



Guía del desarrollador de XML



VERSIÓN 8

Borland®
JBuilder®

Borland Software Corporation
100 Enterprise Way, Scotts Valley, CA 95066-3249
www.borland.com

En el archivo `deploy.html` ubicado en el directorio raíz del producto JBuilder encontrará una lista completa de archivos que se pueden distribuir de acuerdo con la licencia de JBuilder y la limitación de responsabilidad.

Borland Software Corporation puede tener patentes concedidas o en tramitación sobre los temas tratados en este documento. Dirijase al CD del producto o al cuadro de diálogo Acerca de para la lista de patentes. La modificación de este documento no le otorga derechos sobre las licencias de estas patentes.

COPYRIGHT © 1997-2003 Borland Software Corporation. Reservados todos los derechos. Todos los nombres de productos y marcas de Borland son marcas comerciales o registradas de Borland Software Corporation en Estados Unidos y otros países. Las otras marcas pertenecen de sus respectivos propietarios.

Si desea más información acerca de las condiciones de contrato de terceras partes y acerca de la limitación de responsabilidades, consulte las notas de esta versión en su CD de instalación de JBuilder.

Impreso en EE.UU.

JBE0080WW21002xml 5E5R1002

0203040506-9 8 7 6 5 4 3 2 1

PDF

Índice de materias

Capítulo 1

Introducción

1-1

Convenciones de la documentación	1-3
Asistencia a los desarrolladores	1-5
Cómo ponerse en contacto con el servicio técnico de Borland	1-5
Recursos en línea	1-5
World Wide Web	1-6
Grupos de noticias de Borland	1-6
Usenet, grupos de noticias	1-6
Información sobre errores	1-7

Capítulo 2

Características XML de JBuilder

2-1

Funciones XML en la Plataforma Java 2.	2-2
Creación de documentos XML	2-2
Creación manual de documentos XML	2-2
Creación de documentos XML mediante asistentes	2-4
Creación de un documento XML a partir de una DTD	2-4
Creación de una DTD a partir de un documento XML	2-6
Presentación de documentos XML.	2-7
El visor XML	2-7
Configuración de opciones XML	2-10
Opciones generales	2-10
Opciones de seguimiento	2-11
Validación de documentos XML	2-11
Presentación de documentos XML.	2-14
Presentación de XML mediante Cocoon.	2-14
Creación de una aplicación web Cocoon.	2-15
Ejecución de Cocoon	2-19
Transformación de documentos XML	2-20
Aplicación de hojas de estilo internas	2-20
Aplicación de hojas de estilo externas	2-21
Configuración de opciones de seguimiento	2-23
Manipulación de XML mediante programas	2-24
Creación de un gestor de SAX	2-25
Manipulación de XML mediante asociación de datos	2-30
El marco de codificación	2-30
BorlandXML.	2-31
Castor	2-33

Interacción con datos empresariales en bases de datos	2-35
--	------

Capítulo 3

Componentes de base de datos

XML de JBuilder

3-1

Utilización de los componentes basados en modelos	3-2
XML-DBMS	3-2
JBuilder y XML-DBMS	3-4
Creación de un documento de correspondencia y un archivo de script SQL	3-4
Definición de propiedades para los componentes basados en modelos.	3-8
Configuración de propiedades con el personalizador	3-9
Configuración de propiedades con el Inspector	3-12
Utilización de los componentes basados en plantillas	3-13
Configuración de propiedades para las plantillas de bean	3-13
Configuración de propiedades con el personalizador	3-13
Configuración de propiedades con el Inspector	3-21
Configuración de propiedades con un documento de consulta XML	3-21

Capítulo 4

Tutorial: Creación y validación de documentos XML

4-1

Paso 1: Creación de un documento XML.	4-2
Creación de un documento XML manualmente.	4-2
Creación de un documento XML mediante el Asistente para pasar de DTD a XML	4-4
Paso 2: Validación de los documentos XML	4-6
Paso 3: Presentación del documento XML	4-8

Capítulo 5

Tutorial: Transformación de documentos XML

5-1

Paso 1: Activación del visor XML	5-2
Paso 2: Asociación de hojas de estilo al documento	5-3
Paso 3: Transformación del documento mediante hojas de estilo	5-3
Paso 4: Configuración de opciones de seguimiento.	5-4

Capítulo 6

Tutorial: Creación de un gestor de SAX para analizar documentos XML 6-1

Paso 1: El asistente para la gestión de SAX . . .	6-2
Paso 2: Modificación de analizadores SAX . . .	6-5
Paso 3: Ejecución del programa	6-7
Paso 4: Adición de atributos	6-9
Código fuente de MySaxParser.java	6-10

Capítulo 7

Tutorial: Asociación de datos DTD con BorlandXML 7-1

Paso 1: Generación de clases Java a partir de una DTD	7-2
Paso 2: Desmontaje de los datos	7-5
Paso 3: Adición de un registro de empleado . .	7-6
Paso 4: Modificación de un registro de empleado	7-7
Paso 5: Ejecución de la aplicación completa . .	7-8

Capítulo 8

Tutorial: Asociación de datos de esquema con Castor 8-1

Paso 1: Generación de clases Java a partir de un esquema	8-2
Paso 2: Desmontaje de los datos	8-5
Paso 3: Adición de un registro de empleado . .	8-6
Paso 4: Modificación de los datos del nuevo empleado	8-7
Paso 5: Ejecución de la aplicación completa . .	8-7

Capítulo 9

Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos 9-1

Paso 1: Procedimientos iniciales	9-2
Paso 2: Creación de los archivos de asignación y script SQL.	9-4

Introducción de información sobre la conexión JDBC	9-4
Comprobación de la conexión	9-5
Especificación de los nombres de archivo . .	9-6
Paso 3: Creación de tablas de base de datos . .	9-7
Paso 4: Utilización de la aplicación de prueba de ejemplo	9-9
Uso del personalizador de XMLDBMSTable .	9-9
Selección y prueba de una conexión JDBC	9-9
Transferencia de datos de XML a la base de datos	9-10
Transferencia de datos desde la base de datos a XML.	9-12
Uso del personalizador de XMLDBMSQuery	9-14
Selección y prueba de una conexión JDBC	9-15
Transferencia de datos con una sentencia SQL	9-15
El archivo de mapa	9-16

Capítulo 10

Tutorial: Transferir datos con componentes XML de bases de datos basados en plantillas 10-1

Paso 1: Procedimientos iniciales.	10-2
Paso 2: Utilización de la aplicación de prueba de ejemplo	10-2
Paso 3: Utilización del personalizador de XTable	10-3
Introducción de información sobre la conexión JDBC	10-3
Transferencia de datos desde la base de datos a XML	10-4
Paso 4: Utilización del personalizador de XQuery	10-6
Selección de una conexión JDBC.	10-7
Transferencia de datos con una sentencia SQL	10-7

Índice

I-1

Figuras

2.1	DTD con definiciones ATTLIST	2-5	2.10	Errores de validación de XML al aplicar esquemas	2-14
2.2	XML creado por el asistente	2-6	2.11	Código fuente XML para index.xml . . .	2-18
2.3	Vista de XML con la hoja de estilo por defecto	2-8	2.12	Código fuente de hoja de estilo para index.xml	2-18
2.4	Vista de XML sin hojas de estilo	2-8	2.13	Vista Web de index.xml	2-19
2.5	Código fuente de la hoja de estilo en cascada (CSS)	2-9	2.14	Vista del código fuente Web de index.xml	2-20
2.6	Documento XML con instrucciones de hoja de estilo	2-9	2.15	Barra de herramientas Vista transformado . .	2-21
2.7	Documento XML al que se ha aplicado una CSS	2-9	2.16	Vista transformado a la que se ha aplicado una hoja de estilo externa	2-22
2.8	Carpeta de errores del panel de estructura . .	2-12	2.17	Vista transformado sin hoja de estilo. . .	2-23
2.9	Errores de validación de XML al aplicar una DTD	2-13	2.18	Vista transformado con la hoja de estilo por defecto de vista en árbol	2-23
			2.19	Marco de codificación	2-31



Tutoriales

Creación y validación de documentos XML . . .	4-1	Transferencia de datos con componentes XML de	
Transformación de documentos XML	5-1	bases de datos basados en modelos	9-1
Creación de un gestor de SAX para analizar		Transferencia de datos con componentes XML de	
documentos XML	6-1	bases de datos basados en plantillas	10-1
Asociación de datos DTD con BorlandXML . . .	7-1		
Asociación de datos de esquema con Castor . .	8-1		

Introducción

La compatibilidad con XML es una característica de JBuilder SE y Enterprise.

En la *Guía del desarrollador de XML* se explica la forma de utilizar las funciones XML de JBuilder. Pertenecen a esta guía los siguientes capítulos:

- [Capítulo 2, “Características XML de JBuilder”](#)

Explica cómo utilizar las características XML de JBuilder. En este capítulo se tratan los temas siguientes:

- [“Creación de documentos XML”](#)
- [“El visor XML”](#)
- [“Validación de documentos XML”](#)
- [“Presentación de documentos XML”](#) Esta es una característica de JBuilder Enterprise.
- [“Manipulación de XML mediante programas”](#) Esta es una característica de JBuilder Enterprise.

- [Capítulo 3, “Componentes de base de datos XML de JBuilder”](#)

Explica cómo utilizar los componentes bean basados en modelos y plantillas XML para las consultas de bases de datos y transferencia de datos entre documentos XML y bases de datos. Es una característica de JBuilder Enterprise.

- Tutoriales:

Disponible en JBuilder SE y Enterprise:

- [Capítulo 4, “Tutorial: Creación y validación de documentos XML”](#)

Explica la forma de utilizar las funciones de XML de JBuilder para crear y validar un documento XML.

Disponible en JBuilder Enterprise:

- [Capítulo 5, “Tutorial: Transformación de documentos XML”](#)

Explica la forma de utilizar las funciones de XML de JBuilder para transformar documentos XML.

- [Capítulo 6, “Tutorial: Creación de un gestor de SAX para analizar documentos XML”](#)

Ayuda a crear un analizador SAX para efectuar un análisis personalizado de los documentos XML con el Asistente de gestor de SAX de JBuilder.

- [Capítulo 7, “Tutorial: Asociación de datos DTD con BorlandXML”](#)

Explica la forma de utilizar las funciones de asociación de datos XML de JBuilder con DTD y BorlandXML.

- [Capítulo 8, “Tutorial: Asociación de datos de esquema con Castor”](#)

Explica la forma de utilizar las funciones de asociación de datos XML de JBuilder con esquemas y Castor.

- [Capítulo 9, “Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos”](#)

Explica la forma de utilizar los componentes XML de base de datos basados en modelos de JBuilder para transferir datos de un archivo XML a una base de datos y viceversa. También se explica la forma de utilizar el Asistente para XML-DBMS con el fin de crear el archivo de asignación que se utiliza en la transferencia de datos y el archivo de script SQL que se utiliza para crear la base de datos.

- [Capítulo 10, “Tutorial: Transferir datos con componentes XML de bases de datos basados en plantillas”](#)

Explica la forma de utilizar los componentes XML de base de datos basados en plantillas de JBuilder para recuperar datos de una base de datos y pasarlos a un archivo XML.

Si desea información sobre las convenciones de la documentación, consulte [“Convenciones de la documentación”](#) en la página 1-3.

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	<p>El tipo monoespaciado representa lo siguiente:</p> <ul style="list-style-type: none"> • 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 <code>SET PATH</code>. • Código Java. • Tipos de datos de Java, como <code>boolean</code>, <code>int</code> y <code>long</code>. • Los 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 <code>void</code> y <code>static</code>.
Negrita	La negrita se utiliza para las herramientas java, <code>bmj</code> (Borland Make for Java), <code>bcj</code> (Borland Compiler for Java) y opciones del compilador. Por ejemplo: <code>javac</code> , <code>bmj</code> , -vía de acceso a clases .
<i>Cursiva</i>	Las palabras en cursiva indican los términos nuevos que se definen y los títulos de libros; ocasionalmente se usan para indicar énfasis.
<i>Nombres de tecla</i>	Este tipo de letra indica una tecla, como “Pulse <i>Esc</i> 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
< >	<p>Los corchetes angulares se utilizan para indicar variables en las vías de directorios, opciones de comando y ejemplos de código.</p> <p>Por ejemplo, <nombredearchivo> puede utilizarse para indicar dónde tiene que incluir el nombre de un archivo (incluida la extensión) y <usuario> indica normalmente que debe indicar su nombre de usuario.</p> <p>Cuando se sustituyen las variables en las vías de acceso a los directorios, comandos y ejemplos de código, se sustituye la variable completa, incluidos los corchetes (< >). Por ejemplo, reemplazaría <nombredearchivo> con el nombre de un archivo, como <code>employee.jds</code> y omitiría los corchetes.</p> <p>Nota: Los corchetes se utilizan en HTML, XML, JSP y otros archivos basados en etiquetas para demarcar los elementos del documento, como <color de fuente=red> y <ejb-jar>. Las siguientes convenciones describen cómo se especifican las cadenas de variables dentro del ejemplo de código que ya está utilizando corchetes como delimitadores.</p>
<i>Cursiva, serif</i>	<p>Este formato se utiliza para indicar las cadenas de variables en los ejemplos de código que ya están usando corchetes como delimitadores. Por ejemplo, <url="jdbc:borland:jbuilder\\samples\\guestbook.jds">.</p>
...	<p>En los ejemplos de código, los puntos suspensivos (...) indican código que se ha omitido en el ejemplo para ahorrar espacio y aumentar la claridad. Si están en un botón, los puntos suspensivos indican que éste conduce a un cuadro de diálogo de selección.</p>

JBuilder se puede utilizar con diversas plataformas. Consulte la siguiente tabla para ver una descripción de las convenciones de plataforma utilizadas en la documentación.

Tabla 1.2 Convenciones de las plataformas

Elementos	Significado
Vías de acceso	<p>Las vías de acceso a los directorios en la documentación se indican con una barra normal (/).</p> <p>Para la plataforma Windows se utiliza la barra invertida (\).</p>
Directorio de inicio	<p>La ubicación del directorio inicial varía según la plataforma y se indica con una variable <home>.</p> <ul style="list-style-type: none"> • 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\<nombredeusuario>.
Imágenes de pantalla	<p>Las imágenes o capturas de pantalla utilizan el aspecto Metal en diversas plataformas.</p>

Asistencia a los desarrolladores

Borland ofrece una amplia gama de opciones de asistencia técnica y recursos de información para ayudar a los desarrolladores a obtener lo máximo de sus productos Borland. Estas opciones incluyen un rango de programas de asistencia técnica de Borland, así como servicios gratuitos en Internet, donde es posible efectuar búsquedas en nuestra amplia base de información y ponerse en contacto con otros usuarios de productos Borland.

Cómo ponerse en contacto con el servicio técnico de Borland

Borland ofrece varios programas de asistencia para clientes y clientes potenciales. Se puede elegir entre varios tipos de asistencia, que van desde la asistencia para la instalación de los productos Borland hasta el asesoramiento de expertos y la asistencia pormenorizada (servicios no gratuitos).

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 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 proporcionan los medios a todos los clientes de la comunidad Borland para intercambiar sugerencias técnicas acerca de los productos, herramientas relacionadas y tecnologías Borland.

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 jgpubs@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.

Características XML de JBuilder

La compatibilidad con XML es una característica de JBuilder SE y Enterprise.

JBuilder proporciona varias funciones e incorpora diversas herramientas para la integración de Extensible Markup Language (XML). XML es un método de estructuración de la información independiente de la plataforma. Dado que XML separa el contenido del documento de la estructura, puede constituir un sistema útil de intercambio de datos. Por ejemplo, se puede utilizar para intercambiar datos entre bases de datos y programas Java. Otra ventaja de la separación de contenido y estructura es que se pueden aplicar hojas de estilo para mostrar el mismo contenido en distintos formatos, como PDF, HTML, etc.

Las funcionalidades XML disponibles varían según las ediciones de JBuilder. JBuilder SE proporciona estas funciones: creación manual de documentos XML, visualización de documentos XML en el visualizador y validación de documentos XML. Todas estas funciones se incluyen en JBuilder Enterprise.

Cuando trabaja con XML, JBuilder separa las funciones en varias capas:

- Creación y validación de documentos XML.

Son funciones de JBuilder Enterprise:

- Presentación de documentos XML.
- Manipulación del programa en documentos XML.
- Interfaz con datos empresariales en bases de datos.

Consulte

- World Wide Web Consortium (W3C), en <http://www.w3.org/>
- The XML Cover Pages en <http://www.oasis-open.org/cover/sgml-xml.html> o en <http://xml.coverpages.org/>

- XML.org en <http://xml.org/>

El directorio [extras](#) de la instalación completa de JBuilder incluye recursos XML adicionales: Xerces, Xalan, Castor, Borland XML y Coccon. También se incluyen documentación, Javadoc y ejemplos.

Funciones XML en la Plataforma Java 2

JDK 1.4 incluye la API de Java para el procesamiento de XML (JAXP). JAXP incluye las facilidades básicas para trabajar con documentos XML: DOM (Document Object Model – Modelo de objetos de documento), SAX (Simple API for XML Parsing – La API sencilla de XML), XSLT (XSL Transformation – La transformación de XSL) y diferentes niveles de acceso para analizadores. El JDK incluye Crimson como la implementación de análisis por defecto y Xalan-J como el procesador XSLT.

Para obtener más información sobre estas funciones, consulte <http://java.sun.com/j2se/1.4/docs/guide/xml/jaxp/index.html> y <http://java.sun.com/xml/jaxp/index.html>.

Creación de documentos XML

JBuilder proporciona una serie de funciones que permiten crear, modificar, ver y validar documentos XML sin salir del entorno de desarrollo. Se pueden crear documentos XML manualmente, utilizar asistentes para crearlos en JBuilder, verlos en el visualizador de XML, modificar el texto en el editor de JBuilder, buscar errores y, por último, validar documentos. Aunque los documentos DTD no son documentos XML, se incluyen en este análisis debido a la relación entre ambos.

Si desea ver un tutorial sobre la creación de documentos XML, consulte el [Capítulo 4, “Tutorial: Creación y validación de documentos XML”](#).

Creación manual de documentos XML

Es una función de JBuilder SE y Enterprise.

El editor de JBuilder proporciona soporte completo para la creación de documentos XML. Si el nombre de un archivo incluye una extensión relacionada con XML, como DTD, XSD, XSL o XML, el editor lo reconoce automáticamente como documento XML.

Para crear un documento XML en el proyecto:

- 1 Abra un proyecto.
- 2 Seleccione Proyecto | Añadir archivos/paquetes.

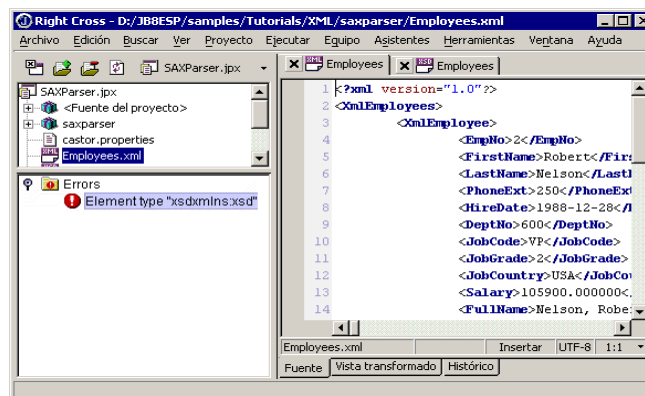
- 3 Seleccione la pestaña Explorador, desplácese hasta el directorio del proyecto y escriba un nombre de archivo con la extensión apropiada en el campo Nombre de archivo, como .dtd, xml o xsd.
- 4 Pulse Aceptar.
- 5 Pulse Aceptar de nuevo cuando se le pregunte si desea crear el archivo. El archivo se añade al proyecto y aparece en el panel de proyecto con el icono de XML.
- 6 Abra el archivo en el editor y escriba el texto. Observe cómo el editor resalta la sintaxis con el fin de diferenciar elementos y atributos. Por defecto, los elementos son azules y los atributos son rojos.
- 7 Guarde el proyecto.

El editor incluye dos características que ayudan a trabajar con documentos XML:

- Resaltado de sintaxis
- Mensajes de error

El editor utiliza el resaltado de la sintaxis para mostrar los elementos y atributos de XML en diferentes colores, de modo que se puedan diferenciar visualmente. Los elementos XML son azules y los atributos son rojos. Si desea personalizar los colores del editor, seleccione Herramientas | Opciones del editor | Color. Si desea cambiar el color de los elementos, seleccione Etiqueta HTML en la lista Elemento de pantalla y elija el color deseado. Si desea cambiar el color de los atributos, seleccione Atributo HTML en la lista Elemento de pantalla y elija el color deseado.

Los mensajes de error se muestran dinámicamente en la carpeta Errores del panel de estructura según se escribe. Pulse sobre un mensaje de error en el panel de estructura para resaltarlo en el editor. Haga doble clic en el mensaje de error para cambiar el foco a la línea de código, en el editor. Es posible que el origen del error no se encuentre en la línea de código a la que señala el mensaje.



Creación de documentos XML mediante asistentes

Son funciones de
JBuilder Enterprise.

JBuilder proporciona asistentes para la creación de documentos XML dentro del IDE:

- Creación de un documento XML a partir de una DTD.
- Creación de una DTD a partir de documentos XML.

Estos asistentes se pueden abrir desde el menú contextual del panel de proyectos y desde la ficha XML de la galería de objetos (Archivo | Nuevo).

Sugerencia

También se pueden crear documentos XML vacíos; después, el editor reconocerá el tipo de archivo y resaltará la sintaxis. Consulte [“Creación manual de documentos XML” en la página 2-2](#).

Creación de un documento XML a partir de una DTD

El asistente DTD a XML constituye una forma rápida de crear documentos XML a partir de una *Definición de tipo de documento* (DTD). Las definiciones DTD son conjuntos de reglas que describen la estructura de los documentos XML. Los analizadores de validación utilizan las DTD para validar las marcas XML. El asistente para pasar de DTD a XML genera una plantilla XML a partir de la DTD, con sustitutos `pcdata`. Sustituya estos `pcdata` por el contenido definitivo.

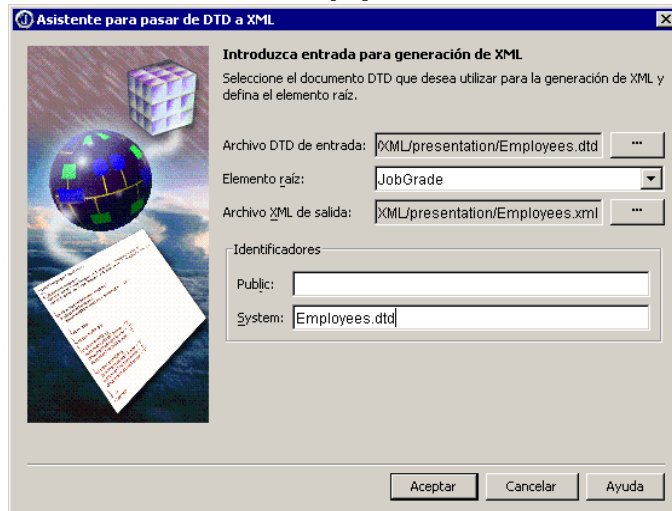
Para crear un documento XML a partir de una DTD:

- 1 Haga clic con el botón derecho del ratón en el archivo DTD, en el panel de proyecto, y elija Generar XML. De esta forma, se introduce automáticamente el nombre de archivo DTD en el campo correspondiente del asistente. Puede encontrar también estos asistentes en la ficha XML de la galería de objetos (Archivo | Nuevo).
- 2 Seleccione el Elemento raíz en la lista desplegable. El *elemento raíz* es el primero del documento, y contiene los demás elementos.
- 3 Acepte el nombre de archivo por defecto del campo Archivo XML de salida o pulse el botón de puntos suspensivos y escriba un nombre distinto.
- 4 **Optativo:** Introduzca los identificadores correspondientes a la declaración DOCTYPE.
 - Public: escriba el URI de la biblioteca de normas especificada.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML3.2 Final//EN">
```

- System: escriba el nombre del archivo DTD.

```
<!DOCTYPE root SYSTEM "Employees.dtd">
```



- 5 Pulse Aceptar para cerrar el asistente. El documento XML se añade al proyecto y aparece en el panel.

Este asistente también gestiona los *atributos*, que se utilizan para definir con más detalle los elementos, y convierte las definiciones **ATTLIST** de la DTD en atributos, en el documento XML. La palabra clave **ATTLIST** de la DTD se utiliza para enumerar los atributos de los elementos. Esta lista incluye, además de los atributos, sus nombres, sus valores y sus opciones por defecto.

Figura 2.1 DTD con definiciones ATTLIST

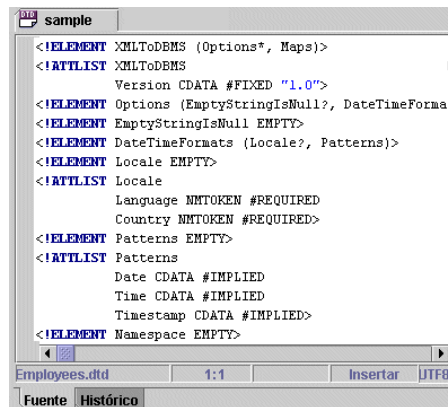
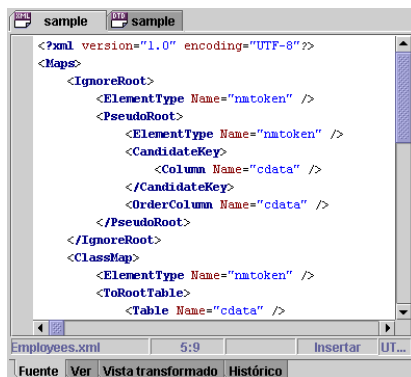


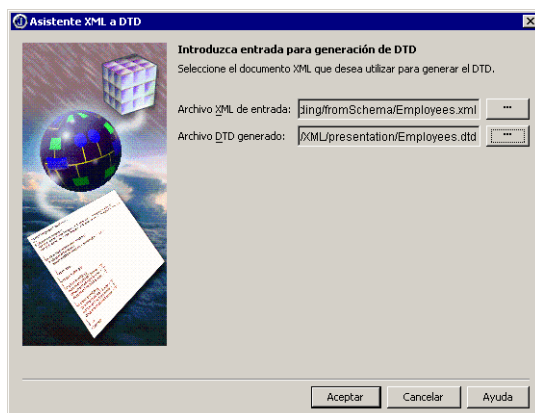
Figura 2.2 XML creado por el asistente

Creación de una DTD a partir de un documento XML

El Asistente XML a DTD constituye una forma rápida de crear una DTD a partir de un documento XML. La DTD, aunque no es un documento XML, describe el documento XML y se utiliza en el analizador de validación para validar las marcas XML.

Para crear una DTD a partir de un documento XML:

- 1 Haga clic con el botón derecho del ratón en el archivo XML; en el panel de proyecto, elija Generar DTD y se abrirá el asistente. De esta forma, se introduce automáticamente el nombre de archivo XML en el campo correspondiente del asistente. Puede encontrar también estos asistentes en la ficha XML de la galería de objetos (Archivo | Nuevo). Otra forma de abrir este asistente consiste en abrir el archivo XML en el editor, pulsar con el botón secundario del ratón en la ficha del nombre de archivo y elegir Generar DTD.
- 2 Acepte el nombre de archivo por defecto del campo Archivo DTD generado o pulse el botón de puntos suspensivos y escriba un nombre distinto.



- 3 Pulse Aceptar para cerrar el asistente. La DTD se añade al proyecto y aparece en el panel.

- Importante** Antes de validar el documento XML contra la DTD, se deberá actualizar el documento XML declarando la DTD con su nombre. Por ejemplo, `<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">`.
- Importante** Si se incluyen atributos en el documento XML, el Asistente XML a DTD genera definiciones `ATTLIST` para ellos en la DTD. Si desea ver ejemplos de atributos, consulte el Asistente para pasar DTD a XML, descrito en el [“Creación de un documento XML a partir de una DTD” en la página 2-4](#).

Presentación de documentos XML

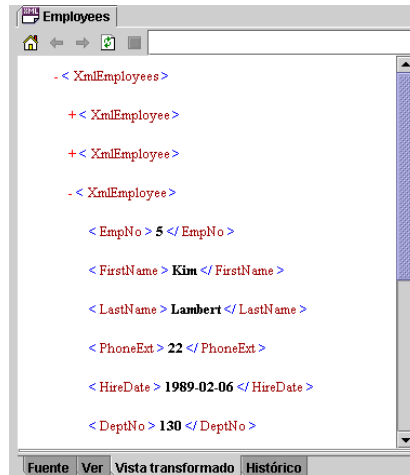
Es una función de
JBuilder SE y Enterprise.

JBuilder proporciona un visor de XML que presenta documentos XML sin salir del entorno de desarrollo. El código XML se puede ver con otra hoja de estilo definida por el usuario, con la predeterminada de JBuilder o sin hoja de estilo. El visor XML de JBuilder, que acepta JavaScript, muestra la hoja de estilo de JBuilder en una vista en árbol plegable. Si desea obtener información sobre configuración de opciones XML en el IDE de JBuilder, consulte [“Configuración de opciones XML” en la página 2-10](#).

El visor XML

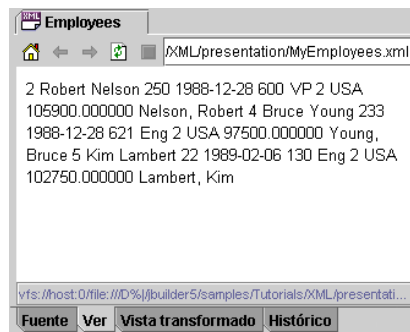
Para ver un documento XML en JBuilder, ábralo y pulse la pestaña Ver del panel de contenido. Si esta pestaña no está disponible, es necesario activarla en la ficha XML del cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE).

Si no hay una CSS disponible, JBuilder aplica una hoja de estilo XSLT por defecto que muestra el documento en una vista en árbol plegable. La ficha Ver pasa por alto las hojas de estilo XSL. Si desea obtener más información acerca de la aplicación de hojas de estilo, consulte [“Transformación de documentos XML” en la página 2-20](#).

Figura 2.3 Vista de XML con la hoja de estilo por defecto

Nota Para ampliar o contraer la vista en árbol, pulse los iconos de signo más (+) y menos (-).

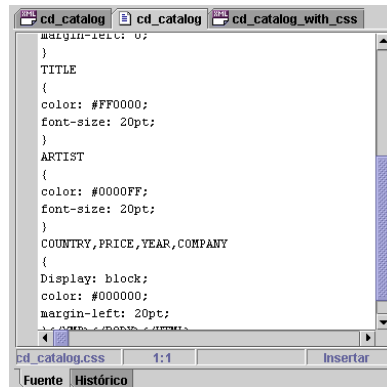
Si se desactiva la opción Aplicar hoja de estilo por defecto, se puede ver el documento XML sin los estilos aplicados. Las hojas de estilo por defecto se desactivan en la ficha XML del cuadro de diálogo Opciones del IDE.

Figura 2.4 Vista de XML sin hojas de estilo

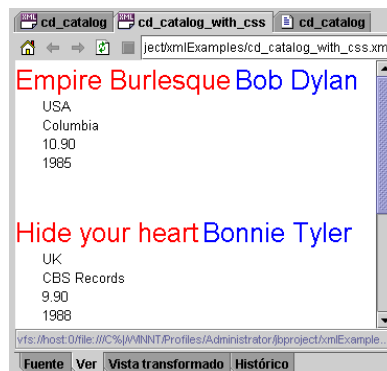
Si el documento XML hace referencia una hoja CSS, el visor XML representa el documento con esta hoja de estilo.

Por ejemplo, si se desea representar directamente un documento XML con una hoja CSS se puede crear un archivo CSS de la forma indicada y colocar una referencia en el documento XML, como sigue:

```
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
```


Figura 2.5 Código fuente de la hoja de estilo en cascada (CSS)**Figura 2.6** Documento XML con instrucciones de hoja de estilo

El resultado de aplicar la hoja de estilo al documento XML se muestra en la imagen siguiente:

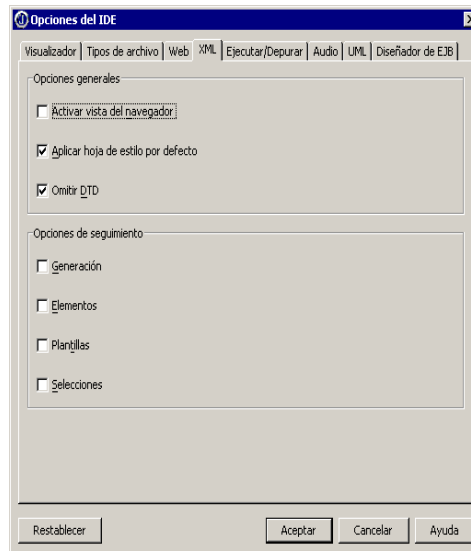
Figura 2.7 Documento XML al que se ha aplicado una CSS

Configuración de opciones XML

Es una función de JBuilder SE y Enterprise.

Las opciones XML para el IDE de JBuilder se pueden configurar en el cuadro de diálogo Opciones del IDE. En JBuilder SE y Enterprise, es posible activar y desactivar el visor XML, así como aplicar una hoja de estilos predeterminada. En JBuilder Enterprise, también se pueden configurar opciones de seguimiento para las transformaciones de un documento XML.

Para abrir el cuadro de diálogo Opciones del IDE, seleccione Herramientas | Opciones del IDE y pulse sobre la pestaña XML con el objeto de configurar las opciones generales y de seguimiento para transformaciones.



Opciones generales

Las opciones generales son las siguientes:

- **Activar vista del navegador:** activa el navegador XML de JBuilder. Si se desactiva esta opción en el panel de contenido aparece la ficha Ver.
- **Aplicar hoja de estilo por defecto:** aplica la hoja de estilo por defecto (XSL), que es una vista de árbol, al documento XML que se muestra en el visualizador XML (pestaña Ver del panel de contenido).

Nota

Es diferente del botón Hoja de estilo por defecto de la barra de herramientas Vista transformado, que aplica una vista de árbol a la transformación mostrada en la ficha Vista transformado del panel de contenido.

- **Omitir DTD:** omiten los DTD cuando analizan archivos XML. Cuando se selecciona esta opción, JBuilder no almacena el DTD y no produce

ningún informe de errores en el panel de estructura. Esto permite trabajar fuera de línea y resulta más rápido si se está trabajando en línea. Si esta opción está activada, el editor debe guardar el DTD cada vez y también enviar un informe de los errores al panel de estructura.

Opciones de seguimiento

Las opciones de seguimiento de la transformación son características de JBuilder Enterprise

Permite configurar las opciones de seguimiento de forma que, cuando se produzca una transformación, se pueda seguir el orden por el que se han aplicado los distintos elementos de la hoja de estilo. Las opciones de seguimiento de la transformación son las siguientes:

- **Generación:** muestra información después de cada suceso de generación de árbol de resultado, como `start document`, `start element`, `characters`, etc.
- **Plantillas:** muestra un suceso cuando se llama a una plantilla. Por ejemplo, `xsl:template match='stocks'`.
- **Elementos:** muestra sucesos que ocurren a medida que se ejecutan los nodos en la hoja de estilo. Por ejemplo, `xsl:value-of select='borland'`.
- **Selecciones:** muestra información después de cada suceso de selección. Por ejemplo `xsl:value-of, select='borland@StockQuote':StockQuote`.

Si desea obtener más información sobre la transformación XML, consulte [“Transformación de documentos XML” en la página 2-20](#).

Validación de documentos XML

Es una función de JBuilder SE y Enterprise.

En XML existen dos tipos de validación: *forma* y *gramática correctas*. Un documento tiene la forma correcta si su estructura y su sintaxis cumplen las normas de XML. Por ejemplo, todos los elementos deben llevar etiquetas de cierre, y se debe colocar una declaración XML al principio del documento. Además, todos los documentos XML deben tener un solo *elemento raíz*, que es el primero del documento y contiene todos los demás elementos.

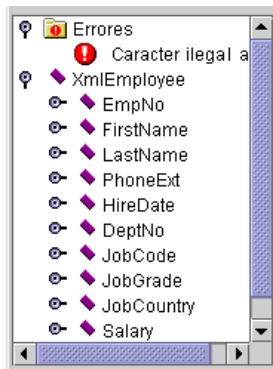
Para ser válidos, los documentos XML deben atenerse a las normas más estrictas de las definiciones de tipo de documentos (DTD) o de esquema (XSD). La DTD describe la estructura del documento, establece los tipos de elementos permitidos y define sus propiedades. Los documentos bien formados no se comparan con una DTD externa.

Los esquemas, al igual que las DTD, describen la estructura del documento. Sin embargo, tienen más funciones que las definiciones de tipo de documento, ya que también pueden describir otras estructuras de información, como las bases de datos. También proporcionan información adicional sobre la herencia y los tipos de datos del documento XML.

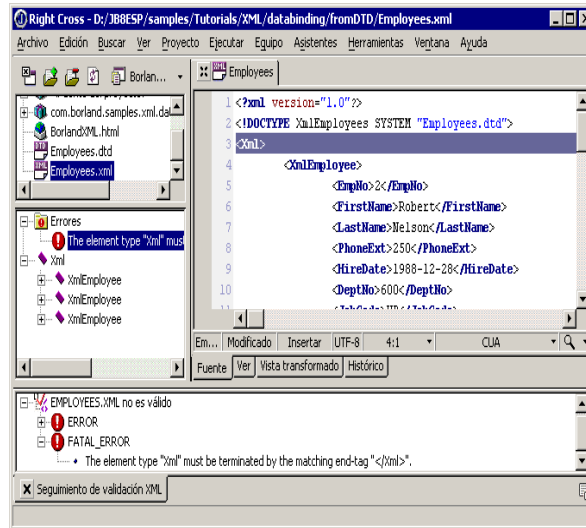
JBuilder incluye el analizador Xerces para validar los documentos XML. Si desea más información sobre Xerces, consulte la documentación y los ejemplos del directorio `extras` de la instalación completa de JBuilder, o visite la página web de Apache, en <http://xml.apache.org/>.

Cuando se abre un documento XML en JBuilder, el panel de estructura muestra la estructura del documento. Si la forma del documento no es correcta, el panel de estructura muestra una carpeta Errores con los mensajes de error. Estudie estos mensajes y corrija los errores de la estructura. Pulse sobre un mensaje de error en el panel de estructura para resaltarlo en el código fuente. Haga doble clic en el mensaje de error para cambiar el foco a la línea de código, en el editor. Es posible que el origen del error no se encuentre en la línea de código a la que señala el mensaje.

Figura 2.8 Carpeta de errores del panel de estructura



JBuilder también puede validar la gramática del código XML del documento comparándola con las definiciones de la DTD. Haga clic con el botón derecho del ratón en el documento XML, en el panel de proyecto, y elija Validar. También puede abrir el archivo XML en el editor, pulsar con el botón secundario del ratón sobre la pestaña del nombre de archivo y elegir Validar. Si el documento es válido aparece un cuadro de diálogo con el mensaje correspondiente. Si el documento contiene errores, el resultado se muestra en una hoja de inspección de la validación XML en el panel de mensajes. Pulse en uno de ellos para resaltar el error en el código fuente. Haga doble clic en un mensaje para desplazar el cursor al código fuente.

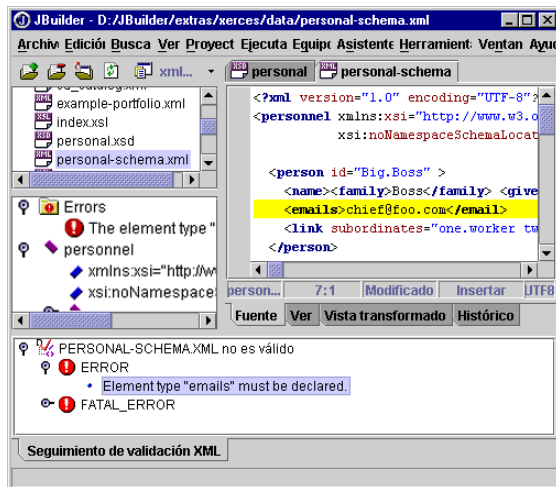
Figura 2.9 Errores de validación de XML al aplicar una DTD

El panel de mensajes muestra los dos tipos de mensajes de error: forma y gramática correctas. Si falta la DTD, el documento no se considera válido, y esto se indica en el panel de mensajes. Después de corregir los errores, es necesario volver a validar el documento.

Si desea seguir un tutorial sobre la validación de documentos XML con archivos DTD, consulte el [Capítulo 4, "Tutorial: Creación y validación de documentos XML"](#).

JBuilder acepta también la validación de archivos de esquema (XSD). Igual que con las DTD, pulse el archivo con el botón derecho del ratón en el panel de proyecto y seleccione Validar. Si un archivo de esquema no está disponible, esto se indica en el panel de mensajes. Aparecen errores en el panel de estructura o en el de mensajes. Si el esquema es válido, esto se indica en un cuadro de diálogo.

Importante la validación de configuraciones XML precisa que se utilice una configuración 2001: `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`. Si se utiliza una versión más antigua, aparece una excepción en el panel de mensajes. Los ejemplos más avanzados de configuración Xerces con esquemas 2001 se encuentran en el directorio `<jbuilder>/extras/xerces/data/`.

Figura 2.10 Errores de validación de XML al aplicar esquemas

Presentación de documentos XML

Es una función de
JBuilder Enterprise.

Puesto que XML permite separar el contenido del documento de su presentación, los documentos se pueden presentar en diversos formatos mediante hojas de estilo. Por ejemplo, un documento XML podría mostrarse como HTML, PDF o WML, según la hoja de estilos aplicada, sin necesidad de volver a escribir el contenido XML. JBuilder proporciona herramientas adicionales para efectuar las tareas de presentación de los documentos XML:

- Cocoon como capa de presentación.
- Transformación de documentos XML.

Presentación de XML mediante Cocoon

Cocoon, que forma parte del proyecto XML Apache, es un marco de publicación para XML, escrito en Java y basado en servlet, que se encuentra integrado en JBuilder. Cocoon permite la separación entre contenido, estilo y lógica, y utiliza transformación XSL para combinarlos. Cocoon también puede utilizar hojas lógicas, páginas XSP, para proporcionar contenido dinámico incrustado con lógica de programación escrita en Java.

El modelo Cocoon divide el contenido web en los siguientes elementos:

- Creación de XML: Los propietarios de contenido que deben entender las DTD pero no necesitan informarse sobre su procesamiento crean archivos XML.
- Procesado de XML: El archivo XML se procesa en consonancia con las hojas lógicas. La lógica se separa del contenido.

- Representación de XSL: El documento XML se representa mediante una hoja de estilo y la aplicación de formato según el tipo de recurso (PDF, HTML, WML, XHTML).

Si desea obtener información completa sobre el uso de Cocoon, consulte la documentación y los ejemplos de Cocoon en el subdirectorio `extras/cocoon` del directorio de instalación de JBuilder, o visite la sede web de Apache, en <http://xml.apache.org/cocoon/index.html>.

Para obtener más información sobre las aplicaciones web, consulte la *Guía del desarrollador de aplicaciones Web*.

Creación de una aplicación web Cocoon

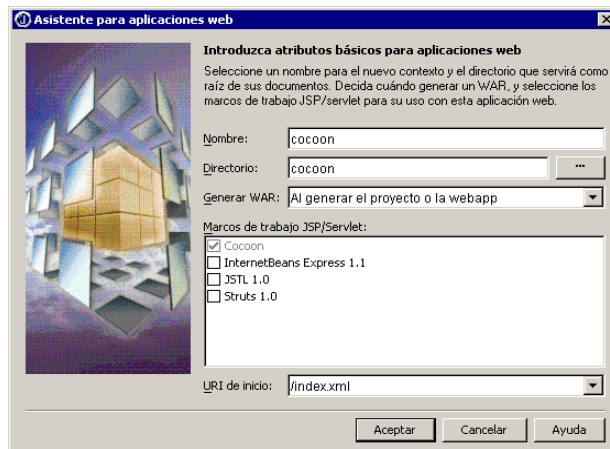
Existen dos formas de crear una aplicación Web Cocoon:

- Asistente para aplicaciones web Cocoon.
- Asistente para aplicaciones web.

Cuando se crea una aplicación web Cocoon con alguno de estos asistentes, se configura esta herramienta de forma que utilice la versión de Cocoon, incluida en JBuilder.

Para crear una aplicación web Cocoon con el Asistente para aplicaciones web Cocoon:

- 1 Cree un proyecto con el Asistente para proyectos (Archivo | Nuevo proyecto).
- 2 Elija Archivo | Nuevo y abra la ficha XML de la galería de objetos.
- 3 Haga doble clic en el icono Aplicación web Cocoon para abrir el Asistente para aplicaciones web Cocoon. Observe que este asistente es el Asistente para aplicaciones web con Cocoon seleccionado como marco de trabajo.

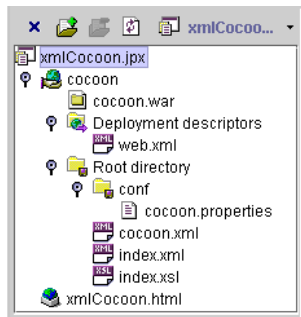


- 4 Acepte los valores predeterminados y pulse Aceptar para cerrar el asistente y generar los archivos de Cocoon.
- 5 Seleccione el archivo en el panel de proyectos, haga clic con el botón derecho y elija Ejecutar Make para generar el archivo WAR.
- 6 Añada los archivos XML y XSL con el botón Añadir al proyecto de la barra de herramientas del panel de proyecto.
- 7 Guarde el proyecto.

Para crear una aplicación web Cocoon con el Asistente para aplicaciones web:

- 1 Cree un proyecto con el Asistente para proyectos (Archivo | Nuevo proyecto).
- 2 Elija Archivo | Nuevo y seleccione la pestaña Web de la galería de objetos.
- 3 Haga doble clic en el icono Aplicaciones web para abrir el Asistente para aplicaciones web.
- 4 Seleccione Cocoon como entorno de trabajo. Los campos Nombre y Directorio, que se pueden modificar, se rellenan automáticamente como cocoon. El valor predeterminado de URI de inicio es `index.xml`, que es la ficha Cocoon predeterminada de JBuilder que aparece cuando se ejecuta el nodo cocoon.
- 5 Acepte los valores predeterminados y pulse Aceptar para cerrar el asistente y generar los archivos de Cocoon.
- 6 Seleccione el archivo en el panel de proyectos, haga clic con el botón derecho y elija Ejecutar Make para generar el archivo WAR.
- 7 Añada los archivos XML y XSL con el botón Añadir al proyecto de la barra de herramientas del panel de proyecto.
- 8 Guarde el proyecto.

Para ver los archivos Cocoon generados por el asistente, despliegue el nodo cocoon y el nodo <Fuente del proyecto> en el panel de proyecto



- `CatalogManager.properties`

Este archivo de propiedades predeterminado de Cocoon, situado en el directorio fuente del proyecto, contiene preferencias y valores del catálogo. Para redefinir la funcionalidad de ese archivo o añadir otros valores, puede utilizar `cocoon.xconf`. Si desea obtener más información, consulte “Using CatalogManager.properties” en la documentación de Cocoon, que se encuentra en `<jbuilder>/extras/cocoon/docs/userdocs/concepts/catalog.html`.

- `cocoon.war`

Archivo recopilatorio web.

- `cocoon.xconf`

Archivo de configuración que contiene registros de hojas lógicas. Este archivo describe los componentes básicos y los optativos. También especifica la ubicación de `sitemap.xmap` y de otros parámetros. Por ejemplo, este archivo podría especificar un analizador, un motor JSP, etc.

- `logkit.xconf`

Archivo de configuración que sirve para controlar los archivos históricos. Los registros históricos proporcionan información de depuración, distribución y funcionamiento. Los archivos de registro histórico se pueden crear mediante el conjunto de herramientas LogKit. Si desea obtener información sobre el conjunto de herramientas LogKit, consulte <http://jakarta.apache.org/avalon/logkit/index.html>.

- `web.xml`

Descriptor de distribución de servlets que especifica la ubicación de `cocoon.xconf`, la ubicación de archivos históricos, y otros parámetros.

- `index.xml`

Un ejemplo de documento XML que se especifica al lanzar URI por defecto. Este documento se muestra automáticamente cuando se ejecuta Cocoon.

- `index.xsl`

Hoja de estilos de ejemplo que se utiliza para aplicar formato HTML a `index.xml`.

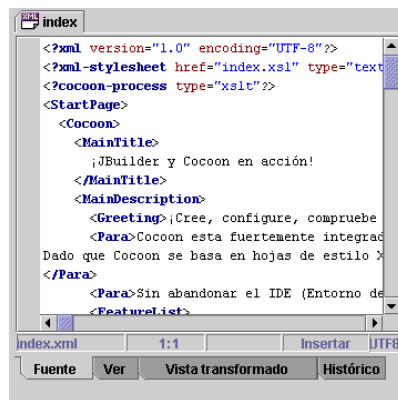
- `sitemap.xmap`

Este archivo asocia el espacio URI con los recursos. Consta de dos partes: componentes y pipelines, que constan, a su vez, de componentes. El Asistente para aplicaciones web Cocoon agrega una asociación a este archivo que apunta a la hoja de estilos para la página de ejemplo Cocoon de JBuilder, `index.xml`.

Si se desea efectuar cambios más adelante, es posible modificar la mayoría de estos archivos directamente en el editor sin necesidad de volver a ejecutar el asistente. También se pueden modificar los archivos originales utilizados por el asistente y ubicados en `<jbuilder>/defaults/cocoon`. Si desea obtener más información sobre cómo configurar Cocoon, consulte la documentación de Cocoon que reside en el directorio `extras/cocoon/` de JBuilder.

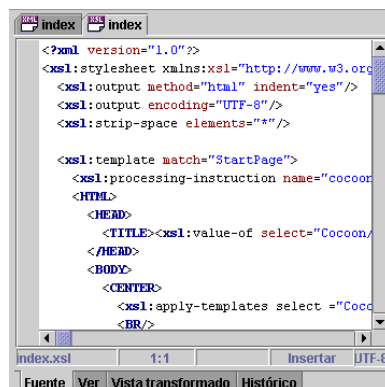
Si desea más información sobre `web.xml` y el editor del descriptor de distribución, consulte los apartados correspondientes a “Descriptores de distribución” de los apartados “Las WebApps y los archivos WAR” y “Distribución de aplicaciones web” de la *Guía del desarrollador de aplicaciones Web*.

Figura 2.11 Código fuente XML para index.xml



La hoja de estilo para `index.xml`, `index.xsl`, contiene reglas de formato HTML. Por lo que, cuando se aplica a `index.xml`, el documento XML se representa como un documento HTML.

Figura 2.12 Código fuente de hoja de estilo para index.xml



Ejecución de Cocoon

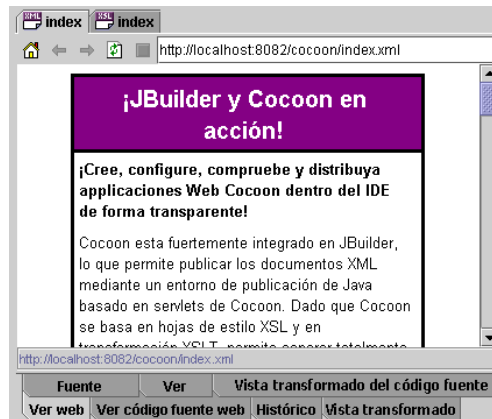
Para ejecutar Cocoon, se debe configurar en el proyecto un servidor web que admita servlets y páginas JSP. El Asistente para aplicaciones web Cocoon configura automáticamente Tomcat y lo utiliza para ejecutar Cocoon. Cocoon también trabaja con otros servidores de aplicaciones web. Para obtener más información sobre la configuración de servidores, consulte “Configuración del servidor web” en la *Guía del desarrollador de aplicaciones Web*.

Para ejecutar Cocoon, realice los siguientes pasos:

- 1 Haga clic con el botón derecho del ratón en el nodo cocoon del panel de proyecto.
- 2 En el menú contextual, seleccione Ejecutar web utilizando los valores por defecto.

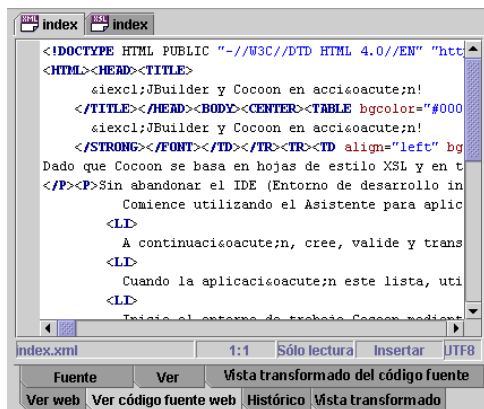
Cocoon ejecuta el motor de servlet configurado actualmente, se introduce en el entorno del servlet, carga `index.xml` en la vista web del panel de contenido y utiliza la información de los archivos `web.xml` y `cocoon.properties` generados por el Asistente para aplicaciones web Cocoon. Es posible modificar `cocoon.properties` para añadir bibliotecas de XSP y recursos a las hojas lógicas.

Figura 2.13 Vista Web de `index.xml`



Para ver el código fuente de la vista Web, abra la ficha Ver código fuente Web.

Figura 2.14 Vista del código fuente Web de index.xml



Transformación de documentos XML

Es una función de
JBuilder Enterprise.

El proceso de conversión de documentos XML a otro tipo de documento se denomina *transformación XML*. JBuilder incorpora el procesador de hojas de estilo Xalan para transformar documentos XML, y utiliza hojas de estilo escritas en XSLT para la transformación. Las hojas de estilo Extensible Style Language (XSL) contienen instrucciones para la transformación de documentos XML de un tipo a otro (XML, HTML, PFD, WML, etc.).

Importante

Si no puede transformar su documento XML, compruebe que está utilizando la versión adecuada de la especificación de hoja de estilos, <http://www.w3.org/1999/XSL/Transform>.

Si desea ver un tutorial sobre la transformación de documentos XML, consulte el [Capítulo 5, “Tutorial: Transformación de documentos XML”](#).

Si desea más información sobre Xalan, consulte la documentación y los ejemplos del directorio `extras` de la instalación completa de JBuilder, o visite la sede web de Apache, en <http://xml.apache.org/xalan-j/index.html>.

Aplicación de hojas de estilo internas

Si desea aplicar hojas de estilo a las que se hace referencia interna en el documento XML, haga lo siguiente:

- 1 Abra el documento XML en JBuilder.
- 2 Haga clic en la pestaña Vista transformada de la parte inferior del panel de contenido.

Nota

Si el documento contiene una instrucción de proceso XSLT y una sola hoja de estilo, ésta se aplica al documento XML. Si en su lugar se muestra una vista en árbol, desactívela pulsando el botón Hoja de estilo por defecto de la barra de herramientas Vista transformado.



El documento transformado, que se recoge en un búfer provisional, se presenta en la ficha Vista transformado del panel de contenido con la hoja de estilo aplicada. También se muestra la ficha Vista fuente de transformado, en la que se puede ver el código fuente de la transformación.

Si desea aplicar otra hoja de estilo interna en la instrucción de hoja de estilo del documento, elíjala en la lista desplegable Hoja de estilo de la barra de herramientas Vista transformado.

Figura 2.15 Barra de herramientas Vista transformado



Tabla 2.1 Botones de la barra de herramientas Vista transformado

Botón	Descripción
Hoja de estilo por defecto	Aplica la hoja de estilo por defecto de JBuilder, una vista de árbol expandible, a la Vista transformado del panel de contenido. Esta opción afecta al resultado de la transformación. Nota: Es diferente a la opción Aplicar hoja de estilo por defecto de la ficha XML del cuadro de diálogo Opciones del IDE (Herramientas Opciones del IDE XML), que aplica una vista de árbol al documento XML que se muestra en el visualizador de XML (ficha Ver del panel de contenido)
Actualizar	Actualiza la vista.
Definir las opciones de seguimiento	Abre el cuadro de diálogo Opciones de seguimiento de la transformación.
Añadir hojas de estilo	Abre el cuadro de diálogo Configurar hojas de estilo de nodos, donde se pueden asociar hojas de estilo a documentos.

Aplicación de hojas de estilo externas

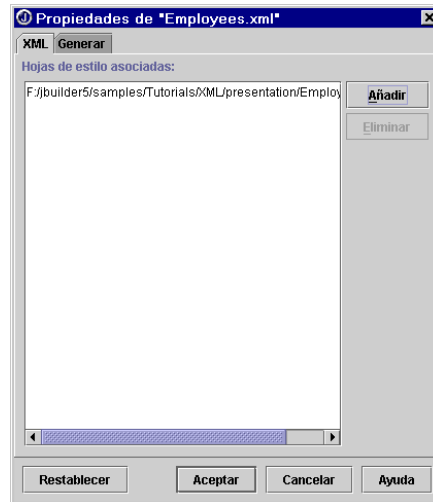
También es posible aplicar hojas de estilo externas a los documentos. En primer lugar, es necesario asociarlas al documento XML.

1 El cuadro de diálogo se puede abrir de las siguientes formas:

- Haga clic con el botón derecho del ratón en el documento XML, en el panel de proyecto, y elija Propiedades.

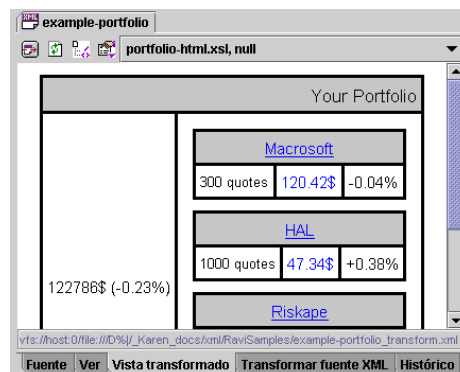


- Pulse el botón Añadir hojas de estilo de la barra de herramientas Vista transformado.

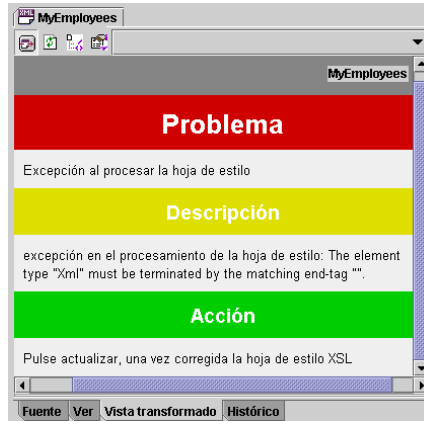


- 2 Seleccione las hojas de estilo y pulse los botones Añadir y Eliminar. Haga clic en Aceptar para cerrar el cuadro de diálogo. Cuando las hojas de estilo se asocian al documento, aparecen en la lista desplegable de la barra de herramientas Vista transformado con las hojas de estilo internas.
- 3 Abra la ficha Vista transformado y seleccione una hoja de estilo externa de la lista desplegable para aplicarla. Si el documento muestra una vista en árbol, desactívela pulsando el botón Hoja de estilo por defecto de la barra de herramientas Vista transformado.

Figura 2.16 Vista transformado a la que se ha aplicado una hoja de estilo externa



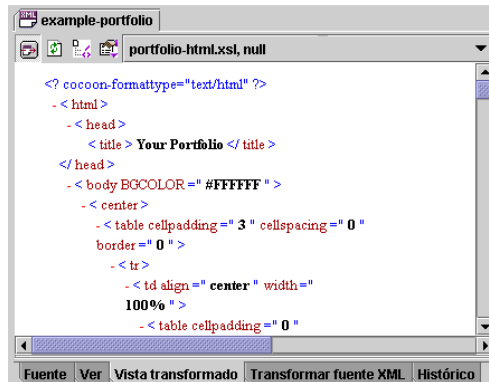
Nota Si no hay una hoja de estilo disponible, en la Vista transformado aparece un mensaje que indica que no hay hojas de estilo asociadas al documento.

Figura 2.17 Vista transformado sin hoja de estilo

- 4 Para mostrar el resultado de la transformación en una vista en árbol con la hoja de estilo por defecto de JBuilder, pulse el botón Hoja de estilo por defecto de la barra de herramientas Vista transformada. Esto resulta útil si la salida de una transformación es otro documento XML sin hoja de estilo.

**Nota**

Si los resultados de la transformación no son un documento XML bien formado, no podrá mostrarlo en la vista de árbol. Esto se produce a menudo si los resultados se encuentran en HTML que no está bien construido.

Figura 2.18 Vista transformado con la hoja de estilo por defecto de vista en árbol**Configuración de opciones de seguimiento**

Las opciones de seguimiento de la transformación se pueden configurar de forma que, cuando ocurra una transformación, se pueda inspeccionar el proceso. Estas opciones son Generación, Plantillas, Elementos y Selecciones.

Para activar la inspección:

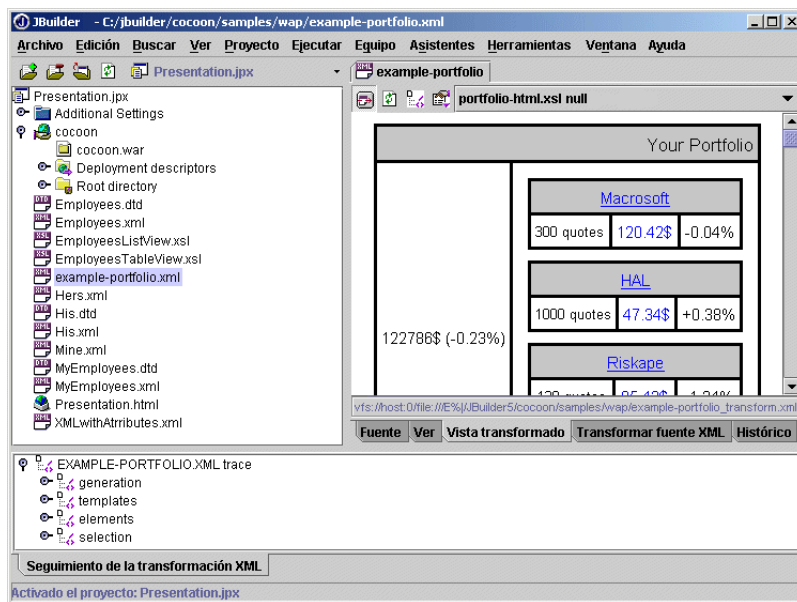
- 1 Seleccione Herramientas | Opciones del IDE.
- 2 Seleccione la pestaña Información.
- 3 Active las opciones de inspección que desee.

Consulte

- Opciones de seguimiento, descritas en “Configuración de opciones XML” en la página 2-10



Si lo prefiere, pulse el botón Definir opciones de seguimiento de la barra de herramientas Vista transformado. Las inspecciones se muestran en el panel de mensajes. Seleccione una inspección para resaltar el correspondiente código fuente en el editor. Si se hace doble clic en una inspección, el foco pasa al código fuente en el editor, donde se puede modificar.



Manipulación de XML mediante programas

La manipulación programática es una característica de JBuilder Enterprise

Normalmente, el programa de XML se manipula mediante la escritura de código, para lo que se utilizan analizadores o soluciones de asociación de datos más especializadas. JBuilder acepta los dos métodos y proporciona herramientas para ambos:

- Asistente Gestor de SAX

- Soluciones para la asociación de datos:
 - BorlandXML para la generación de código fuente Java a partir de DTD.
 - Castor para la generación de código fuente Java a partir de esquemas.

SAX (la API sencilla para XML) se puede utilizar para procesar datos XML, que se tratan de forma muy similar a un flujo de datos de texto. Si bien SAX es relativamente rápido al procesar los datos, presenta varios inconvenientes. SAX es de sólo lectura, valida únicamente la estructura de un documento y no almacena una copia en memoria de los datos. Las soluciones para la asociación de datos, tales como BorlandXML y Castor, pueden leer y escribir datos XML, validar el contenido de los datos y la estructura, procesar los datos mucho más rápidamente que SAX y reducir las necesidades de mantenimiento. No obstante, cada solución tiene su uso apropiado. Por ejemplo, SAX es apropiado para trabajar con aplicaciones sencillas que no necesitan validación de su contenido. Una solución para la asociación de datos es más apropiada para trabajar con aplicaciones más complejas que requieren validación del contenido de los datos y de la escritura XML.

JBuilder proporciona muchas bibliotecas predefinidas que se pueden añadir al proyecto: JDOM, Xerces, BorlandXML, Castor, etc. Si utiliza los asistentes de JBuilder, las bibliotecas apropiadas se añaden automáticamente. Se añaden al proyecto en el cuadro de diálogo Propiedades de proyecto. Elija Proyecto | Propiedades de proyecto y abra la ficha Vías de acceso. Haga clic en la pestaña Bibliotecas necesarias y añada las bibliotecas. Cuando se añaden las bibliotecas, CodeInsight de JBuilder tiene acceso a ellas y puede mostrar ventanas emergentes, sensibles al contexto, dentro del editor. Estas ventanas muestran los miembros de datos accesibles, los métodos, las clases y los parámetros que cabe esperar para el método que se está codificando; además, profundiza en el código fuente. Si utiliza los asistentes de JBuilder, las bibliotecas apropiadas se añaden automáticamente.

Creación de un gestor de SAX

Es una función de
JBuilder Enterprise.

Existen dos tipos de API XML: API basadas en árboles y API basadas en sucesos. Las API basadas en árboles, que compilan documentos XML en una estructura de árbol interno, permite a la aplicación desplazarse por el árbol. Esta API basada en árboles se está normalizando como modelo de objeto de documento (Document Object Model, DOM).

SAX, la API sencilla para XML, es una interfaz estándar para el análisis XML basado en sucesos. SAX informa sobre los sucesos de análisis directamente a la aplicación, por medio de devoluciones de llamada. La

aplicación implementa manejadores que gestionan los distintos sucesos, de forma parecida a la gestión de sucesos de la interfaz de usuario.

Por ejemplo, una API basada en sucesos examina este documento:

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

<page>
  <title>Ejemplo basado en suceso</title>
  <content>¡Hola a todos!</content>
</page>
```

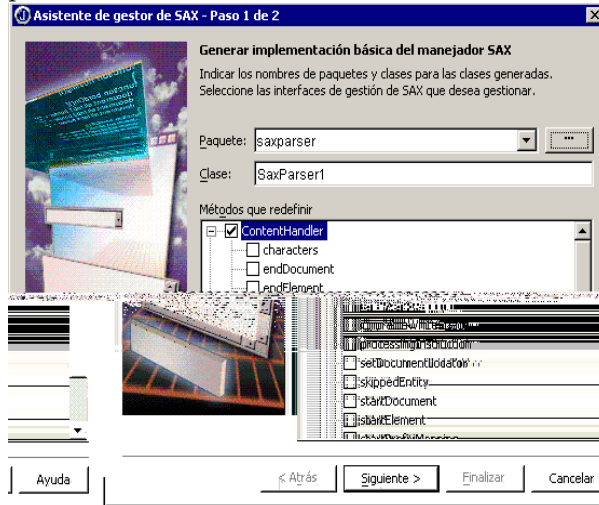
y lo divide en estos sucesos:

```
start document
start element: page
start element: title
characters: Ejemplo basado en suceso
end element: title
start element: content
characters: ¡Hola a todos!
end element: content
end element: page
end document
```

JBuilder facilita la utilización de SAX para la manipulación programática de XML. El Asistente de gestor de SAX utiliza JAXP (API de Java para el procesamiento de XML), incluido en el JDK 1.4, con el fin de crear una plantilla de implementación del analizador SAX que incluye sólo los métodos que se desean implementar en el análisis del código XML. JAXP admite el procesamiento de documentos XML mediante SAX, DOM y XSLT.

- 1 Seleccione Archivo | Nuevo para abrir la galería de objetos, abra la ficha XML y haga doble clic en el icono Gestor de SAX para abrir el asistente.

- 2 Escriba nombres para el paquete y la clase o acepte los predeterminados.



- 3 Seleccione las interfaces y métodos que desea redefinir y pulse Siguiente. Si desea más información acerca de estos métodos e interfaces, consulte la documentación de la API de SAX.
- 4 Seleccione el analizador SAX y cualquier opción que desee. Las opciones varían de acuerdo con el analizador SAX seleccionado.

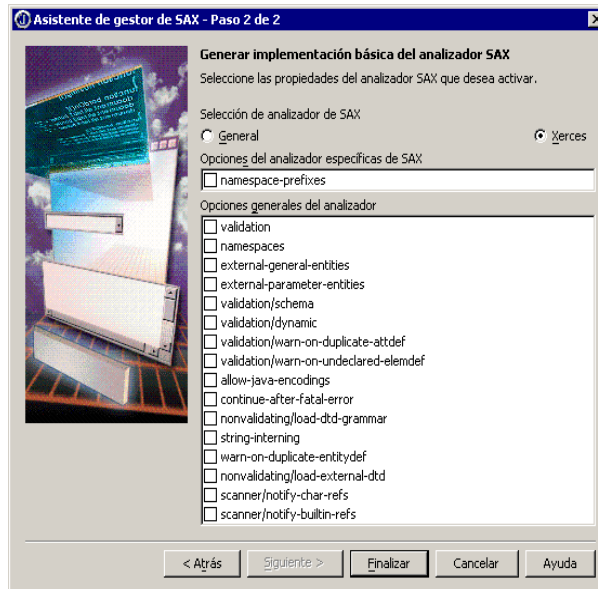
El analizador generar es un analizador compatible con JAXP, que sólo admite funciones JAXP necesarias. JDK 1.4 utiliza Crimson como el analizador por defecto. Para obtener más información, consulte la clase SAXParserFactory.

El analizador Xerces admite todas las funciones de Xerces 2. Para obtener más información sobre Xerces y las opciones que admite, consulte la documentación de <jbuilder>\extras\xerces\docs\features.html o visite la página web de Apache en <http://xml.apache.org/xerces2-j/features.html>.

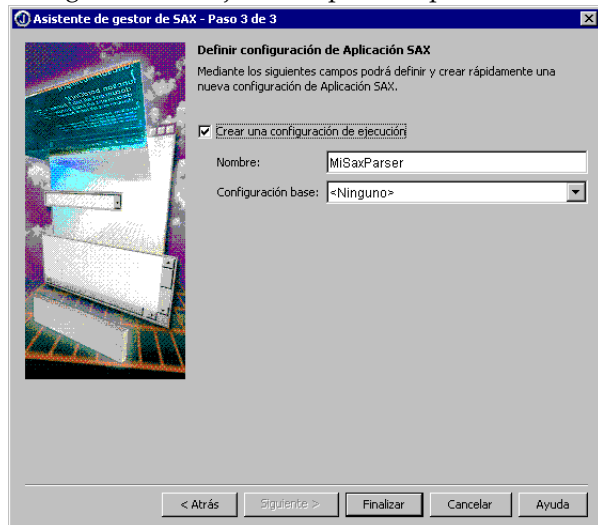
Importante

Si está utilizando el JDK 1.3 para su proyecto utilice Xerces como analizador, dado que JDK 1.3 no incluye un analizador JAXP. Cuando

se selecciona Xerces como el analizador, JBuilder añade automáticamente la biblioteca Xerces al proyecto.

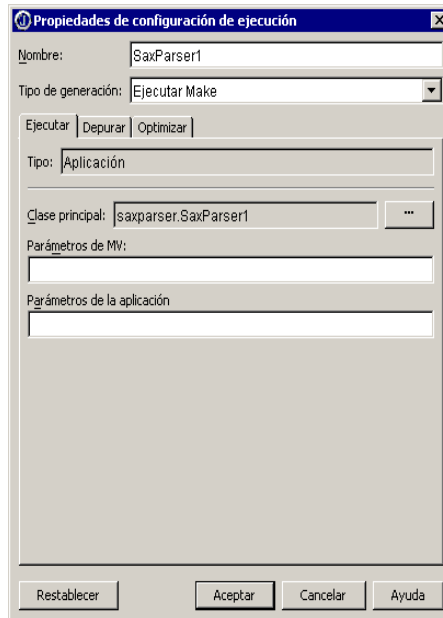


- 5 Haga clic en Siguiete para avanzar al paso siguiente del asistente. Aquí se puede ver que el asistente crea automáticamente una configuración de ejecución para la aplicación SAX.



- 6 Pulse Finalizar para crear una clase que implemente un analizador SAX.
- 7 Modifique el código fuente y añada la lógica necesaria para implementar los métodos seleccionados.
- 8 Modifique la configuración de ejecución creada por el asistente en la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto y, en el campo Parámetros de la aplicación, especifique el archivo XML que desee analizar.
 - a Seleccione Ejecutar | Configuraciones para mostrar la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto.
 - b Seleccione la configuración de ejecución creada por el asistente y pulse Modificar para modificar los parámetros de la aplicación.
 - c Escriba la vía de acceso del documento XML en el campo Parámetros de la aplicación. Por ejemplo:

file:///C:/<jbuilder>\samples\Tutorials\XML\saxparser\Employees.xml



- d Haga dos veces clic en Aceptar para cerrar los cuadros de diálogo. Si desea obtener más información sobre configuraciones de ejecución, consulte "Definición de las configuraciones de ejecución" en *Creación de aplicaciones con JBuilder*.
- 9 Guarde el proyecto y seleccione Ejecutar | Ejecutar proyecto para generar y ejecutar el proyecto.

Consulte

- Los paquetes SAX: `org.xml.sax`, `org.xml.sax.ext` y `org.xml.sax.helpers` en la Especificación API de Java (Ayuda | Referencia de Java)
- [Capítulo 6, “Tutorial: Creación de un gestor de SAX para analizar documentos XML”](#)

Manipulación de XML mediante asociación de datos

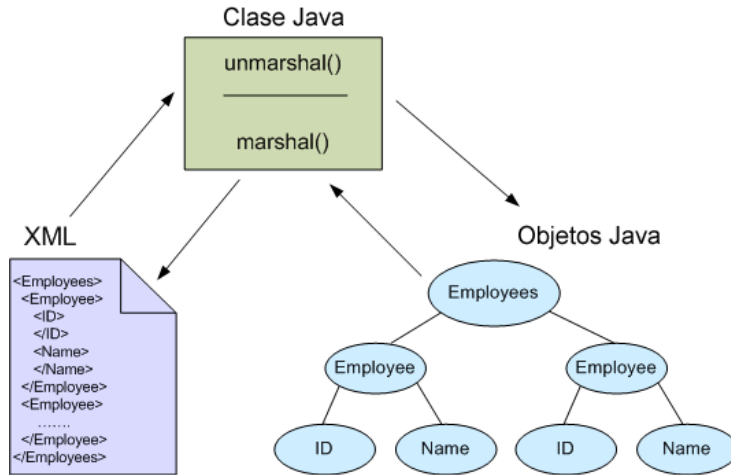
Es una función de
JBuilder Enterprise.

La asociación de datos constituye una forma de acceder a datos, manipularlos y, una vez revisados, devolverlos a la base de datos o mostrarlos en un documento XML. El documento XML se puede utilizar como mecanismo de transferencia entre la base de datos y la aplicación. Para realizar esta transferencia se asocia un objeto Java a un documento XML. Para implementar la asociación de datos se generan clases Java que representan las restricciones que contiene la gramática, tales como los esquemas DTD y XML. Después, se pueden utilizar estas clases para crear y leer documentos XML acordes con la gramática, y validar documentos XML con la gramática cuando se efectúan cambios en ellos.

JBuilder ofrece varios tipos de soluciones para la asociación de datos: BorlandXML y Castor de código abierto. BorlandXML genera clases Java a partir de archivos DTD, mientras que Castor genera clases Java a partir de archivos schema (XSD).

El marco de codificación

BorlandXML y Castor utilizan un marco de codificación para realizar conversiones de datos entre Java y XML. El marco de codificación se divide en dos partes: montaje (*marshalling*) y desmontaje (*unmarshalling*). El *marshalling* escribe un documento XML a partir de objetos JavaBean (Java a XML). En el *desmontaje* (*unmarshalling*) se lee un documento XML y sus objetos se convierten en objetos Java (XML a Java).

Figura 2.19 Marco de codificación

Consulte

- “La especificación de asociación de datos XML” en <http://www.oasis-open.org/cover/xmlDataBinding.html>

BorlandXML

BorlandXML proporciona un mecanismo de asociación de datos que oculta los detalles de XML y reduce la complejidad del código, todo ello con un mantenimiento sencillo. BorlandXML es un generador de clases programables basado en plantillas que se utiliza para generar clases JavaBean a partir de una definición de tipo de documento (DTD). Después, se utiliza la sencilla convención de programación JavaBean para manipular los datos XML sin preocuparse por los detalles.

BorlandXML utiliza las DTD para la generación de clases Java en un proceso de dos pasos. En el primero, se genera un archivo de modelo de clase a partir de una DTD. Este archivo es un documento XML con la extensión `.bom`. Este archivo describe una estructura de alto nivel de las clases de destino y proporciona una forma de personalizarlas. En el segundo paso, BorlandXML genera clases Java a partir del archivo `.bom` (archivo de modelo de clase XML).

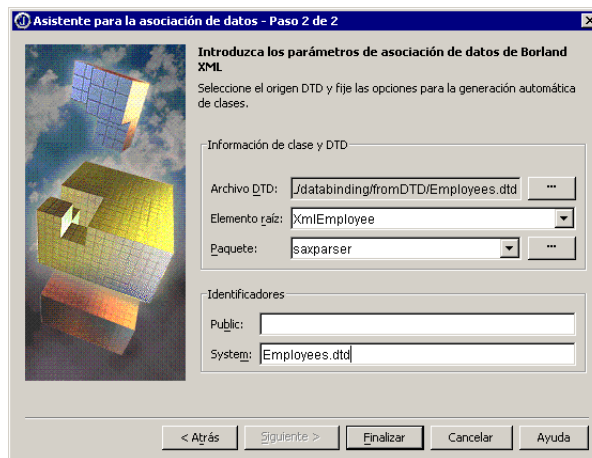
BorlandXML acepta varias funciones:

- Manipulación de JavaBeans: Manipula los beans para construir documentos XML o acceder a los datos que contienen.
- Un marco de codificación para convertir datos entre Java y XML.

- Validación de documento: Valida los objetos JavaBean antes de su montaje en XML o después de desmontar un documento XML, en objetos JavaBean.
- Personalización de PCDATA: Permite personalizar PCDATA con el fin de aceptar distintos tipos de datos primitivos, tales como `integer` y `long`, así como nombres de propiedades personalizadas.
- Nombres de variables: Permite que los nombres de variables generados para los elementos y los atributos tengan prefijos y sufijos personalizados.

Para generar clases Java a partir de un DTD, utilice el Asistente para la asociación de datos de la siguiente forma:

- 1 Haga clic con el botón derecho del ratón en el archivo DTD en el panel de proyecto, elija Generar Java y se abrirá el asistente. El nombre del archivo aparece automáticamente en el campo Archivo DTD. El Asistente para la asociación de datos también se puede abrir desde la ficha XML de la galería de objetos (Archivo | Nuevo).
- 2 Seleccione BorlandXML como tipo de asociación de datos, que se basa únicamente en DTD, y pulse Siguiente.
- 3 Rellene los campos necesarios, tales como el nombre y la ubicación de la DTD que se utiliza, el Elemento raíz y el nombre del Paquete. El *elemento raíz* es el primero del documento, y contiene los demás elementos.
- 4 Escriba un identificador PUBLIC o SYSTEM, que se introduce en la declaración DOCTYPE.



- 5 Pulse el botón Finalizar.
- 6 Amplíe el nodo Paquete generado del panel de proyectos para presentar los archivos `.java` generados por el asistente:

7 Escriba el código que interactúe con estas clases y desmonte (lea) y monte (escriba) los datos. Por ejemplo:

```

Foo foo = Foo.unmarshal("D:\Temp\foo.xml");    \\Lee desde foo.xml
foo.setBar("This is an element");             \\Modifica la barra de
elementos
foo.marshal("D:\Temp\foo-modified.xml");       \\Escribe en foo-modified.xml

```

Para ver un tutorial sobre la asociación de datos con BorlandXML, consulte el [Capítulo 7, “Tutorial: Asociación de datos DTD con BorlandXML”](#).

El directorio `extras` de la instalación completa de JBuilder contiene ejemplos y documentación sobre BorlandXML.

Castor

Castor es un marco de asociación de datos XML que asigna una instancia de un esquema XML a un modelo de objetos que representa los datos. Este modelo de objetos incluye un conjunto de clases y tipos, así como descriptores que se utilizan para obtener información sobre una clase y sus campos.

Castor utiliza un marco de montaje que incluye un conjunto de `ClassDescriptors` (descriptores de clases) y `FieldDescriptor` (descriptores de campos) que describen la forma en que se debe montar y desmontar un objeto desde XML.

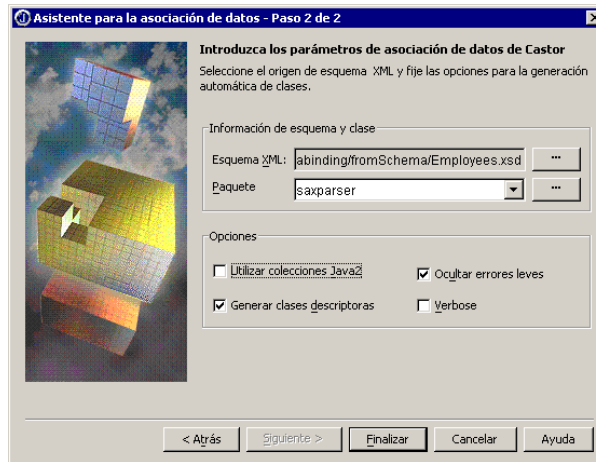
Castor utiliza esquemas en lugar de DTD para crear clases Java. Los esquemas (XSD), más resistentes y flexibles, tienen varias ventajas respecto a las DTD. Los esquemas son documentos XML. Por contra, las DTD contienen sintaxis ajena a XML. Además, los esquemas aceptan los espacios en los nombres (necesarios para evitar los conflictos de denominación), ofrecen tipos de datos más extensos y aceptan la herencia.

Importante Castor necesita soporte para esquema 2001: `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`. Si se utiliza una versión más antigua de schema, aparece una excepción en el panel de mensajes. Esto mismo es válido para una comprobación de esquemas XML. Los ejemplos más avanzados de configuración Xerces con esquemas 2001 se encuentran en el directorio `<jbuilder>/extras/xerces/data/`.

Para generar clases Java a partir de un esquema XML con el Asistente para la asociación de datos:

- 1 Haga clic con el botón derecho del ratón en el archivo de esquema (XSD) en el panel de proyecto, elija Generar Java y se abrirá el asistente. De esta forma, el nombre del archivo aparece automáticamente en el campo Archivo de esquema XML. El Asistente para la asociación de datos también se puede abrir desde la ficha XML de la galería de objetos (Archivo | Nuevo).

- 2 En Tipo de asociación de datos elija Castor, que acepta los esquemas XML, y pulse Siguiente.
- 3 Rellene los campos necesarios, tales como el Paquete, y configure las opciones deseadas.



- 4 Pulse el botón Finalizar.
- 5 Amplíe el nodo Paquete generado del panel de proyectos para presentar los archivos .java generados por el asistente:
- 6 Escriba el código que interactúe con estas clases y desmonte (lea) y monte (escriba) los datos. Por ejemplo:

```
\\Leer archivo
Foo foo = Foo.unmarshal(new FileReader("D:\Temp\foo.xml"));
\\Modificar barra de elementos
foo.setBar("This is an element");
\\Escribir en el archivo
foo.marshal(new java.io.FileWriter("D:\Temp\foo-modified.xml"));
```

Importante Aparecerán advertencias del compilador desaconsejando porque Castor genera código que utiliza Sax 1.0.

Nota Por defecto, el sistema de montaje de Castor escribe documentos XML sin sangrías, ya que éstas aumentan el tamaño de los documentos generados. Si desea activar el sangrado, modifique el archivo `castor.properties` con el siguiente contenido: `org.exolab.castor.indent=true`. Este archivo contiene otras propiedades que se pueden modificar. El Asistente para la asociación de datos crea automáticamente el archivo `castor.properties` en el directorio de código fuente del proyecto.

Para ver un tutorial sobre la asociación de datos con Castor, consulte el [Capítulo 8, "Tutorial: Asociación de datos de esquema con Castor"](#).

El directorio `extras` de la instalación completa de JBuilder contiene ejemplos y documentación sobre Castor. También se encuentran en la página web de Castor, en <http://castor.exolab.org>.

Interacción con datos empresariales en bases de datos

Es una función de
JBuilder Enterprise.

JBuilder admite dos categorías de bases de datos XML: basadas en modelos y basadas en plantillas. La solución basada en modelos utiliza un documento de asignación que determina la transferencia de los datos entre la estructura XML y los metadatos de la base de datos. Los componentes basados en modelos, `XMLDBMSTable` y `XMLDBMSQuery`, se implementan por medio de XML-DBMS, un software intermedio XML de código abierto integrado en JBuilder.

La solución basada en plantillas se basa en conjuntos de normas. `XTable` y `XQuery`, los componentes de la solución basada en plantillas, son muy flexibles, ya que no existe una relación predefinida entre el documento XML y el conjunto de metadatos de base de datos que se consulta.

Para obtener más información sobre la adición de componentes XML de bases de datos, consulte el [Capítulo 3, “Componentes de base de datos XML de JBuilder”](#).

Consulte

- XML-DBMS en <http://www.rpbouret.com/xmldbms/>

Componentes de base de datos XML de JBuilder

Es una función de JBuilder Enterprise.

El soporte para bases de datos XML de JBuilder se encuentra disponible a través de un conjunto de componentes de la ficha XML de la paleta de componentes del diseñador de interfaces. El código durante el tiempo de ejecución para los beans aparece como parte de una biblioteca redistribuible en `xmlbeans.jar`.

Los componentes de base de datos XML de la biblioteca XmlBeans pueden ser de dos tipos:

- Componentes basados en modelos.
- Componentes basados en plantillas.

Los componentes basados en modelos utilizan un documento de mapa que determina cómo se transfieren los datos entre una estructura XML y los metadatos de la base de datos. Gracias a que el usuario especifica una asignación entre un elemento del documento XML y una tabla o columna en concreto de una base de datos, los documentos XML anidados se pueden trasladar desde y hacia un conjunto de tablas de bases de datos. Los componentes basados en modelos se implementan utilizando XML-DBMS, un software intermedio de Open Source XML que se distribuye en un paquete con JBuilder.

Para utilizar componentes basados en plantillas, debe proporcionar una sentencia SQL y el componente generará el documento XML apropiado. Esa sentencia SQL funciona como la plantilla que se sustituye en el documento XML al aplicarla. La solución basada en plantillas es muy flexible, ya que no se establece una relación predeterminada entre el documento XML y el conjunto de metadatos de base de datos que se están consultando. Aunque los componentes basados en plantillas son muy

eficaces a la hora de obtener datos de una base de datos e introducirlos en un documento XML, el formato de dicho documento es bastante simple. Además, los componentes basados en plantillas pueden generar documentos HTML basados en hojas de estilo por defecto o personalizadas.

Consulte

- paquete `com.borland.jbuilder.xml.database.template`, en *XML Database Components Reference* de la *DataExpress Component Library Reference*
- paquete `com.borland.jbuilder.xml.database.xmldbms`, en *XML Database Components Reference* de la *DataExpress Component Library Reference*
- paquete `com.borland.jbuilder.xml.database.common`, en *XML Database Components Reference* de la *DataExpress Component Library Reference*

Utilización de los componentes basados en modelos

JBuilder utiliza XML-DBMS en los componentes basados en modelos. XML-DBMS, que es un software intermedio (middleware) para transferir datos entre documentos XML y bases de datos relacionales, utiliza asignación de objeto relacional para asignar objetos a la base de datos. XML-DBMS se distribuye con JBuilder. El código fuente, los ejemplos y la documentación de XML-DBMS se encuentran en el directorio `<jbuilder>/extras/xmldbms`.

JBuilder suministra dos beans para transferir datos XML-DBMS: `XMLDBMSTable` y `XMLDBMSQuery`, que son el tercer y cuarto bean de la ficha XML de la paleta de componentes que se encuentra en el diseñador de interfaces de JBuilder. El bean `XMLDBMSTable` utiliza la tabla y claves especificadas como criterios de selección para la transferencia, mientras que el bean `XMLDBMSQuery` se ocupa de los resultados de una consulta SQL.

Si desea seguir un tutorial sobre el uso de los componentes XML basados en modelos, consulte el [Capítulo 9, “Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos”](#).

Si desea colocar un bean en su aplicación, seleccione la pestaña Diseño y haga clic en la pestaña XML en la paleta de componentes. Seleccione un bean y suéltelo en el diseñador.

XML-DBMS

La solución XML-DBMS consiste en lo siguiente:

- Una base de datos relacional con un controlador JDBC.
- Un documento XML para la entrada y salida de datos.

- Un documento de asignación XML que define la asignación entre la base de datos y el documento XML.
- Una biblioteca con un conjunto de métodos API para transferir datos entre la base de datos y el documento XML.

En el centro de XML-DBMS se encuentra el documento de correspondencia especificado en XML. Este se define por un lenguaje de asignación y está documentado como parte de la distribución XML-DBMS. Si desea obtener más información, consulte la documentación de XML-DBMS y los archivos de código fuente del directorio *extras* de JBuilder.

Entre los elementos principales del lenguaje de asignación, están:

- **ClassMap**

El ClassMap es la raíz de la asignación. Un ClassMap asigna una tabla de base de datos a elementos XML que contienen otros elementos. Además, un ClassMap anida PropertyMaps y RelatedClassMap.

- **PropertyMap**

Un PropertyMap asigna elementos PCDATA y atributos de valor simple a columnas específicas en una tabla de base de datos.

- **RelatedClassMap**

RelatedClassMap asigna las relaciones entre las clases. Esto se lleva a cabo haciendo referencia a otro tipo de elemento (como, por ejemplo ClassMap, declarado en otro lugar), y especificando la base de la relación. La asignación define las claves principal y ajena utilizadas en la relación y en cuál de las dos tablas se almacena la clave principal. Observe que el identificador CandidateKey se utiliza para representar una clave principal.

Además, se admite la creación de claves. Hay escenarios en los que las claves son datos empresariales como CustNo o EmpNo. En otros, es necesario crear las claves para establecer los enlaces. Esto es posible si se utiliza un atributo de generación como parte de la definición de la clave correspondiente.

Si fuera necesario, también se admite un orderColumn optativo con creación automática de claves como parte de la asignación.

- **MiscMaps y opciones**

Además de las asignaciones ya descritas, existen otras para gestionar nulls, omitir un elemento raíz que no tenga datos correspondientes en la base de datos, sino que sólo sirva como elemento de agrupación, y otras para gestionar formatos de fecha y hora.

JBUILDER y XML-DBMS

JBUILDER suministra el siguiente soporte XML-DBMS:

- Asistente XML-DBMS.
- `XModelBean`: clase base para `XMLDBMSTable` y `XMLDBMSQuery`
- `XMLDBMSTable`: transfiere datos basados en una tabla y una clave.
- `XMLDBMSQuery`: transfiere datos basados en un conjunto de resultados definido por una consulta SQL.

Creación de un documento de correspondencia y un archivo de script SQL

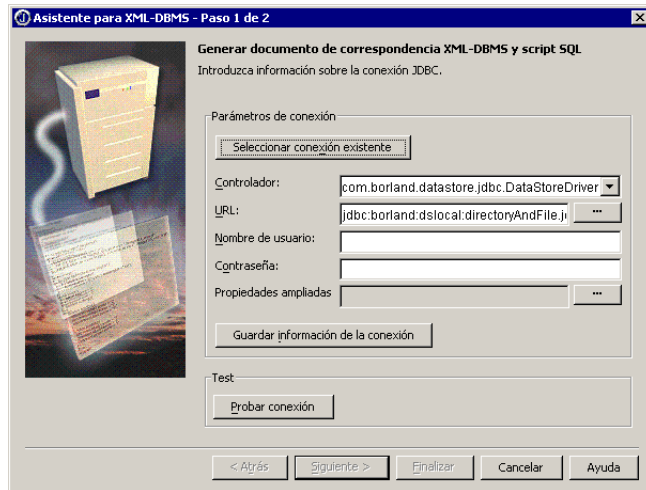
El asistente XML-DBMS de JBUILDER forma parte de la solución basada en modelos o asignaciones que utiliza el API de `Map_Factory_DTD` en XML-DBMS. Si se cuenta con un DTD, el asistente genera una plantilla de documento de mapa y un archivo de script SQL para crear los metadatos. En la mayoría de los casos, el documento de asignación es únicamente el punto de partida para la creación de la asignación necesaria. También es necesario modificar el script SQL (el cual consiste en una serie de sentencias `Create Table`), porque XML-DBMS no regenera los scripts SQL a partir del documento de asignación modificado.

El asistente XML-DBMS no admite la creación de un archivo de mapa a partir de un esquema de base de datos. Si se empieza con una base de datos ya creada, es necesario crear el archivo de asignación de forma manual. Si tiene el documento XML, puede abrirlo, hacer clic en él con el botón derecho del ratón y generar la DTD. A continuación, esta DTD se puede utilizar para generar el archivo de asignación y modificarla para que coincida con el esquema de la base de datos.

Para crear archivos de correspondencia y de script SQL:

- 1 Seleccione Archivo | Nuevo y haga clic en la pestaña XML de la galería de objetos.

- 2 Haga doble clic en el icono XML-DBMS para iniciar el Asistente para XML-DBMS.



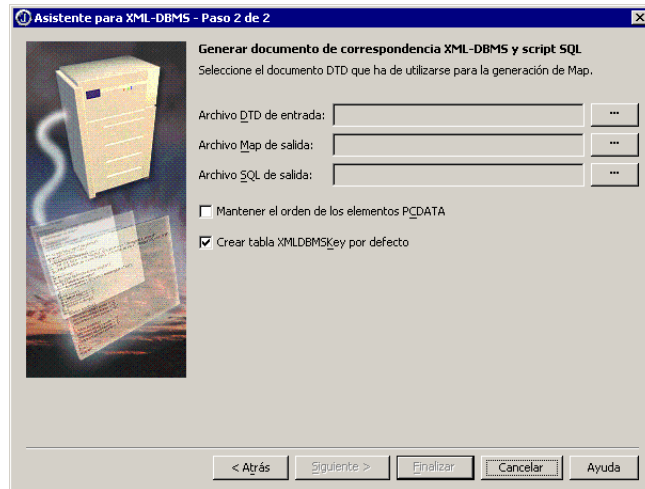
En la primera ficha puede especificar la conexión JDBC con la base de datos que contenga la información que desea utilizar para crear un documento XML. Contiene estos campos:

- **Controlador** Escoja de la lista desplegable el controlador JDBC que se va a usar. Los controladores que aparecen en negro son controladores que ya están instalados. Los controladores que aparecen en rojo no están disponibles.
- **URL** Especifique la dirección URL de la fuente de datos que contenga la información que pretenda usar para crear un documento XML. Cuando se hace clic en el campo, se muestra el patrón que se debe usar para especificar la dirección URL dependiendo del controlador JDBC que se escoja.
- **Nombre de usuario** Si es necesario, introduzca el nombre de usuario para la fuente de datos.
- **Contraseña** Introduzca la contraseña de la fuente de datos, si es que se le solicita.
- **Propiedades ampliadas** Añada cualquier propiedad ampliada que necesite. Haga clic sobre el botón de puntos suspensivos para abrir el cuadro de diálogo de propiedades ampliadas que utilice para añadir propiedades.

Si ya tiene una o más conexiones definidas en JBuilder hacia las fuentes de datos, haga clic sobre el botón Seleccionar conexión existente... y seleccione la conexión que desee. La mayoría de los parámetros de conexión se rellenan automáticamente.

Para comprobar si su conexión JDBC es correcta, haga clic sobre el botón Probar conexión. El personalizador informa sobre el resultado de la conexión. Después de haber probado la conexión JDBC, puede elegir Guardar información de la conexión para su uso en el futuro.

3 Pulse el botón Siguiente tras haber verificado la conexión.



Esta ficha sirve para definir el DTD que va a utilizar para generar el archivo de mapa y el archivo del script SQL que crea la tabla de la base de datos. Rellene los siguientes campos:

- Archivo DTD Especifique un archivo DTD.
- Directorio de salida Puede dejar el nombre por defecto o modificarlo si lo desea.
- Archivo MAP Especifique el nombre del Archivo de mapa que desea generar.
- Archivo de script SQL Especifique el nombre del archivo de script SQL que desea generar.

4 Pulse Aceptar para cerrar el asistente. El asistente crea los archivos de mapa y SQL y los añade a su proyecto.

Por ejemplo, suponga que tiene un DTD, el `request.dtd`:

```
<!ELEMENT request (req_name, parameter*)>
<!ELEMENT parameter (para_name, type, value)>
<!ELEMENT req_name (#PCDATA)>
<!ELEMENT para_name (#PCDATA)>
```

```
<!ELEMENT type (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

El asistente para XML-DBMS generaría este archivo request.map:

```
<?xml version='1.0' ?>
<!DOCTYPE XMLToDBMS SYSTEM "xmldbms.dtd" >

<XMLToDBMS Version="1.0">
  <Options>
  </Options>
  <Maps>
    <ClassMap>
      <ElementType Name="request"/>
      <ToRootTable>
        <Table Name="request"/>
        <CandidateKey Generate="Yes">
          <Column Name="requestPK"/>
        </CandidateKey>
      </ToRootTable>
      <PropertyMap>
        <ElementType Name="req_name"/>
        <ToColumn>
          <Column Name="req_name"/>
        </ToColumn>
      </PropertyMap>
      <RelatedClass KeyInParentTable="Candidate">
        <ElementType Name="parameter"/>
        <CandidateKey Generate="Yes">
          <Column Name="requestPK"/>
        </CandidateKey>
        <ForeignKey>
          <Column Name="requestFK"/>
        </ForeignKey>
      </RelatedClass>
    </ClassMap>
    <ClassMap>
      <ElementType Name="parameter"/>
      <ToClassTable>
        <Table Name="parameter"/>
      </ToClassTable>
      <PropertyMap>
        <ElementType Name="para_name"/>
        <ToColumn>
          <Column Name="para_name"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="type"/>
        <ToColumn>
          <Column Name="type"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="value"/>
```

```

        <ToColumn>
          <Column Name="value"/>
        </ToColumn>
      </PropertyMap>
    </ClassMap>
  </Maps>
</XMLToDBMS>

```

El asistente para XML-DBMS crearía el siguiente archivo, `request.sql`:

```

CREATE TABLE "request" ("req_name" VARCHAR(255), "requestPK" INTEGER);
CREATE TABLE "parameter" ("para_name" VARCHAR(255), "type" VARCHAR(255),
                           "requestFK" INTEGER, "value" VARCHAR(255));
CREATE TABLE XMLDBMSKey (HighKey Integer);
INSERT INTO XMLDBMSKey VALUES (0);

```

Una vez que tenga un archivo de mapa y un archivo de script SQL, puede modificarlos como desee. Por ejemplo, el nombre de un elemento puede ser “HireDate”, pero el nombre de la columna es “Date_Hired”. Puede realizar el cambio directamente modificando el archivo de mapa. Normalmente, el archivo de script SQL es solamente el punto de partida para crear el tipo de tabla que desea, así que, a menudo, también es necesario modificarlo.

Cuando tenga el archivo de script SQL a su gusto, ejecútelo para crear las tablas de la base de datos. Un modo muy sencillo de hacerlo consiste en copiar las sentencias SQL en el Explorador de bases de datos y, a continuación, hacer clic en el botón Ejecutar. Si desea información adicional acerca del uso del Explorador de bases de datos, consulte “Explorador de base de datos” en la *Guía del desarrollador de aplicaciones de base de datos*. Si desea información más específica acerca de la ejecución de sentencias SQL, consulte el tema “Ejecución de sentencias SQL” en el capítulo Tareas de administración de bases de datos de la *Guía del desarrollador de aplicaciones de base de datos*.

Definición de propiedades para los componentes basados en modelos

Una vez que tenga el archivo XML, el archivo de mapa y las tablas de la base de datos, ya puede utilizar los beans basados en modelos para transferir datos en ambos sentidos entre el archivo XML y la tabla.

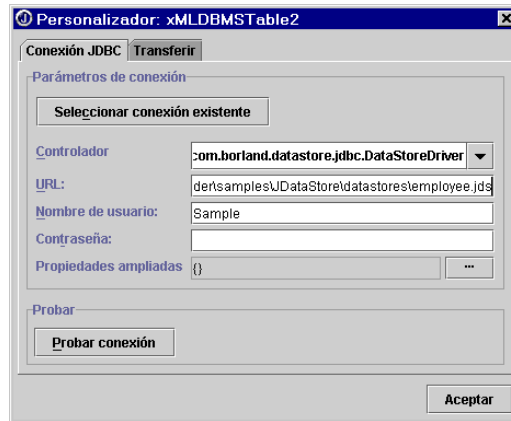
Hay dos modos de configurar las propiedades para un bean basado en modelos:

- Configuración de propiedades con el personalizador.
- Configuración de propiedades con el Inspector.

Configuración de propiedades con el personalizador

Para abrir el personalizador de un componente, haga clic en él, con el botón derecho del ratón en el panel de estructura, y elija Personalizador en el menú contextual.

Este es el personalizador para `XMLDBMSTable`:



Establecimiento de una conexión JDBC

La ficha Conexión JDBC permite definir la conexión JDBC con la base de datos que contenga los datos que desee utilizar para crear el documento XML. Contiene estos campos:

- **Controlador** Escoja de la lista desplegable el controlador JDBC que se va a usar. Los controladores que aparecen en negro son controladores que ya están instalados. Los controladores que aparecen en rojo no están disponibles.
- **URL** Especifique la dirección URL de la fuente de datos que contenga la información que pretenda usar para crear un documento XML. Cuando se hace clic en el campo, se muestra el patrón que se debe usar para especificar la dirección URL dependiendo del controlador JDBC que se escoja
- **Nombre de usuario** Si es necesario, introduzca el nombre de usuario para la fuente de datos.
- **Contraseña** Introduzca la contraseña de la fuente de datos, si es que se le solicita.
- **Propiedades ampliadas** Añada cualquier propiedad ampliada que necesite. Haga clic sobre el botón de puntos suspensivos (...) para abrir el cuadro de diálogo de propiedades ampliadas que utilice para añadir propiedades.

Si ya tiene una o más conexiones definidas en JBuilder hacia las fuentes de datos, haga clic sobre el botón Seleccionar conexión existente... y seleccione la conexión que desee. La mayoría de los parámetros de conexión se rellenan automáticamente.

Para comprobar si su conexión JDBC es correcta, haga clic sobre el botón Probar conexión. El personalizador le informa acerca de si la conexión falló o no. Después de haber probado la conexión JDBC, puede elegir Guardar información de la conexión para su uso en el futuro.

Cuando haya establecido una conexión, abra la ficha Transferir.



Transferencia de datos

La ficha Transferir del asistente permite indicar si se transfieren datos de un documento XML a la base de datos y viceversa, así como aportar la información necesaria para la transferencia.

Para transferir los datos de un archivo XML al archivo de base de datos, siga estos pasos:

- 1 Modifique el archivo XML para que contenga los valores que desee que se transfieran a la tabla de la base de datos.
- 2 En la ficha Transferir del personalizador `XMLDBMSTable`, rellene el campo Archivo XML de entrada con el nombre del archivo XML que contiene la información que se transfiere a la base de datos.
- 3 Especifique el archivo de mapa que ha creado como el valor para el campo Archivo MAP.

Los dos campos restantes están desactivados para este tipo de transferencia.

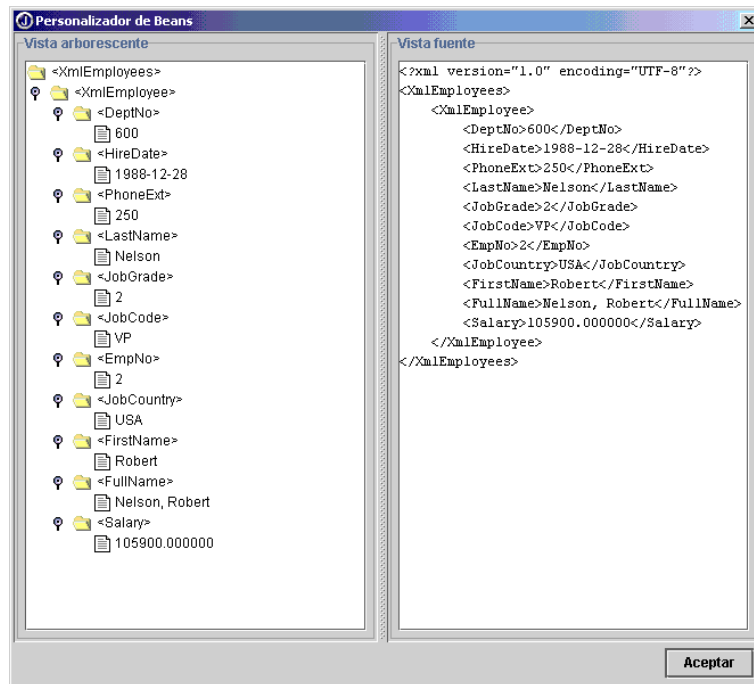
- 4 Pulse el botón Transferir.

Para ver el resultado de la transferencia, elija Herramientas | Explorador de bases de datos para abrir la tabla y ver el contenido.

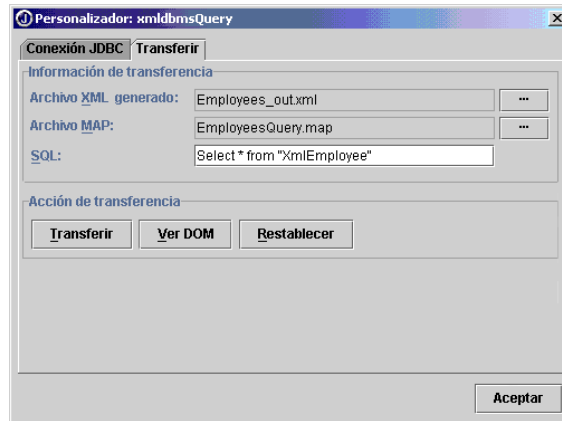
Si desea transferir datos desde la base de datos al archivo XML, siga los siguientes pasos:

- 1 Haga clic en el botón de radio DB a XML.
- 2 Rellene el campo Archivo de salida XML con el nombre del archivo XML que va a recibir los datos transferidos desde la base de datos.
- 3 Especifique el archivo de mapa que ha creado como el valor para el campo Archivo de mapa.
- 4 Especifique el nombre de la tabla desde la que se transfieren los datos como el valor del campo Nombre de la tabla.
- 5 Defina el valor o valores de la clave principal que especifica el registro o registros que se van a transferir. Por ejemplo, si la clave primaria es EMP_NO y se desean transferir los datos de la fila en la que EMP_NO es igual a 5, el valor de la clave indicado debe ser 5. Consulte el archivo de asignación para averiguar la clave. Aparece definida como "CandidateKey", en el nodo <RootTable> de la tabla.
- 6 Seleccione Transferir.

Si desea ver los resultados de la transferencia, seleccione Ver DOM, y podrá consultar cómo queda la estructura del archivo XML:



La ficha Transferir es distinta en el caso de `xmlDBMSQuery`:



El `XMLDBMSQuery` permite especificar una consulta SQL para transferir datos de la base de datos al documento XML.

Para transferir datos de la base de datos al archivo XML:

- 1 Defina un nombre para Archivo generado.
- 2 Especifique un nombre para el Archivo MAP.
- 3 Introduzca su sentencia SQL en el campo SQL.
- 4 Seleccione Transferir.

Puede consultar los resultados de la transferencia seleccionando Ver DOM.

Configuración de propiedades con el Inspector

También se pueden configurar las propiedades de los beans basados en modelos en el Inspector del diseñador. Para abrir el Inspector:

- 1 Seleccione la pestaña Diseño en el panel de contenido. El Inspector aparece a la derecha del diseñador.
- 2 Haga clic en el campo a la derecha de una propiedad e introduzca la información pertinente.

Si desea seguir un tutorial sobre cómo utilizar los componentes `XMLDBMSTable` y `XMLDBMSQuery`, consulte el [Capítulo 9, "Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos"](#).

Utilización de los componentes basados en plantillas

Los dos componentes basados en plantillas son XTable y XQuery, el primero y el segundo de los componentes de la ficha XML de la paleta de componentes de JBuilder.

Si desea seguir un tutorial sobre el uso de los componentes XML basados en plantillas, consulte el [Capítulo 10, “Tutorial: Transferir datos con componentes XML de bases de datos basados en plantillas”](#).

Para comenzar a trabajar con estos componentes, seleccione uno de ellos en la ficha XML de la paleta de componentes. Para añadirlo en su aplicación, suéltelo en el diseñador de interfaces o en el panel de estructura.

Configuración de propiedades para las plantillas de bean

Se pueden configurar las propiedades de los dos componentes basados en plantillas de tres maneras:

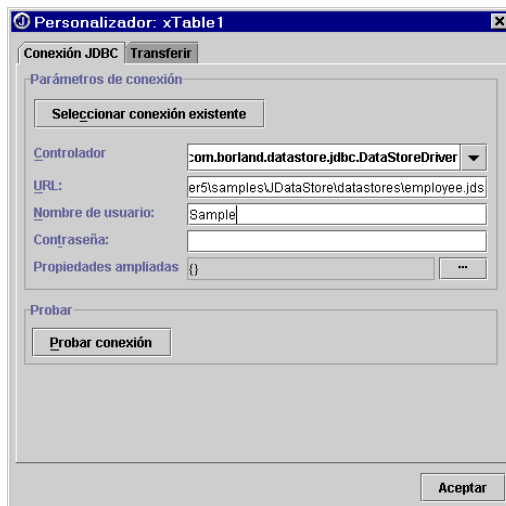
- Configuración de propiedades con el personalizador.
- Configuración de propiedades con el Inspector.
- Configuración de propiedades con un documento de consulta XML.

Configuración de propiedades con el personalizador

Cada componente de bases de datos XML cuenta con su propio personalizador. La forma más sencilla de configurar las propiedades de un componente consiste en utilizar su personalizador. Puede incluso comprobar su conexión JDBC, llevar a cabo la transferencia para visualizar el nuevo documento y ver el Document Object Model (Modelo de Objeto Documento, DOM).

Para abrir el personalizador de un componente, haga clic en él, con el botón derecho del ratón en el panel de estructura, y elija Personalizador en el menú contextual.

Por ejemplo, este es el personalizador para xTable:



Establecimiento de una conexión JDBC

La ficha Conexión JDBC permite definir la conexión JDBC con la base de datos que contenga los datos que desee utilizar para crear el documento XML. Contiene estos campos:

- **Controlador** Escoja de la lista desplegable el controlador JDBC que se va a usar. Los controladores que aparecen en negro son controladores que ya están instalados. Los controladores que aparecen en rojo no están disponibles.
- **URL** Especifique la dirección URL de la fuente de datos que contenga la información que pretenda usar para crear un documento XML. Cuando se hace clic en el campo, se muestra el patrón que se debe usar para especificar la dirección URL dependiendo del controlador JDBC que se escoja.
- **Nombre de usuario** Introduzca el nombre de usuario para la fuente de datos, si lo hay.
- **Contraseña** Introduzca la contraseña de la fuente de datos, si es que se le solicita.
- **Propiedades ampliadas** Añada cualquier propiedad ampliada que necesite. Haga clic sobre el botón de puntos suspensivos (...) para abrir el cuadro de diálogo de propiedades ampliadas que utilice para añadir propiedades.

Si ya tiene una o más conexiones definidas en JBuilder hacia las fuentes de datos, haga clic sobre el botón Seleccionar conexión existente... y seleccione la conexión que desee. La mayoría de los parámetros de conexión se rellenan automáticamente.

Para comprobar si su conexión JDBC es correcta, haga clic sobre el botón Probar conexión. El personalizador le informa acerca de si la conexión falló o no. Después de haber probado la conexión JDBC, puede elegir Guardar información de la conexión para su uso en el futuro.

Cuando haya establecido una conexión, abra la ficha Transferir.

The screenshot shows the 'Personalizador: xTable1' dialog box with the 'Transferir' tab selected. The 'Información de transferencia' section includes fields for 'Archivo de consulta:', 'Archivo de salida:' (set to 'utorials/XML/database/Beans/xTableOut.html'), and 'Archivo XSL:'. Below this, the 'Formato de columna' section has radio buttons for 'Como elementos' (selected) and 'Como atributos'. The 'Formato de salida' section has radio buttons for 'XML' and 'HTML' (selected). The 'Nombres de elementos' section has text boxes for 'Documento:' (set to 'XmlEmployee') and 'Fila:' (set to 'XmlEmployee'). There is a checkbox for 'Omitir nulls'. The 'Nombre de la tabla:' field is set to 'XmlEmployee'. The 'Claves:' field is set to 'EmpNo' and the 'DefaultParams:' field is set to 'EmpNo=2'. At the bottom, there are buttons for 'Transferir', 'Ver', 'Restablecer', and 'Aceptar'.

Transferencia de datos

La ficha Transferir contiene los siguientes campos:

- Archivo de consulta

Especifique un documento XML de consulta (optativo). Si lo hace, no tendrá que rellenar ningún campo del personalizador excepto el campo Nombre de archivo de salida y, si lo desea, el Nombre de archivo XSL, ya que el documento de consulta especifica los valores para las propiedades. Si desea información adicional acerca de la creación y el uso de un documento de consulta XML, consulte [“Configuración de propiedades con un documento de consulta XML”](#) en la página 3-21.
- Archivo de salida

Especifique el nombre del archivo XML o HTML que desee generar.

- Archivo XSL Especifique el nombre del archivo de hoja de estilos XSL que desee utilizar para transformar el archivo de salida, si lo hay. Si no especifica ninguno, se genera una hoja de estilos por defecto y se coloca en el mismo directorio que el archivo de salida. El nombre del fichero XSL generado es `JBuilderDefault.xml`. Este archivo se puede modificar para crear una presentación más personal. Si desea modificar el archivo XSL, asegúrese de que la propiedad Nombre de archivo XSL está configurada para el archivo modificado. Tenga en cuenta que JBuilder no redefine las hojas de estilo por defecto anteriores.
- Formato de columna Especifique si desea que las columnas de la fuente de datos aparezcan como elementos o como atributos en el archivo XML nuevo.
- Formato de salida Especifique si desea que el nuevo archivo esté en formato XML o HTML.
- Nombres de elementos Especifique un nombre para el elemento Documento y otro para el elemento Fila.
- Obviar nulls Marque esta casilla si desea omitir los nulls en su archivo de salida XML. De lo contrario, "null" será un valor admitido.
- Nombre de usuario Escriba el nombre de la tabla que contiene los datos que le interesan. Coloque el nombre entre comillas altas dobles, por ejemplo, `"XmlEmployee"`.
- Contraseña Defina la clave o claves que identifican las filas de la tabla que desee que formen parte del documento XML que ha creado. Para especificar una clave, pulse el botón de puntos suspensivos junto al campo Claves para abrir el Editor de claves. Pulse el botón Añadir para añadir una fila a la rejilla. Cambie el nombre del elemento añadido a una columna de la tabla y coloque el nombre entre comillas altas dobles. Siga añadiendo todas las claves que desee con la ayuda del editor de propiedades. Si especifica un nombre para la tabla pero no las claves, se devolverán todas las filas de la tabla.

- Parámetros por defecto** Utilice este campo para especificar el par nombre/valor de los nombres introducidos en el campo Claves. Si especificó un valor para el campo Claves, debe definir un parámetro por defecto para la columna o columnas que especificó como claves. Pulse el botón de puntos suspensivos junto al campo Parámetros por defecto con el fin de añadir parámetros por defecto a su consulta. En el cuadro de diálogo Parámetros por defecto, pulse el botón Añadir si desea agregar algún parámetro por defecto. En la línea en blanco que se añade, especifique un nombre para el parámetro en Parámetros y su valor en Valor. Por ejemplo, si se desea ver el registro del empleado número 5 se debe indicar “EMP_NO” como nombre del parámetro, y ‘5’ como valor en la columna EMP_NO. Recuerde colocar los valores de cadenas entre comillas simples. Si desea más información sobre los parámetros por defecto, consulte [“Especificación de parámetros” en la página 3-18](#).

El personalizador para *XQuery* se parece mucho. Al igual que *XTable*, cuenta con una ficha Conexión JDBC:

Personalizador: xQuery1

Conexión JDBC | Transferir

Parámetros de conexión

Seleccionar conexión existente

Controlador: com.borland.datastore.jdbc.DataStoreDriver

URL: jdbc:borland:dslocal:F:\builder5\samples\JDataStore\datastores\employee.jds

Nombre de usuario: Sample

Contraseña:

Propiedades ampliadas: {}

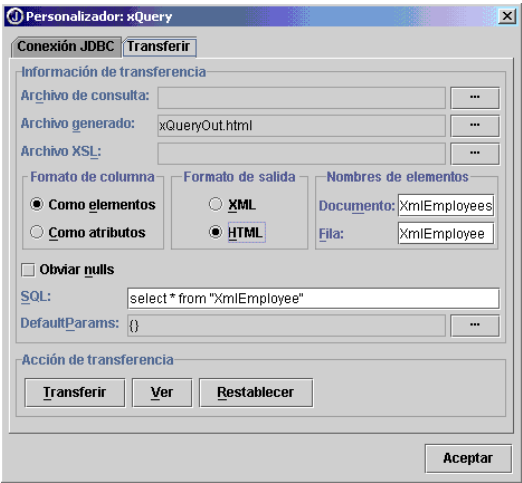
Probar

Probar conexión

Aceptar

Rellene esta ficha tal y como lo hizo para *XTable* y compruebe la conexión. Para guardar la conexión, seleccione Guardar información de la conexión.

La ficha Transferir del personalizador de XQuery se distingue de la de XTable por tener un campo SQL que sustituye a los campos Nombre de tabla y Claves:



En el campo SQL se puede definir cualquier sentencia SQL. Si su sentencia SQL es una consulta parametrizada, debe especificar un parámetro por defecto para cada una de las variables parametrizadas.

Especificación de parámetros

Si la consulta que está utilizando está parametrizada, debe especificar un valor por defecto para el parámetro antes de crear un archivo XML o HTML. Los parámetros por defecto se pasan mediante el método `setDefaultParams()`. Puede redefinir el valor del parámetros por defecto con el valor de otro parámetro añadiendo el método `setParams()` al código.

Si desea ver cómo utilizar parámetros por defecto y otros parámetros, mire esta consulta de ejemplo:

```
Select emp_name from employee where emp_no = :emp_no
```

Supongamos que la tabla Employee contiene las siguientes entradas:

Tabla 3.1 Tabla Employee

emp_no	emp_name
1	Tom
2	Dick

El parámetro `:emp_no` se puede suministrar de dos formas:

- Utilice un parámetro por defecto especificado durante el diseño en el personalizador. Los parámetros por defecto se pasan mediante el método `setDefaultParams()`.

- Redefina el parámetro por defecto durante la ejecución añadiendo el método `setParams()` al código y pasándolo a un parámetro diferente.

Estas son las posibilidades:

- No se especifican parámetros de ningún tipo. El resultado: la consulta devuelve un error.
- `defaultParams` set to `:emp_no = 1`. El resultado: la consulta devuelve Tom.
- `defaultParams` configurado en `:emp_no = 1` y el método `setParams()` utilizado durante la ejecución pasando el argumento,
`:emp_no = 2`. The result: la consulta devuelve Dick.

Por ejemplo, el código, durante la ejecución, debe tener un aspecto semejante a este:

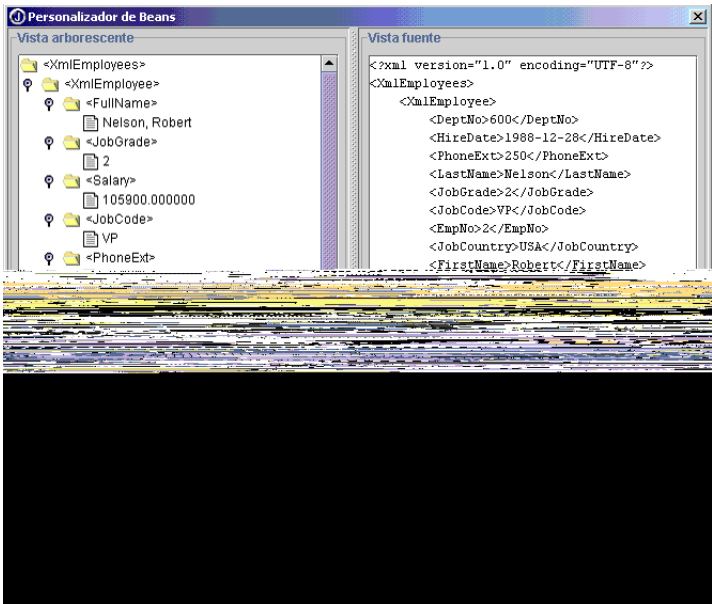
```
xTable.setParams(XData.convertToHashMap (new String [][] {
    {"EmpNo","'2'"},}));
```

En otras palabras, si el parámetro se especifica mediante el método `setParams()` durante la ejecución, redefine el valor del parámetro por defecto, configurado durante el diseño. Los nombres de parámetros distinguen entre mayúsculas y minúsculas.

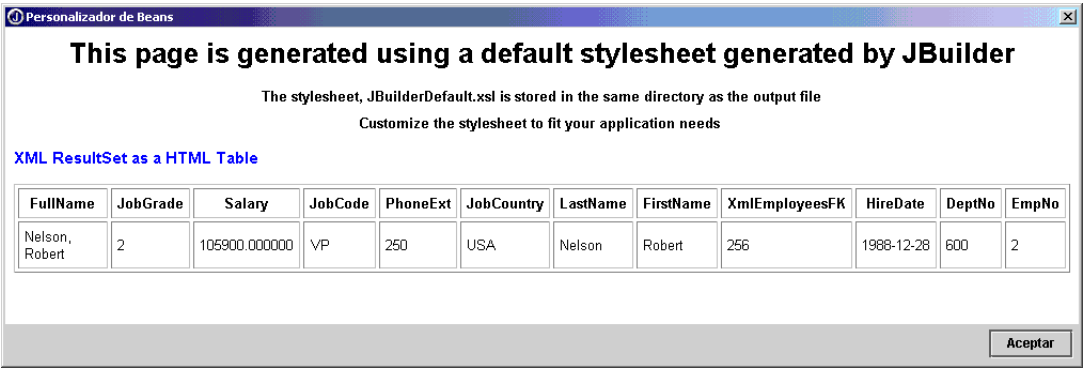
Transferencia a XML o HTML

Si desea ver cuáles serán los resultados de su configuración de propiedades, haga clic en el botón Transferir. Si decide crear un archivo

XML, pulse el botón Ver y aparecerá el modelo de objeto de documento (DOM):



Si decide generar un archivo HTML, puede pulsar el botón Ver archivo para abrir el documento resultante:



Configuración de propiedades con el Inspector

También puede establecer el valor de estas propiedades en el Inspector del diseñador. Para abrir el Inspector:

- 1 Seleccione la pestaña Diseño en el panel de contenido. El Inspector aparece a la derecha del diseñador.
- 2 Haga clic en el campo a la derecha de una propiedad e introduzca la información pertinente.

Configuración de propiedades con un documento de consulta XML

Otro modo de configurar las opciones de conexión y transferencia consiste en hacerlo mediante un documento de consulta XML. Puede crear un documento de consulta XML y especificarlo como el valor del campo Archivo de consulta en el personalizador del componente o en el Inspector.

Nota Los nombres de elementos XML distinguen entre mayúscula y minúscula.

A continuación se muestra un documento de consulta de ejemplo para XTable:

```
<Query>
  <Options
    OutputType="XML"
    ColumnFormat="AsElements"
    IgnoreNulls="True"
    DocumentElement="MyDoc"
    RowElement="YourRow">

  <Connection
    Url="jdbc:odbc:foodb"
    Driver="sun.jdbc.odbc.JdbcOdbcDriver"
    User="me"
    Password="ok"
    ExtendedProperties="name=value;name=value...">

  <Params>
    <Param Name=":Part" Default="'ab-c'">
    <Param Name=":Number" Default="2">
  </Params>

  <table name="LINES">
    <Key Name="Number">
    <Key Name="Part">
  </table>
</Query>
```

La consulta anterior debe devolver el siguiente documento XML:

```
<MyDoc>
  <YourRow>
    <col1>some data</col1>
    <col2>some data</col2>
    <Number>2</Number>
    <Part>ab-c</Part>
  </YourRow>
  <YourRow>
    <col1>some other data</col1>
    <col2>some other data</col2>
    <Number>2</Number>
    <Part>ab-c</Part>
  </YourRow>
</MyDoc>
```

Nota Si el formato de columna del documento de consulta es “AsAttributes”, col1, col2, Number y Part serían atributos de YourRow.

A continuación se muestra un documento de consulta de ejemplo para XQuery:

```
<Query>
  <Options
    OutputType="XML"
    ColumnFormat="AsElements"
    IgnoreNulls="True"
    DocumentElement="MyDoc"
    RowElement="YourRow">

  <Connection
    Url="jdbc:odbc:foodb"
    Driver="sun.jdbc.odbc.JdbcOdbcDriver"
    User="me"
    Password="ok"
    ExtendedProperties="name=value;name=value...">

  <Params>
    <Param Name=":Part" Default="'ab-c'">
    <Param Name=":Number" Default="2">
  </Params>

  <Sql Value="SELECT * FROM LINES where Number >= :Number AND Number
    <= :Number"/>
  <!--The above should use CDATA section or escape with lt/gt entity
    references -->
</Query>
```

Si desea seguir un tutorial sobre cómo utilizar los componentes XTable y XQuery, consulte el [Capítulo 10, “Tutorial: Transferir datos con componentes XML de bases de datos basados en plantillas”](#).

Tutorial: Creación y validación de documentos XML

En este tutorial se utilizan
funciones de JBuilder SE
y Enterprise

En este detallado tutorial se explica la forma de utilizar la funcionalidad XML de JBuilder para la creación y validación de documentos XML. Se facilita un ejemplo en el directorio `samples/tutorials/XML/presentation/` de JBuilder. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura. Se facilitan como ejemplos una DTD y hojas de estilo (XSL).

El tutorial contiene ejemplos específicos que muestran cómo realizar las siguientes tareas:

- Crear manualmente un documento XML en el editor de JBuilder.
- Crear un documento XML a partir de una DTD mediante un asistente de JBuilder Enterprise.
- Añadir datos al documento XML, como identificador del empleado, nombre, ubicación de la oficina, etc.
- Validar el documento XML con la DTD.
- Localizar errores en el documento XML.
- Ver el documento XML con el visor XML y la vista en árbol de hoja de estilo por defecto de JBuilder.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

JBuilder Enterprise incluye funciones de transformación adicionales para documentos XML. Si es usuario de JBuilder Enterprise, una vez completado este tutorial, es aconsejable continuar con el [Capítulo 5, “Tutorial: Transformación de documentos XML”](#) para aprender cómo transformar documentos XML en JBuilder.

El apartado Opciones de accesibilidad en las 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 1-3](#).

Paso 1: Creación de un documento XML

Existen varias formas para crear documentos XML en JBuilder. En JBuilder SE y Enterprise, se pueden crear documentos XML manualmente en el editor. JBuilder Enterprise proporciona el Asistente para pasar de DTD a XML, que permite crear rápidamente un documento XML a partir de una *Definición de tipo de documento* (DTD).

Cree el documento XML según la edición de JBuilder que posea:

- JBuilder SE: Consulte [“Creación de un documento XML manualmente” en la página 4-2](#)
- JBuilder Enterprise: Consulte [“Creación de un documento XML mediante el Asistente para pasar de DTD a XML” en la página 4-4](#)

Creación de un documento XML manualmente

Este paso se aplica a los usuarios de JBuilder SE.

El editor de JBuilder proporciona soporte completo para la creación de documentos XML. Si el nombre de un archivo incluye una extensión relacionada con XML, como DTD, XSD, XSL o XML, el editor lo reconoce automáticamente como documento XML. Varias funciones del editor le ayudan a trabajar con los documentos XML: el resaltado de la sintaxis y los mensajes de error.

Para crear un documento XML en el proyecto:

- 1 Abra `Presentation.jpx`, situado en el directorio `samples` de JBuilder: `samples/Tutorials/XML/presentation/`.
- 2 Seleccione Proyecto | Añadir archivos/paquetes.
- 3 Seleccione la pestaña Explorador, busque el directorio del proyecto, `samples/Tutorials/XML/presentation/`, y escriba `MyEmployees.xml` en el campo Archivo.
- 4 Pulse Aceptar.

- 5 Pulse **Aceptar de nuevo** cuando se le pregunte si desea crear el archivo. `MyEmployees.xml` se añade al proyecto y aparece en el panel de proyecto con el icono de XML apropiado.
- 6 Abra `MyEmployees.xml` en el editor.
- 7 Escriba el siguiente texto o cópielo y péguelo en `MyEmployees.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">
<XmlEmployees>
  <XmlEmployee>
    <EmpNo>2</EmpNo>
    <FirstName>Robert</FirstName>
    <LastName>Nelson</LastName>
    <PhoneExt>250</PhoneExt>
    <HireDate>1988-12-28</HireDate>
    <DeptNo>600</DeptNo>
    <JobCode>VP</JobCode>
    <JobGrade>2</JobGrade>
    <JobCountry>USA</JobCountry>
    <Salary>105900.000000</Salary>
    <FullName>Nelson, Robert</FullName>
  </XmlEmployee>
  <XmlEmployee>
    <EmpNo>4</EmpNo>
    <FirstName>Bruce</FirstName>
    <LastName>Young</LastName>
    <PhoneExt>233</PhoneExt>
    <HireDate>1988-12-28</HireDate>
    <DeptNo>621</DeptNo>
    <JobCode>CEO</JobCode>
    <JobGrade>2</JobGrade>
    <JobCountry>Eng</JobCountry>
    <Salary>97500.000000</Salary>
    <FullName>Young, Bruce</FullName>
  </XmlEmployee>
</XmlEmployees>
```

Observe cómo el editor resalta la sintaxis con el fin de diferenciar elementos y atributos. Por defecto, los elementos son azules y los atributos son rojos. Observe también que el elemento `DOCTYPE` hace referencia a `Employees.dtd`. `MyEmployees.xml` se basa en esta DTD y puede validarse contra ella. Abra la DTD para ver qué elementos debe contener el documento XML para ser válido.

- 8 Guarde el proyecto.

En el paso siguiente, se validará el documento contra la DTD. Proceda con el siguiente paso, [“Paso 2: Validación de los documentos XML” en la página 4-6](#).

Creación de un documento XML mediante el Asistente para pasar de DTD a XML

Este paso se aplica a los usuarios de JBuilder Enterprise.

JBuilder Enterprise proporciona el Asistente para pasar de DTD a XML, el cual permite generar documentos XML a partir de una DTD ya creada. Se utilizará el archivo `Employees.dtd` para crear un documento denominado `MyEmployees.xml`. Después, modificará el código generado utilizando datos reales.

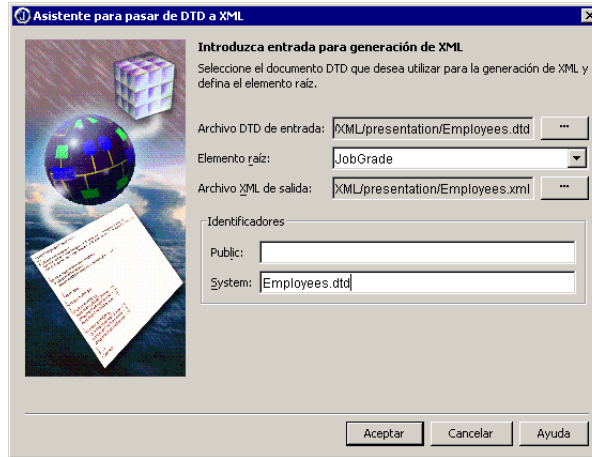
- 1 Abra `Presentation.jpx`, situado en el directorio `samples` de JBuilder: `samples/Tutorials/XML/presentation/`.
- 2 Elija `Employees.dtd` en el Archivo DTD de entrada, haga clic con el botón derecho del ratón y elija Generar XML para abrir el asistente DTD a XML. Cuando se selecciona el archivo DTD se rellena automáticamente el campo Archivo DTD de entrada del asistente. También se puede abrir el asistente desde la galería de objetos (Archivo | Nuevo | XML).
- 3 Pulse la flecha para abrir la lista desplegable Elemento raíz y elija `XmlEmployees` en la lista. Tenga cuidado de **no** elegir `XmlEmployee`, ya que éste no es el elemento raíz. El elemento raíz, el primero del documento, contiene los demás elementos. Si abre la DTD, podrá ver que el elemento raíz se define como el que contiene los restantes elementos:

```
<!ELEMENT XmlEmployees (XmlEmployee+)>
<!ELEMENT XmlEmployee (EmpNo, FirstName, LastName, PhoneExt, HireDate,
DeptNo, JobCode, JobGrade, JobCountry, Salary, FullName)>
```

- 4 Pulse el botón de puntos suspensivos contiguo al campo Archivo XML de salida y escriba `MyEmployees.xml` en lugar del nombre por defecto en el campo Nombre de archivo.
- 5 Haga clic en Aceptar para cerrar el cuadro de diálogo.
- 6 Introduzca el nombre del archivo DTD en el campo System: `Employees.dtd`. De esta forma, se genera la declaración `DOCTYPE`, que indica al documento XML que se está utilizando una DTD:

```
<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">
```

El asistente DTD a XML tiene el siguiente aspecto:



7 Pulse Aceptar para cerrar el asistente.

8 Guarde el proyecto.

El Asistente para pasar de DTD a XML genera un documento XML llamado `MyEmployees.xml` a partir de la DTD. El documento XML se abre en el editor y se añade al proyecto. Observe cómo el editor resalta la sintaxis con el fin de diferenciar elementos y atributos. Por defecto, los elementos son azules y los atributos son rojos.

El Asistente para pasar de DTD a XML genera un texto de marcadores, `pcdata`, para cada elemento de la DTD. Por ejemplo, `<EmpNo>pcdata</EmpNo>`. Es necesario sustituir este texto por los datos reales. Observe también que `Employees.dtd`, ya introducido en el campo de identificador SYSTEM del asistente DTD a XML, se ha introducido en la declaración DOCTYPE.

A continuación, se sustituirá el texto de marcadores `pcdata` por datos reales.

- 1 Cree otro registro de empleado; para ello, copie las etiquetas `<XmlEmployee>` `</XmlEmployee>` y su contenido.
- 2 Pegue la copia por debajo del primer registro.
- 3 Vaya sustituyendo los marcadores `pcdata` por los datos, de la forma que se indica aquí:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">
<XmlEmployees>
  <XmlEmployee>
    <EmpNo>2</EmpNo>
    <FirstName>Robert</FirstName>
    <LastName>Nelson</LastName>
    <PhoneExt>250</PhoneExt>
```

```
<HireDate>1988-12-28</HireDate>
<DeptNo>600</DeptNo>
<JobCode>VP</JobCode>
<JobGrade>2</JobGrade>
<JobCountry>USA</JobCountry>
<Salary>105900.000000</Salary>
<FullName>Nelson, Robert</FullName>
</XmlEmployee>
<XmlEmployee>
  <EmpNo>4</EmpNo>
  <FirstName>Bruce</FirstName>
  <LastName>Young</LastName>
  <PhoneExt>233</PhoneExt>
  <HireDate>1988-12-28</HireDate>
  <DeptNo>621</DeptNo>
  <JobCode>CEO</JobCode>
  <JobGrade>2</JobGrade>
  <JobCountry>Eng</JobCountry>
  <Salary>97500,000000</Salary>
  <FullName>Young, Bruce</FullName>
</XmlEmployee>
</XmlEmployees>
```

4 Guarde el proyecto.

En el paso siguiente, se validará el documento contra la DTD.

Paso 2: Validación de los documentos XML

En XML existen dos tipos de validación: *forma y gramática correctas*. Un documento tiene la forma correcta si su estructura y su sintaxis cumplen las normas de XML. Por ejemplo, todos los documentos XML deben tener un solo *elemento raíz*, que es el primero del documento y contiene todos los demás elementos. Los documentos bien formados no se comparan con una DTD externa.

Para ser válidos, los documentos XML deben atenerse a las normas más estrictas de las definiciones de tipo de documentos (DTD). La DTD describe la estructura del documento, establece los tipos de elementos permitidos y define sus propiedades. Si no hay una DTD, el documento XML no es válido.

JBuilder efectúa las dos validaciones. Si la forma del documento no es correcta, el panel de estructura muestra una carpeta Errores con los mensajes de error. Si la gramática del documento no es correcta, el panel de mensajes presenta los mensajes de error.

El documento creado en el paso anterior tiene la forma correcta porque en el panel de estructura no aparece la carpeta Errores. Si se elimina el elemento raíz, necesario para que la forma del documento XML sea correcta, en el panel de estructura aparece la carpeta Errores.

Ahora, introduzca un error en este documento de forma correcta para ver la forma que tiene JBuilder de mostrar los errores.

- 1 Seleccione en el editor el elemento raíz, `<XmlEmployees>` y córtelo del documento. Para estar bien formado, un documento debe tener etiquetas de inicio y fin, por lo que ahora mostrará un error. Observe que en el panel de estructura aparece una carpeta Errores.
- 2 Abra la carpeta Errores y seleccione el mensaje de error para resaltarlo en el código fuente. Haga doble clic en el error para cambiar el foco a la línea de código, en el editor. Es posible que el origen del error no se encuentre en la línea de código a la que señala el mensaje. En este ejemplo, el error tiene lugar porque falta la etiqueta inicial del elemento raíz.
- 3 Vuelva a escribir el elemento raíz en el documento XML. Verá que la carpeta Errores desaparece. El documento vuelve a tener una forma correcta.

A continuación, compruebe si la gramática del documento, comparada con la DTD, es válida.

- 1 Haga clic con el botón derecho del ratón en `MyEmployees.xml`, en el panel de proyecto, y elija Validar. Aparece un cuadro de diálogo llamado Correcto, con un mensaje que indica que el documento es válido.
- 2 Introduzca un error de validación. Para ello, seleccione la declaración `DOCTYPE` y córtela del documento:

```
<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">
```

- 3 Vuelva a efectuar la validación. La ficha Inspección de validación XML se muestra en el panel de mensajes con un nodo ERROR.
- 4 Amplíe el nodo ERROR en el panel de mensajes para mostrar los errores:

```
MYEMPLOYEES.XML is invalid
ERROR
    There is no DTD or Schema present in this document
```

- 5 Vuelva a escribir la declaración `DOCTYPE` o péguela en el documento.
- 6 Introduzca otro error; para ello, cambie la 'N' mayúscula de `<FirstName>` por una 'n' minúscula: `<Firstname>`.
- 7 Haga clic con el botón derecho del ratón en el archivo XML y seleccione Validar. Observe los mensajes de error.

```
MYEMPLOYEES.XML is invalid
ERROR
    Element type "Firstname" must be declared.
FATAL_ERROR
    The element type "Firstname" must be terminated by the matching
    end-tag "</Firstname>".
```

Aquí hay dos errores: el elemento `Firstname` no se declara en la DTD y no tiene una etiqueta de cierre.

- 8 Cambie `<Firstname>` de nuevo a `<FirstName>`.
- 9 Haga clic con el botón derecho del ratón en el archivo XML y seleccione Validar. Con ello, logrará que el documento vuelva a ser válido.

Paso 3: Presentación del documento XML

Los documentos XML se pueden ver en el Visor XML con la hoja de estilos por defecto de JBuilder. Esta hoja de estilo está escrita en XSLT y muestra los documentos en una vista en árbol que se puede ampliar y contraer, en la ficha Ver del panel de contenido. Para ver un documento XML en el Visor XML, primero debe activar el Visor XML. Una vez activado el visor XML, aparece una pestaña Ver en la parte inferior del panel de contenido.

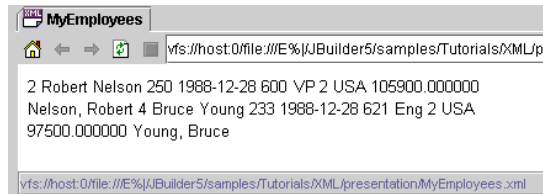
En este paso, se activará el Visor XML y, a continuación, se mostrará primero el documento XML sin la hoja de estilo por defecto de JBuilder y, después, con la hoja de estilo. En primer lugar, es necesario habilitar el Visor XML.

- 1 Seleccione Herramientas | Opciones del IDE, elija la página XML y seleccione la opción Activar vista del navegador. De este modo, el Visor XML se muestra en la pestaña Ver. De forma predefinida se aplica la opción Aplicar hoja de estilo por defecto. La hoja de estilo por defecto muestra los documentos XML en una vista en árbol que se puede ampliar y contraer.
- 2 Haga clic en Aceptar para cerrar el cuadro de diálogo Opciones del IDE. La pestaña Ver muestra `MyEmployees.xml` en el Visor XML, con la hoja de estilo por defecto aplicada. Para ampliar y contraer la vista en árbol, pulse los iconos de signo más (+) y menos (-).



A continuación, examine el documento XML sin hoja de estilo, como sigue:

- 1 Desactive la opción Aplicar hoja de estilo por defecto en la ficha XML del cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE).
- 2 Haga clic en Aceptar para cerrar el cuadro de diálogo. Si no ve ningún cambio, abra la ficha Fuente y después la ficha Ver para actualizar la vista. Observe que el documento se muestra ahora sin estilos, en una línea continua.



Enhorabuena. Ha finalizado este tutorial. En él, ha aprendido a crear un documento XML, modificarlo, validarlo y a activar el Visor XML para verlo en JBuilder.

JBuilder Enterprise también proporciona funciones de transformación adicionales. Para aprender más acerca de estas funciones, consulte el [Capítulo 5, “Tutorial: Transformación de documentos XML”](#). Si desea ver más tutoriales sobre XML, consulte [“Tutoriales”](#) en la página [v](#).

Tutorial: Transformación de documentos XML

En este tutorial se utilizan
funciones de JBuilder
Enterprise

En este detallado tutorial se explica la forma de utilizar la funcionalidad XML de JBuilder para la transformación de un documento XML. Se facilita un ejemplo en el directorio `samples/tutorials/XML/presentation/` de JBuilder. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura. Se facilitan como ejemplos una DTD y hojas de estilo (XSL).

Antes de empezar este tutorial, debe crear un documento XML según se explica en [“Paso 1: Creación de un documento XML” en la página 4-2](#).

El tutorial contiene ejemplos específicos que muestran cómo realizar las siguientes tareas:

- Activar el visor XML.
- Asociar hojas de estilo al documento XML.
- Transformar el documento XML aplicando varias hojas de estilo.
- Definir las opciones de seguimiento.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

El apartado Opciones de accesibilidad en las 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 1-3.

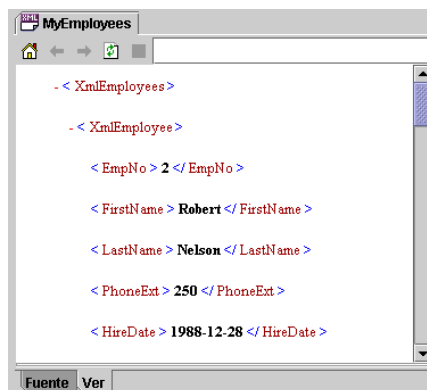
Paso 1: Activación del visor XML

Los documentos XML se pueden ver en JBuilder con una hoja de estilo definida por el usuario, con la predeterminada de JBuilder o sin hoja de estilo. Por defecto, el visor XML de JBuilder muestra documentos XML mediante una hoja de estilo predeterminada escrita en XSLT, la cual muestra XML en una vista en árbol plegable. Una vez activado el visor XML, aparece una pestaña Ver en la parte inferior del panel de contenido.

En este paso, se activará el visor XML para verla con la hoja de estilo por defecto de JBuilder.

- 1 Cree el documento XML, `MyEmployees.xml`, según se describe en “Paso 1: Creación de un documento XML” en la página 4-2.
- 2 Abra `MyEmployees.xml` en el editor.
- 3 Active el visor XML como se indica a continuación:
 - a Seleccione Herramientas | Opciones del IDE, elija la página XML y seleccione la opción Activar vista del navegador. De este modo, el Visor XML se muestra en la pestaña Ver. De forma predefinida se aplica la opción Aplicar hoja de estilo por defecto. La hoja de estilo por defecto muestra los documentos XML en una vista en árbol que se puede ampliar y contraer.
 - b Haga clic en Aceptar para cerrar el cuadro de diálogo Opciones del IDE.

Observe que `MyEmployees.xml` aparece ahora en el Visor XML, con la hoja de estilo por defecto aplicada. Para ampliar y contraer la vista en árbol, pulse los iconos de signo más (+) y menos (-).




Paso 2: Asociación de hojas de estilo al documento

Además de ver documentos en una vista en árbol, también es posible aplicar hojas de estilo personalizadas al documento XML. Este proceso de conversión de documentos XML a otro tipo de documento se denomina *transformación XML*.

Con el fin de aplicar hojas de estilo personalizadas en JBuilder es necesario asociarlas al documento. Si lo prefiere, incluya una instrucción de proceso XSLT en el documento XML que haga referencia a las hojas de estilo.


A continuación, asocie las hojas de estilo al documento como sigue:

- 1 Abra la pestaña Vista transformado. Observe que aparece un mensaje que indica que no hay ninguna hoja de estilo asociada al documento.
 - 2  Pulse el botón Añadir hojas de estilo de la barra de herramientas Vista transformado, para abrir el cuadro de diálogo Configuración de las hojas de estilo del nodo.
 - 3 Pulse el botón Añadir del cuadro de diálogo y abra el directorio `samples/Tutorials/XML/presentation/xsls`, en el que se encuentran las hojas de estilo. Seleccione `EmployeesListView.xml` y pulse Aceptar.
 - 4 Pulse de nuevo Añadir para agregar la segunda hoja de estilo, `EmployeesTableView.xml`. Ahora, las hojas de estilo XSL están asociadas al documento. Haga clic en Aceptar para cerrar el cuadro de diálogo.
- Nota** También es posible añadir hojas de estilo en el cuadro de diálogo Propiedades. Haga clic con el botón derecho del ratón en el documento XML, en el panel de proyecto, y elija Propiedades.

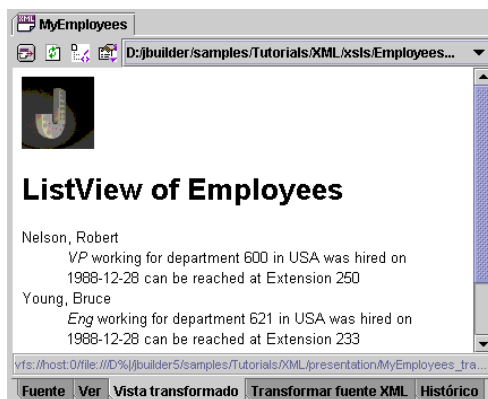
Paso 3: Transformación del documento mediante hojas de estilo

Ahora que las hojas de estilo están asociadas al documento XML es posible transformarlo mediante distintos estilos. Ahora, las hojas de estilo asociadas están disponibles en la lista desplegable de la barra de herramientas Vista transformado.

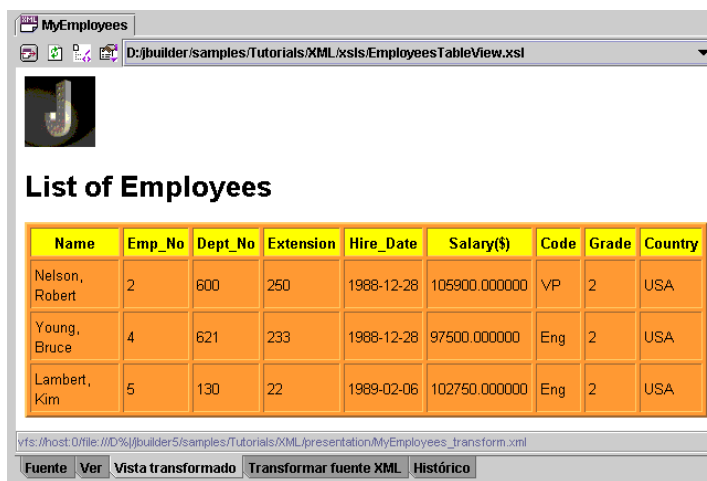
Observe que en la vista en árbol por defecto aparece `MyEmployees.xml`. De forma predeterminada, la vista transformada utiliza la hoja de estilo por defecto para presentar el documento, si no hay otra disponible. Desactive esta vista y aplique la hoja de estilo.

- 1  Pulse Aplicar hoja de estilo por defecto para desactivar la vista en árbol.
- 2 Seleccione `EmployeesListView.xml` en la lista desplegable de hojas de estilo. Ahora, Vista transformado muestra el documento transformado en forma de lista, en aplicación de la hoja de estilo. La pestaña Fuente

de vista transformado muestra el código fuente del documento transformado. El documento XML se asemejará a:



- 3 Aplique la segunda hoja de estilo. Para ello, elija `EmployeesTableView.xsl` en la lista desplegable y examine la Vista transformado. Comprobará que se ha convertido en una tabla.



Paso 4: Configuración de opciones de seguimiento

Las opciones de seguimiento se pueden configurar de forma que, cuando ocurra una transformación, se pueda seguir el flujo a medida que se aplica la hoja de estilo. Estas opciones son Generación, Plantillas, Elementos y Selecciones. Las inspecciones se muestran en el panel de mensajes. Cuando se hace clic en una inspección se resalta el código fuente correspondiente. Si se hace doble clic en una inspección, el foco pasa al código fuente en el editor, donde se puede modificar.

Para definir las opciones de seguimiento:



- 1 Pulse el botón Definir opciones de inspección de la barra de herramientas Vista transformado o elija Herramientas | Opciones del IDE y abra la pestaña XML.
- 2 Seleccione todas las opciones de seguimiento y pulse Aceptar.

Ahora, transforme `MyEmployees.xml`. Para ello, aplique `EmployeesListView.xml`. Observe lo que ocurre en el panel de mensajes.

- 1 Seleccione `EmployeesListView.xml` en la lista desplegable de hojas de estilo. Observe que, cuando tiene lugar la transformación, se abre el panel de mensajes y aparecen cuatro nodos: generación, plantillas, elementos y selecciones.
 - generación: muestra información después de cada suceso de generación de árbol de resultado, como start document, start element, characters, etc.
 - plantillas: muestra un suceso cuando se llama a una plantilla.
 - elementos: muestra sucesos que ocurren a medida que se ejecutan los nodos en la hoja de estilo.
 - selecciones: muestra información después de cada suceso de selección.
- 2 Amplíe los nodos para mostrar el flujo de transformación del documento.

Enhorabuena. Ha finalizado este tutorial. En él, ha aprendido a activar el visor XML, asociar hojas de estilo con el documento, transformar el documento mediante hojas de estilo y configurar opciones de seguimiento de las transformaciones.

Si desea seguir más tutoriales sobre XML, consulte [“Tutoriales”](#) en la página [v](#).

Tutorial: Creación de un gestor de SAX para analizar documentos XML

En este tutorial se utilizan
funciones de JBuilder
Enterprise

En este detallado tutorial se explica la forma de utilizar el Asistente para el gestor de SAX de JBuilder para analizar documentos XML. Se facilitan ejemplos en el directorio `samples/Tutorials/XML/saxparser` de JBuilder. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura. En este tutorial se utiliza como ejemplo un documento XML que contiene datos de empleados, como el número de empleado, el nombre, el apellido, etc.

Existen dos tipos de API XML: API basadas en árboles y API basadas en sucesos. SAX, la API sencilla para XML, es una interfaz estándar para el análisis XML basado en sucesos. Informa sobre los sucesos de análisis directamente a la aplicación, por medio de devoluciones de llamada. La aplicación implementa manejadores que gestionan los distintos sucesos, de forma parecida a la gestión de sucesos de la interfaz de usuario.

JBuilder facilita la utilización de SAX para la manipulación del programa de XML. El asistente para la gestión de SAX crea una plantilla de implementación de analizador SAX que incluye los métodos que se desean implementar para analizar el documento XML.

El tutorial contiene ejemplos específicos que muestran cómo realizar las siguientes tareas:

- Creación de un analizador SAX con el Asistente para el gestor de SAX.
- Modificación del código del analizador SAX con el fin de personalizar el análisis.

- Ejecución del programa y examen del resultado.
- Añadir atributos al documento XML, introducir el código necesario para gestionarlos y volver a analizar el documento.

Para ver el código fuente de `MySaxParser.java`, consulte [“Código fuente de MySaxParser.java” en la página 6-10](#).

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte [“El entorno de JBuilder”](#). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción.”](#)

Consulte

- SAX (Simple API for XML), en <http://www.saxproject.org/>.
- Los paquetes `org.xml.sax`, `org.xml.sax.ext` y `org.xml.sax.helpers` en la Especificación API de java (Ayuda | Referencia de Java)
- La documentación y los ejemplos sobre Xerces, disponibles en el directorio `extras` de la instalación completa de JBuilder
- Xerces en la sede web de Apache, en <http://xml.apache.org/>

El apartado Opciones de accesibilidad en las 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 1-3](#).

Paso 1: El asistente para la gestión de SAX

El asistente para la gestión de SAX de JBuilder ayuda a crear un analizador SAX con el objeto de efectuar un análisis personalizado de los documentos XML con el motor de análisis Xerces.

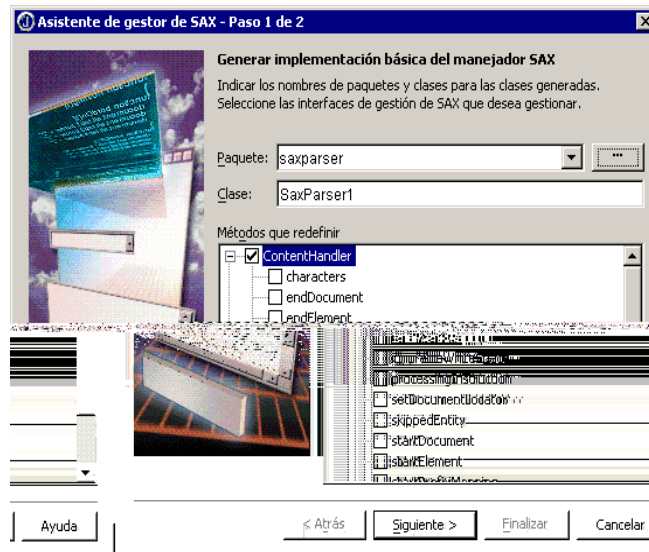
Para crear un analizador con el Asistente para la gestión de SAX:

- 1 Abra el archivo de proyecto, `SAXParser.jpx`, que se encuentra en el subdirectorio `samples/Tutorials/XML/saxparser/` del directorio de JBuilder.
- 2 Abra `Employees.xml` y examine los datos del documento XML. Observe que hay tres empleados, y que sus registros contienen datos como el número de empleado, el nombre, el apellido y el nombre completo.
- 3 Para abrir la galería de objetos, elija Archivo | Nuevo o pulse el botón Nuevo de la barra de herramientas principal.



- 4 Para iniciar el asistente, abra la ficha XML y haga doble clic en el icono Gestor de SAX.
- 5 Efectúe los siguientes cambios en los nombres de clase y paquete:
 - Paquete: `com.borland.samples.xml.saxparser`
 - Nombre de clase: `MySaxParser`
- 6 Elija `ContentHandler` como interfaz que se va a redefinir y amplíe el nodo `ContentHandler`. Active estas cinco opciones para crear métodos: `characters`, `endDocument`, `endElement`, `startDocument` y `startElement`.

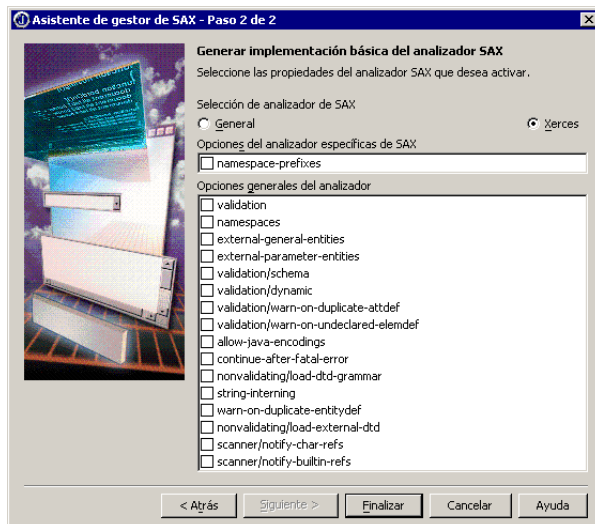
El Paso 1 debería tener este aspecto.



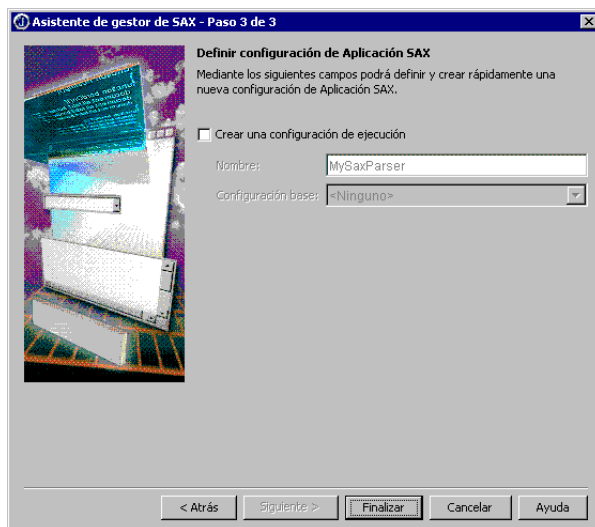
- 7 Pulse **Siguiete** para pasar al Paso 2, que enumera los analizadores disponibles y sus opciones. En este tutorial, se aceptará el analizador Xerces por defecto, y no será necesario seleccionar ninguna de sus opciones. Si desea obtener más información sobre estas opciones, pulse el botón **Ayuda** del asistente.

Paso 1: El asistente para la gestión de SAX

El Paso 2 tiene este aspecto.



- 8 Haga clic en Siguiente para avanzar al último paso del asistente. Esta ficha crea una configuración de ejecución para el manejador de SAX. El proyecto ya tiene una configuración de ejecución, por lo tanto no necesita crear otra.



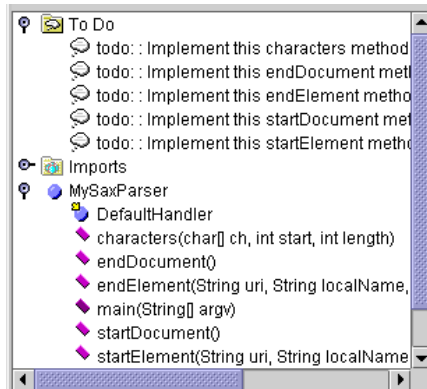
- 9 Pulse Finalizar para cerrar el asistente.

El asistente genera un archivo de análisis llamado `MySaxParser.java`, lo añade al proyecto y lo abre en el editor. Examine este archivo y verá que el asistente ha generado métodos vacíos, que es necesario completar.

Sugerencia Para localizar una de las clases importadas de este archivo, abra la carpeta Importaciones en el panel de estructura. Haga doble clic en un paquete para abrir el cuadro de diálogo Buscar símbolo Import y buscar la clase deseada. Para ver la documentación disponible, seleccione la pestaña Doc en el panel de contenido.

Paso 2: Modificación de analizadores SAX

El asistente genera métodos vacíos que es necesario implementar. Observe que la estructura `MySaxParser.java` se ve en el panel de estructura a la izquierda del editor, y que la carpeta Por Hacer contiene cinco métodos que se deben implementar: `characters()`, `endDocument()`, `endElement()`, `startDocument()` y `startElement()`.



Examine el bloque `try` del método `main()` que ha generado el asistente:

```
try {
    SAXParserFactory parserFactory = SAXParserFactory.newInstance();
    parserFactory.setValidating(false);
    parserFactory.setNamespaceAware(false);
    MySaxParser MySaxParserInstance = new MySaxParser();
    SAXParser parser = parserFactory.newSAXParser();
    parser.parse(uri, MySaxParserInstance);
}
```

Este bloque de código crea una instancia de un analizador y, a continuación, pasa el archivo XML especificado en el URI al analizador para que lo analice. El archivo XML se especificará en la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto más adelante en este mismo tutorial.

En primer lugar, añade sentencias `print` a los métodos `startDocument()` y `endDocument()` que muestran en la pantalla los mensajes de análisis inicial y final.

- 1 Añade una sentencia `print` al método `startDocument()`:


```
System.out.println("PARSING begins...");
```
- Sugerencia** Haga doble clic en un método del panel de estructura o en la carpeta Por Hacer, para desplazar el cursor a este método en el editor.
- 2 Añade una sentencia `print` al método `endDocument()`:


```
System.out.println("...PARSING ends");
```
- 3 Elimine las sentencias `throw` y los comentarios `@todo` de los métodos, ya que no se necesitan.
- 4 Cree una variable para identificar la salida analizada y declárela justo antes del método `characters()`:


```
private int idx = 0; //indent
public void characters(char[] ch, int start, int length) throws
    SAXException {
```
- 5 Cree una constante `INDENT` con el valor 2 justo antes del método `main()`.


```
private static int INDENT = 2;

public static void main(String[] argv) {
```
- 6 Cree un método `getIndent()` al final de la clase `MySaxParser`, después del método `startElement()`. Este método coloca una sangría en la salida del análisis, para que resulte más fácil de leer.


```
private String getIndent() {
    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < idx; i++)
        sb.append(" ");
    return sb.toString();
}
```
- 7 Añade el siguiente código marcado en negrita a cada uno de los métodos, para aplicar la sangría a la salida:


```
public void characters(char[] ch, int start, int length)
    throws SAXException {
    //instancia s, sangra la salida, imprime valores de carácter en el
    elemento
    String s = new String(ch, start, length);
    if (!s.startsWith("\n"))
        System.out.println(getIndent() + " Value: " + s);
}
public void endDocument() throws SAXException {
    idx-= INDENT;
    System.out.println(getIndent() + "end document");
    System.out.println("...PARSING ends");
}
```



```

public void endElement(String uri, String localName, String qName)
    throws SAXException {
    System.out.println(getIndent() + "end element");
    idx -= INDENT;
}

public void startDocument() throws SAXException {
    idx += INDENT;
    System.out.println("PARSING begins...");
    System.out.println(getIndent() + "start document: ");
}

public void startElement(String uri, String localName, String qName,
    Attributes attributes) throws SAXException {
    idx += INDENT;
    System.out.println('\n' + getIndent() + "start element: " + qName);
}

```

Sugerencia

Buscar símbolo, del editor, permite buscar clases, interfaces, sucesos, métodos, propiedades e identificadores, para obtener más información sobre ellos. Sitúe el cursor sobre uno de estos nombres, pulse el botón derecho y seleccione Buscar definición. Con el fin de que una clase se encuentre de forma automática, debe estar en la vía de acceso import. Los resultados de la búsqueda aparecen en el panel de contenido del Visualizador de aplicaciones. También es posible buscar clases en el editor, desde el menú Buscar (Buscar | Buscar clases).

8 Guarde el proyecto.

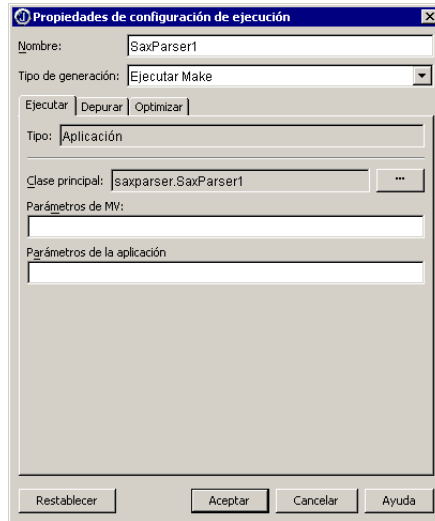
Paso 3: Ejecución del programa

Antes de ejecutar el programa, es necesario indicar la vía del documento XML como parámetro de ejecución, de forma que la aplicación de análisis identifique el archivo que debe analizar. Para ello, ha de abrir el cuadro de diálogo Propiedades de configuración de ejecución.

- 1** Seleccione Ejecutar | Configuraciones para mostrar la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto. Aquí puede ver la configuración de ejecución del proyecto existente, denominada Default SaxParser.
- 2** Seleccione Default SaxParser en la lista pulse Modificar para modificar los parámetros de la aplicación.
- 3** Modifique la ubicación de JBuilder en la vía de acceso del documento `Employees.xml` en el campo Parámetros de la aplicación. Por ejemplo:

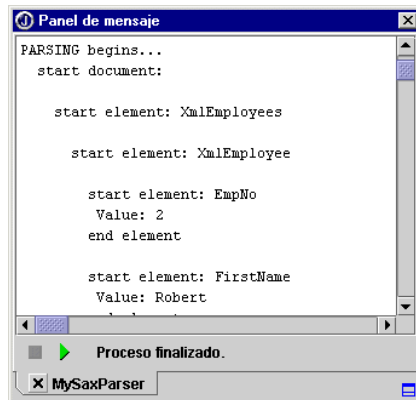
Paso 3: Ejecución del programa

file:///C:/jbuilder8/samples/Tutorials/XML/saxparser/Employees.xml



- 4 Haga dos veces clic en Aceptar para cerrar los cuadros de diálogo.
- 5 Haga clic con el botón derecho del ratón en `MySaxParser.java` en el panel de proyecto y pulse Ejecutar utilizando "Default SaxParser".

Se abre el panel de mensaje, que muestra la salida del análisis.



Paso 4: Adición de atributos

A continuación, añade atributos al documento XML. Los atributos sirven para definir los elementos con más detalle. Después se puede añadir código al analizador, para que pueda gestionar los atributos.

1 Cambie a `Employees.xml` en el editor.

2 Añada un atributo al primer elemento `EmpNo` de `Employees.xml`:

```
<EmpNo att1="a" att2="b">2</EmpNo>
```

3 Añada atributos al primer elemento `FirstName` del documento XML:

```
<FirstName z="z1" d="d1" k="k1">Robert</FirstName>
```

4 Cambie a `MySaxParser.java` en el editor.

5 Añada la variable `attList` justo encima del método `main()`:

```
public class MySaxParser extends DefaultHandler {

    private static int INDENT = 2;
    private static String attList = "" ;
    public static void main(String[] argv) {
```

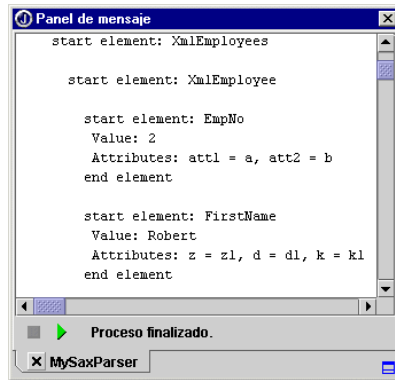
6 Añada el siguiente código de gestión del atributo al método `startElement()`:

```
public void startElement(String uri, String localName, String qName,
    Attributes attributes) throws SAXException {
    idx += INDENT;
    System.out.println('\n' + getIndent() + "start element: " +
        qName);
    if (attributes.getLength() > 0) {
        idx += INDENT;
        for (int i = 0; i < attributes.getLength(); i++) {
            attList = attList + attributes.getQName(i) + " = " +
                attributes.getValue(i);
            if (i < (attributes.getLength() - 1))
                attList = attList + ", ";
        }
        idx -= INDENT;
    }
}
```

7 Añada el siguiente código al método `endElement()`:

```
public void endElement(String uri, String localName, String qName)
    throws SAXException {
    if (!attList.equals(""))
        System.out.println(getIndent() + " Attributes: " + attList);
    attList = "";
    System.out.println(getIndent() + "end element");
    idx -= INDENT;
}
```

- 8 Guarde el proyecto y vuelva a ejecutar el programa. Observe que, ahora, la salida del análisis incluye los atributos.



Enhorabuena. Ha finalizado este tutorial. Ha creado un analizador SAX con el Asistente de gestor de SAX, ha modificado el código del analizador para personalizar el análisis, ha añadido atributos al documento XML y por último lo ha analizado.

Si desea seguir más tutoriales sobre XML, consulte [“Tutoriales”](#) en la página [v](#).

Código fuente de MySaxParser.java

El código fuente completo de MySaxParser.java después de completar el tutorial es el siguiente:

```
package com.borland.samples.xml.saxparser;

import java.io.IOException;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;

public class MySaxParser extends DefaultHandler {

    private static int INDENT = 2;
    private static String attList = "" ;

    public static void main(String[] argv) {
        if (argv.length != 1) {
            System.out.println("Usage: java MySaxParser [URI]");
            System.exit(0);
        }
        System.setProperty("javax.xml.parsers.SAXParserFactory",
            "org.apache.xerces.jaxp.SAXParserFactoryImpl");
        String uri = argv[0];
        try {
```

```

        SAXParserFactory parserFactory = SAXParserFactory.newInstance();
        parserFactory.setValidating(false);
        parserFactory.setNamespaceAware(false);
        MySaxParser MySaxParserInstance = new MySaxParser();
        SAXParser parser = parserFactory.newSAXParser();
        parser.parse(uri, MySaxParserInstance);
    }
    catch(IOException ex) {
        ex.printStackTrace();
    }
    catch(SAXException ex) {
        ex.printStackTrace();
    }
    catch(ParserConfigurationException ex) {
        ex.printStackTrace();
    }
    catch(FactoryConfigurationError ex) {
        ex.printStackTrace();
    }
}

private int idx = 0;

public void characters(char[] ch, int start, int length) throws
    SAXException {
    String s = new String(ch, start, length);
    if (!s.startsWith("\n"))
        System.out.println(getIndent() + " Value: " + s);
}

public void endDocument() throws SAXException {
    idx-= INDENT;
    System.out.println(getIndent() + "end document");
    System.out.println("...PARSING ends");
}

public void endElement(String uri, String localName, String qName)
    throws SAXException {
    if (!attList.equals(""))
        System.out.println(getIndent() + " Attributes: " + attList);
    attList = "";
    System.out.println(getIndent() + "end element");
    idx-= INDENT;
}

public void startDocument() throws SAXException {
    idx += INDENT;
    System.out.println("PARSING begins...");
    System.out.println(getIndent() + "start document: ");
}

public void startElement(String uri, String localName, String qName,
    Attributes attributes) throws SAXException {
    idx += INDENT;
    System.out.println('\n' + getIndent() + "start element: " + qName);
    if (attributes.getLength() > 0) {
        idx += INDENT;
        for (int i = 0; i < attributes.getLength(); i++) {

```

Código fuente de MySaxParser.java

```
        attList = attList + attributes.getQName(i) + " = " +  
            attributes.getValue(i);  
        if (i < (attributes.getLength() - 1))  
            attList = attList + ", ";  
    }  
    idx-= INDENT;  
}  
  
private String getIndent() {  
    StringBuffer sb = new StringBuffer();  
    for (int i = 0; i < idx; i++)  
        sb.append(" ");  
    return sb.toString();  
}  
}
```

Tutorial: Asociación de datos DTD con BorlandXML

En este tutorial se utilizan
funciones de JBuilder
Enterprise

En este detallado tutorial se explica la forma de aprovechar las funciones de asociación de datos XML de JBuilder usando DTD y BorlandXML para generar clases Java. El ejemplo se encuentra en el directorio `samples` de JBuilder: `samples/Tutorials/XML/databinding/fromDTD/`. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura. En este tutorial se utiliza como ejemplo registros de empleados con campos tales como el número de empleado, el nombre, el apellido, etc. Se facilitan como ejemplos un documento XML y una DTD, así como una aplicación de prueba para la manipulación de los datos.

La asociación de datos constituye una forma de acceder a datos, manipularlos y, una vez revisados, devolverlos a la base de datos o mostrarlos en un documento XML. El documento XML se puede utilizar como mecanismo de transferencia entre la base de datos y la aplicación. Para realizar esta transferencia, se asocia objetos Java a un documento XML. Para implementar la asociación de datos se generan clases Java que representan las restricciones que contiene la gramática, tales como los esquemas DTD y XML. Después, se pueden utilizar estas clases para crear y leer documentos XML acordes con la gramática, y validar documentos XML con la gramática.

El tutorial contiene ejemplos específicos que muestran cómo realizar las siguientes tareas:

- Generar clases Java a partir de una DTD con BorlandXML.
- Desmontar (unmarshal) los datos de los objetos XML y convertirlos en objetos Java.

- Modificar los datos, añadiendo un registro de empleado y modificando el nombre de otro.
- Montar (marshal) los objetos Java en el documento XML.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

El apartado Opciones de accesibilidad en las 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 1-3.

Paso 1: Generación de clases Java a partir de una DTD

El primer paso para trabajar con los datos consiste en generar clases Java a partir de la DTD con el Asistente para la asociación de datos. Si se elige BorlandXML como tipo de asociación de datos, el Asistente para la asociación de datos examina la DTD y crea una clase Java para cada uno de sus elementos.

Para generar clases Java a partir de una DTD con el Asistente para la asociación de datos:

- 1 Abra el archivo de proyecto, `BorlandXML.jpx`, que se encuentra en el directorio `samples/Tutorials/XML/databinding/fromDTD` de JBuilder.
- 2 Abra `Employees.xml` y examine los datos del documento XML. Observe que hay tres empleados. Robert Nelson, Bruce Young y Kim Lambert. Los registros contienen datos tales como el número de empleado, el nombre, el apellido y el nombre completo. Éstos son los datos que se van a manipular.

Nota También se puede examinar el documento XML en el Visor XML. Active la vista de navegador en la ficha XML del cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE). Abra la ficha Vista en el panel de contenido para ver el documento en la vista en árbol por defecto.

- 3 Abra `Employees.dtd` y examine los elementos del documento XML. `XmlEmployee`, `EmpNo`, `FirstName`, etc. El Asistente para la asociación de datos genera una clase Java para cada uno de estos elementos.
- 4 Haga clic con el botón derecho del ratón en `Employees.dtd` y elija Generar Java para abrir el asistente. Observe que el tipo de asociación de datos

seleccionado es BorlandXML. BorlandXML genera clases Java a partir de DTD.

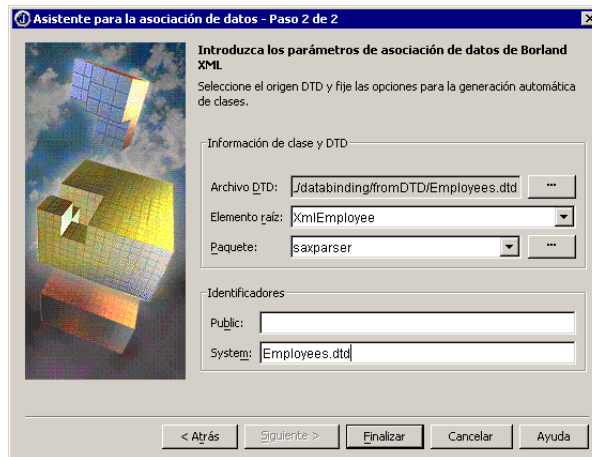
Nota El Asistente para la asociación de datos también se puede abrir desde la ficha XML de la galería de objetos (Archivo | Nuevo).

5 Pulse Siguiente para pasar al Paso 2.

6 Rellene los campos siguientes de la segunda ficha del asistente:

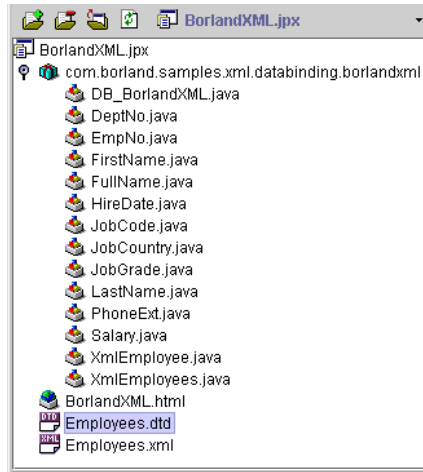
- **Archivo DTD:** acepte la vía de acceso al archivo DTD: `<jbuilder>/samples/Tutorials/XML/databinding/fromDTD/Employees.dtd`. Este campo se rellena automáticamente porque se ha seleccionado el archivo DTD en el panel de proyecto antes de abrir el asistente.
- **Elemento raíz:** Seleccione un `XmlEmployees` de la lista desplegable. Asegúrese de seleccionar `XmlEmployees` y no el elemento que no tiene una ese al final, `XmlEmployee`.
- **Paquete:** Cambie el nombre del paquete a `com.borland.samples.xml.databinding.borlandxml`
- **Identificador de sistema:** Escriba `Employees.dtd` como Identificador de sistema.

Ahora, éste es el aspecto del Asistente para la asociación de datos:



7 Pulse el botón Finalizar.

8 Amplíe el nodo de paquete de código fuente automático, `com.borland.samples.xml.databinding.borlandxml` en el panel de proyecto para ver los archivos `.java` generados por el asistente. Observe que cada elemento de la DTD tiene su propia clase. El nodo de paquetes también incluye la aplicación de pruebas, `DB_BorlandXML.java`, que forma parte del ejemplo. La aplicación de pruebas se va a utilizar para manipular los datos.



9 Seleccione Proyecto | Ejecutar Make del proyecto para compilar las clases.

10 Guarde el proyecto.

Antes de seguir, examine algunas de las clases generadas.

- 1 Abra `EmpNo.java` y examine el código. Observe que hay un constructor que crea un objeto `EmpNo` a partir del elemento `EmpNo`, y métodos para desmontar (`unmarshal`) el elemento `EmpNo` en el objeto `EmpNo` y obtener el nombre de etiqueta del elemento.
- 2 Abra `XmlEmployee.java`: El elemento `XmlEmployee` del documento XML contiene todos los registros de esta persona, tales como `EmpNo` (número de empleado), `FirstName` (nombre) y `LastName` (apellido). En esta clase, hay un constructor que crea un objeto `XmlEmployee` a partir del elemento `XmlEmployee`, declaraciones que definen los elementos y métodos que definen y obtienen los elementos contenidos en `XmlEmployee`. Además, el método `unmarshal()` lee los objetos XML y los convierte en objetos Java. A continuación, el método `marshal()` vuelve a convertir los objetos Java en objetos XML después de manipularlos en la aplicación Java.
- 3 Abra `XmlEmployees.java`. La clase `XmlEmployees` representa el elemento raíz del documento, `XmlEmployees`. Esta clase tiene métodos que definen y obtienen el elemento `XmlEmployee`, además de métodos que añaden y eliminan empleados, obtienen y definen los identificadores `PUBLIC` y `SYSTEM`, y montan y desmontan los datos.

Sugerencia Desde el editor, puede utilizar Buscar definición, para buscar clases, interfaces, sucesos, métodos, propiedades, paquetes e identificadores. Sitúe el cursor sobre uno de estos nombres, pulse el botón derecho y seleccione Buscar definición. Con el fin de que una clase se encuentre de

forma automática, debe estar en la vía de acceso import. Los resultados de la búsqueda aparecen en el panel de contenido del Visualizador de aplicaciones. También es posible buscar clases en el editor, desde el menú Buscar (Buscar | Buscar clases).

Paso 2: Desmontaje de los datos

Ahora que se han creado los objetos Java a partir de los objetos XML, examine la aplicación de prueba, `DB_BorlandXML.java`. Esta aplicación pasa los datos entre el documento XML y los objetos Java. Utiliza el marco de montaje para gestionar la conversión entre Java y XML. En primer lugar, los datos se desmontan y se convierten de XML a Java. A continuación se vuelven a montar y se convierten de Java a XML.

- 1 Haga doble clic en `DB_BorlandXML.java`, en el panel de proyecto, para abrirlo en el editor. Observe que en el método `main()` de la aplicación está la variable de clase `db_BorlandXML`, que llama a distintos métodos. Tres de ellos se han excluido por medio de signos de comentario. Las llamadas a estos métodos se implementarán más adelante.

```
public class DB_BorlandXML {

    public DB_BorlandXML() {
    }

    public static void main(String[] args) {

        db_BorlandXML = new DB_BorlandXML();
        db_BorlandXML.readEmployees();

        // db_BorlandXML.addEmployee();
        // db_BorlandXML.modifyEmployee();
        // db_BorlandXML.readEmployees();
    }
    ....
}
```

En el próximo paso se ejecuta la aplicación sin modificar el código. La aplicación lee los empleados del documento XML y los convierte en objetos Java. Más adelante, se manipularán los datos mediante la modificación del código, para que la aplicación pueda añadir y modificar empleados. El primer paso consiste en leer los empleados del documento XML.

- 2 Ejecute la aplicación; para ello, pulse con el botón derecho del ratón en `DB_BorlandXML.java` en el panel de proyecto y seleccione Ejecutar utilizando “BorlandXML”. La aplicación se ejecuta, lee la información sobre el empleado y muestra lo siguiente en el panel de mensajes:

```
== unmarshalling "Employees.xml" ==
Total Number of Employees read = 3
```

First Employee's Full Name is Nelson, Robert
Last Employee's Full Name is Lambert, Kim

Paso 3: Adición de un registro de empleado

En este paso se añade un registro de empleado y se montan los datos en el documento XML. Para esto, es necesario eliminar las marcas de comentario que llama al método `addEmployee()`.

También se añade otra llamada al método `readEmployees()`, que lee los nuevos datos tras añadir al empleado. Después, al ejecutar el programa, se añade el empleado Charlie Chaplin con el método `addEmployee()`.

- 1 Elimine los comentarios de la llamada al método:

```
db_BorlandXML.addEmployee();
```

- 2 Añada otra llamada al método `readEmployees()` justo debajo de la línea cuyas marcas de comentario acaba de retirar. Su código debería tener este aspecto:

```
public static void main(String[] args) {  
  
    db_BorlandXML = new DB_BorlandXML();  
    db_BorlandXML.readEmployees();  
  
    db_BorlandXML.addEmployee();  
    db_BorlandXML.readEmployees();  
    // db_BorlandXML.modifyEmployee();  
    // db_BorlandXML.readEmployees();  
}
```

Observe los métodos `addEmployee()` y `readEmployees()` para familiarizarse con su comportamiento.

- 3 Guarde el proyecto.
- 4 Ejecute de nuevo la aplicación. Observe el panel de mensajes.

```
== unmarshalling "Employees.xml" ==  
Total Number of Employees read = 3  
First Employee's Full Name is Nelson, Robert  
Last Employee's Full Name is Lambert, Kim  
== unmarshalling "Employees.xml" ==  
Total Number of Employees read = 4  
First Employee's Full Name is Nelson, Robert  
Last Employee's Full Name is Chaplin, Charlie
```

- 5 Cambie a `Employees.xml` y compruebe que se ha añadido al empleado Charlie Chaplin en cuarto lugar.

Paso 4: Modificación de un registro de empleado

Ahora se va a modificar el nombre de Charlie Chaplin. Para esto, se deben volver a anteponer marcas de comentario a las llamadas a los métodos `addEmployee()` y `readEmployees()` y retirar las de `modifyEmployee()` y `readEmployees()`.

- 1 Vuelva a `DB_BorlandXML.java` y anteponga signos de comentario a estas dos líneas:

```
// db_BorlandXML.addEmployee();
// db_BorlandXML.readEmployees();
```

- 2 Elimine las marcas de comentario de estas dos líneas:

```
db_BorlandXML.modifyEmployee();
db_BorlandXML.readEmployees();
```

Su código debería tener este aspecto:

```
public static void main(String[] args) {

    db_BorlandXML = new DB_BorlandXML();
    db_BorlandXML.readEmployees();

    // db_BorlandXML.addEmployee();
    // db_BorlandXML.readEmployees();
    db_BorlandXML.modifyEmployee();
    db_BorlandXML.readEmployees();
}
```

Ahora que se han retirado las marcas de comentario de la llamada al método `modifyEmployee()`, cuando se vuelva a ejecutar el programa, el nombre de Charlie Chaplin será sustituido por la información del método `modifyEmployee()`. Examine el método `modifyEmployee()` para observar su funcionamiento.

- 1 Haga clic con el botón derecho del ratón en `DB_BorlandXML.java`, en el panel de proyecto, y elija Ejecutar. Examine el contenido del panel de mensajes y verá que Charlie se ha sustituido por Andy Scott.

```
== unmarshalling "Employees.xml" ==
Total Number of Employees read = 4
First Employee's Full Name is Nelson, Robert
Last Employee's Full Name is Chaplin, Charlie
== unmarshalling "Employees.xml" ==
Total Number of Employees read = 4
First Employee's Full Name is Nelson, Robert
Last Employee's Full Name is Scott, Andy
```

- 2 Vuelva a `Employees.xml` y observe que los datos de Andy Scott se han montado (marshal) desde los objetos Java a los objetos XML y se han vuelto a escribir en el documento XML. De esta forma, Andy Scott sustituye a Charlie Chaplin.

Paso 5: Ejecución de la aplicación completa

Ahora que ya se ha explicado lo que hace la aplicación, elimine los datos nuevos del documento XML y los comentarios del programa, añada una sentencia print para leer el nuevo empleado y ejecute el programa como se explica en estos pasos.

- 1 Quite los datos de empleado y las etiquetas XML de Andy Scott que se muestran aquí (tenga cuidado de no eliminar otros datos o etiquetas XML):

```
<XmlEmployee>
    <EmpNo>9000</EmpNo>
    <FirstName>Andy</FirstName>
    <LastName>Scott</LastName>
    <PhoneExt>1993</PhoneExt>
    <HireDate>2/2/2001</HireDate>
    <DeptNo>600</DeptNo>
    <JobCode>VP</JobCode>
    <JobGrade>3</JobGrade>
    <JobCountry>USA</JobCountry>
    <Salary>145000,00</Salary>
    <FullName>Scott, Andy</FullName>
</XmlEmployee>
```

Ahora, el documento XML sólo contiene los tres registros de empleados originales.

- 2 Vuelva a DB_BorlandXML.java y elimine las marcas de comentario de todas las llamadas a métodos de las variables de clase. Su código debería tener este aspecto:

```
public static void main(String[] args) {

    db_BorlandXML = new DB_BorlandXML();
    db_BorlandXML.readEmployees();
    db_BorlandXML.addEmployee();
    db_BorlandXML.readEmployees();
    db_BorlandXML.modifyEmployee();
    db_BorlandXML.readEmployees();
}
```

- 3 Guarde el proyecto.
- 4 Ejecute el programa para mostrar los datos y examine el contenido del panel de mensajes:

```
== unmarshalling "Employees.xml" ==
Total Number of Employees read = 3
First Employee's Full Name is Nelson, Robert
Last Employee's Full Name is Lambert, Kim
== unmarshalling "Employees.xml" ==
Total Number of Employees read = 4
First Employee's Full Name is Nelson, Robert
Last Employee's Full Name is Chaplin, Charlie
```

```
== unmarshalling "Employees.xml" ==  
Total Number of Employees read = 4  
First Employee's Full Name is Nelson, Robert  
Last Employee's Full Name is Scott, Andy
```

- 5 Cambie a `Employees.xml` y compruebe que los datos se han montado (marshal) en el documento XML. Observe que Charlie Chaplin, el nuevo empleado, ha sido sustituido por Andy Scott.

¡Enhorabuena! Ha finalizado este tutorial. Ya ha leído, añadido y modificado los datos de los empleados de un documento XML con una aplicación Java y clases Java generadas por el Asistente para la asociación de datos a partir de una DTD.

Si desea seguir más tutoriales sobre XML, consulte [“Tutoriales”](#) en la página [v](#).

Tutorial: Asociación de datos de esquema con Castor

En este tutorial se utilizan
funciones de JBuilder
Enterprise

En este detallado tutorial se explica la forma de aprovechar las funciones de asociación de datos XML de JBuilder usando esquemas y Castor para generar clases Java. El ejemplo se encuentra en el directorio `samples` de JBuilder: `samples/Tutorials/XML/databinding/fromSchema`. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura. En este tutorial se utiliza como ejemplo registros de empleados con campos tales como el número de empleado, el nombre, el apellido, etc. Se facilitan como ejemplos un documento XML y un archivo de configuración (XSD), así como una aplicación de prueba para la manipulación de los datos.

La asociación de datos constituye una forma de acceder a datos, manipularlos y, una vez revisados, devolverlos a la base de datos o mostrarlos en un documento XML. El documento XML se puede utilizar como mecanismo de transferencia entre la base de datos y la aplicación. Para realizar esta transferencia se asocia un objeto Java a un documento XML. Para implementar la asociación de datos se generan clases Java que representan las restricciones que contiene la gramática, tales como los esquemas DTD y XML. Después, se pueden utilizar estas clases para crear y leer documentos XML acordes con la gramática, y validar documentos XML con la gramática.

El tutorial contiene ejemplos específicos que muestran cómo realizar las siguientes tareas:

- Generar clases Java a partir de un archivo de esquema usando Castor y el Asistente para la asociación de datos:

- Desmontar (unmarshal) los datos de los objetos XML y convertirlos en objetos Java.
- Modificar los datos, añadiendo un registro de empleado y modificando otro.
- Montar (marshal) los objetos Java en el documento XML.

Si desea más información sobre Castor, consulte la documentación en el directorio `extras` de la instalación completa de JBuilder o en la sede web de Castor, www.castor.org.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

El apartado Opciones de accesibilidad en las 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 1-3](#).

Paso 1: Generación de clases Java a partir de un esquema

El primer paso para trabajar con los datos consiste en generar clases Java a partir del archivo de esquema. Si se elige Castor como tipo de asociación de datos, el Asistente para la asociación de datos examina el archivo de esquema seleccionado y crea clases Java basadas en él.

Para generar clases Java a partir de un esquema con el Asistente para la asociación de datos:

- 1 Abra el archivo de proyecto, `castor.jpx`, que se encuentra en el subdirectorio `samples/Tutorials/XML/databinding/fromSchema` del directorio de JBuilder.
- 2 Abra `Employees.xml` y examine los datos del documento XML. Observe que hay tres empleados. Robert Nelson, Bruce Young y Kim Lambert. Los registros contienen datos tales como el número de empleado, el nombre, el apellido y el nombre completo. Éstos son los datos que se van a manipular.

Nota También se puede examinar el documento XML en el Visor XML. Para activar el Visor XML, seleccione Herramientas | Opciones del IDE y abra la pestaña XML. En Opciones generales, seleccione la opción Activar vista del navegador. De forma predefinida se habilita la opción Aplicar hoja de estilo por defecto, que muestra los documentos XML en

una vista en árbol. Ahora, abra la ficha Vista en el panel de contenido para ver el documento en la vista de árbol.

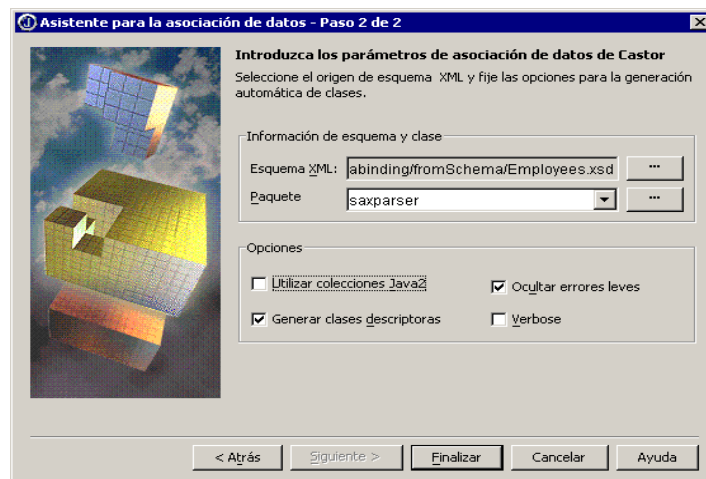
- 3 Abra `Employees.xsd` y examine el archivo. El Asistente para asociación de datos genera clases Java acordes con el archivo de esquema (XSD).
- 4 Haga clic con el botón derecho del ratón en `Employees.xsd`, en el panel de proyecto, elija Generar Java y se abrirá el asistente.
- 5 Acepte Castor como tipo de asociación de datos y pulse Siguiente para abrir la segunda ficha.

Castor utiliza esquemas (XSD) para crear clases Java. Los esquemas, más resistentes y flexibles que las DTD, tienen varias ventajas respecto a éstas. Los esquemas son documentos XML. Por contra, las DTD contienen sintaxis ajena a XML. Además, los esquemas aceptan los espacios en los nombres (necesarios para evitar los conflictos de denominación), ofrecen tipos de datos más extensos y aceptan la herencia.

- 6 Rellene los campos de la siguiente manera:

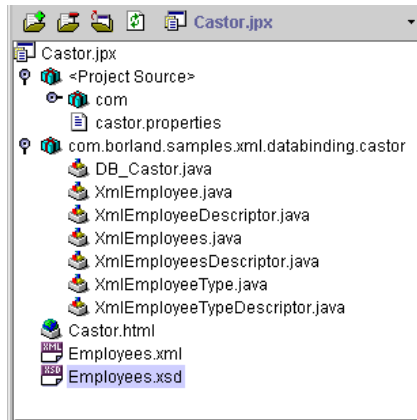
- Esquema XML: Vaya a la vía de acceso del archivo de esquema: `<jbuilder>/samples/Tutorials/XML/databinding/fromSchema/Employees.xsd`. Este campo ya está lleno.
- Paquete: cambie el nombre del paquete a `com.borland.samples.xml.databinding.castor`
- Acepte las opciones por defecto.

El segundo paso del Asistente ofrecerá el siguiente aspecto:



- 7 Pulse el botón Finalizar.
- 8 Amplíe el nodo de paquete `com.borland.samples.xml.databinding.castor` en el panel de proyecto para ver los archivos `.java` generados por el

asistente. El nodo de paquetes también incluye la aplicación de pruebas que forma parte del ejemplo, `DB_Castor.java`. La aplicación de pruebas se va a utilizar para manipular los datos.



9 Abra `castor.properties`. En este archivo se indican los valores de ejecución de Castor. Por ejemplo, en él se puede configurar el sangrado. Observe que el valor de `sangrado` es `true`. Consulte la nota “Asociación de datos: Castor” en la [página 2-34](#) si desea más información sobre el archivo `castor.properties`.

10 Seleccione Proyecto | Ejecutar Make del proyecto para compilar las clases.

Nota Aparecerán advertencias desaconsejando porque Castor genera código que utiliza Sax 1.0.

11 Guarde el proyecto.

Antes de dirigirse al paso siguiente, examine algunas de las clases Java generadas por el asistente. Castor ejecuta la asociación de datos asignando una instancia de un esquema XML a un modelo de objetos adecuado. Este modelo de objetos incluye un conjunto de clases y tipos que representan los datos. Para obtener información sobre una clase y sus campos se utilizan descriptores. En la mayoría de los casos, el marco de montaje (marshalling) utiliza un conjunto de descriptores `ClassDescriptor` y `FieldDescriptor` para describir la forma en que debe montar y desmontar un objeto `Object`.

En primer lugar, observe que hay varios tipos de archivos Java:

- `XmlEmployee.java`: monta y desmonta los datos entre el documento XML y los objetos Java.
- `XmlEmployeeDescriptor.java`: obtiene namespaces, identity, clases Java, etc.
- `XmlEmployees.java`: obtiene y define el elemento `XmlEmployee`, y monta y desmonta los datos entre el documento XML y los objetos Java.

- `XmlEmployeesDescriptor.java`: obtiene namespaces, identity, clases Java, etc.
- `XmlEmployeeType.java`: define el tipo de datos de los objetos.
- `XmlEmployeeTypeDescriptor.java`: inicializa los descriptores de elemento y contiene varios métodos para la obtención del nombre del archivo XML, la clase Java y el namespace.

Sugerencia Desde el editor, puede utilizar Buscar definición, para buscar clases, interfaces, sucesos, métodos, propiedades, paquetes e identificadores. Sitúe el cursor sobre uno de estos nombres, pulse el botón derecho y seleccione Buscar definición. Con el fin de que una clase se encuentre de forma automática, debe estar en la vía de acceso import. Los resultados de la búsqueda aparecen en el panel de contenido del Visualizador de aplicaciones. También es posible buscar clases en el editor, desde el menú Buscar (Buscar | Buscar clases).

Consulte

- Castor API en el directorio `extras` de la instalación completa de JBuilder.
- La sede web de Castor, en <http://www.castor.org/javadoc/overview-summary.html>

Paso 2: Desmontaje de los datos

Ahora que se han creado los objetos Java para los objetos XML, examine la aplicación de prueba, `DB_Castor.java`. Esta aplicación transfiere los datos entre el documento XML y los objetos Java mediante el marco de montaje, que gestiona la conversión entre Java y XML. En el desmontaje se leen los objetos XML y se convierten en objetos Java; en el montaje, los objetos Java se convierten de nuevo en XML.

- 1 Haga doble clic en `DB_Castor.java` en el panel de proyectos para abrirla en el editor.
- 2 Examine el código fuente y observe que existen varias llamadas a métodos para la manipulación de los datos. En primer lugar, la aplicación desmonta o lee los datos del documento XML. A continuación representa en la pantalla el número de empleados y sus nombres. A continuación, se añade y se modifica un empleado. En último lugar, los datos se vuelven a montar en el documento XML.

Sugerencia Para examinar cualquiera de las clases importadas de este archivo, abra la carpeta `Imports` del panel de estructura, haga doble clic en un paquete para abrirlo en el cuadro de diálogo Buscar símbolo de paquete y busque la clase que desea examinar.

Si se ejecutara la aplicación sin realizar cambios, añadiría y modificaría un empleado, pero antes se va a dividir este proceso en dos pasos para observar su funcionamiento. En el siguiente paso se va a añadir un empleado. Más adelante se modificarán sus datos.

Paso 3: Adición de un registro de empleado

En primer lugar, modifique la aplicación para que sólo añada un empleado. Para ello se deben anteponer signos de comentario para evitar que se ejecute la llamada al método que modifica los datos de los empleados en `DB_Castor.java`. En el siguiente paso se modifican estos datos.

- 1 Anteponga signos de comentario a la llamada al método `setXmlEmployee()` para que no se ejecute. Cuando se colocan marcas de comentario delante de este método, la aplicación desmonta los datos, los muestra en la pantalla, añade un empleado y monta los datos nuevos en el documento XML, sin modificar el nuevo empleado. El empleado se modificará más adelante.

```
//Modificar el último XmlEmployee
//xmlEmployees.setXmlEmployee(xmlEmployees.getXmlEmployeeCount()-1,
//    getXmlEmployee("8000","600","Peter","Castor","3/3/2001",
//        "VP","USA","3","2096","125000.00"));
```

- 2 Añada una sentencia `print` a la llamada al método `addXmlEmployee()`: La sentencia `print` representa en la pantalla el nombre del empleado que se acaba de añadir.

```
// Añadir un XmlEmployee
xmlEmployees.addXmlEmployee(getXmlEmployee("8000","400","Charlie","Castor",
    "3/3/2001","VP","USA","2","1993","155000.00"));
System.out.println("New XmlEmployee's Full Name is " +
    xmlEmployees.getXmlEmployee(
    xmlEmployees.getXmlEmployeeCount()-1).getFullName());
```

- 3 Haga clic con el botón derecho del ratón en `DB_Castor.java` en el panel de proyecto y pulse Ejecutar utilizando "Castor". La aplicación se ejecuta, lee la información sobre el empleado y muestra lo siguiente en el panel de mensajes:

```
== unmarshalling "Employees.xml" ==
Total Number of XmlEmployees read = 3
First XmlEmployee's Full Name is Nelson, Robert
Last XmlEmployee's Full Name is Lambert, Kim
New XmlEmployee's Full Name is Castor, Charlie
```

- 4 Pase a `Employees.xml` para comprobar que los nuevos datos se han montado en el documento XML. Observe que se ha añadido el empleado Charlie Castor.

Paso 4: Modificación de los datos del nuevo empleado

A continuación, modifique el registro de empleado de Charlie Castor con la llamada al método `setXmlEmployee()` y ejecute de nuevo el programa para actualizar el registro en el documento XML.

- 1 Vuelva a `DB_Castor.java` y coloque marcas de comentario (`//`) delante de la llamada al método `addEmployee()` y la sentencia `print` que se encuentra a continuación:

```
// Añadir un XmlEmployee
//xmlEmployees.addXmlEmployee(getXmlEmployee("8000","400","Charlie",
//    "Castor","3/3/2001","VP","USA","2","1993","155000.00"));
//System.out.println("New XmlEmployee's Full Name is " +
//    xmlEmployees.getXmlEmployee(
//    xmlEmployees.getXmlEmployeeCount()-1).getFullName());
```

- 2 Elimine las marcas de comentario de la llamada al método `setXmlEmployee()` y añada la sentencia `print`:

```
// Modificar el último XmlEmployee
xmlEmployees.setXmlEmployee(xmlEmployees.getXmlEmployeeCount()-1,
    getXmlEmployee("8000","600","Peter","Castor","3/3/2001",
    "VP","USA","3","2096","125000.00"));
System.out.println("New XmlEmployee's Modified Full Name is " +
    xmlEmployees.getXmlEmployee(
    xmlEmployees.getXmlEmployeeCount()-1).getFullName());
```

- 3 Guarde el proyecto.
- 4 Haga clic con el botón derecho del ratón en `DB_Castor.java` y seleccione Ejecutar utilizando "Castor". La aplicación se ejecuta, lee la información del empleado e imprime el número de empleados leídos y los nombres completos del primero y el último, además del empleado modificado:

```
== unmarshalling "Employees.xml" ==
Total Number of XmlEmployees read = 4
First XmlEmployee's Full Name is Nelson, Robert
Last XmlEmployee's Full Name is Castor, Charlie
New XmlEmployee's Modified Full Name is Castor, Peter
```

Paso 5: Ejecución de la aplicación completa

Ahora que ya se ha explicado lo que hace la aplicación, elimine los comentarios, añada una sentencia `print` para leer el nuevo empleado, elimine los datos nuevos del documento XML y ejecute la aplicación completa.

- 1 Elimine los comentarios de la llamada al método `addXmlEmployee()` y su sentencia `print`. Su código debería tener este aspecto:

```
// Añadir un XmlEmployee
xmlEmployees.addXmlEmployee(getXmlEmployee("8000","400","Charlie","Castor",
```

Paso 5: Ejecución de la aplicación completa

```
"3/3/2001", "VP", "USA", "2", "1993", "155000.00"));
System.out.println("New XmlEmployee's Full Name is " +
    xmlEmployees.getXmlEmployee(
        xmlEmployees.getXmlEmployeeCount()-1).getFullName());

// Modificar el último XmlEmployee
xmlEmployees.setXmlEmployee(xmlEmployees.getXmlEmployeeCount()-1,
    getXmlEmployee("8000", "600", "Peter", "Castor", "3/3/2001",
        "VP", "USA", "3", "2096", "125000.00"));
System.out.println("New XmlEmployee's Modified Full Name is " +
    xmlEmployees.getXmlEmployee(
        xmlEmployees.getXmlEmployeeCount()-1).getFullName());
```

- 2 Añada una sentencia print después del método `setEmployee()` y antes de `marshal()`, para volver a leer los datos después de que el nuevo dato haya sido añadido y modificado. Su código debería tener este aspecto:

```
// Modificar el último XmlEmployee
xmlEmployees.setXmlEmployee(xmlEmployees.getXmlEmployeeCount()-1,
    getXmlEmployee("8000", "600", "Peter", "Castor", "3/3/2001", "VP",
        "USA", "3", "2096", "125000.00"));
System.out.println("New XmlEmployee's Modified Full Name is "
    + xmlEmployees.getXmlEmployee(
        xmlEmployees.getXmlEmployeeCount()-1).getFullName());
//Lee de nuevo los empleados
System.out.println("Total Number of XmlEmployees read =
    + xmlEmployees.getXmlEmployeeCount());
// Montar (marshal) los datos en el mismo archivo XML
xmlEmployees.marshal(new java.io.FileWriter(fileName));
```

- 3 Vaya a `Employees.xml` y elimine las etiquetas XML y los datos del empleado Peter Castor, con cuidado de no eliminar otros datos ni etiquetas XML.
- 4 Guarde el proyecto.
- 5 Vuelva a `DB_Castor.java` en el editor, pulse con el botón secundario del ratón en la pestaña de nombre de archivo de `DB_Castor.java` y seleccione Ejecutar utilizando “Castor” para ver el siguiente resultado en el panel de mensajes:

```
== unmarshalling "Employees.xml" ==
Total Number of XmlEmployees read = 3
First XmlEmployee's Full Name is Nelson, Robert
Last XmlEmployee's Full Name is Lambert, Kim
New XmlEmployee's Full Name is Castor, Charlie
New XmlEmployee's Modified Full Name is Castor, Peter
Total Number of XmlEmployees read = 4
```

- 6 Vuelva a `Employees.xml` y compruebe que los datos se han montado (marshal) en el documento XML. Si no puede ver los nuevos datos, seleccione otra pestaña de archivo y, a continuación, vuelva a `Employees.xml` con el fin de actualizar el archivo. Observe que Charlie Castor se ha añadido y se ha modificado. Cuando los empleados se leen por segunda vez, se incluye al nuevo.

¡Enhorabuena! Ha finalizado este tutorial. Ya ha leído, añadido y modificado los datos de los empleados de un documento XML con una aplicación Java y clases Java generadas por el Asistente para la asociación de datos a partir de un archivo de esquema.

Si desea seguir más tutoriales sobre XML, consulte “[Tutoriales](#)” en la [página v](#).

Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos

Es una función de JBuilder Enterprise.

En este tutorial se explica la forma de utilizar los componentes XML de base de datos basados en modelos de JBuilder para transferir datos de un archivo XML a una base de datos y viceversa. También se explica la forma de utilizar el Asistente para XML-DBMS con el fin de crear el archivo de asignación que se utiliza en la transferencia de datos y el archivo de script SQL que se utiliza para crear la base de datos.

Los componentes basados en modelos utilizan un documento de mapa que determina cómo se transfieren los datos entre una estructura XML y los metadatos de la base de datos. Gracias a que el usuario especifica una asignación entre un elemento del documento XML y una tabla o columna en concreto de una base de datos, los documentos XML anidados se pueden trasladar desde y hacia un conjunto de tablas de bases de datos. Los componentes basados en modelos se implementan utilizando XML-DBMS, un software intermedio de Open Source XML que se distribuye en un paquete con JBuilder.

En este tutorial se explica la forma de hacer lo siguiente.

- Crear un archivo .map que asigna los elementos de un archivo DTD a las columnas de una tabla de base de datos.
- Crear un archivo de script SQL que genera los metadatos de la tabla de base de datos.
- Utilizar el componente `XDBMSTable` para transferir datos de un documento XML a la tabla de una base de datos.

- Utilizar el mismo componente `XDBMSTable` para transferir datos de la tabla de la base de datos a un documento XML.
- Utilizar el componente `XDBMSQuery` para transferir datos de la tabla de la base de datos a un documento XML.
- Utilizar los personalizadores de `XDBMSTable` y `XDBMSQuery` para definir las propiedades y ver el resultado de su configuración en la transferencia de los datos.

Este tutorial utiliza el ejemplo `XMLDBMSBeans.jpx` del directorio `/<jbuilder>/samples/Tutorials/XML/database/XMLDBMSBeans/`. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

El apartado Opciones de accesibilidad en las 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 1-3](#).

Paso 1: Procedimientos iniciales

En este tutorial se crea una tabla de base de datos `XmlEmployee` que contiene información básica sobre los empleados, número de historial, nombre, sueldo, fecha de contratación, categoría, etc. Para crear la tabla es necesario tener una DTD que defina los metadatos de la tabla de base de datos a la que se van a transferir y desde la que se van a recuperar los datos. La DTD se puede crear de forma manual o a partir de un documento XML con la estructura correcta, por medio del asistente XML a DTD.

En JBuilder, abra el proyecto `/<jbuilder>/samples/Tutorials/XML/database/XMLDBMSBeans/XMLDBMSBeans.jpx`, que contiene el archivo DTD que necesita, `Employees.dtd`. Tiene el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT XmlEmployee (EmpNo, FirstName, LastName, PhoneExt, HireDate, DeptNo,
    JobCode, JobGrade, JobCountry, Salary, FullName)>
<!ELEMENT DeptNo (#PCDATA)>
<!ELEMENT EmpNo (#PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT FullName (#PCDATA)>
<!ELEMENT HireDate (#PCDATA)>
<!ELEMENT JobCode (#PCDATA)>
```

```

<!ELEMENT JobCountry (#PCDATA)>
<!ELEMENT JobGrade (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT PhoneExt (#PCDATA)>
<!ELEMENT Salary (#PCDATA)>
<!ELEMENT XmlEmployees (XmlEmployee+)>

```

El proyecto de ejemplo tiene también un archivo `Employees.xml`. Si el proyecto no tuviera un archivo `Employees.xml`, es posible crearlo por medio del Asistente para pasar de DTD a XML, con el archivo `Employees.dtd` como DTD de entrada. Después, modifique la estructura XML resultante y añada los datos.

Más adelante se va a utilizar `Employees.xml` para llenar la tabla de la base de datos y modificar su contenido. `Employees.xml`, como aparece en el proyecto de ejemplo, contiene datos sobre tres empleados. Tiene el siguiente aspecto:

```

<?xml version="1.0"?>
<!DOCTYPE XmlEmployees SYSTEM "Employees.dtd">
<XmlEmployees>
  <XmlEmployee>
    <EmpNo>2</EmpNo>
    <FirstName>Robert</FirstName>
    <LastName>Nelson</LastName>
    <PhoneExt>250</PhoneExt>
    <HireDate>1988-12-28</HireDate>
    <DeptNo>600</DeptNo>
    <JobCode>VP</JobCode>
    <JobGrade>2</JobGrade>
    <JobCountry>USA</JobCountry>
    <Salary>105900,000000</Salary>
    <FullName>Nelson, Robert</FullName>
  </XmlEmployee>
  <XmlEmployee>
    <EmpNo>4</EmpNo>
    <FirstName>Bruce</FirstName>
    <LastName>Young</LastName>
    <PhoneExt>233</PhoneExt>
    <HireDate>1988-12-28</HireDate>
    <DeptNo>621</DeptNo>
    <JobCode>Eng</JobCode>
    <JobGrade>2</JobGrade>
    <JobCountry>USA</JobCountry>
    <Salary>97500,000000</Salary>
    <FullName>Young, Bruce</FullName>
  </XmlEmployee>
  <XmlEmployee>
    <EmpNo>5</EmpNo>
    <FirstName>Kim</FirstName>
    <LastName>Lambert</LastName>
    <PhoneExt>22</PhoneExt>
    <HireDate>1989-02-06</HireDate>
    <DeptNo>130</DeptNo>
  </XmlEmployee>

```

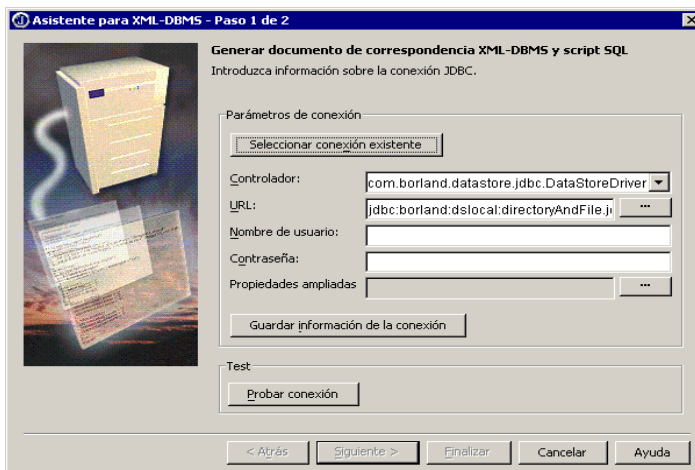
```
<JobCode>Eng</JobCode>
<JobGrade>2</JobGrade>
<JobCountry>USA</JobCountry>
<Salary>102750.000000</Salary>
<FullName>Lambert, Kim</FullName>
</XmlEmployee>
</XmlEmployees>
```

Paso 2: Creación de los archivos de asignación y script SQL

Ya está creada la estructura de la tabla de la base de datos, definida por la DTD y el documento XML con los datos que se desean almacenar se encuentra en la tabla de la base de datos. Sin embargo, aún no existe la tabla de base de datos; es necesario crearla. También es necesario crear un archivo de asignación que describa la forma de transferir los datos de los elementos XML a las columnas correspondientes de la nueva base de datos. El Asistente para XML-DMBS de JBuilder puede crear el archivo de guión SQL que se ejecuta para generar la tabla mientras crea el archivo de asignación necesario para transferir los datos.

Para abrir el asistente XML-DBMS:

- 1 Abra la galería de objetos seleccionando Archivo | Nuevo.
- 2 Pulse la pestaña XML y haga doble clic en el icono XML-DBMS.



Introducción de información sobre la conexión JDBC

Este tutorial utiliza la base de datos `employee.jds` de `JDataStore`, ubicada en el directorio `/<jbuilder>/samples/JDataStore/datastores`. Si ha trabajado con los ejemplos de `JDataStore`, es posible que ya tenga una conexión JDBC con esta base de datos. Si es así, pulse el botón **Seleccionar conexión**

existente y elíjala. Si se emplea este método, los parámetros de conexión se completan automáticamente. Si no utiliza una conexión ya establecida, deberá introducir la información según se describe en los siguientes pasos:

- 1 Seleccione `com.borland.datastore.jdbc.DataStoreDriver` como controlador, en la lista desplegable. Es necesario tener instalado JDataStore, que se incluye en JBuilder Enterprise. Si desea obtener más información acerca del trabajo con JDataStore, consulte "Fundamentos de JDataStore" en la *Guía del desarrollador de JDataStore*.
- 2 Indique la URL del almacén de datos que está utilizando, `employee.jds`. Cuando se elige DataStoreDriver como controlador aparece un modelo para ayudar a introducir la URL. Pulse el botón de puntos suspensivos (...) situado junto al campo URL para desplazarse hasta la dirección URL correcta. Suponiendo que haya instalado JBuilder en la unidad C de su sistema, la URL del almacén de datos `employee.jds` del directorio `samples` es la siguiente:

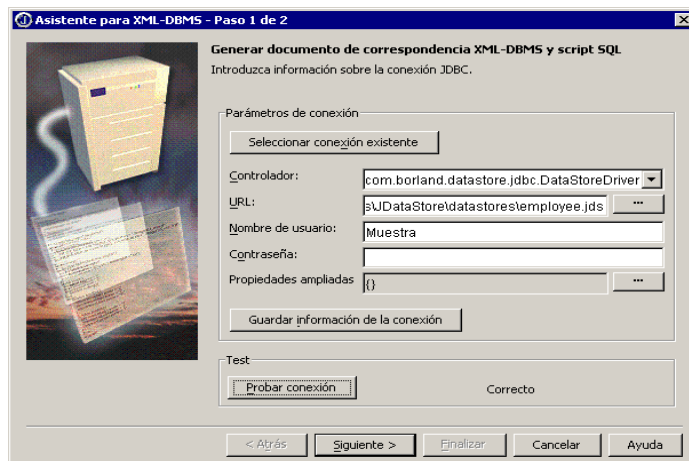
```
jdbc:borland:dslocal:C:\<jbuilder>\samples\JDataStore\datastores\employee.jds
```

- 3 Escriba `Sample` en el campo Nombre de usuario.
- 4 Introduzca un valor en el campo Contraseña o déjelo en blanco ya que `employee.jds` no requiere contraseña.
- 5 No rellene el campo Propiedades ampliadas.
- 6 Seleccione el botón Guardar información de la conexión para guardar la conexión recién creada.

Comprobación de la conexión

Después de establecer la conexión, es necesario probarla para verificar si la conexión JDBC se ha especificado correctamente.

- 1 Pulse el botón Probar conexión. Junto al botón aparece un mensaje que indica si la conexión se ha establecido correctamente.



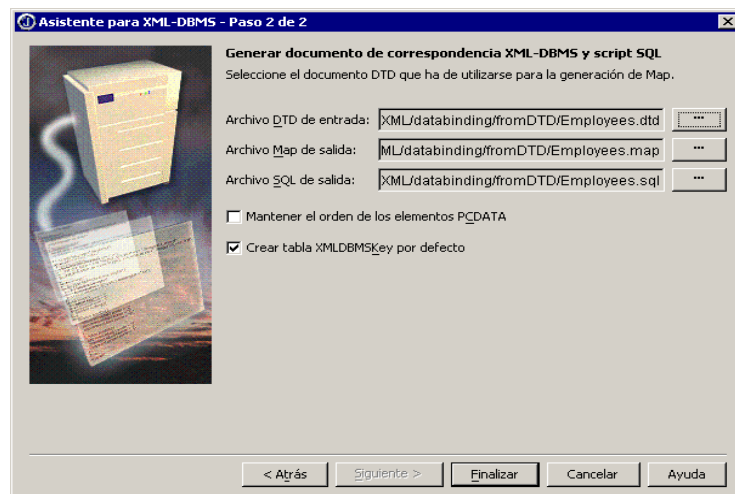
2 Haga clic en Siguiente para avanzar al Paso 2 del asistente.

Especificación de los nombres de archivo

En la segunda ficha del Asistente para XML-DBMS, escriba el archivo DTD que utiliza para crear el archivo de asignación e indique los nombres del archivo de asignación y el del archivo script SQL, tal como se describe en los siguientes pasos:

- 1 Pulse el botón de puntos suspensivos ubicado junto al campo Archivo DTD de entrada para llegar a él y seleccionar el archivo `Employees.dtd` del proyecto `XMLDBMSBeans.jpx`. Pulse Aceptar.
- 2 Acepte los nombres de archivo predeterminados para los campos Archivo Map de salida y Archivo SQL de salida.
- 3 Marque la casilla de verificación Crear tabla XMLDBMSKey por defecto, en el caso de que no lo estuviera.

El Asistente para XML-DBMS tendrá el siguiente aspecto:



4 Pulse Finalizar para cerrar el asistente.

El Asistente para XML-DBMS genera `Employees.map` y `Employees.sql`, los cuales se muestran en el panel de proyecto y aparecen abiertos en el editor. Mediante el editor, compruebe en `Employees.map` la forma en que se asignan los elementos XML a las columnas de la tabla de la base de datos que se va a crear. Si desea modificar el nombre de las columnas de esta tabla, hágalo en el archivo de asignación. Si decide hacerlo, también es necesario efectuar los mismos cambios en los nombres de columnas en el archivo de guión SQL. Para este tutorial no es necesario hacer cambios. Sin embargo, con frecuencia es conveniente modificar los archivos de

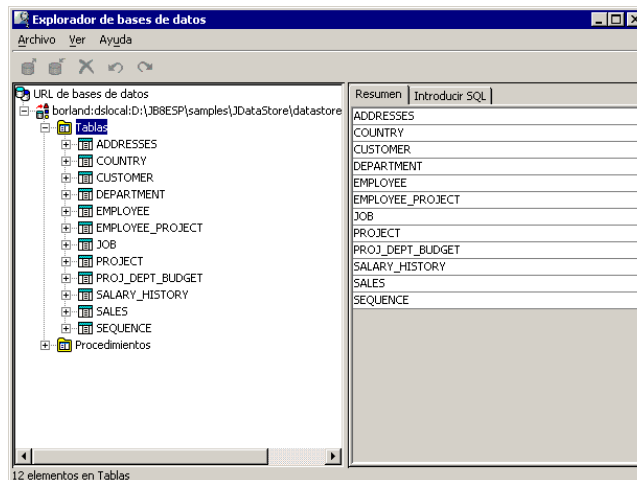
asignación y guión SQL generados por el asistente, para adaptarlos a sus necesidades.

Paso 3: Creación de tablas de base de datos

Mediante el editor, compruebe en `Employees.sql` las sentencias SQL generadas por el Asistente para XML-DBMS. Observe que contiene tres sentencias CREATE TABLE, no sólo una. La primera crea una tabla `XmlEmployee` que contiene los metadatos especificados por la DTD, además de una columna "`XmlEmployeesFK`" (FK indica que la clave es externa). La segunda sentencia CREATE TABLE crea una tabla `XmlEmployees` que sólo contiene una columna, "`XmlEmployeesPK`" (PK indica la clave primaria). La tercera crea una tabla `XMLDBMSKey`. XML-DBMS utiliza estas tablas para representar la estructura del archivo DTD de entrada.

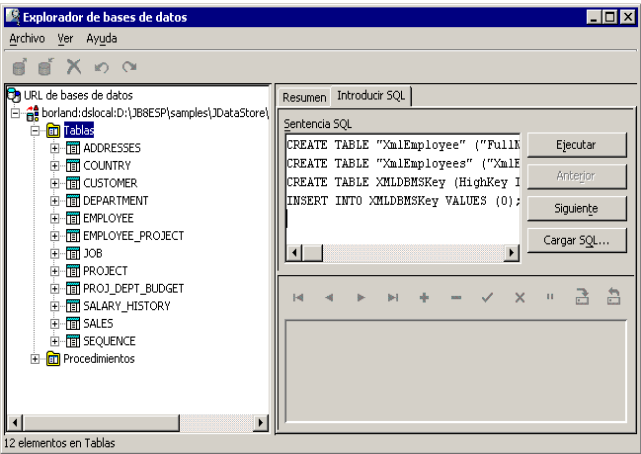
El explorador de bases de datos de JBuilder permite ejecutar sentencias SQL cómodamente:

- 1 Seleccione todo el texto del archivo `Employees.sql` y elija Edición | Copiar para pasarlo al portapapeles.
- 2 Elija Herramientas | Explorador de base de datos.
- 3 Haga doble clic en la URL de base de datos de `employee.jds` especificada en el Asistente para XML-DBMS con el fin de conectarse con la base de datos.
- 4 Escriba cualquier texto en el campo Nombre de usuario y deje el campo Contraseña en blanco. Pulse Aceptar para conectar con la base de datos.
- 5 Despliegue el nodo Tablas y observe que la base de datos contiene varias tablas. A continuación, creará tres tablas en esta base de datos.

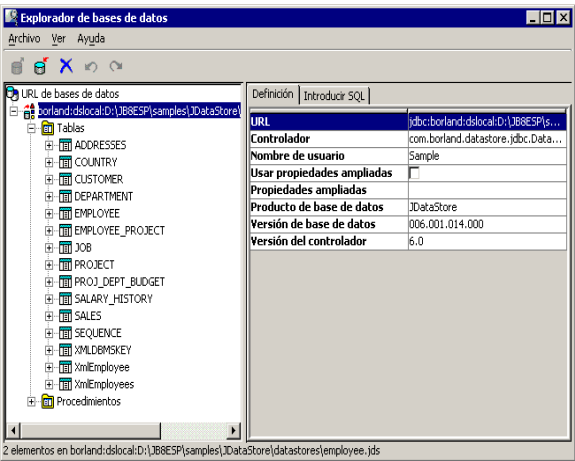


Paso 3: Creación de tablas de base de datos

- 6 Pulse sobre la pestaña Introducir SQL situada en la parte derecha del Explorador de bases de datos.
- 7 Pegue la sentencia SQL copiada en el cuadro Sentencia SQL:



- 8 Pulse Ejecutar. El Explorador de base de datos crea tres tablas en el almacén de datos employee.jds.
- 9 Seleccione Ver | Actualizar en el Explorador de bases de datos y expanda el nodo Tablas otra vez para ver las tres nuevas tablas añadidas a la base de datos.



- 10 Cierre el explorador de bases de datos.

Paso 4: Utilización de la aplicación de prueba de ejemplo

Normalmente, cuando se utilizan los componentes de base de datos XML de JBuilder, se desarrolla al mismo tiempo una aplicación que presenta a los usuarios una interfaz gráfica para manejar el programa. En este tutorial no se va a hacer esto, sino que se va a utilizar la aplicación de ejemplo, `XMLDBMS_Test.java`, que simplemente es una clase Java que contiene los componentes de base de datos XML basados en modelos, cuyas propiedades define. En este tutorial se explica la forma de trabajar con los personalizadores de los componentes para definir propiedades y ver el resultado de la transferencia de datos. Cuando esté seguro de que la transferencia funciona correctamente puede crear la aplicación de interfaz.

- 1 Expanda el nodo del paquete `com.borland.samples.xml.XMLDBMS` en el panel de proyecto para ver el contenido. Se muestra un archivo, `XMLDBMS_Test.java`.
- 2 Haga doble clic en `XMLDBMS_Test.java` para abrirlo en el editor.
- 3 Examine el código fuente y verá que contiene los dos componentes, `XMLDBMSTable` y `XMLDBMSQuery`.

Uso del personalizador de `XMLDBMSTable`

Para empezar a trabajar con el componente `XMLDBMSTable`:

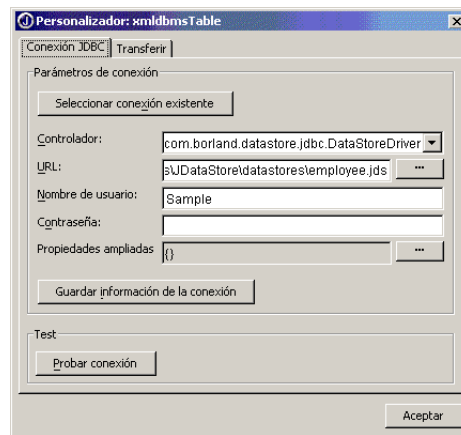
- 1 Haga clic en la pestaña Diseño de la aplicación de prueba abierta, `XMLDBMS_Test.java`, para abrir el diseñador de interfaces de usuario. Verá que la carpeta Otros del panel de estructura contiene los dos componentes basados en modelos.
- 2 Haga clic con el botón derecho del ratón en `xmldbmsTable`, en el panel de estructura, y seleccione Personalizador en el menú. Aparece el personalizador de `XMLDBMSTable`.

Selección y prueba de una conexión JDBC

La aplicación de prueba de ejemplo ya define todas las propiedades en el código fuente. Si estuviera creando una aplicación propia, debería rellenar personalmente todos los campos del personalizador: imagine que los campos están en blanco y observe cómo se llenarían. Como se ha hecho en el primer paso del Asistente para XML-DBMS, es necesario seleccionar la conexión JDBC.

- 1 Pulse el botón Seleccionar conexión existente y seleccione la misma conexión establecida con el almacén de datos `employee.jds`. Apenas

seleccionada la conexión, el personalizador utiliza sus datos para llenar el resto de los campos.

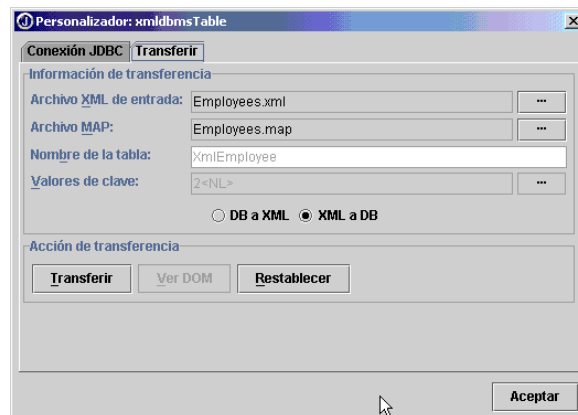


- 2 Para comprobar que la conexión es correcta, pulse el botón Probar conexión. A la derecha del botón Probar conexión, aparecerá un mensaje indicando el resultado de la operación.

Transferencia de datos de XML a la base de datos

A continuación, se transferirán los datos desde el documento XML hasta la base de datos.

- 1 Abra la ficha Transferir para ver la siguiente página del personalizador:



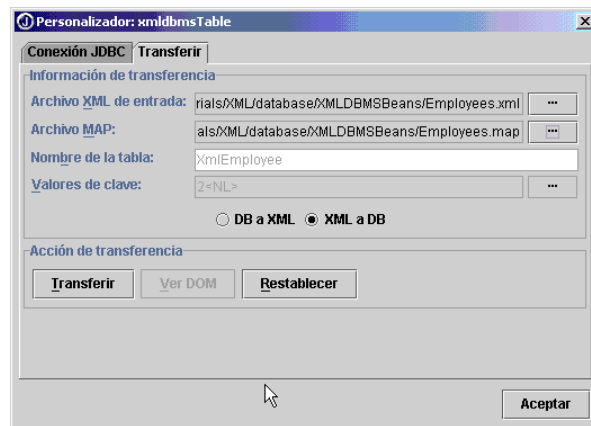
- 2 Introduzca la información de transferencia necesaria:
 - a Pulse el botón de puntos suspensivos que se encuentra junto al campo Archivo XML de entrada para llegar a él y seleccione el archivo `Employees.xml` del proyecto. Siempre se debe especificar el nombre completo. Si utiliza el botón de puntos suspensivos, el

nombre del archivo incluirá siempre la vía de acceso completa. Pulse Aceptar.

- b Pulse el botón de puntos suspensivos que se encuentra junto al campo Archivo MAP para llegar a él y seleccione el archivo `Employees.map` del proyecto. Pulse Aceptar.
- c Si aún no está seleccionada, pulse la opción XML a DB. Se está preparando para transferir los datos del documento `Employees.xml` a la tabla `XmlEmployee`.

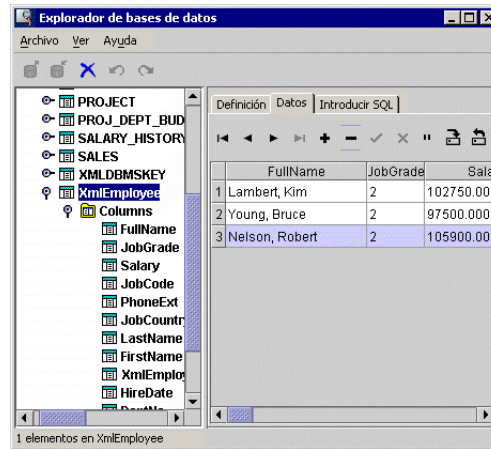
Éstos son los únicos campos necesarios para la transferencia. Los otros se encuentran desactivados. En ellos aparecen valores de datos, porque la aplicación de prueba de ejemplo, `XMLDBMSBeans_Test.java`, define los valores de las propiedades de estos campos en el código fuente. Estos campos no se utilizan para transferir datos del documento XML a la tabla de base de datos `XmlEmployees`. Para obtener más información sobre la ficha Transferir, consulte [“Configuración de propiedades con el personalizador” en la página 3-9](#).

El personalizador debe parecerse a:



- 3 Para transferir los datos del documento `XmlEmployees.xml` a la tabla de la base de datos `XmlEmployees` creada, pulse Transferir. Tiene lugar la transferencia de los datos.
- 4 Pulse Aceptar para cerrar el personalizador.
- 5 Abra el Explorador de bases de datos (Herramientas | Explorador de bases de datos), expanda el nodo Tablas, seleccione el nodo de tabla

XmlEmployee y haga clic en la pestaña Datos con el fin de confirmar si los datos se han transferido a la tabla.



6 Cierre el explorador de bases de datos.

Transferencia de datos desde la base de datos a XML

Para poder transferir datos de una tabla de base de datos a un documento XML es necesario modificar el archivo `Employees.map`. El elemento `XmlEmployee` se debe comportar como raíz. Por ello, `Employees.map` debe indicar a XML-DBMS que pase por alto la raíz actual, el elemento plural `XmlEmployees`, y utilice en su lugar el elemento singular `XmlEmployee`.

- 1 Seleccione la pestaña del archivo `Employees.map` en el editor para ver el archivo de mapa.
- 2 Introduzca el texto en negrita:

```
<?xml version='1.0' ?>
<!DOCTYPE XMLToDBMS SYSTEM "xmlbms.dtd" >

<XMLToDBMS Version="1.0">
  <Options>
  </Options>
  <Maps>
    <IgnoreRoot>
      <ElementType Name="XmlEmployees"/>
      <PseudoRoot>
        <ElementType Name="XmlEmployee"/>
        <CandidateKey Generate="No">
          <Column Name="EmpNo"/>
        </CandidateKey>
      </PseudoRoot>
    </IgnoreRoot>
  </Maps>
  <ClassMap>
    ...
```

```
</ClassMap>
</Maps>
</XMLToDBMS>
```

3 Quite este bloque de código ubicado al final del archivo:

```
<ClassMap>
  <ElementType Name="XmlEmployees"/>
  <ToRootTable>
    <Table Name="XmlEmployees"/>
    <CandidateKey Generate="Yes">
      <Column Name="XmlEmployeesPK"/>
    </CandidateKey>
  </ToRootTable>
  <RelatedClass KeyInParentTable="Candidate">
    <ElementType Name="XmlEmployee"/>
    <CandidateKey Generate="Yes">
      <Column Name="XmlEmployeesPK"/>
    </CandidateKey>
    <ForeignKey>
      <Column Name="XmlEmployeesFK"/>
    </ForeignKey>
  </RelatedClass>
</ClassMap>
```

4 Guarde el proyecto.

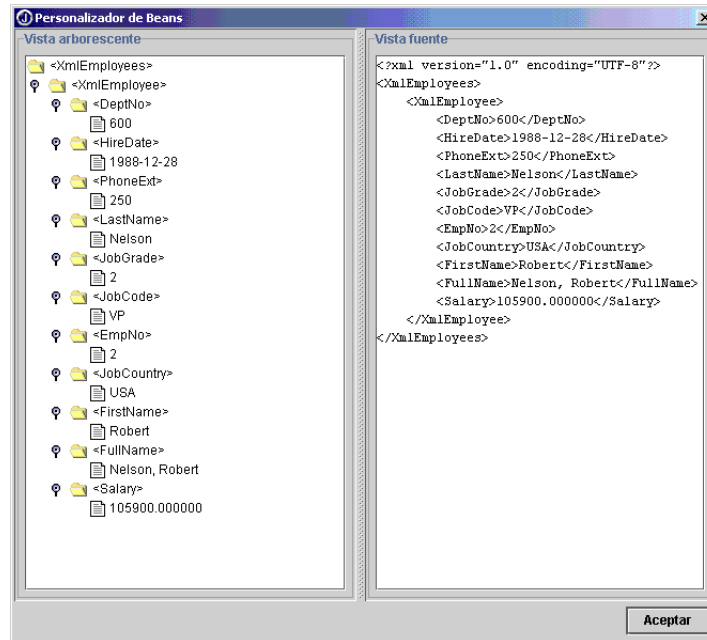
Ahora que existen datos en la tabla XmlEmployee, se pueden transferir datos desde la base de datos a un documento XML. Esto se puede hacer desde la ficha Transferir del personalizador de XMLDBMSTable.

Para transferir datos de la tabla XmlEmployee a un documento XML llamado Employees_out.xml:

- 1 Vuelva a la pestaña Diseño de XMLDBMS_Test.java.
- 2 Haga clic con el botón derecho del ratón en xmldbmsTable, en el panel de estructura, y seleccione Personalizador en el menú.
- 3 Seleccione la pestaña Transferir.
- 4 Active la opción XML a DB. Los campos del personalizador cambian ligeramente. Los campos Nombre de la tabla y Valores de clave aparecen ahora activados.
- 5 Pulse el botón de puntos suspensivos situado junto al campo Archivo XML de salida y cambie el nombre a Employees_out.xml. Pulse Aceptar.
- 6 Conserve el nombre por defecto del campo Archivo Map, Employees.map (el archivo modificado), incluida la vía de acceso.
- 7 Acepte el nombre de la tabla, XmlEmployee.
- 8 Acepte el valor de clave 2 en el campo Valores de clave. Este es el valor en la columna "EmpNo" del empleado que desea transferir de la tabla de base de datos al documento Employees_out.xml. La columna "EmpNo" es la clave primaria de XmlEmployee. Si desea transferir los registros de

varios empleados, utilice el campo Valores de clave y su editor de propiedades para introducir los diferentes números.

- 9 Seleccione Ver DOM para transferir datos desde la tabla XmlEmployee hasta Employees_out.xml. Es posible ver el resultado de la solicitud de transferencia: En el panel de la izquierda aparece una vista en árbol de Employees_out.xml, y en el panel de la derecha se muestra el código fuente XML:



- 10 Pulse Aceptar dos veces para cerrar la vista DOM y el personalizador.
- 11 Añada Employees_out.xml al proyecto y ábralo para ver si los datos se han transferido:
 - a Seleccione Proyecto | Añadir archivos/paquetes.
 - b Seleccione Employees_out.xml y pulse Aceptar.
 - c En el panel de proyecto, haga doble clic en Employees_out.xml.

Uso del personalizador de XMLDBMSQuery

También se puede utilizar una sentencia SQL para recuperar datos de una tabla de base de datos con el componente XMLDBMSQuery.

Para empezar a trabajar con el componente XMLDBMSQuery:

- 1 Vuelva a XMLDBMS_test.java en el editor y haga clic en la pestaña Diseño.

- 2 Haga clic con el botón derecho del ratón en `xmlDBMSQuery`, en el panel de estructura, y seleccione Personalizador en el menú. Aparece el personalizador de `XMLDBMSQuery`:

Selección y prueba de una conexión JDBC

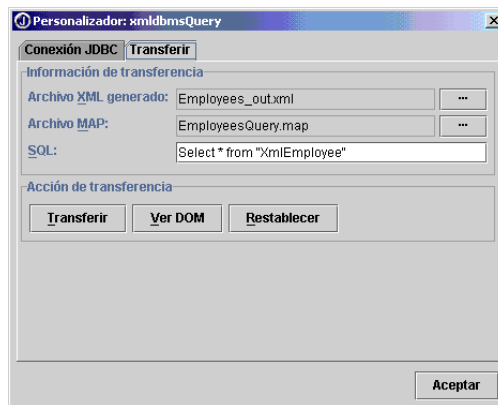
Al igual que se hizo con el componente `XMLDBMSTable`, a continuación se seleccionará una conexión JDBC y se someterá a prueba.

- 1 Pulse el botón Seleccionar conexión existente y especifique la conexión con `employee.jds` en el campo URL de la base de datos. Por ejemplo, `jdbc:borland:dslocal:C:\JBuilder\samples\JDataStore\datastores\employee.jds`.
- 2 Pulse Probar conexión para comprobar si la conexión funciona correctamente.

Transferencia de datos con una sentencia SQL

Ahora, transferirá los datos mediante una sentencia SQL.

- 1 Pulse la pestaña Transferir para pasar a la siguiente ficha.



- 2 Introduzca la información de transferencia necesaria:
 - a Pulse el botón de puntos suspensivos situado junto al campo Archivo XML de salida para llegar a él y seleccione el archivo `Employees_out.xml` creado anteriormente. Siempre se debe introducir el nombre completo y, si se pulsa el botón de puntos suspensivos, el nombre de archivo siempre incluye la vía de acceso. Pulse Aceptar.
 - b Pulse el botón de puntos suspensivos situado junto al campo Archivo MAP para llegar a él y seleccione el archivo `EmployeesQuery.map` del proyecto. Haga clic en Aceptar para cerrar el cuadro de diálogo. El componente `XMLDBMSQuery` utiliza un archivo de mapa diferente. Consulte ["El archivo de mapa" en la página 9-16](#) para obtener más información.

c Escriba la sentencia SQL por defecto en el campo SQL.

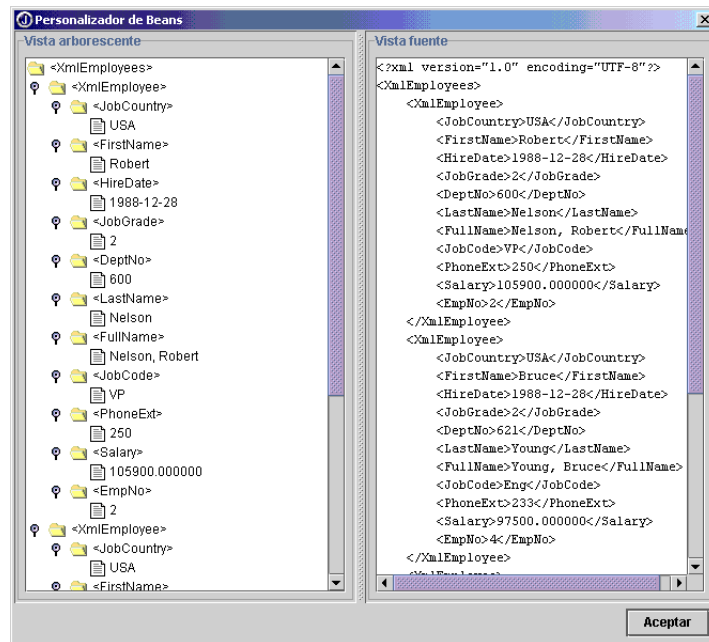
```
Select * from "XmlEmployee"
```

El nombre de la tabla debe encerrarse entre comillas dobles. Esta sentencia recupera todas las filas de la tabla XmlEmployee y las transfiere a Employees_out.xml. Por supuesto, para realizar una consulta en la tabla XmlEmployee se puede utilizar cualquier sentencia SQL válida. Por ejemplo:

```
Select * from "XmlEmployee" where "JobCode" = 'VP'
```

Para obtener más información sobre la ficha Transferir, consulte [“Configuración de propiedades con el personalizador” en la página 3-9.](#)

3 Elija Ver DOM para consultar el resultado. El resultado de la primera consulta sería similar al siguiente:



4 Pulse Aceptar dos veces para cerrar la vista DOM y el personalizador.

5 Abra Employees_out.xml en el editor para comprobar si todos los registros de empleados se han transferido desde la base de datos al documento XML.

El archivo de mapa

Habrá observado que el componente XMLDBMSQuery utiliza el archivo de asignación EmployeesQuery.map, que no coincide con el que utiliza el

componente XMLDMSTable. Éste es el aspecto del archivo EmployeesQuery.map, con las diferencias respecto al archivo Employees.map resaltadas en negrita:

```
<?xml version='1.0' ?>
<!DOCTYPE XMLToDBMS SYSTEM "xmlDbms.dtd" >

<XMLToDBMS Version="1.0">
  <Options>
  </Options>
  <Maps>
    <IgnoreRoot>
      <ElementType Name="XmlEmployees"/>
      <PseudoRoot>
        <ElementType Name="XmlEmployee"/>
        <CandidateKey Generate="No">
          <Column Name="EmpNo"/>
        </CandidateKey>
      </PseudoRoot>
    </IgnoreRoot>

    <ClassMap>
      <ElementType Name="XmlEmployee"/>
      <ToClassTable>
        <Table Name="Result Set"/>
      </ToClassTable>
      <PropertyMap>
        <ElementType Name="FullName"/>
        <ToColumn>
          <Column Name="FullName"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="JobGrade"/>
        <ToColumn>
          <Column Name="JobGrade"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="Salary"/>
        <ToColumn>
          <Column Name="Salary"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="JobCode"/>
        <ToColumn>
          <Column Name="JobCode"/>
        </ToColumn>
      </PropertyMap>
      <PropertyMap>
        <ElementType Name="PhoneExt"/>
        <ToColumn>
          <Column Name="PhoneExt"/>
        </ToColumn>
      </PropertyMap>
    </ClassMap>
  </Maps>
</XMLToDBMS>
```

Paso 4: Utilización de la aplicación de prueba de ejemplo

```
</PropertyMap>
<PropertyMap>
  <ElementType Name="JobCountry"/>
  <ToColumn>
    <Column Name="JobCountry"/>
  </ToColumn>
</PropertyMap>
<PropertyMap>
  <ElementType Name="LastName"/>
  <ToColumn>
    <Column Name="LastName"/>
  </ToColumn>
</PropertyMap>
<PropertyMap>
  <ElementType Name="FirstName"/>
  <ToColumn>
    <Column Name="FirstName"/>
  </ToColumn>
</PropertyMap>
<PropertyMap>
  <ElementType Name="HireDate"/>
  <ToColumn>
    <Column Name="HireDate"/>
  </ToColumn>
</PropertyMap>
<PropertyMap>
  <ElementType Name="EmpNo"/>
  <ToColumn>
    <Column Name="EmpNo"/>
  </ToColumn>
</PropertyMap>
<PropertyMap>
  <ElementType Name="DeptNo"/>
  <ToColumn>
    <Column Name="DeptNo"/>
  </ToColumn>
</PropertyMap>
</ClassMap>
</Maps>
</XMLToDBMS>
```

Cuando se utiliza el componente `XMLDBMSQuery` para consultar la tabla de base de datos `XmlEmployee`, el elemento `XmlEmployee` debe ser la raíz. Por ello, el archivo de asignación debe indicar a XML-DBMS que pase por alto la raíz actual, el elemento plural `XmlEmployees`, y utilice en su lugar el elemento singular `XmlEmployee`.

Si el archivo `EmployeesQuery.map` no existe, como en el proyecto de ejemplo, es necesario efectuar personalmente los cambios en el archivo de asignación. Se debe añadir el bloque de código que comienza por `<IgnoreRoot>` y termina por `</IgnoreRoot>`. También se debe cambiar el nombre de la tabla de salida a "Conjunto resultado". Por último, se elimina este bloque de código:

Paso 4: Utilización de la aplicación de prueba de ejemplo

```
<ClassMap>
  <ElementType Name="XmlEmployees"/>
  <ToRootTable>
    <Table Name="XmlEmployees"/>
    <CandidateKey Generate="Yes">
      <Column Name="XmlEmployeesPK"/>
    </CandidateKey>
  </ToRootTable>
  <RelatedClass KeyInParentTable="Candidate">
    <ElementType Name="XmlEmployee"/>
    <CandidateKey Generate="Yes">
      <Column Name="XmlEmployeesPK"/>
    </CandidateKey>
    <ForeignKey>
      <Column Name="XmlEmployeesFK"/>
    </ForeignKey>
  </RelatedClass>
</ClassMap>
```

Enhorabuena. Ha finalizado este tutorial. Si desea más información sobre la compatibilidad XML en JBuilder, consulte el [Capítulo 1, "Introducción"](#).

Tutorial: Transferir datos con componentes XML de bases de datos basados en plantillas

Es una función de JBuilder Enterprise.

En este tutorial se explica la forma de utilizar los componentes XML de base de datos basados en plantillas de JBuilder para recuperar datos de una base de datos y pasarlos a un archivo XML. Se utiliza el ejemplo `XBeans.jpx`, que reside en el directorio `<jbuilder>/samples/Tutorials/XML/database/`. Los usuarios con acceso de sólo lectura a los ejemplos de JBuilder deberán copiar el directorio de ejemplos en un directorio con permisos de lectura y escritura.

Los componentes basados en plantillas generan los documentos XML correspondientes a las consultas formuladas. La consulta se sustituye por el documento XML como resultado de la aplicación de la plantilla. La solución basada en plantillas es muy flexible, ya que no existe una relación predefinida entre el documento XML y el conjunto de metadatos de base de datos que se consulta. El formato del documento XML es relativamente sencillo. Se puede elegir si se desea presentar el documento XML resultante con la hoja de estilo por defecto o con una personalizada.

En este tutorial se explica la forma de hacer lo siguiente:

- Transferir datos de una tabla de base de datos a un documento XML utilizando el componente `XTable`.
- Transferir datos de una tabla de base de datos a un documento XML por medio de una sentencia SQL creada por el componente `XQuery`.
- Utilizar los personalizadores de `XTable` y `XQuery` para definir las propiedades y ver el resultado de su configuración en la transferencia de los datos.

Para seguir este tutorial se necesitan ciertos conocimientos sobre el funcionamiento de JBuilder y XML. Si ésta es la primera vez que utiliza JBuilder, consulte “El entorno de JBuilder” (Ayuda | Entorno de JBuilder). Si desea más información sobre las funciones XML de JBuilder, consulte el [Capítulo 1, “Introducción”](#).

El apartado Opciones de accesibilidad en las 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 1-3.

Paso 1: Procedimientos iniciales

En este tutorial se utiliza la tabla de base de datos XmlEmployee que se ha creado en el [Capítulo 9, “Tutorial: Transferir datos con componentes XML de bases de datos basados en modelos”](#). Si aún no ha seguido este tutorial, hágalo ahora. Como mínimo, utilice el asistente XML-DBMS para crear los archivos de asignación y script SQL descritos en el tutorial, y ejecute las sentencias SQL para crear la tabla XmlEmployee. En el [Capítulo 9](#) se explica la forma de hacerlo.

- 1 Abra el proyecto de ejemplo `<jbuilder>/samples/Tutorials/XML/database/XBeans.jpx`.
- 2 Expanda el nodo de paquetes `com.borland.samples.xml.XBeans` para localizar la aplicación de prueba de ejemplo `XBeans_Test.java`. Haga doble clic en esta aplicación de prueba de ejemplo para abrirla en el editor.

Paso 2: Utilización de la aplicación de prueba de ejemplo

Normalmente, cuando se utilizan los componentes de base de datos XML de JBuilder, se desarrolla al mismo tiempo una aplicación que presenta a los usuarios una interfaz para manejar el programa. En este tutorial no se va a hacer esto, sino que se va a utilizar la aplicación de prueba de ejemplo, `XBeans_Test.java`, que simplemente es una clase Java que contiene los componentes de base de datos XML basados en plantillas y define sus propiedades. En este tutorial se explica la forma de trabajar con los personalizadores de los componentes para definir propiedades y ver el resultado de la transferencia de datos. Cuando esté seguro de que la transferencia funciona correctamente puede crear la aplicación de interfaz.

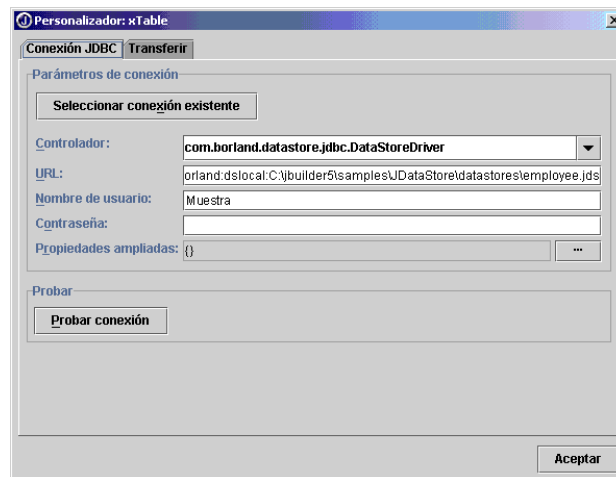
Examine el código fuente y verá que contiene los dos componentes, `xTable` y `xQuery`. Si estuviera creando una aplicación de prueba propia, le bastaría con añadirle estos componentes, de la forma siguiente:

- 1 Abra en el editor la clase de la aplicación.
- 2 Haga clic en la pestaña Diseño.
- 3 Pulse la pestaña XML de la paleta de componentes.
- 4 Seleccione el componente `XTable` y arrástrelo al diseñador de interfaces.
- 5 Seleccione el componente `XQuery` y arrástrelo al diseñador de interfaces.

Paso 3: Utilización del personalizador de XTable

Para empezar a trabajar con el componente `XTable`:

- 1 Abra `XBeans_Test.java` en el editor y pulse sobre la pestaña Diseño para abrir el Diseñador de interfaces de usuario. Verá que la carpeta Otros del panel de estructura contiene los dos componentes basados en plantilla.
- 2 Haga clic con el botón derecho del ratón en `xTable`, en el panel de estructura, y seleccione Personalizador en el menú. Aparece el personalizador de `xTable`.



Introducción de información sobre la conexión JDBC

Este tutorial utiliza la base de datos `employee.jds` de `JdataStore`, ubicada en el directorio `<jbuilder>/samples/JDataStore/datastores`. Si ha trabajado con los ejemplos de `JDataStore`, es posible que ya tenga una conexión JDBC con esta base de datos. Si es así, pulse el botón **Seleccionar conexión**

existente y elíjala. Si se emplea este método, los parámetros de conexión se completan automáticamente. Si no utiliza una conexión ya establecida, deberá introducir la información según se describe en los siguientes pasos:

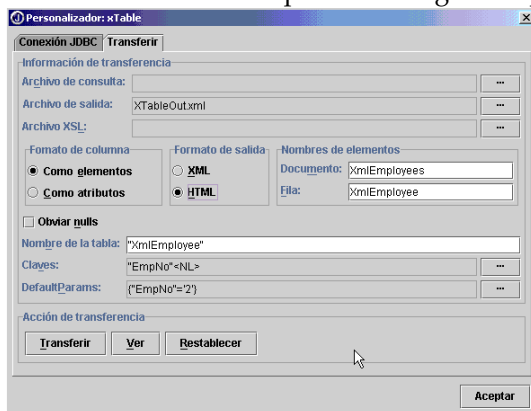
- 1 Seleccione `com.borland.datastore.jdbc.DataStoreDriver` como controlador, en la lista desplegable. Es necesario tener `JDataStore` instalado en el sistema. Si desea obtener más información acerca del trabajo con `JDataStore`, consulte “Fundamentos de `JDataStore`” en la *Guía del desarrollador de `JDataStore`*.
- 2 Indique la URL del almacén de datos que está utilizando, `employee.jds`. Cuando se elige `DataStoreDriver` como controlador aparece un modelo para ayudar a introducir la URL. Suponiendo que haya instalado `JBuilder` en la unidad C de su sistema, la URL del almacén de datos `employee.jds` del directorio `samples` es la siguiente:

```
jdbc:borland:dslocal:C:\<jbuilder>\samples\JDataStore\datastores\employee.jds
```
- 3 Escriba `Sample` en Nombre de usuario.
- 4 Introduzca un valor en el campo Contraseña o déjelo en blanco ya que `employee.jds` no requiere contraseña.
- 5 No rellene el campo Propiedades ampliadas.
- 6 Para comprobar que la conexión JDBC es correcta, pulse el botón Probar conexión. Junto al botón aparece un mensaje que indica si la conexión se ha establecido correctamente.
- 7 Para guardar la conexión recién creada, seleccione Guardar información de la conexión.

Transferencia de datos desde la base de datos a XML

A continuación, se transferirán los datos desde la base de datos hasta un documento XML.

- 1 Abra la ficha Transferir para ver la siguiente página del personalizador:

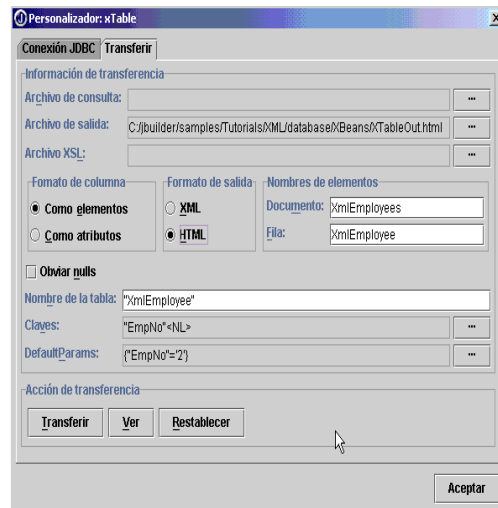


La aplicación de prueba de ejemplo rellena la mayor parte de la información necesaria. Sólo tendrá que aceptar la mayoría de las opciones predeterminadas y modificar el campo Archivo de salida, de modo que incluya la ruta de acceso completa al archivo.

2 Introduzca la siguiente información de transferencia:

- a Deje en blanco el campo Archivo de consulta, ya que no es necesario en este tutorial. Para obtener más información acerca de la depuración de archivos, consulte [“Configuración de propiedades con un documento de consulta XML” en la página 3-21](#).
- b Seleccione el botón puntos suspensivos (...), situado junto al campo Archivo de salida, con el fin de especificar el nombre del documento al que desee transferir los datos. Desplácese hasta el proyecto y acepte el nombre de archivo predeterminado, `XTableOut.html`. Debe especificar siempre la ruta de acceso completa para el archivo de salida. Pulse Aceptar.
- c No rellene el campo Archivo XSL. En este tutorial se utiliza la hoja de estilo por defecto del componente.
- d Acepte todos los valores por defecto.

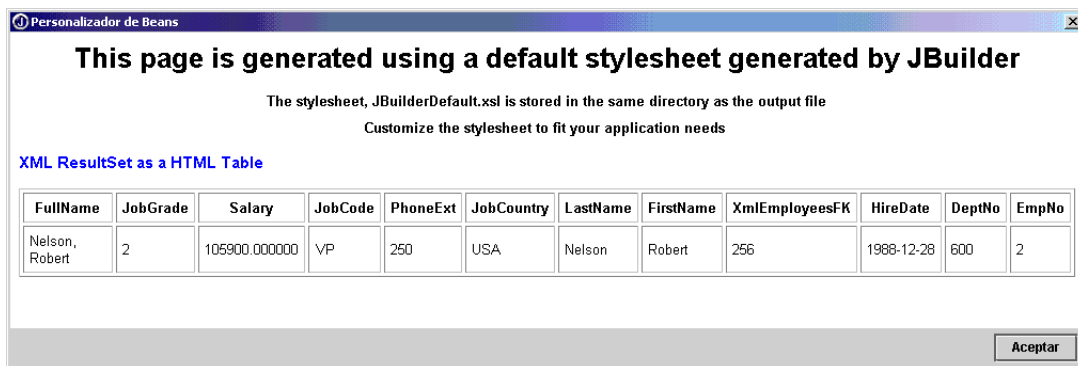
La ficha Transferir presenta un aspecto similar al siguiente:



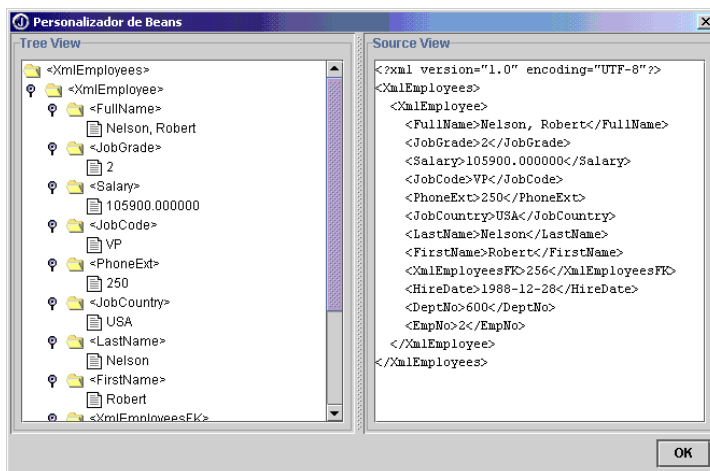
Ahora se pueden transferir los datos.

1 Seleccione Transferir.

- 2 Pulse Ver para observar el aspecto de la transferencia. El resultado, que utiliza la hoja de estilo HTML por defecto, tiene el siguiente aspecto:



- 3 Pulse Aceptar para cerrar la ficha.
- 4 Seleccione la opción Archivo de salida XML. Observe también que, ahora, el archivo de salida tiene la extensión .xml. Pulse Ver para ver la estructura de árbol XML por defecto.



- 5 Pulse Aceptar dos veces para cerrar la vista y el personalizador.

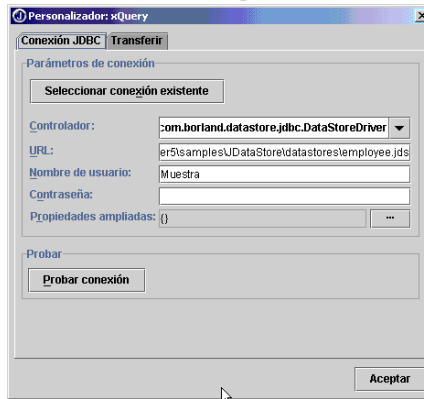
Paso 4: Utilización del personalizador de XQuery

También se puede utilizar una sentencia SQL para recuperar datos de una tabla de base de datos mediante el componente XQuery.

Para empezar a trabajar con el componente XQuery de la aplicación de prueba de ejemplo:

- 1 Vuelva a XBeans_Test.java y al diseñador de interfaces de usuario.

- 2 Haga clic con el botón derecho del ratón en `xQuery`, en el panel de estructura, y seleccione Personalizador en el menú. Aparece el personalizador de `xQuery`.



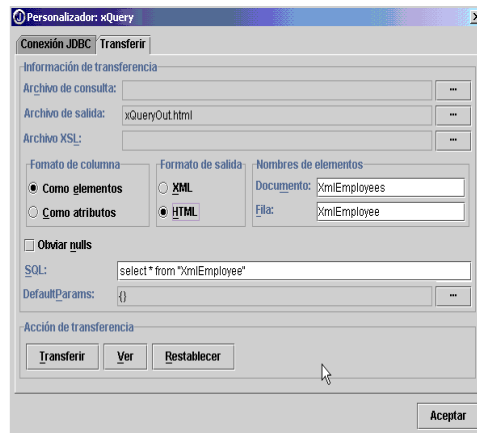
Selección de una conexión JDBC

Al igual que hizo con el componente `XTable`:

- 1 Pulse el botón Seleccionar conexión existente y especifique la conexión con `employee.jds` establecida anteriormente.
- 2 Pulse la pestaña Transferir para pasar a la siguiente ficha.

Transferencia de datos con una sentencia SQL

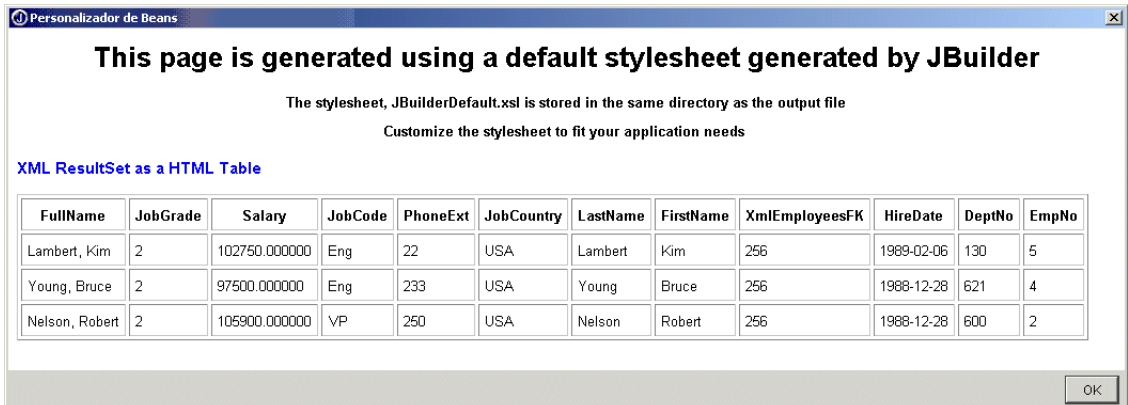
La ficha Transferir aparece con el siguiente aspecto:



La aplicación de prueba de ejemplo rellena la información necesaria. Deberá modificar el campo Archivo de salida de modo que incluya la ruta de acceso completa al archivo.

1 Introduzca la siguiente información de transferencia:

- a Deje en blanco el campo Archivo de consulta, ya que no es necesario en este tutorial. Para obtener más información acerca de la depuración de archivos, consulte el [“Configuración de propiedades con un documento de consulta XML” en la página 3-21](#)
 - b Seleccione el botón puntos suspensivos (...), situado junto al campo Archivo de salida, con el fin de especificar el nombre del documento al que desee transferir los datos. Desplácese hasta el proyecto y acepte el nombre de archivo predeterminado, `XQueryOut.html`. Debe especificar siempre la ruta de acceso completa para el archivo de salida.
 - c No rellene el campo Archivo XSL. En este tutorial se utiliza la hoja de estilo por defecto del componente.
 - d Acepte los restantes valores por defecto. Con el valor especificado en el campo SQL, `select * from "XmlEmployee"`, se ejecutará una consulta en la base de datos y se devolverán todos los registros de empleados. Para obtener más información sobre la ficha Transferir, consulte [“Configuración de propiedades con el personalizador” en la página 3-9](#).
- 2 Elija Ver para mostrar el resultado con la hoja de estilo HTML por defecto:



This page is generated using a default stylesheet generated by JBuilder

The stylesheet, JBuilderDefault.xsl is stored in the same directory as the output file

Customize the stylesheet to fit your application needs

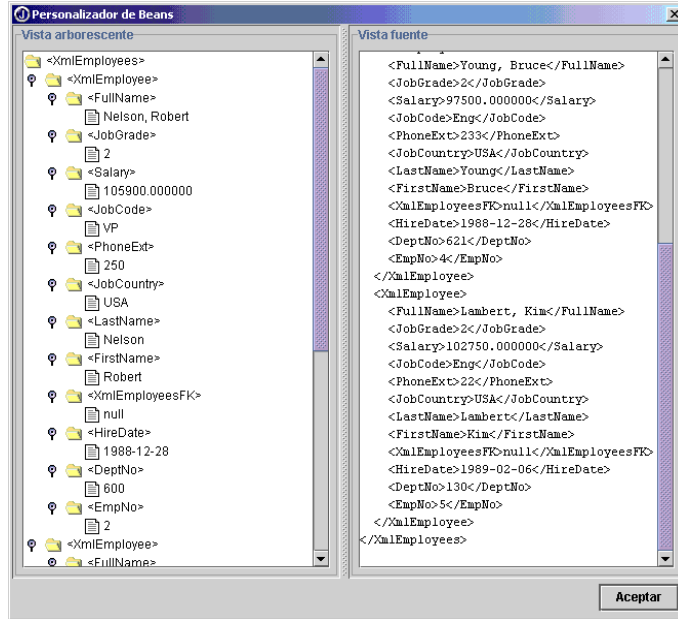
[XML ResultSet as a HTML Table](#)

FullName	JobGrade	Salary	JobCode	PhoneExt	JobCountry	LastName	FirstName	XmlEmployeesFK	HireDate	DeptNo	EmpNo
Lambert, Kim	2	102750.000000	Eng	22	USA	Lambert	Kim	256	1989-02-06	130	5
Young, Bruce	2	97500.000000	Eng	233	USA	Young	Bruce	256	1988-12-28	621	4
Nelson, Robert	2	105900.000000	VP	250	USA	Nelson	Robert	256	1988-12-28	600	2

OK

3 Pulse Aceptar para cerrar la ficha.

- 4 Ahora, seleccione la opción Formato de salida XML. Observe también que, ahora, el archivo de salida tiene la extensión .xml. Pulse Ver para ver la estructura de árbol XML por defecto.



- 5 Pulse Aceptar para cerrar la vista.

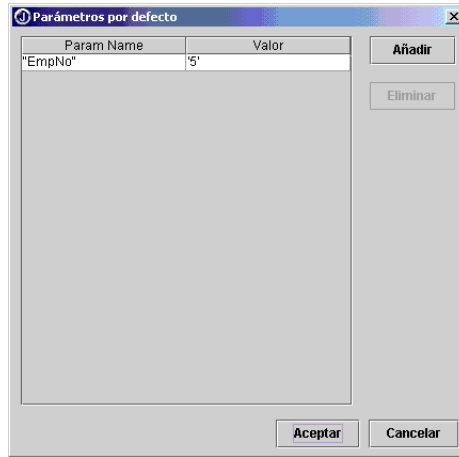
A continuación, pruebe con una consulta parametrizada como sentencia SQL:

- 1 Escriba esta sentencia en el campo SQL.

```
select * from "XmlEmployee" where "EmpNo" = : "EmpNo"
```

Importante Escriba un espacio **antes**, pero no después, de los dos puntos.

- 2 Pulse el botón de puntos suspensivos (...) junto al campo Parámetros por defecto para abrir el cuadro de diálogo del mismo nombre y añada. "EmpNo" como nombre del parámetro y '5' como valor:



- 3 Pulse Aceptar para cerrar el cuadro de diálogo Parámetros por defecto.
- 4 Seleccione Ver. Los resultados muestran el registro del empleado 5.

Enhorabuena. Ha finalizado este tutorial. Si desea más información sobre la compatibilidad XML en JBuilder, consulte el [Capítulo 1, "Introducción"](#).

Índice

A

- analizadores
 - Crimson 2-20
 - Xerces 2-11, 2-27
- analizadores Xerces 2-14
- analizar sintácticamente
 - documentos XML 2-11
 - SAX 2-33
 - Xerces 2-27
- API sencilla para XML (SAX) 2-33
- asignación relacional de objeto 3-8
- asistencia para bases de datos XML 2-35
- asistentes
 - Gestión de SAX 2-33
 - Gestor de SAX 2-33
 - para aplicaciones web Cocoon 2-14
 - para la asociación de datos 2-31, 3-1
 - para pasar de DTD a XML 2-4, 2-6, 3-4
 - para XML 2-4
 - XML-DBMS 2-5
 - para XML
 - Consulte también* asistentes
- asociación de datos 2-30
 - BorlandXML 2-31
- atributos 2-24
- ATTLIST (definición) 2-24

B

- bases de datos
 - compatibilidad con XML 2-35
- bibliotecas XML 1-5
- Borland
 - asistencia
 - a desarrolladores 1-7
 - técnica 1-7
 - contacto 1-7
 - e-mail 1-6
 - grupos de noticias 1-5
 - informar sobre errores 1-6
 - recursos en línea 1-6
 - World Wide Web 2-31

C

- Castor
 - archivo de propiedades 2-33
 - asociación de datos 2-30, 3-1
- clases
 - de Java

- generación desde esquema 2-30, 3-1
- generar con el Asistente para la asociación de datos 2-31
- generar desde DTD 2-30
- generación desde esquema 2-20, 2-31
- generar con el Asistente para la asociación de datos 2-31
- generar desde DTD 2-20
- componentes
 - de base de datos
 - basados en modelo XML 3-13
 - XML 3-2, 3-13
 - XML basados en plantillas 3-1
 - XML
 - basados en modelos 2-35, 3-2, 3-8
 - asignar valores a propiedades 3-12
 - definir propiedades con el inspector 3-10
 - introducir información de transferencia 3-9
 - personalizadores 3-13
 - basados en plantillas 2-35, 3-2, 3-13
 - asignar valores a propiedades 3-21
 - definir propiedades con documento de consulta 2-2
 - definir propiedades con el inspector 3-14
- XMLDBMSQuery 2-35, 3-8
 - introducción de información de transferencia 3-9
- XMLDBMSTable 2-35, 3-8
- XQuery 2-35
- Xtable 2-35
- conexiones JDBC
 - establecer 3-2
- consultas parametrizadas 3-15
- controladores JDBC
 - especificar 3-2
- convenciones
 - de la documentación 1-4, 2-33
 - de plataformas 2-33
- Crimson
 - analizar XML 2-20

D

- datos
 - desmontaje de parámetros 2-31, 3-21
 - montaje de parámetros 2-31, 3-21
 - transferir entre XML y bases de datos 2-20
- Definición de tipo de documento (DTD) 2-4, 2-11
 - Consulte también* DTD

distinción entre mayúsculas y minúsculas
elementos XML 3-21

documentos

de consulta XML 2-2, 3-19

XML

- analizar sintácticamente 2-33
- creación en el editor 2-2, 2-6
- crear desde una DTD 2-4
- crear DTD desde XML 2-12
- errores 2-24
- forma correcta 2-12
- manipular mediante programa 2-20
- transformar 2-11
- validar con DTD 2-7
- validar con esquemas (XSD) 2-11
- visualizar 2-4

DTD

- crear a partir de documentos XML 3-4
- crear XML desde DTD 2-11
- definición 2-4, 2-11
- validación de XML 2-12

E

elemento raíz

- definición 2-31

errores de validación 2-11

esquemas 2-30

- definición 2-13
- validar 2-11

F

forma correcta de XML

- definición 2-31

G

generación de clases Java

- BorlandXML 1-6
- Castor 3-1

gestores de SAX

- crear 2-33

gramática XML

- validar 2-12

grupos de noticias

- public 2-8

H

hojas de estilo

- aplicar hojas de estilo
 - en cascada a documentos XML 2-7
 - XSL a documentos XML 2-11
- en cascada (CSS) 2-26

- hojas de estilo
 - por defecto XSLT 2-7

J

JAXP 2-14, 2-20

JDK 1,4

- procesado de XML 2-20

L

La API de Java para el procesado de XML
(JAXP) 2-20

M

Marco de publicación de XML Cocoon 2-10

mensajes de error de XML 2-12

montaje de parámetros

- conversión entre Java y XML 2-34

O

opciones

- del IDE

- configurar XML 3-9

- hoja de estilo por defecto 3-9

- XML 3-9

P

personalizadores

- componentes de base de datos XML 2-2

- XMLDBMSQuery 3-13

- establecer una conexión JDBC 3-18

- XMLDBMSTable 3-13

- establecer una conexión JDBC 3-18

- introducción de información de
transferencia 3-9

XQuery

- consultas parametrizadas 3-15

- establecer conexión JDBC 3-2

- introducción de información de
transferencia 3-19

- transferir a HTML 3-13

- transferir a XML 3-13

XTable

- establecer conexión JDBC 3-2

- introducción de información de
transferencia 3-19

- transferir a HTML 3-13

- transferir a XML 3-13

- utilización de parámetros 3-15

procesado de XML

- JDK 1,4 2-20

procesador de hojas de estilo Xalan 2-14
publicación XML 2-25

S

SAX (API sencilla para XML) 2-33

T

transferencia de datos

- de Java a XML 2-23, 2-31

- de XML a Java 2-23, 2-31

transformación de XML

- transformar opciones de inspección 3-9, 8-1

- Xalan 2-30

tutoriales

- asociación de datos de esquema con Castor 7-1

- asociación de datos DTD con BorlandXML 6-1

- creación de un gestor de SAX 4-1

- creación y validación de documentos XML 9-1

- transferencia de datos con componentes XML

 - de base de datos basados en modelos 10-1

- transferencia de datos con componentes XML
- de base de datos basados en plantillas 5-1
- transformar documentos XML 2-13

U

Usenet, grupos de noticias 2-8

V

validar documentos XML

- con esquemas (XSD) 2-2

visor XML

- activar 2-26, 3-9

X

XML (transformación) 3-4

XML-DBMS 2-35, 3-2, 3-8

- ubicación 3-8

