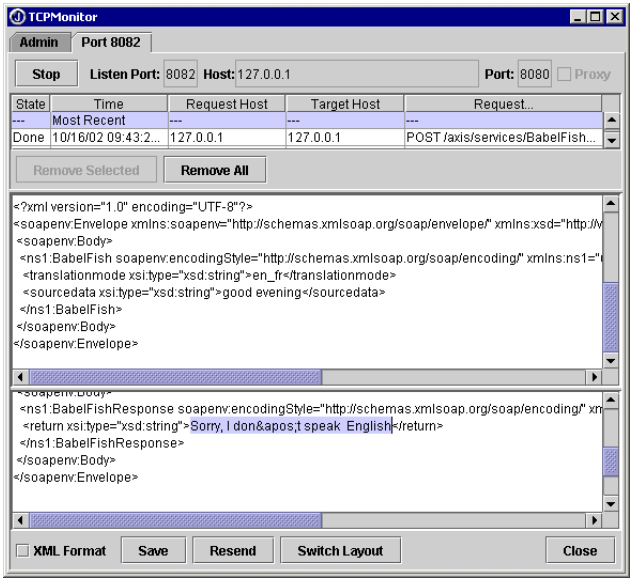


7 Modifique la solicitud en el panel de peticiones del TCPMonitor y vuelva a enviarla como se indica a continuación:

a Cambie el idioma en el elemento <translationmode> de `en_de` a `en_fr` para cambiar el idioma a francés:
`<translationmode xsi:type="xsd:string">en_fr</translationmode>`.

b Cambie el texto `good evening` en el elemento <sourcedata> a `goodbye`:
`<sourcedata xsi:type="xsd:string">goodbye</sourcedata>`

c Pulse Resend para enviar el mensaje revisado. El histórico situado en la parte superior del TCPMonitor muestra que el mensaje fue reenviado; además, una nueva respuesta se registra y aparece también en el panel de respuestas. La respuesta procedente del servicio es ahora: `Lo siento, no hablo inglés`. El TCPMonitor presenta un aspecto similar al siguiente:



8 Pare y cierre el TCPMonitor.

¡Enhorabuena! Ha completado este tutorial y ha creado un servicio web que traduce cadenas de texto por medio de BabelFish de AltaVista. También ha consumido el servicio albergado en el sitio de XMethod, ha consumido el servicio local que creó y ha realizado un seguimiento de los mensajes SOAP entre el cliente y el servidor mediante el TCPMonitor. Para obtener más información acerca de WSDL, consulte el Capítulo 5, "Utilización de WSDL". Para obtener más información sobre configuración de proyectos de JBuilder para servicios web, consulte el Capítulo 3, "Configuración de proyectos para servicios web".

Capítulo

15

Tutorial: Crear un servicio web desde una aplicación EJB con Borland Enterprise Server

En este tutorial se utilizan funciones de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Este tutorial, que supone un cierto conocimiento de Enterprise JavaBeans (EJB), muestra cómo crear un servicio web a partir de una aplicación EJB utilizando Borland Enterprise Server 5.0.2-5.1.x como servidor de aplicaciones y Axis como kit de herramientas de servicios web. El tutorial utiliza un ejemplo que reside en el directorio de ejemplos de JBuilder, `<jbuilder>/samples/tutorials/webservices EJBEmployeeBES/Employee.jpx`. El ejemplo contiene una aplicación EJB que accede a una base de datos de registros de empleados.

El tutorial incluye los siguientes pasos:

- Configuración del proyecto de ejemplo.
- Creación de un servidor de servicios web que albergue el servicio.
- Distribución del servicio web en el servidor de servicios web y exportación de una aplicación EJB como servicio web.
- Generación de código para el cliente y el servidor a partir del documento WSDL.
- Comprobación del funcionamiento del servicio
- Escritura del código del cliente y consumo del servicio.

Para obtener más información sobre los EJB, consulte *Guía del desarrollador de Enterprise JavaBeans*. Si desea obtener más información sobre los servicios web y los EJB, consulte el Capítulo 6, "Desarrollo de EJB como servicios web".

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Configuración del proyecto de ejemplo

Si aún no ha configurado el servidor Borland Enterprise Server 5.0.2-5.1.x, consulte "Configuración del servidor de aplicaciones de destino", en la *Guía del desarrollador de Enterprise JavaBeans*, para obtener instrucciones sobre cómo hacerlo.

1. Abra el ejemplo, `Employee.jpx`, situado en `<jbuilder>/samples/Tutorials/webservices/EJBEmployeeBES/`.
2. Antes de continuar con el siguiente paso, complete las instrucciones que aparecen en el archivo de proyecto del ejemplo, `EmployeeProject.html`, a fin de actualizar el servidor y conectar con la base de datos.

Paso 2: Crear un servidor de servicios web y distribuir en el servidor de aplicaciones

En este paso, utilizará el asistente Configuración de servicios web con el fin de crear una configuración de servicios web para una WebApp que alberga un servidor de servicios web. A continuación, distribuirá el servicio web en un archivo EAR. Al distribuir el servicio en el servidor, JBuilder expone automáticamente como servicio web cualquier bean sesión sin estado que contenga métodos en la interfaz remota.

1. Abra la galería de objetos seleccionando Archivo | Nuevo.
2. Haga clic en la pestaña Servicios web y doble clic en el icono Configuración de servicios web. El Asistente para configuración de servicios web configura un kit de herramientas para el servicio web y genera un servidor de servicios web para albergar el servicio. Para poder configurar el proyecto, deberá crear antes un recopilatorio EAR para la distribución del EJB y una WebApp para albergar el servidor de servicios web.

3. Cree un recopilatorio EAR como se indica a continuación. Distribuirá todo en un archivo EAR, incluido el servidor de servicios web de Axis.
   a. Pulse el botón Nuevo junto al campo EAR.
   b. Acepte el nombre por defecto para el EAR, `Employee`, y pulse Siguiente.
   c. En la pestaña Módulos EJB del paso 2, active ejb20module y pulse Finalizar para volver al asistente Configuración de servicios web.
4. Cree una WebApp como se indica a continuación:
   a. Pulse el botón Nuevo junto al campo WebApp.
   b. Escriba `axis` en los campos Nombre y Directorio.
   c. Pulse Aceptar para crear la WebApp y volver al asistente Configuración de servicios web.
5. Seleccione Axis de Apache como kit de herramientas y pulse Siguiente para ver la configuración de ejecución del servidor de servicios web que el asistente creará para generar el proyecto y ejecutar el servidor.
6. Pulse Finalizar para cerrar el asistente.
7. Expanda el nuevo nodo WebApp de Axis para ver los nodos y el archivo de distribución generados por el Asistente para configuración de servicios web. Cuando se ejecuta la configuración de ejecución del servidor de servicios web creada por el asistente, todos los módulos EJB y cualquier bean sesión sin estado del proyecto se distribuyen automáticamente como servicios web. En la ficha Propiedades, puede seleccionar qué módulos desea distribuir. Como en este ejemplo se van a exponer todos los elementos del proyecto, no es necesario hacer ningún cambio.
   a. Expanda el nodo Componentes del servicio web.
   b. Pulse con el botón secundario del ratón sobre el nodo Servicios basados en EJB y seleccione Propiedades. Podrá ver que el módulo

EJB y el bean aparecen expuestos como servicios. En este ejemplo, va a exponer todos los elementos del proyecto.

c Acepte los valores por defecto y pulse Aceptar para cerrar el cuadro de diálogo.

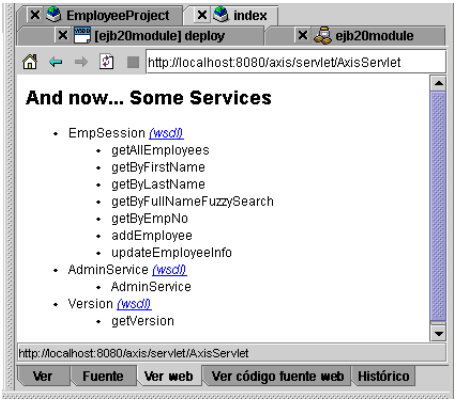
8 Guarde el proyecto y seleccione Proyecto | Ejecutar Make del proyecto para generarlo.

9 Seleccione Herramientas | Agente de gestión de Borland Enterprise Server. Este agente debe estar en funcionamiento para poder ejecutar el servidor Borland Enterprise Server.

10 En el panel del proyecto, pulse con el botón secundario del ratón sobre el nodo Employee.eargrp y seleccione Ejecutar utilizando "Servidor de servicios web" para ejecutar el servidor con la configuración de ejecución creada por el asistente Configuración de servicios web. Cuando se utiliza esta configuración de ejecución, los beans EJB se exportan automáticamente como servicios, el proyecto se genera y el servicio se distribuye en el servidor de aplicaciones. Una vez que el servidor está ya en funcionamiento, la ficha Apache-Axis aparece en el panel de contenido.

11 Pulse sobre el enlace Vista de la ficha Apache-Axis para comprobar si el servicio se ha distribuido correctamente. Si recibe un mensaje del tipo "500 Sin contexto", espere hasta que el servidor esté en funcionamiento y, a continuación, pulse el botón Actualizar, situado en la barra de

herramientas, para abrir la ficha Axis. Observe que todos los métodos de EmpSession se han distribuido.



El documento WSDL generado por el kit de herramientas también aparece aquí. Pulse sobre el enlace EmpSession (WSDL) para ver el documento WSDL.

Paso 3: Generar de código para el cliente y el servidor a partir del documento WSDL

Ahora que ha distribuido la aplicación EJB como servicio web, utilizará el Explorador de servicios web para buscar el servidor Axis que alberga el servicio, mostrar el servicio EJB, importar el WSDL que lo describe y generar clases de servidor, cliente y prueba para el servicio.

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Expanda el nodo Servidores WSDL y seleccione el nodo Por defecto en el árbol.
3. En la ficha Detalles, acepte la dirección URL por defecto para el servidor, `http://localhost:8080/axis/servlet/AxisServlet` (si no utiliza el número de puerto 8080, modifíquelo convenientemente).
4. Pulse el botón Mostrar servicios para mostrar el nuevo servicio EJB, EmpSession, en el árbol.
5. Seleccione el servicio EJB EmpSession en el árbol y pulse el botón Importar un servicio web; como consecuencia, el asistente Importar un servicio web se abre y permite importar el documento WSDL y generar las clases Java a partir de él.

6 Seleccione estas opciones en el asistente:

   a Seleccione Axis de Apache como kit de herramientas y pulse Aceptar.

   b Acepte la URL del WSDL y pulse Siguiente. En este ejemplo, no es necesario rellenar los campos Nombre de usuario y Contraseña.

   c Active Generar clases esqueleto. Esta opción crea un cliente que consume el servicio.

   d Pulse Siguiente y cambie el nombre del paquete de `localhost` a `com.borland.samples.emp.webservices`.

   e Acepte el resto de los valores por defecto del asistente. Observe que la opción Generar un test JUnit se encuentra activada. Más adelante, en este mismo tutorial, utilizará ese test para probar el funcionamiento del servicio.

   f Pulse el botón Finalizar.

   g En el panel del proyecto, expanda el nodo `com.borland.samples.emp.webservices` para ver los archivos generados por el asistente.

Para obtener información sobre los archivos generados por el asistente Importar un servicio web, consulte "Importación de archivos WSDL" en la página 7-6.

Paso 4: Comprobación del funcionamiento del servicio

En este paso, añadirá código al test JUnit generado por el asistente y probará el servicio. En primer lugar, añadirá nueva información de empleado al método `test6EmpSessionAddEmployee()`. A continuación, añadirá código al método `test7EmpSessionUpdateEmployeeInfo()`.

1 Expanda el nodo del paquete `com.borland.samples.emp.webservices` y haga doble clic en `EmpSessionServiceTestCase.java` para abrirlo en el editor.

2 Cambie el valor que aparece en el método `test5EmpSessionGetByEmpNo()` de 0 a 2 (en la base de datos, existe un empleado número 2, pero no un empleado número 0):

```
try {
            com.borland.samples.emp.data.EmployeeInfo value = null;
            value = binding.getByEmpNo((short)2);
        }
```

3 Añada los nuevos datos de empleado indicados en negrita al método `test6EmpSessionAddEmployee()`:

```
try {
            boolean value = false;
com.borland.samples.emp.data.EmployeeInfo empInfo =
                  new com.borland.samples.emp.data.EmployeeInfo();
            empInfo.setEmpNo(new Short((short)301));
            empInfo.setFirstName("Jim");
            empInfo.setLastName("Shorts");
            empInfo.setFullName("Shorts, Jim");
            value = binding.addEmployee(empInfo);
        }
```

4 Añada el siguiente código indicado en negrita al método `test7EmpSessionUpdateEmployeeInfo()`:

```
try {
            boolean value = false;
com.borland.samples.emp.data.EmployeeInfo empInfo =
                  new com.borland.samples.emp.data.EmployeeInfo();
            empInfo.setFirstName("Sally");
            empInfo.setEmpNo(new Short((short)2));
            value = binding.updateEmployeeInfo(empInfo);
        }
```

5 En el panel del proyecto, pulse con el botón secundario del ratón sobre `EmpSessionServiceTestCase.java` y seleccione Ejecutar test utilizando valores por defecto para probar el servicio. El proyecto se genera y, a continuación, se ejecutan los tests. Observe que los tests resultan satisfactorios. Si desea más información sobre los tests JUnit, consulte "Tests de módulos" en *Creación de aplicaciones con JBuilder*.

6 Cierre la pestaña EmpSessionServiceTestCase en el panel de mensajes.

Paso 5: Escritura del código del cliente y consumo del servicio

A continuación, escribirá un cliente sencillo que obtiene todos los registros de empleados de la base de datos y los muestra en el panel de mensajes. Este cliente sólo constará de un método, `empSessionGetAllEmployees()`.

1 Seleccione Archivo | Nueva clase para abrir el Asistente para clases.

2 Escriba `com.borland.samples.emp.client` en el campo Paquete.

3 Escriba `Client` en el campo Clase.

4 Elimine el contenido de `Client.java` y reemplácelo por este código:

```
package com.borland.samples.emp.client;

public class Client {
  public static void empSessionGetAllEmployees() {
      com.borland.samples.emp.webservices.EmpSession binding;
```

```
        try {
            binding = new com.borland.samples.emp.
                webservices.EmpSessionServiceLocator().getEmpSession();
        }
        catch (javax.xml.rpc.ServiceException jre) {
            jre.printStackTrace();
     return;
        }
     try {
            com.borland.samples.emp.data.EmployeeInfo[] value = null;
            value = binding.getAllEmployees();
            if (value != null) {
              for (int i=0; i < value.length; i++) {
                System.out.println(value[i]);
              }
            }
        }
        catch (java.rmi.RemoteException re) {
          re.printStackTrace();
     return;
        }
    }
    public static void main(String[] args) {
       empSessionGetAllEmployees();
    }
  }
```

El cliente tiene que buscar en enlace para el servicio, que se encuentra en la clase Locator. Observe que la primera sentencia try especifica el enlace para el servicio y llama a la clase `Locator`. La clase Locator es la implementación del servidor, para la parte cliente, de la interfaz del servicio.

```
try {
          binding = new com.borland.samples.emp.
              webservices.EmpSessionServiceLocator().getEmpSession();
      }
```

5 En el panel del proyecto, pulse con el botón secundario del ratón sobre `Client.java` y seleccione Ejecutar utilizando el cliente. Los registros de empleados se muestran en el panel de mensajes.

También podría añadir otros métodos al cliente para obtener más información sobre los empleados.

Una vez distribuida la aplicación EJB y creado y comprobado el servicio web, puede publicarla en un registro UDDI desde un servidor Axis. Para obtener más información, consulte "Difusión de servicios web desde un servidor Axis" en la página 11-26.

Enhorabuena. Ha finalizado este tutorial. Acaba de configurar el proyecto para servicios web, exportar automáticamente una aplicación EJB como servicio web, distribuir el servicio a un servidor web, hospedarlo en un servidor Axis y comprobarlo.

Capítulo

16

Tutorial: Importación de servicios web como aplicaciones EJB

En este tutorial se utilizan funciones de JBuilder Enterprise. Con JBuilder WebLogic Edition no se proporciona asistencia para el kit de herramientas Axis de Apache

Este tutorial, en el que presupone un cierto conocimiento de Enterprise JavaBeans (EJB), muestra cómo crear una aplicación EJB a partir de un servicio web existente utilizando Borland Enterprise Server 5.0.2-5.1.x como servidor de aplicaciones. El tutorial utiliza un documento WSDL (Lenguaje de descripción de servicios web) de ejemplo que reside en el directorio `samples` de JBuilder: `<jbuilder>/samples/tutorials/webservices/wsdl`.

El tutorial incluye los siguientes pasos:

- Adición de un documento WSDL al proyecto.
- Creación de una aplicación EJB a partir del documento WSDL.
- Comprobación del funcionamiento de la aplicación EJB.

Para obtener más información sobre los EJB, consulte *Guía del desarrollador de Enterprise JavaBeans*. Si desea obtener más información sobre los servicios web y los EJB, consulte el Capítulo 6, "Desarrollo de EJB como servicios web".

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Adición de un documento WSDL al proyecto

En este paso, se crea un nuevo proyecto, se establece Borland Enterprise Server 5.0.2-5.1.x como servidor y se añade un documento WSDL al documento. Si aún no ha configurado el servidor Borland Enterprise Server 5.0.2-5.1.x, consulte "Configuración del servidor de aplicaciones de destino", en la *Guía del desarrollador de Enterprise JavaBeans*, para obtener instrucciones sobre cómo hacerlo.

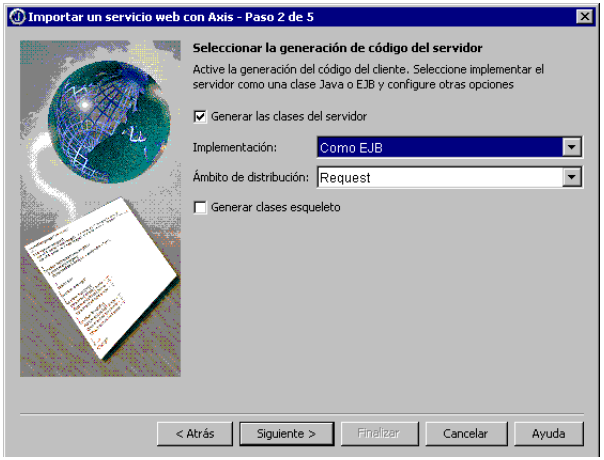
1. Elija Archivo | Nuevo proyecto para iniciar el Asistente para proyectos.
2. Escriba `interop` como nombre del proyecto y pulse Aceptar.
3. Configure el proyecto de modo que utilice Borland Enterprise Server. Se debe configurar el servidor (Herramientas | Configura servidores) para que aparezca en la lista de servidores.
   a. Seleccione Proyecto | Propiedades de proyecto, y a continuación abra la pestaña Servidor.
   b. Seleccione Borland Enterprise Server AppServer Edition 5.0.2-5.1.x en la lista desplegable de servidor único y pulse Aceptar.
4. Seleccione la opción de menú Proyecto | Añadir archivos/paquetes y busque y seleccione el archivo `InteropTest.wsdl` en el directorio `<jbuilder>/samples/webservices/axis/wsdl`.
5. Pulse Aceptar para añadir el archivo WSDL al proyecto.

Paso 2: Creación de una aplicación EJB a partir del documento WSDL

A continuación, importará el documento WSDL y utilizará el asistente Importar un servicio web con el fin de generar una aplicación EJB a partir del WSDL.

1. En el panel del proyecto, pulse con el botón secundario del ratón sobre `Interoptest.wsdl` y seleccione Importar un servicio web para abrir el asistente.
2. Seleccione Axis de Apache como kit de herramientas y pulse Aceptar.
3. Acepte `InteropTest.wsdl` como la URL del WSDL y pulse Siguiente. No se requiere nombre de usuario ni contraseña.

4 En el paso 2, en la lista desplegable Implementación, seleccione Como EJB y pulse Siguiente.



5 Pulse Siguiente para aceptar las opciones por defecto del Paso 3.

6 En el paso 4, pulse Nuevo y se abrirá el Asistente para grupos EJB vacíos, con el que podrá crear un módulo para la aplicación EJB.

7 Acepte el nombre por defecto para el módulo, seleccione Compatible con EJB 2.0 como versión y pulse Aceptar para cerrar el asistente.

8 En el asistente Importar un servicio web, pulse Finalizar.

El asistente Importar un servicio web permite crear una aplicación EJB, así como la implementación de la parte del servidor del servicio web a partir del documento WSDL. En el panel del proyecto, expanda el paquete `org.soapinterop` y podrá ver los archivos EJB generados. Observe también que se ha creado un módulo EJB.

Paso 3: Comprobación del funcionamiento de la aplicación EJB

En este paso del tutorial, creará e implementará un cliente de prueba EJB para comprobar el funcionamiento de la aplicación EJB.

1 Seleccione Archivo | Nuevo | Enterprise y haga doble clic sobre el icono Cliente de prueba EJB.

2 Seleccione Aplicación como tipo de cliente de prueba.

3 Pulse Siguiente.

4 Acepte todos los valores por defecto y haga clic en Finalizar. El cliente de prueba, `InteropSessionPortTypeTestClient1.java`, se crea en el paquete `org.soapinterop`.

5. Añada al método `main()` de `InteropSessionPortTypeTestClient1.java` el código que aparece en negrita:

```
public static void main(String[] args) {
  InteropTestPortTypeSessionTestClient1 client =
      new InteropTestPortTypeSessionTestClient1();
client.create();
  client.echoString("Test");
}
```

6. Abra `InteropTestPortTypeSessionBean.java` en el editor e implemente el método `echoString()` de modo que devuelva un valor.

```
public java.lang.String echoString(java.lang.String inputString)
    throws java.rmi.RemoteException  {
   return inputString;
  }
```

7. Ejecute el servidor:
   a. Seleccione Herramientas | Borland Enterprise Server Agente de gestión.
   b. Haga clic con el botón derecho del ratón sobre el módulo EJB y seleccione Ejecutar utilizando valores por defecto.
8. En el panel del proyecto, pulse con el botón derecho del ratón sobre `InteropTestPortTypeSessionTestClient1.java` y seleccione Ejecutar utilizando InteropTestPortTypeSessionTestClient1. El valor devuelto se muestra en el panel de mensajes.

```
Valor devuelto por echoString(Test): Test.
```

Enhorabuena. Ha finalizado este tutorial. En este tutorial, ha aprendido a crear una aplicación EJB a partir de un documento WSDL, a generar las clases de la parte del servidor mediante el asistente Importar un servicio web y a crear y ejecutar un test para EJB.

Capítulo

17

Tutorial: Creación de un servicio web sencillo con WebLogic

En este tutorial se utilizan funciones de JBuilder Enterprise

Este tutorial muestra cómo utilizar el asistente Exportar como servicio web con el fin de exportar un JavaBean como servicio web, generar un documento WSDL que describa el servicio y crear un servidor de servicios web utilizando el servidor WebLogic Server para albergar el servicio y el kit de herramientas WebLogic para generarlo.

En este tutorial, se completan las siguientes tareas:

- Creación de un JavaBean de ejemplo.
- Exportación del bean de ejemplo como servicio web y configuración del proyecto para servicios web.
- Distribución y ejecución del servicio.
- Comprobación del funcionamiento del servicio en un navegador web.
- Escritura del código de un cliente para probar el funcionamiento del servicio.

Este tutorial asume que usted está familiarizado con Java y con el IDE (entorno integrado de desarrollo) de JBuilder. Para obtener más información sobre las JSP, consulte *Procedimientos iniciales con Java.* Si desea obtener más información sobre el IDE de JBuilder, consulte "El entorno de JBuilder" en *Introducción a JBuilder*.

Si desea obtener más información sobre el servidor WebLogic y los servicios web, consulte la documentación de BEA en `http://edocs.bea.com/wls/docs70/webservices.html`.

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Creación de un JavaBean de ejemplo

En este paso, creará un proyecto mediante el Asistente para proyectos con WebLogic como servidor. A continuación, creará un JavaBean que exportará como servicio web.

1. Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
2. Introduzca `exportbean` en el campo Nombre.

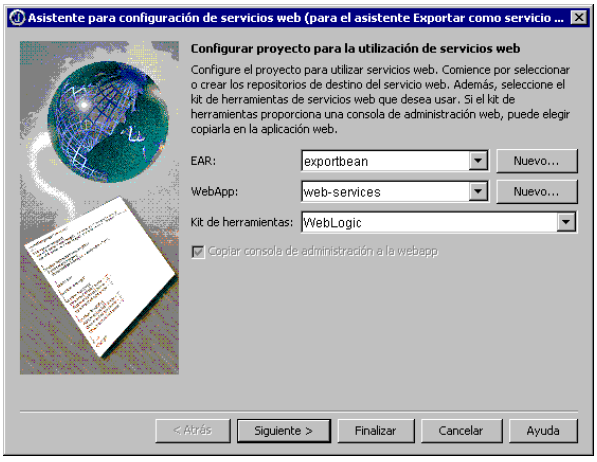
Importante Asegúrese de que crea el proyecto en un directorio **sin** espacios en el nombre o, de lo contrario, el servidor WebLogic generará errores.

3. Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.
4. Seleccione Proyecto | Propiedades de proyecto, abra la pestaña Servidor y seleccione WebLogic Application Server 7.x como servidor único para el proyecto. Si no aparece en la lista, deberá activarlo en el cuadro de diálogo Configurar servidores (Herramientas | Configurar servidores).
5. Elija Archivo | Nuevo para abrir la galería de objetos y pulse la pestaña General.
6. Seleccione JavaBean y pulse Aceptar para iniciar el Asistente para JavaBean.
7. Seleccione `java.lang.Object` como clase base, active Generar propiedad de ejemplo y acepte todos los demás valores por defecto.
8. Pulse Aceptar para cerrar el asistente. En el paquete `exportbean` se genera un archivo `Bean1.java`.

Paso 2: Exportación del bean de ejemplo como servicio web

En este paso, configurará el proyecto para servicios web y utilizará el asistente Exportar como servicio web con el fin de exportar el bean como un servicio web. El asistente expone todos los métodos del bean en el servicio y crea un archivo de distribución de servicios web.

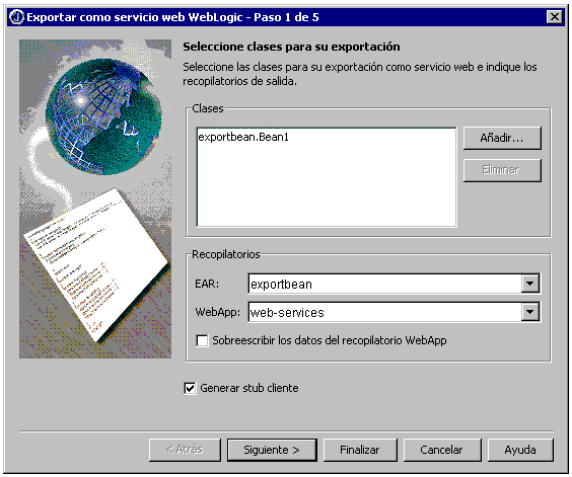
1. En el panel del proyecto, pulse con el botón derecho del ratón sobre `Bean1.java` y seleccione Exportar como servicio web. Como el proyecto no está configurado aún para servicios web, se abrirá el Asistente para configuración de servicios web de modo que pueda crear una WebApp para hospedar el servicio web y un EAR. Después de haber configurado el proyecto, aparecerá el asistente Exportar como servicio web.
2. Configure el proyecto para servicios web mediante el asistente Configuración de servicios web como se indica a continuación:
   - **a** Pulse el botón Nuevo, situado junto al campo EAR, y cree un EAR mediante el asistente que aparece.
   - **b** Acepte `exportbean` como nombre por defecto para el EAR y pulse Finalizar para volver al Asistente de configuración de servicios web.
   - **c** Seleccione WebLogic en la lista desplegable Kit de herramientas.
   - **d** Pulse el botón Nuevo, situado junto al campo WebApp, y cree una WebApp mediante el Asistente para aplicaciones web. Es necesario que haya una WebApp para hospedar el servicio.
   - **e** Acepte `web-services` como nombre y directorio de la WebApp y pulse Aceptar para volver al Asistente de configuración de servicios web. El asistente Configuración de servicios web presentará el siguiente aspecto:



f Pulse Siguiente y podrá ver la configuración de ejecución del servidor de servicios web creada por el asistente. Utilizará esta configuración de ejecución más adelante para ejecutar el proyecto.

g Pulse Finalizar para cerrar el Asistente de configuración de servicios web y continuar con el asistente Exportar como servicio web.

Ahora que ya ha configurado el proyecto para servicios web, utilizará el asistente Exportar como servicio web con el fin de exportar la clase como un servicio web. El asistente Exportar como servicio web presenta el siguiente aspecto:



1 Acepte los valores por defecto del paso 1 del asistente, tales como la clase que se va a exportar, el EAR y la WebApp. Observe que la opción Generar stub cliente se encuentra activada. Cuando esta opción está activada, el asistente genera un JAR que contiene las interfaces, stubs y clases que una aplicación cliente necesita para llamar el servicio.

2 Pulse Siguiente para pasar al Paso 2.

3 Acepte el resto de los valores por defecto y observe que el nombre del identificador ServiceURI es `exportbean`. Utilizará el identificador ServiceURI más adelante para probar el servicio distribuido.

4 Continúe con el paso 4 para ver los nombres por defecto del paquete y del JAR cliente que se generan para el servicio: `exportbean.generated` y `Bean1_client.jar`.

5 Pulse Finalizar para cerrar el asistente y generar el servicio web.

6 Expanda el nodo WebApp de servicios web y el nodo Componentes del servicio web creado por el Asistente de configuración de servicios web. A continuación, expanda el nodo Servicios basados en Java para ver el archivo de distribución WLDU creado por el asistente. Este archivo de distribución proporciona información sobre la distribución de servicios

web para el servidor WebLogic, como es el nombre del EAR, del WAR y de la clase que se va a exportar. Cuando el servicio se distribuye en el servidor, se utiliza la información de distribución con el fin de crear el servicio web y el documento WSDL a partir de la clase seleccionada.

7 Expanda el nodo GeneratedWebServicesClients y haga doble clic sobre el archivo `Bean1_client.jar` creado por el asistente para ver su contenido en el panel de estructura.

8 En el panel de estructura, expanda el exportbean y los nodos generados y podrá ver las clases y el documento WSDL generado por el kit de herramientas WebLogic.

Paso 3: Ejecución del servidor y distribución del servicio

En este paso, ejecutará el servidor WebLogic con la configuración de ejecución para servidor de servicios web y distribuirá el servicio en el servidor.

1 Elija Ejecutar | Ejecutar proyecto para ejecutar el servidor. Cuando se ejecuta el proyecto, se utiliza la configuración de ejecución creada por el Asistente de configuración de servicios web, Servidor de servicios web. Esta configuración de ejecución genera el proyecto y ejecuta el servidor WebLogic como servidor de servicios web. Para obtener más información sobre las configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.

2 En el panel del proyecto, pulse con el botón derecho del ratón sobre el nodo exportbean.eargrp después de que el servidor se haya ejecutado y seleccione Opciones de distribución | Distribuir para distribuir el servicio en el servidor WebLogic.

Importante Si realiza cambios en `Bean1.java` después de haberlo exportado, debería volver a compilarlo, exportarlo como servicio web y distribuirlo para que los cambios surtan efecto.

Paso 4: Comprobación del servicio distribuido

Una vez distribuido el servicio, puede probarlo desde la página de inicio de servicios web de WebLogic con un navegador web.

1 Abra un navegador web y escriba `http://localhost:7001/web-services/exportbean`, que es la dirección del servidor donde se ha distribuido el servicio. El puerto en el que se ejecuta el servidor se especifica en la configuración de ejecución del servidor de servicios web, en la ficha Ejecutar de Propiedades de proyecto. Para ver los valores de la

configuración de ejecución del servidor de servicios web, seleccione Ejecutar | Configuraciones.

Los métodos expuestos en la clase se muestran en la página de inicio de servicios web de WebLogic: `getSample()` y `setSample()`. En esta página, también puede ver el documento WSDL generado si pulsa sobre el enlace Services Description. La página principal de pruebas dispone también de código de ejemplo para llamar al servicio; puede copiar ese código, modificarlo y utilizarlo para probar el servicio que acaba de distribuir.

2. Pulse sobre el enlace setSample y observe que el valor de setSample es "sample string".
3. Pulse el botón Invoke y podrá ver la solicitud SOAP dirigida al servidor, así como su respuesta.
4. Vuelva a la primera ficha, haga clic sobre el enlace getSample y pulse el botón Llamar para obtener el valor del ejemplo. La respuesta del servidor devuelve el valor "sample string".
5. Vuelva a la primera página y pulse sobre el enlace Services Description para ver el WSDL generado para el servicio. Observe que tanto el método `getSample()` como el método `setSample()` se exponen como operaciones en la descripción del servicio.

Paso 5: Escritura del código de un cliente para probar el funcionamiento del servicio

A continuación, escribirá un cliente con el que probar el servicio web exportado.

1. Vuelva a la primera página de la página de inicio y copie el código de ejemplo que llama al servicio:

```
import {package}.Bean1;
    ...

    String wsdlUrl = "http://localhost:7001/web-services/exportbean?WSDL";
    Bean1 service = new Bean1_Impl( wsdlUrl );
    Bean1Port port = service.getBean1Port();

    result = port.getSample( ... );
```

2. Seleccione Archivo | Nueva clase para crear una clase.
3. Escriba `test` en el campo Paquete.
4. Escriba `Test` en el campo método Clase.
5. Active Generar Main.
6. Desactive Generar constructor por defecto.

7 Pulse Aceptar para crear la nueva clase de test.

8 Pegue el código de ejemplo en la clase y modifíquelo del siguiente modo:

```
package test;

import exportbean.generated.*;
import java.io.*;

public class Test {
  public static void main(String[] args) {

   try {
    String wsdlUrl = "http://localhost:7001/web-services/exportbean?WSDL";
    Bean1 service = new Bean1_Impl( wsdlUrl );
    Bean1Port port = service.getBean1Port();

          System.out.println("Output is "+ port.getSample());
    } catch (IOException ex) {
       ex.printStackTrace();
        }
  }
}
```

9 Haga clic con el botón derecho del ratón en `Test.java` en el panel del proyecto y pulse Ejecutar utilizando valores por defecto. El resultado aparece en el panel de mensajes: `El resultado es Sample`.

¡Enhorabuena! Ha finalizado este tutorial. En él, ha aprendido a exportar un JavaBean como servicio web, a distribuirlo en el servidor WebLogic, a probarlo desde la página de inicio de servicios web de WebLogic y a crear un cliente para probarlo. Para obtener más información sobre el kit de herramientas WebLogic, consulte el Capítulo 8, “El kit de herramientas WebLogic”.

Capítulo

18

Tutorial: Creación de un servicio web desde una aplicación EJB con WebLogic Server

En este tutorial se utilizan funciones de JBuilder Enterprise

Este tutorial, en el que se presupone un cierto conocimiento de Enterprise JavaBeans (EJB), muestra cómo crear un servicio web a partir de una aplicación EJB utilizando WebLogic Application Server 7.x como servidor de aplicaciones. El tutorial utiliza un ejemplo que reside en el directorio de `ejemplos` de JBuilder, `<jbuilder>/samples/tutorials/webservices/EJBEmployeeWL/Employee.jpx`. El ejemplo contiene una aplicación EJB que accede a una base de datos de registros de empleados.

El tutorial incluye los siguientes pasos:

- Configuración del proyecto para servicios web.
- Distribución de una aplicación EJB como servicio web.
- Generación de código para el cliente a partir del documento WSDL.
- Escritura de una aplicación cliente que consume el servicio de forma local.

Para obtener más información sobre los EJB, consulte la *Guía del desarrollador de Enterprise JavaBeans*. Si desea obtener más información sobre los servicios web y los EJB, consulte el Capítulo 6, "Desarrollo de EJB como servicios web".

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Configuración del proyecto

Si aún no ha configurado el servidor WebLogic Server 7.x, consulte "Configuración del servidor de aplicaciones de destino", en la *Guía del desarrollador de Enterprise JavaBeans*, para obtener instrucciones sobre cómo hacerlo.

1. Abra el ejemplo `Employee.jpx`, que reside en `<jbuilder>/samples/Tutorials/webservices/EJBEmployeeWL/`.
2. Antes de continuar con el siguiente paso, complete las instrucciones que aparecen en el archivo de proyecto del ejemplo, `EmployeeProject.html`, para actualizar el servidor, añadir bibliotecas necesarias, conectar con la base de datos y crear una fuente de datos transaccional de WebLogic.

Paso 2: Configuración del proyecto para servicios web

En este paso, utilizará el asistente Configuración de servicios web con el fin de crear una configuración de servicios web para una WebApp que alberga un servidor de servicios web.

1. Abra la galería de objetos seleccionando Archivo | Nuevo.
2. Haga clic en la pestaña Servicios web y doble clic en el icono Configuración de servicios web. El Asistente de configuración de servicios web configura un kit de herramientas para el servicio web y genera un servidor de servicios web para albergar el servicio. Para poder configurar el proyecto, deberá crear antes un recopilatorio EAR para la distribución del EJB y una WebApp para albergar el servidor de servicios web.

3. Cree un recopilatorio EAR como se indica a continuación. Todos los elementos se distribuirán en un archivo EAR.
   a. Pulse el botón Nuevo junto al campo EAR.
   b. Acepte el nombre por defecto para el EAR, `Employee`, y pulse Siguiente.
   c. En la pestaña Módulos EJB del paso 2, active ejb20module y pulse Finalizar para volver al asistente Configuración de servicios web.
4. Seleccione WebLogic como kit de herramientas y pulse Siguiente para ver la configuración de ejecución del servidor de servicios web que el asistente creará para generar el proyecto y ejecutar el servidor WebLogic.
5. Cree una WebApp como se indica a continuación:
   a. Pulse el botón Nuevo junto al campo WebApp.
   b. Acepte `web-services` en los campos Nombre y Directorio y pulse Aceptar para crear la WebApp y volver al Asistente para configuración de servicios web.
6. Pulse Finalizar para cerrar el asistente.

Paso 3: Distribución de un EJB como servicio web

En este paso, aprenderá a distribuir los EJB como servicios web a un archivo EAR. Al distribuir el servicio en el servidor WebLogic, JBuilder expone automáticamente como servicio web cualquier bean sesión sin estado que contenga métodos en la interfaz remota.

1. Expanda el nuevo nodo WebApp de servicios web para ver los nodos y el archivo de distribución generados por el Asistente para configuración de servicios web. Cuando se ejecuta la configuración de ejecución del servidor de servicios web creada por el asistente, todos los módulos EJB y cualquier bean sesión sin estado del proyecto se distribuyen automáticamente como servicios web. En la ficha Propiedades, puede seleccionar qué módulos desea distribuir. Como en este ejemplo se van a exponer todos los elementos del proyecto, no es necesario hacer ningún cambio.
   a. Expanda el nodo Componentes del servicio web.

b Pulse con el botón secundario del ratón sobre el nodo Servicios basados en EJB y seleccione Propiedades. Podrá ver que el módulo EJB y el bean aparecen expuestos como servicios. En este ejemplo, va a exponer todos los elementos del proyecto.

c Acepte los valores por defecto y pulse Aceptar para cerrar el cuadro de diálogo.

2 Seleccione Proyecto | Ejecutar Make del proyecto para generarlo.

3 Inicie el servidor pulsando sobre la flecha de despliegue que aparece junto al botón Ejecutar de la barra de herramientas principal y seleccioner Servidor de servicios web. Esta configuración de ejecución genera el proyecto, distribuye automáticamente los EJB e inicia el servidor WebLogic.

4 En el panel del proyecto, pulse con el botón secundario del ratón sobre el nodo Employee.eargrp y seleccione Opciones de distribución | Distribuir para distribuir el servicio en el servidor WebLogic.

Paso 4: Generación de código para el cliente a partir del documento WSDL

Ahora que ya ha distribuido la aplicación EJB como un servicio web, va a importar el servicio web desde el archivo recopilatorio EAR y generará clases para el cliente, así como un documento WSDL que describe el servicio.

1 Pulse con el botón secundario del ratón sobre Employee.eargrp y seleccione Importar un servicio web para abrir el asistente

correspondiente, el cual genera las clases Java de cliente y un documento WSDL.

2. Seleccione estas opciones en el asistente:
   a. Acepte el nombre del EAR, el nombre de la WebApp y el servicio que se va a importar y pulse Siguiente.
   b. Cambie el nombre del paquete a `com.borland.samples.emp.generated`
   c. Acepte el resto de los valores por defecto del asistente.
   d. Pulse el botón Finalizar.
   e. Amplíe el nodo GeneratedWebServiceClients y haga doble clic sobre el archivo `EmpSession_client.jar`, que contiene el servicio web generado y el WSDL.

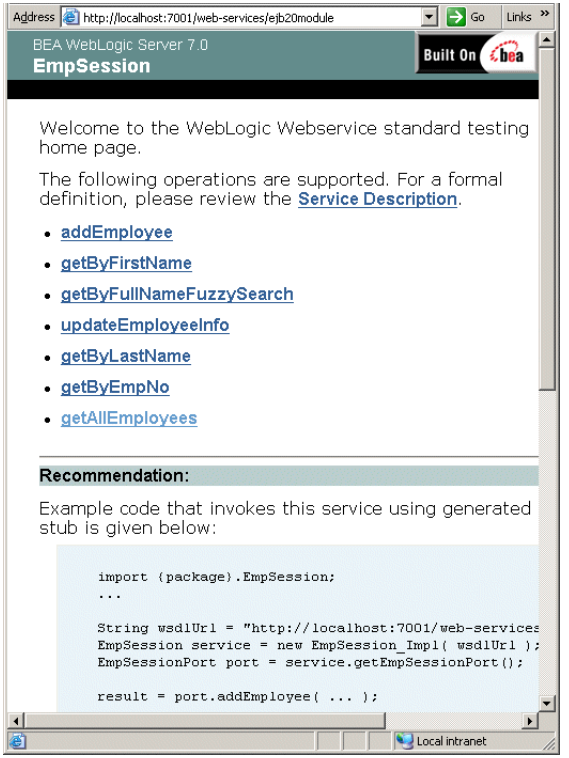
Paso 5: Escritura del código del cliente y consumo del servicio de forma local

A continuación, escribirá un cliente sencillo que obtiene todos los registros de empleados de la base de datos y los muestra en el panel de mensajes. En primer lugar, vaya a la página de inicio de servicios web de WebLogic para compiar el código de ejemplo que llama al servicio. A continuación, modifíquelo.

1. Abra un navegador web y especifique en él la dirección del servicio distribuido: `http://localhost:7001/web-services/ejb20module`. Si está ejecutando el servidor WebLogic en otro puerto, modifique la dirección del puerto. El valor de la dirección URI del servicio (ServiceUri) se encuentra en el archivo de distribución `servicegen.wldu`: `serviceURI="ejb20module"`.

   Si la distribución se realiza correctamente, los métodos expuestos en EmpSession aparecen en la página de inicio. También existe una enlace

al WSDL del servicio y al código de ejemplo que ha de copiar y modificar para llamar al servicio de forma local.



2 Copie estas líneas de código de la página de inicio:

```
String wsdlUrl ="http://localhost:7001/web-services/ejb20module?WSDL";
EmpSession service = new EmpSession_Impl( wsdlUrl );
EmpSessionPort port = service.getEmpSessionPort();
```

3 Vuelva a JBuilder y seleccione Archivo | Nueva clase para crear una clase.

4 Realice estas modificaciones en el Asistente para clases:

   a Escriba `com.borland.samples.emp.client` en el campo Paquete.

   b Escriba `Client` en el campo Clase.

   c Active Generar método Main.

   d Desactive Generar constructor por defecto.

   e Pulse Aceptar para cerrar el asistente y crear la clase.

5 Añada estas sentencias de importación bajo el nombre del paquete:

```
import com.borland.samples.emp.generated.*;
import java.io.*;
```

6. Pegue el código que se copió de la página de inicio de servicios web de WebLogic en el método principal de la nueva clase Cliente.
7. Seleccione el código que acaba de añadir y utilice `Mayús+Ctrl+C` para insertar el código en una sentencia try/catch.
8. Añada el código en negrita en la sentencia catch.

```
catch (IOException ex) {
       ex.printStackTrace();
        }
```

9. Añada una llamada al método `getAllEmployees()`, sentencias para extraer información de la matriz y un sentencia print para imprimir todos los informes de los empleados de la base de datos.

```
//llamada a getAllEmployees
EmployeeInfo [] employees = port.getAllEmployees();
//extraer información de la matriz.
for (int i=0; i < employees.length; i++) {
  //comprobar si se han devuelto valores null desde la base de datos.
  if (employees[i] != null) {
    //si no se ha devuelto ningún null, imprimir el nombre completo del
empleado.
    System.out.println(employees[i].getFullName());
  }
```

Toda la clase Cliente debe tener el siguiente aspecto:

```
public class Client {
  public static void main(String[] args) {
   try {
      String wsdlUrl ="http://localhost:7001/web-services/ejb20module?WSDL";
    EmpSession service = new EmpSession_Impl( wsdlUrl );
    EmpSessionPort port = service.getEmpSessionPort();
      //llamada a getAllEmployees
EmployeeInfo [] employees = port.getAllEmployees();
//extraer información de la matriz.
for (int i=0; i < employees.length; i++) {
  //comprobar si se han devuelto valores null desde la base de datos.
  if (employees[i] != null) {
    //si no se ha devuelto ningún null, imprimir el nombre completo del
empleado.
    System.out.println(employees[i].getFullName());
        }
      }
    } catch (IOException ex) {
       ex.printStackTrace();
    }
  }
}
```

10 Haga clic con el botón derecho del ratón sobre la pestaña del archivo `Client.java` en el editor y seleccione Ejecutar Make para compilar la clase.

11 Haga clic con el botón derecho del ratón sobre la pestaña del archivo `Client.java` en el editor y seleccione Ejecutar utilizando el cliente para ejecutar la clase. Los registros de empleados se muestran en el panel de mensajes.

Enhorabuena. Ha finalizado este tutorial. Ha aprendido a configurar el proyecto para servicios web, a distribuir la aplicación EJB como servicio web, a generar clases de cliente y a comprobarlo.

Para obtener más información sobre el kit de herramientas WebLogic, consulte el Capítulo 8, “El kit de herramientas WebLogic”.

Capítulo

19

Tutorial: Búsqueda de servicios web UDDI

En este tutorial se utilizan funciones de JBuilder Enterprise

JBuilder incluye el Explorador de servicios Web, que sirve para examinar los registros UDDI y realizar consultas en ellos. Los registros UDDI son bases de datos que utilizan las empresas para registrar sus servicios Web y buscar otros servicios disponibles. Estos registros se basan en el marco de trabajo *UDDI* (Universal Description, Discovery, and Integration, Descripción, detección e integración universales). UDDI describe los servicios mediante XML (Extensible Markup Language, lenguaje de marcas ampliable), y en ocasiones, Web Services Description language (WSDL Web Services Description Language, Lenguaje de descripción de servicios web).

En este tutorial se explica la forma de utilizar el Explorador de servicios web por medio de las siguientes tareas.

- Examinar servicios web en el sitio Xmethods.
- Examinar tModels.
- Buscar empresas de publicación de software en el sitio UDDI de Microsoft.
- Generar clases Java.

Importante La función UDDI del Explorador de servicios web requiere el uso de la extensión Java Secure Sockect Extension (JSSE) de Sun. Si está utilizando el JDK 1.3 o una versión anterior, debe descargar e instalar la Extensión JSSE de Sun en `http://java.sun.com/products/jsse/index-103.html`.

Consulte

- "Términos y definiciones UDDI" en la página 11-4
- La especificación UDDI en `http://www.uddi.org/specification.html`
- Capítulo 11, "Búsqueda y publicación de servicios web"

Consulte el apartado Tutoriales en las Sugerencias de JBuilder si desea información sobre cómo ver e imprimir tutoriales. El apartado Opciones de accesibilidad de Sugerencias de JBuilder contiene sugerencias sobre la utilización de las funciones de JBuilder para mejorar la facilidad de uso de JBuilder para personas con discapacidades.

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte "Convenciones de la documentación" en la página 1-4.

Paso 1: Examen de servicios web en el sitio Xmethods

En este paso se utiliza el Explorador de servicios web para buscar los servicios web de XMethods, una organización que promueve el desarrollo, la distribución y el uso de los servicios web. Si desea más información sobre XMethods, visite `http://www.xmethod.net`.

1. Seleccione Herramientas | Explorador de servicios web para abrir el Explorador.
2. Amplíe el nodo Sitios UDDI del árbol de la izquierda y elija el nodo del sitio del operador XMethods.
3. Seleccione el nodo Lista de negocios para abrir la ficha Consulta por negocio a la derecha.

4 Escriba la letra **s** en el campo Buscar por nombre para buscar todas las empresas incluidas en el sitio que empiezan por "s"; para ejecutar la consulta, pulse el botón Ejecutar, situado en la ficha Consultar empresa. También puede seleccionar Consulta | Ejecutar o pulsar el botón Ejecutar consulta de la barra de herramientas.

Sugerencia Si desea buscar todas las empresas de un sitio, deje en blanco el campo Buscar por nombre.

Importante Es posible aumentar el tiempo de espera del Explorador de servicios web. Elija UDDI | Configuración de conexión y escriba un nuevo valor en el campo Tiempo de espera. Para aplicar este valor a las conexiones, cierre el Explorador y vuelva a abrirlo. Es posible que el servidor detenga la búsqueda si determina que el resultado es demasiado voluminoso.

5 Amplíe el nodo Lista de negocios de XMethods para ver todos los negocios cuyo nombre empieza por “s” en el sitio XMethods.

6 Busque el negocio SQLData como se indica a continuación:

a Vuelva a seleccionar el nodo Lista de negocios de Xmethods.

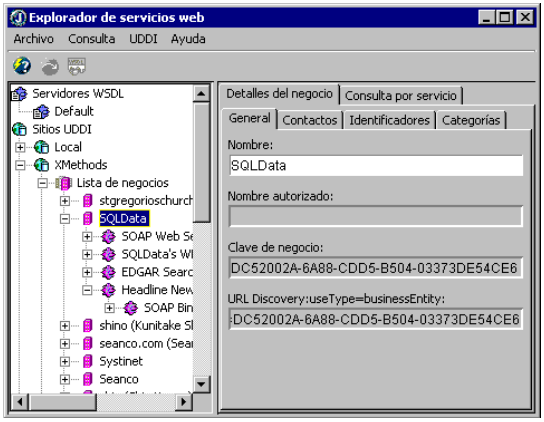
b Abra la pestaña Nombre de la ficha Consulta por negocio.

c Escriba `SQLData` en el campo Buscar por nombre.

d Pulse la tecla *Intro* o el botón Ejecutar.

7 Seleccione el nodo SQLData en el árbol.

A la derecha, aparece la ficha Detalles del negocio, que muestra información general y datos de contacto, identificación y categoría, si están disponibles. A la derecha aparece también la ficha Consulta por servicio. Si conoce el nombre del servicio, puede escribirlo en el campo Buscar por nombre de esta ficha para limitar más aún la búsqueda.



8. Amplíe el nodo el nodo SQLData. Verá que hay varios servicios disponibles.
9. Seleccione el nodo del servicio Headline News. Ahora, a la derecha aparece la ficha Detalles de servicio, con información general sobre el servicio web, como el nombre de la empresa, la descripción y la clave del servicio.
10. Amplíe el nodo Headline News y seleccione el nodo siguiente, SOAP Binding. La ficha Detalles de enlace, a la derecha, explica la forma de crear un enlace con el servicio, y muestra el punto de acceso al servicio y su tipo de URL. Algunos de estos campos pueden aparecer en blanco si no son necesarios.
11. Amplíe el nodo SOAP Binding y seleccione el siguiente, <Sin descripción>. A la derecha aparece la ficha Detalles de instancia de TModel, con detalles sobre el tModel, como la descripción del servicio, el nombre y la clave del tModel, la descripción de la instancia, la descripción general del documento con su URL y los parámetros de la instancia. Muchos de estos campos están en blanco porque son optativos.
12. Desplácese hasta el último nodo SQLData bajo el nodo <Sin descripción>, que es el nodo Headline News. Se trata del nodo TModel. Examine los detalles del TModel (descripción, clave del TModel, descripción y URL del documento general) que aparecen a la derecha, en la pestaña Descripción general. Parte de la información aparece también en la ficha Detalles de instancia del TModel. En el campo Descripción se explica lo que hace el servicio web. El campo URL del documento general contiene la dirección URL de un documento WSDL que describe la especificación del servicio web en Web Services Description Language (Lenguaje de descripción de servicios web). El documento WSDL, escrito en XML, describe la interfaz del servicio y los mensajes XML que entran y salen del servicio web. El examen del documento WSDL permite escribir programas que llaman al servicio web e interaccionan con él.

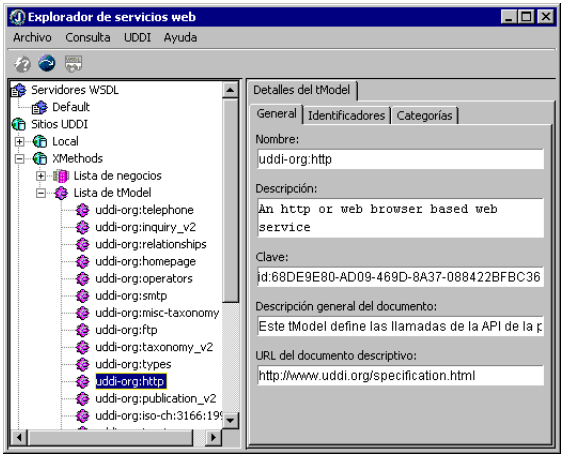
En este punto, podría utilizar el botón Importar servicio web, situado en la barra de herramientas, para abrir el asistente Importar un servicio web. Este asistente genera clases Java a partir del archivo WSDL. Consulte el "Paso 4: generación de clases Java" en la página 19-8. Este servicio en concreto necesita dos parámetros:

a. SRLFile: El nombre del archivo de lenguaje para la solicitud de servicio, utilice NEWS.SRI para las noticias de cabecera
b. RequestName: Como fuente de noticias, utilice, por ahora, yahoo o newslinx; se añadirán más.

Paso 2: Examen de tModels

En este paso buscará todos los tModels cuyo nombre empieza por uddi-org. Los TModels son referencias a una especificación técnica de un servicio. UDDI utiliza los tModels para apuntar a la especificación en lugar de contener todos los datos dentro del registro. Cada tModel tiene un identificador irrepetible, llamado Universal Unique Identifier (Identificador exclusivo universal, UUID). UDDI asigna estos identificadores la primera vez que se publica un servicio web. Los programadores que desarrollan aplicaciones que consumen servicios web utilizan este UUID para acceder a la información sobre la especificación del servicio web.

1. Amplíe el nodo Microsoft del árbol Sitios UDDI y elija en el árbol el nodo Lista de tModels, para mostrar a la derecha la ficha Consulta por tModel.
2. Escriba `uddi-org` en el campo Buscar por nombre para buscar todos los tModels de este tipo que ofrece Microsoft.
3. Pulse el botón Ejecutar para ejecutar la consulta.
4. Amplíe el nodo Lista de tModels y observe las distintas categorías de uddi-org, como http, types, publication, ftp, fax, taxonomy, etc.
5. Seleccione el segundo nodo uddi-org:http, no el primero. Aquí se indica que este tModel describe un servicio web al que se llama mediante un navegador web y el protocolo http.



6. Pulse la ficha Categorías para ver las correspondientes al tModel. La clave de este tipo de tModel es el transporte.

7 Seleccione el nodo uddi-org:publication. Lea el campo Descripción general del documento, en la ficha General, para ver una descripción del tModel: "Este tModel define las llamadas de la API de la publicación para interaccionar con el registro UDDI." Observe también que la dirección URL del documento señala a un documento WSDL. Este documento WSDL define las llamadas de la API de la publicación para interaccionar con el registro de negocios UDDI. Si copia la dirección URL del documento y la pega en un navegador que acepte XML, como Internet Explorer, podrá ver las llamadas definidas.

8 Pulse la pestaña Compilador. Este tipo de tModel tiene cuatro valores clave y apunta a cuatro especificaciones: specification, xmlSpec, soapSpec y wsdlSpec.

Paso 3: Búsqueda de empresas de publicación de software en el UDDI de Microsoft

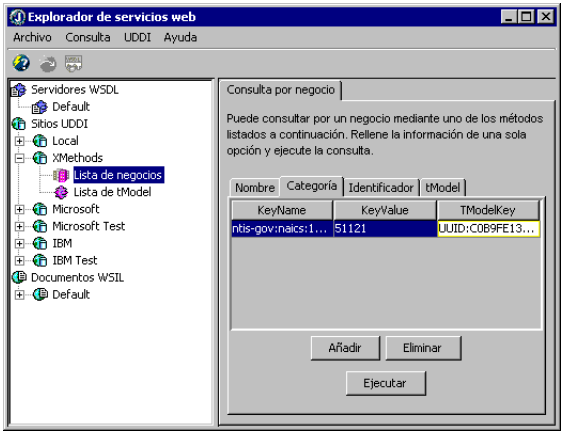
A continuación se va a utilizar una búsqueda de categorías para localizar todas las empresas de publicación de software del registro UDDI de Microsoft.

1 Seleccione el nodo Lista de negocios Microsoft y se abrirá la ficha Consulta por empresa.

2 Abra la pestaña Categoría.

3 Seleccione `ntis-gov:naics:1997` en la lista desplegable KeyName (nombre de la clave). El North American Industry Classification System (Sistema de clasificación industrial norteamericano, NAICS) es un sistema de clasificación industrial que ha sustituido al sistema estadounidense Standard Industrial Classification (Clasificación industrial estándar, SIC). El campo TModelKey (clave del tModel) se completa automáticamente.

4 Escriba `51121` en el campo KeyValue. Éste es el código NAIC de las empresas publicadoras de software.



5 Pulse *Intro* para cambiar el valor.

6 Pulse el botón Ejecutar para ejecutar la consulta.

7 Amplíe el nodo Lista de negocios para ver el resultado de la consulta.

8 Elija un negocio de la lista y abra la pestaña Categorías de la ficha Detalles del negocio. Una de las categorías es “software publisher”, con un KeyValue (valor de la clave) de 51121. Como puede ver, es frecuente que las empresas tengan varias categorías.

9 Seleccione el nodo IBM en la lista de negocios Microsoft y pulse sobre la pestaña Identificadores en la ficha Detalles del negocio. Cada compañía también tiene sus identificadores. Aquí puede ver que IBM tiene un identificador D-U-N-S, que es un sistema identificador numérico de Dun & Bradstreet que utiliza un código de nueve dígitos. El nombre del identificador (keyName) es D-U-N-S y su valor (keyValue) es el código de nueve dígitos: 00-136-8083. Asimismo, cada identificador tiene asignado un UUID (identificador exclusivo universal), tModelKey.

10 Amplíe el nodo de IBM Corporation y elija Buy From IBM.

11 Abra la pestaña Categorías de la ficha Detalles del servicio para ver a cuál pertenece IBM. Dado que se trata de una gran empresa con una amplia gama de productos, está registrada en varias categorías o KeyNames (nombres clave). Observe que muchos de los nombres clave empiezan por NAICS (North American Industry Classification System, Sistema de clasificación de industrias de Norteamérica) y por UNSPSC (Universal Standard Products and Services Classification, Clasificación universal estándar de productos y servicios), que son dos sistemas de clasificación distintos. Cada KeyName tiene una tModelKey y un KeyValue.

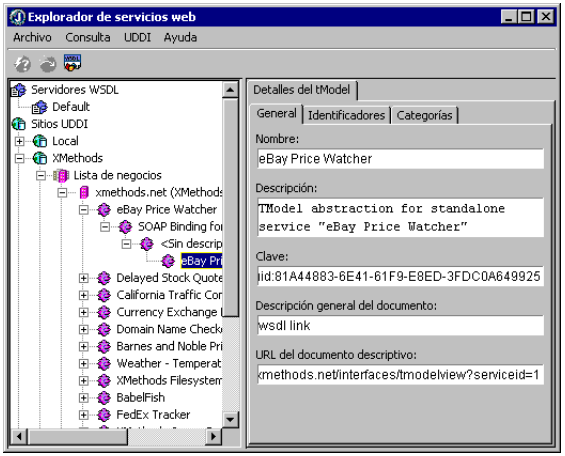
Paso 4: generación de clases Java

JBuilder proporciona el asistente Importar un servicio web, que permite generar rápidamente archivos Java a partir de un documento WSDL. Un documento WSDL (Web Services Description Language, Lenguaje de descripción de servicios web) describe la interfaz del servicio y los mensajes XML que entran y salen del servicio Web. Los TModels pueden señalar a una descripción del WSDL, escrita en XML.

En este paso se crea un proyecto, se busca un servicio web con el Explorador de servicios web y se utiliza el asistente Importar un servicio web para generar clases Java a partir del documento WSDL del servicio web.

1. Vuelva al IDE de JBuilder y seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos y crear un proyecto.
2. Escriba `WsdlToJava` en el campo Nombre y pulse Finalizar.
3. Vuelva al Explorador de servicios web y expanda el nodo XMethods en el árbol.
4. Seleccione el nodo Lista de negocios y escriba `xmethods` en el campo Buscar por nombre de la pestaña Nombre de la ficha Consulta por negocio. Pulse el botón Ejecutar o la tecla *Intro* para ejecutar la consulta.
5. Expanda el nodo XMethods, situado bajo el nodo Lista de negocios.
6. Seleccione el servicio web, eBay Price Watcher.
7. Vaya hasta el final del nodo eBay Price Watcher y seleccione el último subnodo, Detalles del tModel. Si el nodo tModel está seleccionado, el campo WSDL del asistente Importar un servicio web se completa automáticamente con el documento WSDL correspondiente al servicio. Para poder usar el asistente Importar un servicio web es necesario que el nodo tModel se encuentre seleccionado, ya que contiene el

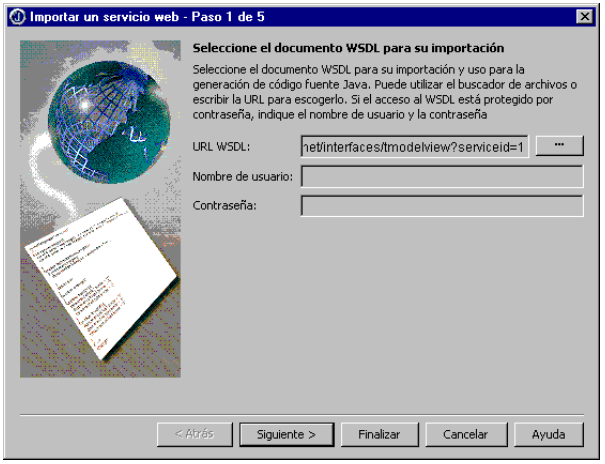
documento WSDL que aparece en el campo URL del documento general.



8 Para abrir el asistente Importar un servicio web, pulse el botón Importar servicio web, situado en la barra de herramientas, o seleccione Archivo | Importar servicio web.

9 Seleccione Axis como kit de herramientas y pulse Aceptar. Si selecciona WebLogic como kit de herramientas, los pasos a seguir serán diferentes. Consulte el Capítulo 8, "El kit de herramientas WebLogic". Observe que el documento WSDL correcto se introduce automáticamente en el campo URL de WSDL. Este WSDL no requiere nombre de usuario o contraseña, por lo que debe dejar los campos en blanco.

10 Pulse Siguiente para pasar al Paso 2.



11 Elija las opciones de código de servidor que desee y pulse Siguiente.

12 Acepte el nombre por defecto del paquete, que se basa en targetNameSpace (espacio de nombres de destino), y pulse Finalizar.

   Las clases Java se generan en un nodo de paquete, en el panel del proyecto. El asistente Importar un servicio web genera clases Java a partir del documento WSDL. Genera un nombre de paquete basado en el espacio de nombres del archivo WSDL o en el indicado por el usuario. También añade al proyecto las bibliotecas necesarias. Si desea más información sobre este asistente, consulte la documentación de JBuilder sobre el kit de herramientas que está utilizando, el Capítulo 7, "El kit de herramientas Axis de Apache", o el Capítulo 8, "El kit de herramientas WebLogic".

13 Vuelva al Explorador de servicios web y resalte la dirección URL del documento WSDL en el campo URL del documento general.

14 Copie la dirección URL y péguela en un navegador que acepte XML, como Internet Explorer, para abrir el documento WSDL. Aquí se muestran los métodos que se utilizan para acceder al servicio web. Los métodos se enumeran en el elemento `<operation>` de XML. Por ejemplo, uno de los métodos, `getCurrent price`, se representa en el documento WSDL como: `<operation name="getCurrentPrice">`.

Consulte

- Capítulo 5, "Utilización de WSDL"
- Capítulo 14, "Tutorial: Creación de un servicio web desde un documento WSDL"

¡Enhorabuena! Ha finalizado este tutorial. El Explorador de servicios web también incluye las siguientes funciones:

- Publicación de servicios web
- Búsqueda de servicios web en servidores Axis, una función del kit de herramientas Axis
- Publicación de servicios web desde servidores Axis, una función del kit de herramientas Axis
- Búsqueda de servicios web con documentos WSIL
- Seguimiento de mensajes UDDI

Para aprender más sobre el Explorador de servicios web y sus características, consulte el Capítulo 11, "Búsqueda y publicación de servicios web",, o bien, seleccione Ayuda | Indice de materias en el Explorador de servicios web.

Índice

A

accessPoint, UDDI 11-4
archivos
  de clase
    exportar como servicio web, Axis 7-2
    exportar como servicio web, WebLogic 8-2
  deploy.wsdd 7-6, 7-15, 8-6
    volver a generar al generar 3-7
  server-config.wsdd 3-7
  WLDU 8-15
  WSDD 7-6, 7-15, 8-6
    volver a generar al generar 3-7
asistentes
  Exportar
    como servicio web Axis 7-2
    como servicio web WebLogic 8-2
  Importar
    un servicio web con Axis 7-6
    un servicio web con WebLogic 8-6
  Importar un servicio web 11-29
  para configuración de servicios web 3-1
Axis de Apache 7-1

B

Borland
  asistencia
    a desarrolladores 1-6
    técnica 1-6
  contacto 1-6
  correo electrónico 1-8
  grupos de noticias 1-7
  informar sobre errores 1-8
  recursos en línea 1-7
  World Wide Web 1-7
businessKey, UDDI 11-4

C

category, UDDI 11-4, 11-10
clases Java
  exportar
    como servicio web, Axis 7-2
    como servicio web, WebLogic 8-2
  generar
    a partir de un documento WSDL, Axis 7-6
    a partir de un documento WSDL, WebLogic 8-6
configuraciones de ejecución
  servidor de servicios web 3-6
convenciones de la documentación 1-4, 1-5

D

detalles de tModelInstance 11-5
distribuir servicios web
  a un servidor
    de servicios web, Axis 7-15
    WebLogic 8-1
  archivos
    WLDU, WebLogic 8-15
    WSDD, Axis 7-15
  opción Volver a generar distribución 3-7
  probar servicios web, WebLogic 8-9
D-U-N-S (Dun & Bradstreet's Data Universal Numbering System) ID 11-12

E

EAR
  importar servicio web, WebLogic 8-6
EJB
  exportar
    como servicio web 6-1
    como servicio web, Axis 7-10
    como servicio web, WebLogic 8-12
  importar desde un documento WSDL, Axis 7-12
Elemento Envelope, SOAP 3-1
Explorador de servicios web
  añadir nodos 11-6
  buscar
    con documentos WSIL 11-20
    en servidores Axis 11-17
    negocios 11-8
    por categoría 11-10
    por identificador 11-12
    por nombre de negocio 11-8
    servicios 11-1, 11-8, 11-13
    tModels 11-14
  eliminar nodos 11-6
  fichas de detalles 11-15
  importar servicios como EJB, Axis 7-13
  JSEE 11-1
  publicar
    desde servidores Axis 11-26
    servicios web 11-24
    tModels 11-26
  registrar servicios 11-1
  resultados de las consultas UDDI 11-15
  servidores Axis 11-5

tutorial 19-1
WSIL 11-6

F

fichas
Detalles 11-16
de enlace 11-16
de la instancia de tModel 11-16
de tModel 11-16
del negocio 11-16
del servicio 11-16
Generar de Propiedades de proyecto
pestaña Servicios web 3-7
fuentes
Convenciones empleadas en la documentación de JBuilder 1-4

G

grupos de noticias
Borland 1-7
public 1-8

I

identifier, UDDI 11-4, 11-12

J

JSEE
Explorador de servicios web 11-1

K

keyName, UDDI 11-4
kit de herramientas Axis, servicios web 7-1
distribuir servicios web 7-15, 7-16
exportar
clase Java 7-2
EJB como servicios web 7-10, 7-16
importar
EJB como servicio web 7-12
servicio como EJB con el Explorador de servicios web 7-13
un WSDL 7-6
opción Copiar consola de administración a la webapp 3-3
kit de herramientas SOAP 2 de Apache, servicios web 3-3
exportar clase como servicio web 9-2
importar un WSDL 9-2
modificar código 9-1
kit de herramientas WebLogic, servicios web 8-1
configurar nombres de servicios por defecto 8-18
distribuir servicios web 8-15
exportar
clase como servicio web 8-2
EJB como servicio web 8-12
varias clases 8-6
importar servicio web 8-6
probar servicios web distribuidos 8-9
kits de herramientas de servicios web 3-1
seleccionar 3-3

M

Monitor de mensajes UDDI 11-28
Monitor JDBC 4-1

O

opción Volver a generar distribución 3-7

P

plataformas
convenciones 1-5
proyectos
configurar para servicios web 3-1
publicar
negocios 11-24
servicios
desde servidores Axis 11-26
web 11-24
web albergados por Axis 11-18
tModels 11-26

R

registros UDDI 11-3
buscar con el Explorador de servicios web 11-8
publicar
desde servidores Axis 11-26
servicios 11-24
tModels 11-26

S

serviceKey, UDDI 11-5
servicios web
arquitectura 2-2
buscar 11-1, 11-14
categorías de negocios 11-10
configuración del servidor de aplicaciones 10-1
descripción general 2-1

descriptor
de distribución, Axis 7-6, 7-15
de distribución, WebLogic 8-6, 8-15
distribuir
EJB, Axis 7-16
EJB, WebLogic 8-15
distribuir a
Axis 7-15
WebLogic 8-1
Explorador de servicios web 11-1
exportar
clase Java como, Axis 7-2
clase Java como, WebLogic 8-2
EJB como 6-1
EJB, Axis 7-10
EJB, WebLogic 8-12
generar a partir de
un documento WSDL, Axis 7-6
un documento WSDL, WebLogic 8-6
un EAR, WebLogic 8-6
identificadores de negocios 11-12
importar servicio como EJB, Axis 7-12
J2EE 6-1
kits de herramientas 3-1
Axis de Apache 7-1
SOAP 2 de Apache 9-1
probar servicios distribuidos, WebLogic 8-9
publicar 11-1, 11-24
publicar tModels 11-26
registrar 11-1
servidores
de aplicaciones admitidos 2-8
SOAP (Simple Object Access Protocol, Protocolo de acceso a objetos sencillos) 2-3
tecnologías 2-3
UDDI (Descripción, detección e integración universales) 2-5
Web Services Description Language (Lenguaje de descripción de servicios web, WSDL) 2-4, 5-1
WSIL (Lenguaje de inspección de servicios web) 11-6
WSIL (Web Services Inspection Language, Lenguaje de inspección de servicios web) 2-6
servidores
Axis
activar acceso remoto a 11-19
importar servicio en el Explorador de servicios web 11-18
mostrar servicios web 7-13, 11-5
publicar servicios 11-18, 11-26
de aplicaciones
configurar servicios web 10-1
servicios web admitidos 2-8
de servicios web
configuración de ejecución 3-6
iniciar 3-6
relaciones con webapp 3-1
servicios web admitidos 2-8
Simple Object Access Protocol (Protocolo de acceso a objetos sencillos, SOAP) 2-3, 3-1
sitios de operadores, UDDI 11-4
SOAP 2-3, 3-1
archivo server-config.wsdd 3-7
monitorizar mensajes UDDI 11-28
nodo del kit de herramientas 3-1
opción generar de Propiedades del proyecto 3-7
seguir mensajes con el monitor TCP 4-1

T

Thomas Register 11-12
tModelKey 11-5
tModels, UDDI 11-5
buscar 11-14
publicar 11-26
tutoriales
creación de un servicio web
a partir de un documento WSDL, Axis 14-1
a partir de una aplicación EJB mediante el servidor WebLogic Server 18-1
a partir de una aplicación EJB, Axis 15-1
creación de un servicio web sencillo
con Axis 13-1
con WebLogic 17-1
importar servicios web como aplicaciones EJB, Axis 16-1

U

UDDI 2-5
descripción general 11-3
JSEE 11-1
seguir mensajes SOAP UDDI 11-28
sitios de operadores 11-3, 11-4
términos y definiciones 11-4
URL del documento explicativo 11-5
Usenet, grupos de noticias 1-8

W

Web Services Description Language (Lenguaje de descripción de servicios web, WSDL) 2-4
WebApp
hospedar servicios web 3-1
WebLogic Web Services Home Page 8-9

WSDL 5-1
ejemplos 5-3
generar a partir de una clase Java, Axis 7-2
importar servicio web, WebLogic 8-6
importar, Axis 7-6
términos definidos 5-2
WSIL (Lenguaje de inspección de servicios web) 2-6, 11-6