



Guía del desarrollador de aplicaciones Web



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. Diríjase 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.

JBE0080WW21002webapps 3E3R1002
0203040506-9 8 7 6 5 4 3 2 1
PDF

Índice de materias

Capítulo 1		
Introducción	1-1	
Convenciones de la documentación	1-4	
Asistencia y recursos para desarrolladores	1-6	
El servicio de asistencia técnica de Borland	1-6	
Recursos en línea	1-7	
World Wide Web	1-7	
Grupos de noticias de Borland	1-7	
Usenet, grupos de noticias	1-8	
Información sobre errores	1-8	
Capítulo 2		
Introducción general al proceso de desarrollo de aplicaciones web	2-1	
Servlets	2-2	
Páginas JavaServer (JSP)	2-3	
InternetBeans Express	2-5	
Struts	2-5	
JSTL (Biblioteca de etiquetas estándar para Páginas JavaServer)	2-6	
Applets	2-7	
Qué tecnologías utilizar en su aplicación Web	2-8	
El proceso básico de desarrollo de aplicaciones web	2-9	
Aplicaciones web y aplicaciones distribuidas	2-9	
Capítulo 3		
Las WebApps y los archivos WAR	3-1	
La WebApp	3-1	
Archivos recopilatorios Web (WAR)	3-2	
Herramientas para trabajar con WebApps y archivos WAR	3-3	
Creación de una WebApp con el Asistente para aplicaciones web	3-3	
La WebApp y sus propiedades	3-5	
Directorio raíz	3-6	
Descriptores de distribución	3-7	
Propiedades de WebApp	3-7	
La ficha WebApp	3-8	
La ficha Directorios	3-9	
La ficha Clases	3-11	
La ficha Dependencias	3-13	
La ficha Descriptor	3-14	
El archivo WAR	3-15	
Applets en el archivo WAR	3-18	
Capítulo 4		
Los servlets	4-1	
Servlets y JSP	4-2	
Servlets y servidores web	4-3	
La API de servlet	4-4	
El paquete servlet.HTTP	4-5	
El ciclo de vida de los servlets	4-6	
Construcción e inicialización del servlet	4-6	
Gestión de las peticiones del cliente	4-7	
Servlets y multihilo	4-7	
Destrucción de un servlet	4-7	
Servlets con enlace a HTML	4-8	
Servlets específicos para HTTP	4-8	
Cómo se utilizan los servlets	4-9	
Distribución de servlets	4-9	
Capítulo 5		
Los servlets en JBuilder	5-1	
Opciones del Asistente para servlets	5-1	
Ficha Seleccione el nombre y el tipo de servlet	5-1	
Ficha Modificar detalles del servlet estándar	5-3	
Opción Generar contenido de tipo	5-4	
Opciones de Implementar métodos	5-5	
Opciones de los archivos SHTML	5-6	
Ficha Introduzca los detalles de webapp	5-7	
Ficha Introduzca los parámetros de la petición del servlet	5-9	
Ficha Introduzca los detalles del servlet monitor	5-10	
Ficha Definir configuración de servlet	5-11	
Cómo llamar a los servlets	5-12	
Llamada a un servlet desde una ventana de un visualizador	5-12	
Llamada a un servlet desde una página HTML	5-13	
Internacionalización de servlets	5-14	
Escrutura de servlets enlazados a datos	5-15	
Capítulo 6		
Desarrollo de Páginas JavaServer	6-1	
Etiquetas JSP	6-3	
Marcos de trabajo y bibliotecas de etiquetas JSP	6-4	
JSP en JBuilder	6-5	

Las bibliotecas de etiquetas JSP y los marcos de trabajo en JBuilder	6-5	Asistente para JSP a partir de ActionForm	8-12
Utilización del cuadro de diálogo		Ficha WebApp, JSP y ActionForm	8-12
Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario	6-6	Ficha Tipos de etiquetas para campos ActionForm	8-13
Desarrollo de las JSP	6-10	Ficha Indique las opciones para crear esta JSP Struts	8-14
El Asistente para JSP	6-10	Asistente para conversión de Struts	8-15
Compilación de las JSP	6-13	Ficha Indique las fichas para convertir a Struts	8-16
Ejecución web de las JSP.	6-13	Ficha Etiquetas que se han de convertir	8-16
Depuración web de las JSP	6-14	Ficha Especifique las opciones para convertir etiquetas a Struts	8-17
Distribución de las JSP.	6-14	Editor de configuración de Struts	8-18
Recursos adicionales de JSP	6-14	Implementaciones del marco de trabajo Struts en JBuilder	8-19
Capítulo 7		Creación de una aplicación web preparada para Struts en JBuilder	8-20
InternetBeans Express	7-1		
Descripción general de las clases de InternetBeans Express	7-2		
Utilización de InternetBeans Express con servlets	7-3		
Visualización de páginas web dinámicas con servlets mediante InternetBeans Express.	7-3		
Envío de datos con servlets mediante InternetBeans Express	7-5		
Análisis de páginas.	7-5		
Generación de tablas.	7-6		
Utilización de InternetBeans Express con JSP	7-6		
Tabla de etiquetas de InternetBeans	7-9		
Formato de internetbeans.tld	7-10		
Capítulo 8			
El marco de trabajo Struts en JBuilder	8-1		
Versiones beta Struts 1.0 y 1.1	8-3		
Herramientas de JBuilder para Struts	8-3		
Aceptación del marco de trabajo Struts	8-4		
Asistente para aplicaciones web preparado para Struts.	8-5		
Asistente para JSP preparado para Struts	8-6		
Asistente para ActionForm	8-7		
Ficha Aplicación web e información de clases para ActionForm.	8-8		
Ficha Definición de campos para ActionForm	8-8		
Ficha Selección de opciones adicionales	8-9		
Asistente para Action	8-10		
Ficha Nombre y WebApp para Action	8-10		
Ficha Información de configuración	8-11		
Capítulo 9			
Configuración del servidor web	9-1		
Visualización de las configuraciones de Tomcat	9-1		
Configuración de otros servidores web	9-3		
Elección de un servidor para su proyecto	9-4		
Configuración del IDE para ejecutar/depurar web	9-6		
Capítulo 10			
Las aplicaciones web en JBuilder	10-1		
Creación de una configuración de ejecución.	10-2		
Creación de una configuración de ejecución con los asistentes.	10-2		
Creación de una configuración de ejecución de applets.	10-3		
Creación de una configuración de ejecución del servidor.	10-5		
Cómo las URL ejecutan los servlets	10-11		
Configuración de propiedades de ejecución	10-14		
Compilación de servlets o JSP	10-15		
Ejecución web de un servlet o una JSP	10-16		
Inicio del servidor web	10-16		
Vista web	10-17		
Ver código fuente web	10-18		
Detención del servidor web	10-19		
Depuración web del servlet o de la JSP.	10-19		

Capítulo 11	
Distribución de aplicaciones web	11-1
Aspectos generales	11-1
Archivos recopilatorios	11-1
Descriptores de distribución	11-2
Applets	11-2
Servlets	11-2
JSP	11-3
Comprobación de su aplicación web.	11-4
Modificación del descriptor de distribución	11-4
Edición de descriptores de distribución	
específicos de un fabricante	11-5
Mas información sobre descriptoros de	
distribución	11-5
Capítulo 12	
Modificación del archivo web.xml	12-1
Menú contextual del Editor DD de WebApp	12-2
Ficha Descriptor de distribución para la	
WebApp	12-3
Ficha Parámetros contextuales	12-4
Ficha Filtros	12-5
Ficha Monitores	12-8
Ficha Servlets	12-9
Ficha Bibliotecas de etiquetas.	12-12
Ficha Tipo de MIME	12-13
Ficha Páginas de error	12-15
Ficha Entradas de entorno	12-15
Ficha Referencias EJB	12-16
Ficha Referencias EJB locales	12-17
Ficha Referencias de la factoría de conexión	
del gestor de recursos	12-17
Referencias del entorno de recursos	12-18
Ficha Conexión	12-19
Ficha Seguridad	12-20
Restricciones de seguridad	12-21
Colección de recursos web	12-22
Capítulo 13	
Modificación del archivo struts-config.xml	13-1
Selección de una ficha del Editor de	
configuración de Struts	13-2
El menú contextual del Editor de	
configuración de Struts	13-3
Ficha Fuentes de datos	13-3
Menú contextual de la ficha Fuentes de	
datos	13-5
Configuración de los atributos de las	
propiedades	13-6
Ficha Form Beans	13-6
Menú contextual de la ficha Form Beans . .	13-9
Configuración de los atributos de las	
propiedades	13-9
Ficha Reenvíos globales	13-9
Menú contextual de la ficha Reenvíos	
globales	13-12
Configuración de los atributos de las	
propiedades	13-13
Ficha Correspondencias Action	13-13
Menú contextual de la ficha	
Correspondencias Action.	13-17
Configuración de los atributos de las	
propiedades	13-18
Capítulo 14	
Las applets	14-1
Cómo funcionan	14-2
La etiqueta <applet>	14-2
Etiqueta de ejemplo <applet>	14-3
Atributos de las etiquetas <applet>	14-3
Errores habituales en la etiqueta <applet> .	14-4
Los navegadores	14-5
Compatibilidad con Java	14-6
Obtención del navegador apropiado	14-6
Compatibilidad con varios navegadores . .	14-7
Diferencias en la implementación de Java .	14-7
Soluciones a los problemas con los	
navegadores	14-8
Consejos adicionales para que funcionen las	
applets	14-9
La seguridad y el Administrador de	
seguridad	14-11
El espacio de contención	14-11
Restricciones de las applets	14-12
Soluciones a problemas de seguridad. . .	14-12
Utilización de bibliotecas de terceros.	14-14
Distribución de las applets.	14-14
Comprobación de las applets	14-15
Pasos básicos de la comprobación.	14-15
Comprobación en los navegadores	14-16
JBuilder y las applets	14-17
Creación de applets con el Asistente para	
applets	14-17
Ejecución de applets	14-21
El Applet TestBed de JBuilder y el	
appletviewer de Sun.	14-22

Ejecución de applets JDK 1.1.x en JBuilder	14-23
Ejecución de applets JDK 1.2 en JBuilder	14-23
Depuración de applets	14-23
Depuración de applets en el Plug-in de Java	14-24
Distribución de applets en JBuilder	14-26
Capítulo 15	
Ejecución de aplicaciones web con Java Web Start	15-1
Consideraciones sobre las aplicaciones Java Web Start	15-2
Instalación de Java Web Start	15-3
Modificación de la definición de bibliotecas	
Web Start de JBuilder	15-4
Web Start y JDK 1.3 ó 1.2	15-4
Java Web Start y JBuilder	15-5
El archivo JAR de la aplicación	15-7
El archivo JNLP y la página de inicio de la aplicación	15-7
Capítulo 16	
Tutorial: Creación de un servlet sencillo	16-1
Paso 1: Creación de un proyecto	16-2
Paso 2: Selección de un servidor	16-2
Paso 3: Creación de una WebApp	16-3
Paso 4: Creación del servlet con el Asistente para servlets	16-4
Paso 5: Adición de código al servlet	16-7
Paso 6: Compilación y ejecución del servlet	16-8
Capítulo 17	
Tutorial: Creación de un servlet que actualiza un libro de visitas	17-1
Paso 1: Creación de un proyecto	17-2
Paso 2: Selección de un servidor	17-2
Paso 3: Creación de una WebApp	17-3
Paso 4: Creación de los servlets	17-4
Paso 5: Creación del módulo de datos	17-9
Paso 6: Cómo añadir componentes de bases de datos al módulo de datos	17-10
Paso 7: Creación de la conexión de datos con el DBServlet	17-14
Paso 8: Creación de un formulario de entrada en el FormServlet	17-15
Capítulo 18	
Tutorial: Creación de una JSP mediante el Asistente para JSP	18-1
Paso 1: Creación de proyectos	18-2
Paso 2: Selección de un servidor	18-2
Paso 3: Creación de una WebApp	18-3
Paso 4: Creación de la JSP	18-4
Paso 5: Adición de funciones al JavaBean	18-5
Paso 6: Modificación del código de la JSP	18-6
Paso 7: Ejecución de la página JSP	18-7
Utilización de la pestaña Ver web	18-9
Depuración de las JSP	18-9
Distribución de la JSP	18-9
Capítulo 19	
Tutorial: Creación de un servlet con InternetBeans Express	19-1
Paso 1: Creación de proyectos	19-2
Paso 2: Selección de un servidor	19-2
Paso 3: Creación de una WebApp	19-2
Paso 4: Creación del servlet	19-3
Paso 5: Creación del módulo de datos	19-5
Paso 6: Diseño de la página de plantilla HTML	19-6
Paso 7: Conexión del servlet al DataModule	19-8
Paso 8: Diseño del servlet	19-9
Paso 9: Edición del servlet	19-11
Paso 10: Configuración de dependencias de la WebApp	19-12
Paso 11: Ejecución del servlet	19-12
Distribución del servlet	19-13
Capítulo 20	
Tutorial: Creación de una página JSP con InternetBeans Express	20-1
Paso 1: Creación de proyectos	20-2
Paso 2: Selección de un servidor	20-2
Paso 3: Creación de una WebApp	20-2
Paso 4: Utilización del Asistente para JSP	20-3

Paso 5: Diseño de la parte HTML de la JSP . . .	20-5	
Paso 6: Adición de la etiqueta de base de datos de InternetBeans	20-6	
Paso 7: Adición de la etiqueta consulta de InternetBeans	20-7	
Paso 8: Adición de la etiqueta de tabla de InternetBeans	20-7	
Paso 9: Adición de las etiquetas de control de InternetBeans	20-8	
Paso 10: Adición de la etiqueta de envío de InternetBeans	20-9	
Paso 11: Adición del método submitPerformed()	20-9	
Paso 12: Adición del código para insertar una fila	20-10	
Paso 13: Adición de la biblioteca JDataStore Server al proyecto	20-10	
Paso 14: Ejecución de la página JSP	20-11	
Distribución de la JSP	20-12	
		Capítulo 21
		Tutorial: Ejecución de la aplicación de ejemplo CheckBoxControl con Java Web Start
		21-1
Paso 1: Apertura y configuración del proyecto	21-2	
Paso 2: Creación de la WebApp de la aplicación	21-3	
Paso 3: Creación del archivo JAR de la aplicación	21-4	
Paso 4: Creación de la página de inicio y del archivo JNLP de la aplicación	21-6	
Paso 5: Creación de una configuración de ejecución del servidor.	21-8	
Paso 6: Inicio de la aplicación	21-10	
		Índice
		I-1

Tablas

1.1	Convenciones tipográficas y de símbolos.	1-4
1.2	Convenciones de las plataformas	1-5
2.1	Tecnologías para aplicaciones web	2-1
3.1	Herramientas de JBuilder para WebApp y archivos WAR	3-3
4.1	Introducción general a la API de Servlet .	4-4
4.2	Interfaces y paquetes de clases servlet utilizados habitualmente	4-5
5.1	Opciones de tipo de Servlet.	5-2
6.1	Etiquetas JSP habituales.	6-3
7.1	Clases de InternetBeans Express.	7-2
7.2	Etiquetas InternetBeans Express	7-9
9.1	Parámetros del cuadro de diálogo Configurar servidores para Tomcat	9-3
10.1	Árboles URI	10-9
10.2	Patrones de URL.	10-12
12.1	Ficha Descriptor de distribución para la WebApp del Editor DD de WebApp . .	12-3
12.2	Ficha Filtros del Editor DD para Webapp.	12-5
12.3	Ficha Filtro individual del Editor DD para Webapp.	12-7
12.4	Ficha Servlet individual del Editor DD de Webapp	12-11
12.5	Ficha Colección de recursos web del Editor DD de WebApp.	12-23
13.1	Atributos de Fuente de datos	13-5
13.2	Menú contextual de la ficha Fuentes de datos	13-5
13.3	Atributos de Form Bean	13-8
13.4	Menú contextual de la ficha Form Beans.	13-9
13.5	Atributos de Reenvío.	13-11
13.6	Menú contextual de la ficha Reenvíos globales	13-12
13.7	Atributos de Action	13-15
13.8	Menú contextual de la ficha Correspondencias Action	13-17
14.1	Atributos de las etiquetas <applet> . . .	14-3
15.1	Perspectiva general de la API del JNLP .	15-2
15.2	Opciones del Creador de recopilatorios .	15-5
15.3	Opciones del asistente para el inicio de Web Start	15-6
16.1	Opciones de parámetros del Asistente para servlets.	16-6

Figuras

3.1	Asistente para aplicaciones web	3-5
3.2	Panel de proyecto que muestra un nodo WebApp	3-6
3.3	Ficha WebApp del cuadro de diálogo Propiedades de WebApp	3-9
3.4	Ficha Directorios del cuadro de diálogo Propiedades de WebApp	3-11
3.5	Ficha Clases del cuadro de diálogo Propiedades de WebApp	3-13
3.6	Ficha Dependencias del cuadro de diálogo Propiedades de WebApp.	3-14
3.7	La ficha Descriptor del cuadro de diálogo Propiedades de WebApp.	3-15
3.8	El nodo del archivo WAR abierto en JBuilder IDE	3-16
3.9	Cuadro de diálogo Propiedades del archivo WAR	3-17
5.1	Asistente para servlets — ficha Seleccione el nombre y el tipo de servlet.	5-3
5.2	Asistente para servlets— ficha Modificar detalles del servlet estándar	5-3
5.3	Asistente para servlets – Servlet estándar, ficha Introduzca los detalles de webapp .	5-8
5.4	Asistente para servlets – Servlet de filtro, ficha Introduzca los detalles de webapp .	5-9
5.5	Asistente para servlets – ficha Introduzca los parámetros de la petición del servlet .	5-10
5.6	Asistente para servlets - ficha Introduzca los detalles del servlet monitor	5-11
5.7	Asistente para servlets – Ficha Definir configuración de servlet	5-12
8.1	Struts — antes y después	8-3
8.2	Marco de trabajo Struts en Configurar bibliotecas	8-4
8.3	Asistente para aplicaciones Web	8-5
8.4	Ficha Modificar los detalles del archivo JSP – Asistente para JSP	8-7
8.5	Ficha Aplicación web e información de clases para ActionForm – Asistente para ActionForm	8-8
8.6	Ficha Definición de campos para ActionForm – Asistente para ActionForm	8-9
8.7	Ficha Selección de opciones adicionales – Asistente para ActionForm	8-9
8.8	Ficha Nombre y WebApp para Action — Asistente para Action.	8-11
8.9	Ficha Información de configuración — Asistente para Action	8-12
8.10	Ficha WebApp, JSP y ActionForm — Asistente para JSP a partir de ActionForm.	8-13
8.11	Ficha Tipos de etiquetas para campos ActionForm en JSP — Asistente para JSP a partir de ActionForm	8-14
8.12	Ficha Indique las opciones para crear esta JSP Struts — Asistente para JSP a partir de ActionForm	8-15
8.13	Ficha Indique las fichas para convertir a Struts – Asistente para conversión de struts	8-16
8.14	Ficha Etiquetas que se han de convertir — Asistente para conversión de struts	8-17
8.15	Ficha Especifique las opciones para convertir etiquetas a Struts — Asistente para conversión de struts	8-18
8.16	Editor de configuración de Struts	8-19
10.1	Mensajes Tomcat.	10-17
10.2	Salida de Ver web	10-18
10.3	Ver código fuente web	10-18
12.1	Ficha Descriptor de distribución para la WebApp del Editor DD de WebApp .	12-4
12.2	Ficha Parámetros contextuales del Editor DD de WebApp	12-5
12.3	Ficha Mapas de filtro del Editor DD de Webapp	12-6
12.4	Nodo de filtro individual en el Editor DD de Webapp.	12-8
12.5	Ficha Monitores del Editor DD de Webapp	12-9
12.6	Ficha Mapas del servlet del Editor DD de Webapp	12-10
12.7	Nodo de servlet individual en el Editor DD de WebApp	12-12
12.8	Ficha Bibliotecas de etiquetas en el Editor DD de WebApp	12-13
12.9	Ficha Tipos de MIME en el Editor DD de WebApp.	12-14
12.10	Ficha Páginas de error en el Editor DD de WebApp.	12-15
12.11	Ficha Entorno en el Editor DD de WebApp	12-16

12.12	Ficha Referencias EJB en el Editor DD de WebApp	12-17
12.13	Ficha Referencias de la factoría de conexión del gestor de recursos en Editor DD de WebApp	12-18
12.14	Ficha Referencias de variables de entorno en el Editor DD de WebApp	12-19
12.15	Ficha Login en el Editor DD de WebApp	12-20
12.16	Ficha Competencias de seguridad en el Editor DD de WebApp	12-21
12.17	Restricción de seguridad en el Editor DD de WebApp	12-22
12.18	Nodo de colección de recursos Web en el Editor DD de WebApp	12-23
13.1	Ficha de visión general de Fuentes de datos	13-4
13.2	Ficha de atributos de Fuentes de datos	13-4
13.3	Ficha de visión general de Form Beans	13-7
13.4	Ficha de atributos de Form Bean	13-7
13.5	Ficha de visión general de Reenvíos globales	13-10
13.6	Ficha de atributos de Reenvío	13-11
13.7	Ficha de visión general de Correspondencias Action	13-14
13.8	Ficha de atributos de Action	13-14
15.1	Vista web de Java Web Start	15-8
15.2	Navegador externo para Java Web Start	15-8
16.1	Servlet que se ejecuta en la vista Web	16-8
16.2	Servlet en ejecución después del envío del nombre	16-9
18.1	Nodo WebApp en el panel de proyecto	18-4
18.2	JSP en Vista Web	18-8
19.1	Nodo WebApp en el panel de proyecto	19-3
20.1	Nodo WebApp en el panel de proyecto	20-3
20.2	Ejecución de la JSP en la Vista Web	20-12

Tutoriales

Creación de un servlet simple	16-1	Express	19-1
Creación de un servlet que actualiza un libro de visitas	17-1	Creación de una JSP con InternetBeans Express	20-1
Creación de una JSP mediante el Asistente para JSP	18-1	Ejecución de la aplicación de ejemplo CheckBoxControl con Java Web Start.	21-1
Creación de un servlet con InternetBeans			

1

Introducción

El desarrollo web es una función de JBuilder Enterprise. El desarrollo de applets es una función de todas las ediciones de JBuilder

La *Guía del desarrollador de aplicaciones Web* presenta algunas de las tecnologías disponibles para desarrollar aplicaciones multinivel basadas en la tecnología Web. Una aplicación web es un conjunto de documentos HTML/XML, componentes web (servlets y páginas JSP) y otros recursos existentes en una estructura de directorio o en un formato de archivo conocido como Web ARchive (WAR). Las aplicaciones web se encuentran en un servidor central y prestan servicios a distintos clientes.

Este manual detalla cómo aparecen estas tecnologías en JBuilder y cómo trabajar con ellas en el IDE y en el editor. También explica cómo estas tecnologías se complementan entre sí en una aplicación web. Si desea obtener más información, seleccione uno de los siguientes temas:

- [Capítulo 2, “Introducción general al proceso de desarrollo de aplicaciones web”](#)

Presenta las tecnologías analizadas en este libro, incluyendo servlets, JavaServer Pages (JSP) e InternetBeans Express, Struts y applets.

- [Capítulo 3, “Las WebApps y los archivos WAR”](#)

Explica cómo crear un archivo y una aplicación web en un archivo WAR en JBuilder. Este capítulo analiza también la estructura y los conceptos generales de las WebApp.

- [Capítulo 4, “Los servlets”](#)

Presenta los servlets y la API de servlet.

- [Capítulo 5, “Los servlets en JBuilder”](#)
Explica las opciones del Asistente para servlets, cómo ejecutar los servlets, cómo internacionalizarlos y cómo crear servlets de enlace a datos.
- [Capítulo 6, “Desarrollo de Páginas JavaServer”](#)
Presenta las JSP y la API de JSP. Explica cómo utilizar el asistente para JSP para crear una JSP.
- [Capítulo 7, “InternetBeans Express”](#)
Explica la biblioteca InternetBeans y cómo utilizar los componentes con servlets y JSP.
- [Capítulo 8, “El marco de trabajo Struts en JBuilder”](#)
Explica el marco de trabajo Struts y el modo de crear una aplicación web preparada para Struts.
- [Capítulo 9, “Configuración del servidor web”](#)
Explica cómo configurar el servidor web para trabajar con JBuilder.
- [Capítulo 10, “Las aplicaciones web en JBuilder”](#)
Explica cómo compilar, ejecutar y depurar servlets y JSP.
- [Capítulo 11, “Distribución de aplicaciones web”](#)
Explica los diferentes aspectos de la distribución de las aplicaciones web.
- [Capítulo 12, “Modificación del archivo web.xml”](#)
Explica cómo se debe utilizar el Editor DD de la webapp para modificar el archivo `web.xml`.
- [Capítulo 13, “Modificación del archivo struts-config.xml”](#)
Explica el modo de utilización del Editor DD Struts con el fin de modificar el archivo `struts-config.xml`.
- [Capítulo 14, “Las applets”](#)
Explica cómo crear applets en JBuilder. Trata de los principales inconvenientes implicados en el desarrollo de un applet y su distribución, con sus soluciones.
- [Capítulo 15, “Ejecución de aplicaciones web con Java Web Start”](#)
Explica cómo utilizar Web Start para lanzar aplicaciones que no sean aplicaciones web desde un navegador web.

Puede consultar los siguientes tutoriales sobre aplicaciones web:

- [Capítulo 16, “Tutorial: Creación de un servlet sencillo”](#)

Muestra los pasos para escribir un servlet sencillo que acepta entradas de usuarios y cuenta el número de visitantes a una web.

- [Capítulo 17, “Tutorial: Creación de un servlet que actualiza un libro de visitas”](#)

Muestra los pasos para escribir un servlet que conecta con una base de datos JDataStore, acepta la introducción de datos por parte de los usuarios y guarda de nuevo los datos en la base de datos.

- [Capítulo 18, “Tutorial: Creación de una JSP mediante el Asistente para JSP”](#)

Muestra los pasos para escribir una JSP que acepta y presenta la entrada de los usuarios y cuenta cuántas veces ha sido visitada una página web.

- [Capítulo 19, “Tutorial: Creación de un servlet con InternetBeans Express”](#)

Muestra los pasos para escribir un servlet que utiliza componentes InternetBeans con el fin de consultar una tabla de una base de datos, mostrar su contenido, aceptar la introducción de datos por parte de los usuarios y guardarlos de nuevo en la base de datos.

- [Capítulo 20, “Tutorial: Creación de una página JSP con InternetBeans Express”](#)

Muestra los pasos para escribir una JSP que utiliza componentes InternetBeans con el fin de consultar una tabla de una base de datos, mostrar su contenido, aceptar la introducción de datos por parte de los usuarios y guardarlos de nuevo en la base de datos.

- [Capítulo 21, “Tutorial: Ejecución de la aplicación de ejemplo CheckBoxControl con Java Web Start”](#)

Le acompaña a través de los pasos necesarios para iniciar una aplicación de ejemplo basada en Swing con Web Start.

Este documento contiene multitud de direcciones de sedes web externas. Las direcciones y vínculos facilitados son válidos a la fecha de impresión de este manual. Las sedes web no pertenecen a Borland, por lo que no se puede garantizar su contenido ni su continuidad.

Si tiene preguntas concretas sobre el desarrollo de aplicaciones web en JBuilder, consulte el grupo de noticias Servlet-JSP borland.public.jBuilder.servlet-jsp, en <http://www.borland.com/newsgroups/>.

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 <code>Hola</code> 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> • 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 <code>java</code> , <code>bmj</code> (Make de Borland Make para Java), <code>bcj</code> (Compilador de Borland para Java) y opciones del compilador. Por ejemplo: javac , bmj , -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 <code>Esc</code> para salir de un menú”.
[]	Los corchetes, en las listas de texto o sintaxis, encierran elementos optativos. En estos casos no se deben escribir los corchetes.

Tabla 1.1 Convenciones tipográficas y de símbolos (continuación)

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

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

Tabla 1.2 Convenciones de las plataformas

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

Tabla 1.2 Convenciones de las plataformas (continuación)

Elementos	Significado
Directorio de inicio	<p>La ubicación del directorio inicial varía según la plataforma y se indica con la 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	Las imágenes o capturas de pantalla utilizan el aspecto Metal en diversas plataformas.

Asistencia y recursos para desarrolladores

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

El servicio de asistencia técnica de Borland

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

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

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

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

Recursos en línea

También puede obtener información de los siguientes recursos en línea:

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

World Wide Web

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

En particular, pueden resultar interesantes las siguientes direcciones:

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

Grupos de noticias de Borland

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

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

Usenet, grupos de noticias

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

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

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

Información sobre errores

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

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

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

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

Introducción general al proceso de desarrollo de aplicaciones web

El desarrollo web es una función de JBuilder Enterprise. El desarrollo de applets es una función de todas las ediciones de JBuilder

Este apartado presenta tecnologías para aplicaciones web y algunas de las diferencias que existen entre ellas, y analiza la forma de decidir qué tecnologías hay que utilizar. Comienza con un resumen básico de las tecnologías que se presentan en este libro:

Tabla 2.1 Tecnologías para aplicaciones web

Tecnología	Descripción
Servlets	Una aplicación Java en el servidor que puede procesar peticiones de los clientes.
Páginas JavaServer (JSP)	Una ampliación de la tecnología servlet. Las páginas JavaServer utilizan bibliotecas de etiquetas personalizadas y ofrecen un modo sencillo de desarrollar servlets. Parecen diferentes durante el desarrollo pero, cuando se ejecutan por primera vez, el servidor web las compila como servlets.
InternetBeans Express	Un conjunto de componentes y una biblioteca de etiquetas suministrados por Borland, que se utilizan para simplificar la presentación y manipulación de los datos de una base de datos. Esta tecnología se usa en conjunción con tecnología servlet o JSP y simplifica el desarrollo de servlets o JSP enlazadas a datos.
Struts	Un marco de trabajo de código abierto, suministrado por el Jakarta Project, que se utiliza para generar aplicaciones web. Struts ofrece una capa de control flexible basada en tecnologías estándar como los Servlets, JSP, JavaBeans, ResourceBundles (conjunto de recursos) y XML.

Tabla 2.1 Tecnologías para aplicaciones web (continuación)

Tecnología	Descripción
JSTL (Biblioteca de etiquetas estándar para Páginas JavaServer)	Una biblioteca de etiquetas, suministrada por Sun, que forma parte de Java Web Services Development Pack 1.0 (WSDP). Ofrece un conjunto de etiquetas que permiten a los desarrolladores llevar a cabo las tareas comunes de un modo estándar. La JSTL cuenta con cuatro áreas, cada una con su propio TLD (descriptor de bibliotecas de etiquetas) y espacio de nombre.
Cocoon	Un marco de trabajo para difusión de Java basado en servlets para XML, que se integra dentro de JBuilder. Cocoon permite la separación entre contenido, estilo y lógica, y utiliza transformación XSL para combinarlos. Cocoon también puede utilizar hojas lógicas, páginas XSP, con el fin de proporcionar contenido dinámico incrustado con lógica de programación escrita en Java. Si desea obtener más información acerca del marco de trabajo Cocoon, consulte "Presentación de XML mediante Cocoon" en la <i>Guía del desarrollador de aplicaciones XML</i> .
Applets	Un tipo especializado de aplicación Java que se puede descargar desde un navegador cliente y ejecutarse en la máquina del cliente.

El resumen ofrece alguna idea de la naturaleza de cada una de estas tecnologías, pero, ¿en qué se puede basar la elección de una de ellas? ¿Cuáles son las ventajas y desventajas de cada una de estas tecnologías? Se responderá a estas y a más preguntas en el siguiente estudio.

Servlets

Los servlets son programas Java que se integran en un servidor web para suministrar el procesamiento en el servidor de las peticiones de un navegador web cliente. Requieren un servidor web que admita tecnología JavaServer, tal como el servidor web Tomcat que se distribuye con las ediciones de JBuilder. (Tomcat también puede estar integrado con servidores Web que no admitan tecnología JavaServer, otorgándoles de esta forma dicha funcionalidad. Un ejemplo de ello es IIS.) Los servlets de Java pueden utilizarse para sustituir programas de Interfaz de enlace común (Common Gateway Interface - CGI), o en las mismas situaciones en las que previamente se habría utilizado una CGI. La utilización de servlets, en lugar de CGI, presenta algunas ventajas:

- Reducción de la sobrecarga de la memoria.
- Independencia de la plataforma.
- Independencia del protocolo.

Se utiliza un programa cliente escrito en cualquier lenguaje para enviar peticiones al servlet. El cliente puede ser tan simple como una página HTML. También se podría usar una applet para el cliente o un programa escrito en un lenguaje que no sea Java. En el servidor, el servlet procesa la petición y genera una salida dinámica que se envía de nuevo al cliente. Habitualmente, los servlets no tienen una IU (interfaz de usuario), pero opcionalmente se puede proporcionar una en el ordenador cliente.

Los servlets tienen algunas ventajas sobre las applets:

- No hay que preocuparse de en qué JDK está ejecutándose en el navegador Web cliente. El navegador Web cliente no necesita admitir Java. Todo el código escrito en Java se ejecuta completamente en el servidor. Esto da más control al administrador del servidor.
- Después de que se inicie el servlet, las peticiones de los navegadores web de los clientes se limitan a invocar el método `service()` de ejecución del servlet. Esto significa que los clientes no experimentan ningún impacto en el rendimiento mientras esperan que se cargue el servlet. Además, resulta mucho más rápido que descargar una applet.

La distribución de un servlet en el servidor Web puede resultar problemática, pero ciertamente no es imposible. JBuilder proporciona algunas herramientas que facilitan la distribución. Estas se analizan en el [Capítulo 11, “Distribución de aplicaciones web”](#).

Si desea obtener más información sobre la depuración, consulte el [Capítulo 4, “Los servlets”](#) y el [Capítulo 5, “Los servlets en JBuilder”](#).

Páginas JavaServer (JSP)

Las Páginas JavaServer (JSP) son una extensión de la tecnología servlet. Básicamente, son una forma simplificada de escribir servlets que pone más énfasis en el aspecto de la presentación de la aplicación.

La diferencia principal entre servlets y JSP es que, con los servlets, la lógica de la aplicación está en un archivo Java y está totalmente separada del HTML en la capa de presentación. Con JSP, Java y HTML se combinan en un archivo que tiene una extensión `.jsp`. Las JSP están evolucionando de tal forma que ya no contienen nada de código Java y sólo utilizan bibliotecas de etiquetas. Por ejemplo, JSTL dispone de etiquetas para condiciones y bucles que podrían sustituir al código Java.

Cuando el servidor web procesa el archivo JSP, se genera un servlet, pero el desarrollador no verá este servlet. De hecho, cuando se compila y se ejecuta JSP en el IDE de JBuilder, se pueden ver excepciones o errores que se producen en el servlet generado. Esto puede ser un poco confuso, ya que puede ser difícil definir qué parte de su JSP está provocando un problema cuando el mensaje de error se refiere a una línea de código que

en realidad es parte del código generado. (Tenga en cuenta que el depurador de JBuilder le permite visualizar el código fuente original o el código fuente generado.)

Las JSP también admiten el uso de bibliotecas de etiquetas. Una *biblioteca de etiquetas* es un conjunto de etiquetas personalizadas relacionadas. Una *etiqueta personalizada* llama a una *acción personalizada*, también conocida como un módulo JSP reutilizable. El Asistente para JSP de JBuilder admite la integración automática de las siguientes bibliotecas de etiquetas en su JSP:

- InternetBeans Express 1.1: Una biblioteca de etiquetas suministrada por Borland, utilizada para presentar y manipular fácilmente los datos de una base de datos.
- JSTL 1.0: Una biblioteca de etiquetas, suministrada por Sun, que ofrece un conjunto de etiquetas que permiten a los desarrolladores realizar tareas comunes de un modo estándar.
- Struts 1.0: Una biblioteca de etiquetas de código abierto, suministrada por el Jakarta Project, que se utiliza para generar aplicaciones web.

Nota También puede definir sus propias bibliotecas de etiquetas y utilizarlas en JBuilder. Además, terceras partes pueden suministrar bibliotecas de etiquetas más avanzadas y otros marcos de trabajo mediante OpenTools.

Las JSP tienen algunas ventajas y algunos inconvenientes en comparación con los servlets. Algunas de las ventajas son:

- Hay que escribir menos código.
- Fácil incorporación de JavaBeans existentes.
- La distribución es más sencilla. La mayoría de las emisiones de distribución se ocupan automáticamente de sí mismas, ya que las JSP se asocian con un servidor web de la misma forma que lo hacen los archivos HTML.
- Puede utilizar sólo etiquetas y, de este modo, no necesita incrustar código HTML en la JSP. Esto implicaría que el autor de la página no necesitaría tener que saber programar en Java. (Evidentemente, puede incrustar código HTML en la JSP. Con una planificación cuidadosa, el código HTML se puede separar claramente del código Java, facilitando la lectura de la JSP.)

Algunas de las desventajas de las JSP son:

- El código invisible del servlet generado puede ser confuso, como ya se indicó previamente.
- Dado que los códigos HTML y Java no están en archivos separados, un desarrollador de Java y un diseñador de páginas web que trabajen juntos en una aplicación deben ser cuidadosos para no sobreescribir los cambios entre sí.

- HTML y Java combinados en un solo archivo pueden hacer que éste sea difícil de leer y este problema se incrementa si no se practica una programación cuidadosa y limpia.
- El diseñador de JBuilder no admite el diseño de archivos .jsp.

El sistema JSP se parece mucho a las ASP (páginas del servidor Active) de Microsoft. La principal diferencia entre las JSP y las ASP es que los objetos manejados en el primero de los sistemas son JavaBeans, que son independientes de plataforma. ASP maneja objetos COM, por lo que depende completamente de las plataformas Microsoft.

Para obtener más información sobre las JSP, consulte el [Capítulo 6, “Desarrollo de Páginas JavaServer”](#).

InternetBeans Express

La tecnología InternetBeans Express de JBuilder se integra en la tecnología servlet y JSP con el fin de ofrecer un valor añadido a sus aplicaciones y simplificar las tareas de desarrollo de servlet y JSP. InternetBeans Express es un conjunto de componentes y extensiones de etiquetas de páginas JavaServer (JSP) utilizado para la generación de presentaciones y la interacción con aplicaciones web. Toma páginas estáticas de plantilla, inserta contenido dinámico desde un modelo dinámico de datos y lo presenta al cliente; a continuación, escribe en el modelo los datos que envía el cliente. Esto facilita la creación de servlets y JSP enlazados a datos.

InternetBeans Express contiene soporte incorporado para DataExpress DataSets y DataModules. También puede utilizarse con modelos genéricos de datos y EJB.

Si desea obtener más información sobre la depuración, consulte el [Capítulo 7, “InternetBeans Express”](#).

Struts

El marco de trabajo de código abierto Struts se basa en el concepto de diseño de software Modelo 2 o Controlador Modelo-Vista. En este marco de trabajo, el modelo contiene los datos, la vista es la presentación de los datos y el controlador supervisa la interacción entre el modelo y la vista.

- La vista es normalmente una página JSP.
- El modelo puede ser cualquier tecnología de acceso a datos, desde JDBC a un EJB.
- El controlador es un servlet Struts de nombre ActionServlet.

Este marco de trabajo, que es una colección de clases, servlets, etiquetas JSP, una biblioteca de etiquetas y clases de utilidades, separa limpiamente el HTML del código Java y la presentación visual de la lógica empresarial.

La ventaja más importante de utilizar Struts es la división entre el código Java y las etiquetas HTML. Gracias a los Struts, la aplicación web es más fácil de comprender. Un diseñador web ya no tiene que buscar por el código Java para realizar cambios en la presentación, y un desarrollador ya no tiene que volver a compilar el código al diseñar de nuevo el flujo de la aplicación.

A parte de su complejidad, el marco de trabajo Struts presenta pocos inconvenientes al desarrollador en Java. JBuilder ofrece un robusto conjunto de herramientas que simplifican la complejidad y que mantienen sincronizados las clases y los archivos `xml`.

Para obtener más información sobre los Struts en JBuilder, consulte el [Capítulo 8, “El marco de trabajo Struts en JBuilder”](#). Para obtener más información sobre los Struts, consulte “The Jakarta Project: Struts” en <http://jakarta.apache.org/struts/index.html>

JSTL (Biblioteca de etiquetas estándar para Páginas JavaServer)

La JSTL es una biblioteca de etiquetas de Sun que forma parte de Java Web Services Development Pack 1.0 (WSDP). Ofrece un conjunto de etiquetas que permiten a los desarrolladores llevar a cabo las tareas comunes de un modo estándar. Por ejemplo, en lugar de utilizar distintas etiquetas de iteración de varios fabricantes, JSTL define una etiqueta estándar que funciona del mismo modo en todas partes. Esta estandarización permite conocer una etiqueta y utilizarla en varios contenedores JSP.

La biblioteca JSTL está dividida en cuatro áreas, cada una con su propio TLD (descriptor de bibliotecas de etiquetas) y espacio de nombre. Estas cuatro áreas son Núcleo, procesamiento de XML, internacionalización y acceso a bases de datos.

Si desea obtener más información acerca de JSTL, consulte sobre la biblioteca JSTL en <http://java.sun.com/products/jsp/jstl/>.

Applets

Cuando apareció el lenguaje Java, había mucha confusión acerca de las applets. En aquellos tiempos, la tecnología Web estaba menos desarrollada y las applets prometían soluciones a algunos de los problemas a los que se enfrentaban los desarrolladores en ese momento. De hecho, las applets se hicieron tan populares que, hasta el día de hoy, el desarrollo de una applet es a menudo una de las primeras tareas previstas en el comienzo de los cursos de Java. Como resultado, un error común entre los desarrolladores de Java es confiar demasiado en las applets. Las applets evidentemente tienen su utilidad, pero no son una solución mágica a todos los problemas de desarrollo web.

Algunos de los inconvenientes de las applets son:

- Su distribución y comprobación puede ser difícil.
- Se confía que en que la máquina del cliente tenga activado Java en su navegador web.
- Los diferentes navegadores web soportan diferentes versiones del JDK, y habitualmente no la última versión.
- El applet puede tardar en iniciarse la primera vez, ya que necesita ser descargada por el cliente.

Algunos de estos problemas tienen soluciones que se tratan con más detalle en el [Capítulo 14, “Las applets”](#). Cuando se considere la utilización de una applet, hay que pensar si alguna otra tecnología de Java puede servir mejor a nuestros propósitos.

Existen algunas ventajas en la utilización de applets. Entre otras, destacan las siguientes:

- Las applets pueden proporcionar interfaces de usuario (IU) más complejas que los servlets o las JSP.
- Desde el momento en que las applets se descargan y se ejecutan en el ordenador del cliente, el servidor web no necesita ser compatible con Java. Esto puede ser importante si se está escribiendo una aplicación Web para una Web en la que no se tenga control sobre el servidor Web (tal como una Web alojada por un ISP externo).
- Las applets pueden validar localmente los datos introducidos por el usuario, en lugar de validarlos en el servidor. Esto también se puede llevar a cabo con JavaScript en conjunción con un servlet o una JSP.
- Después de la descarga inicial del applet, puede reducirse el número de peticiones del navegador web al servidor, dado que una gran cantidad de procesamiento puede llevarse a cabo en la máquina del cliente.

Para obtener más información acerca de las applets y de cómo solucionar los problemas de las applets, consulte el [Capítulo 14, “Las applets”](#).

Qué tecnologías utilizar en su aplicación Web

Ahora que ya se ha hecho una introducción general de las diferentes tecnologías Web, ¿en qué debemos basarnos para decidir qué tecnología vamos a usar en una aplicación Web? Las sugerencias siguientes pueden ayudar a tomar esa decisión:

- ¿Se necesita una IU (interfaz de usuario) muy compleja? Si la IU requiere elementos más complejos que los componentes habituales para la introducción de datos (como campos de texto, botones de radio, cuadros combinados o cuadros de lista, botones de envío, etc.) e imágenes, se debe utilizar un applet.
- Si se pretende otorgar mucha carga de trabajo al servidor, se debe utilizar un servlet o una JSP.
- Si se quiere evitar que los usuarios tengan que descargar un montón de código y acelerar el inicio de la aplicación, se debe utilizar un servlet o una JSP.
- Si se quiere controlar la versión del JDK para la aplicación (sin necesidad de usar descargas) o si se está preocupado por los usuarios que no tienen Java activado en sus navegadores web, se debe utilizar un servlet o una JSP.
- Si se busca un substituto para CGI, con menos sobrecarga de memoria, se debe utilizar un servlet o una JSP.
- Si se quiere algo similar a un ASP, pero se prefiere que sea independiente de la plataforma, se debe utilizar una JSP.
- Si se necesita una IU compleja, pero también se quieren algunas de las características de los servlets o de JSP, se debe considerar combinar un applet y un servlet. Se puede tener una applet en el navegador web cliente que se comunique con un servlet en el servidor.
- Si se quiere utilizar un servlet o una JSP y se quiere hacer enlace de datos, se debe utilizar InternetBeans Express.
- Si está pensando en desarrollar una biblioteca de etiquetas de rutinas estándar, tales como control de estructuras o datos y formato de números, verifique primero si JSTL dispone de las rutinas que necesita.
- Si desea separar el código HTML del código Java, utilice una aplicación web Struts.

Tenga en cuenta que los servlets y las JSP son lo suficientemente parecidos como para que la decisión entre uno y otra dependa, ante todo, de las preferencias personales. Además, recuerde que muchas aplicaciones web utilizarán una combinación de dos o más de estas tecnologías.

El proceso básico de desarrollo de aplicaciones web

Independientemente de la tecnología web que se escoja, todavía se tendrán que seguir los mismos pasos básicos para desarrollar una aplicación web y para mantenerla en funcionamiento en el servidor web. Los pasos son:

Paso	Descripción
Diseñar la aplicación	Decida cómo va a estructurar su aplicación y qué tecnologías utilizará. Defina qué es lo que tiene que hacer la aplicación y el aspecto que tendrá.
Configurar el servidor web en el IDE de JBuilder	Opcionalmente se puede configurar el servidor web para que trabaje en el IDE de JBuilder, de forma que se puedan compilar, ejecutar y depurar las aplicaciones con el mismo servidor web que se piensa utilizar para su distribución. También puede utilizar Tomcat, el servidor web que se suministra con JBuilder, para compilar, ejecutar y depurar.
Desarrollar la aplicación	Cree una WebApp y, a continuación, escriba el código de la aplicación. Siempre que su aplicación esté compuesta de applets, servlets, Páginas JavaServer o clases Struts, la utilización de muchas de las herramientas de JBuilder simplifica las tareas de desarrollo.
Compilar la aplicación	Compile la aplicación en el IDE de JBuilder.
Ejecutar la aplicación	Ejecute la aplicación en el IDE de JBuilder. Esto le da una vista previa de la aplicación, sin necesidad de distribuirla primero. En esta etapa, se deben hacer algunas comprobaciones locales de la aplicación.
Distribuir la aplicación	Modifique los archivos web.xml y struts-config.xml y distribuya la aplicación en el servidor web. (Si se utilizan las herramientas de JBuilder, quizás no sea necesario modificar estos archivos.) Las particularidades de su aproximación a este paso dependerán en gran medida del servidor que se esté utilizando.
Comprobar la aplicación	Compruebe su aplicación ejecutándola en el servidor web. Esto ayudará a identificar cualquier problema que se dé con la distribución o con la aplicación misma. Se deben efectuar comprobaciones desde el navegador web cliente en una máquina diferente del servidor web. También se pueden probar diferentes navegadores web, ya que la aplicación puede aparecer de forma ligeramente diferente en cada uno de ellos.

Aplicaciones web y aplicaciones distribuidas

Se podría considerar el uso de una aplicación distribuida para el programa en vez de una aplicación web. Ambas pueden gestionar la programación cliente/servidor. Existen, sin embargo, ciertas diferencias entre las dos tecnologías.

Las aplicaciones web requieren un navegador web en el ordenador cliente y un servidor web en el servidor. Por ejemplo, las applets se descargan a múltiples plataformas cliente, donde se ejecutan en una máquina virtual (MV) de Java suministrada por el navegador instalado en el ordenador cliente. Los servlets y las JSP se ejecutan dentro de un servidor web con Java que admite las especificaciones de JSP/Servlet.

Las aplicaciones web pueden estar disponibles para cualquiera que tenga acceso a Internet, o se pueden poner detrás de un cortafuegos y utilizarlas solamente dentro de la intranet de la empresa.

Una aplicación web puede ser parte de una aplicación distribuida mayor que, a su vez, puede ser parte de una aplicación enterprise o J2EE. Si desea ver un ejemplo de una aplicación J2EE y la documentación de apoyo, consulte *Java 2 Platform, Enterprise Edition Blueprints* en <http://java.sun.com/j2ee/blueprints/>.

En general, las aplicaciones distribuidas administran y recuperan datos de sistemas previamente instalados. El sistema de herencia puede existir en numerosos ordenadores que ejecuten diferentes sistemas operativos. Una aplicación distribuida utiliza un servidor de aplicaciones, como el Borland Enterprise Server, para la gestión de aplicaciones. Las aplicaciones distribuidas no tienen que estar basadas en Java; de hecho, una aplicación distribuida puede contener muchos programas diferentes, sin tener en cuenta en qué lenguaje están escritos o dónde residen.

Las aplicaciones distribuidas están habitualmente confinadas a una red dentro de una empresa. Ciertas partes de la aplicación distribuida pueden estar a disposición de los usuarios a través de Internet, pero entonces se estará combinando una aplicación distribuida con una aplicación web.

Las tecnologías que se usan en una aplicación distribuida incluyen CORBA (Common Object Request Broker Architecture - La Arquitectura de Intermediador de Objetos Remotos) y RMI (Remote Method Invocation - Llamada a métodos remotos):

- La ventaja primaria de CORBA es que clientes y servidores pueden escribirse en cualquier lenguaje de programación. Esto es posible porque los objetos se definen utilizando el lenguaje de definición de interfaces (IDL: Interface Definition Language), y la comunicación entre los objetos, los clientes y los servidores se realiza por medio de ORB (Object Request Brokers – Intermediador de objetos). Si desea más información sobre CORBA, consulte “Aplicaciones distribuidas basadas en CORBA” en la *Guía del desarrollador de Enterprise JavaBeans* .
- La llamada a métodos remotos permite crear aplicaciones distribuidas “Java a Java” las que se puede llamar a los métodos de los objetos remotos de Java desde otras máquinas virtuales de Java, incluso en distintos anfitriones. Si desea obtener más información acerca de RMI, consulte el ejemplo de RMI en la carpeta <jbuilder>/samples.

Las WebApps y los archivos WAR

El desarrollo web es una función de JBuilder Enterprise

JBuilder posee algunas características importantes para gestionar la estructura de las aplicaciones web. Existen dos conceptos principales que es necesario comprender con el fin de hacer un uso efectivo de estas características: Las WebApp y los archivos recopilatorios web (WAR).

La WebApp

Una WebApp describe la estructura de una aplicación web. Básicamente, es un árbol de directorios que contiene el contenido de la web utilizada en su aplicación. Se asocia a un `ServletContext`. Un archivo descriptor de distribución llamado `Web.xml` está asociado siempre con cada WebApp. Este descriptor de distribución contiene la información que se necesita suministrar al servidor web cuando se distribuye una aplicación.

Es necesaria la utilización de una WebApp si usted tiene servlets o JSP en su proyecto. Aunque probablemente no se quiera usar una WebApp si la aplicación web contiene una sola applet, debe utilizar una en una aplicación web que contuviera una applet y un servlet o una JSP. De esta forma, se puede almacenar la WebApp completa en un único archivo WAR. Se podrían tener varias WebApp en una sola web. JBuilder admite este concepto, permitiendo tener múltiples WebApp en un solo proyecto.

Es importante meditar sobre la estructura de las aplicaciones web durante la etapa de planificación. ¿Cuantas WebApps tendrá? ¿Como se llamarán? ¿Estas WebApps se almacenarán en archivos WAR? ¿Las WebApps necesitan admitir algún marco de trabajo? Proyectando la estructura desde el principio, la distribución terminará por ser más fácil. JBuilder admite el concepto de una WebApp por defecto, alojada en el directorio <directoriodelproyecto>/defaultroot. Si usted no especifica ninguna WebApp para sus aplicaciones web, éstas entrarán en la WebApp por defecto.

Para obtener más información acerca de cómo trabaja JBuilder con las WebApps, consulte “[Creación de una WebApp con el Asistente para aplicaciones web](#)” en la página 3-3 y “[La WebApp y sus propiedades](#)” en la página 3-6.

Archivos recopilatorios Web (WAR)

Un archivo WAR es un archivo recopilatorio de una aplicación web. Es similar a un archivo JAR. Al almacenar su aplicación completa y los recursos que ésta necesita en un archivo WAR, se hace más fácil la distribución. Copie solamente el archivo WAR en su servidor web, en vez de tener que asegurarse de que multitud de pequeños archivos se copien en el sitio adecuado. JBuilder puede generar automáticamente un archivo WAR cuando se genera el proyecto, cuando se genera la WebApp o cuando se generan ambos. También puede decidir no crear nunca un archivo WAR, lo que puede ser de gran utilidad durante las etapas iniciales de desarrollo, antes de que esté preparado para comprobar la distribución.

Para obtener más información acerca de cómo trabaja JBuilder con archivos WAR, consulte “[El archivo WAR](#)” en la página 3-15.

Herramientas para trabajar con WebApps y archivos WAR

He aquí una lista de herramientas que proporciona JBuilder para trabajar con WebApps y archivos WAR:

Tabla 3.1 Herramientas de JBuilder para WebApp y archivos WAR

Herramienta	Descripción
Asistente para aplicaciones web	Un sencillo asistente para crear una WebApp. Permite especificar el nombre de la WebApp, el directorio raíz de los documentos de la aplicación web, el momento de generación de un archivo WAR, los marcos de trabajo JSP/Servlet que debe admitir la WebApp y la URI de inicio por defecto.
Nodo WebApp	Es un nodo en el panel del proyecto de JBuilder IDE que representa la WebApp. Este nodo tiene un cuadro de diálogo de propiedades para configurar la WebApp. Dentro del nodo WebApp existen otros nodos para los descriptores de distribución, el directorio raíz y un archivo WAR opcional.
Nodo de archivo WAR	Es un nodo en el panel del proyecto de JBuilder IDE que representa el archivo WAR. Tiene un cuadro de diálogo de propiedades y un visualizador que permite observar su contenido.
Editor DD para WebApp	Es una interfaz y un editor para el archivo descriptor de distribución <code>web.xml</code> que se necesita para cada WebApp. En JBuilder, también se pueden editar descriptores de distribución específicos de servidor, tales como <code>Weblogic.xml</code> de WebLogic. Los descriptores de distribución se tratan con más detalle en " Modificación del descriptor de distribución " en la página 11-4.
Editor de configuración de Struts	Una interfaz de usuario y un editor para el archivo del descriptor de distribución <code>struts-config.xml</code> necesario para admitir el marco de trabajo Struts.

Creación de una WebApp con el Asistente para aplicaciones web

El Asistente para aplicaciones web crea una nueva WebApp. Antes de que pueda utilizar el Asistente para aplicaciones web, deberá seleccionar un servidor para el proyecto. Para crear una WebApp:

- 1 Seleccione Archivo | Nuevo. Haga clic en la pestaña Web de la galería de objetos.
- 2 Seleccione Aplicación web. Pulse Aceptar.
- 3 Escriba un nombre para la WebApp. El campo Directorio se rellena automáticamente mientras escribe.

- 4 Introduzca la ubicación del Directorio que va a ser la raíz de los documentos de la WebApp. Este campo se rellena automáticamente a medida que se va escribiendo. El nombre de directorio crea un subdirectorio del directorio de proyectos. También se puede pulsar el botón de puntos suspensivos con el fin de examinar y crear un directorio, o para elegir un directorio existente. No se recomienda la elección del directorio raíz del proyecto ni el directorio `src`.
- 5 Especifique el momento de generar un archivo WAR seleccionando una de las siguientes configuraciones de la lista desplegable Generar WAR.
 - Al generar el proyecto o la Webapp.
 - Sólo al generar la Webapp.
 - Sólo al generar el proyecto.
 - Nunca.

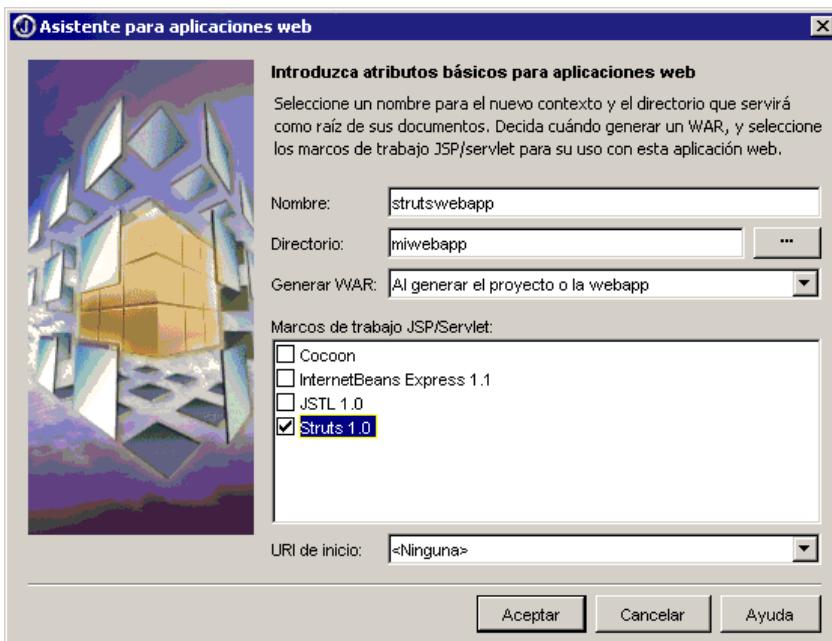
Siempre que se crea un archivo WAR, tiene el mismo nombre que la WebApp y se coloca en el directorio que contiene el directorio raíz de los documentos. Si selecciona Nunca, no se preocupe. Siempre se podrá cambiar de opinión posteriormente. Esta lista desplegable se corresponde con una configuración en el cuadro de diálogo Propiedades de la WebApp.

- 6 Seleccione los marcos de trabajo JSP/Servlet que desea que admita la WebApp. Los siguientes marcos se incluyen con JBuilder:
 - Cocoon – un marco de trabajo basado en servlets de difusión de Java para XML. Consulte “Presentación de documentos XML” en la *Guía del desarrollador de aplicaciones XML*.
 - InternetBeans Express – una biblioteca de componentes que facilita la creación de servlets y JSP enlazados a datos. Consulte el [Capítulo 7, “InternetBeans Express”](#).
 - JSTL (Biblioteca de etiquetas estándar para páginas JavaServer) — JSTL ofrece un modo estándar de llevar a cabo las tareas más comunes de codificación mediante sencillas etiquetas. Tiene a su disposición más información en <http://java.sun.com>.
 - Struts – un marco de trabajo conocido como el concepto de diseño de software Modelo 2 o Controlador Modelo-Vista. Consulte el [Capítulo 8, “El marco de trabajo Struts en JBuilder”](#).

Puede hacer que las bibliotecas de etiquetas JSP definidas por el usuario estén disponibles en este asistente si las añade en el cuadro de diálogo Configurar bibliotecas. Para obtener más información, consulte [“Utilización del cuadro de diálogo Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario”](#) en la página 6-6.

- 7 Defina una URI de inicio por defecto, si lo desea. Si el marco de trabajo seleccionado admite una URI de inicio, se debe llenar este campo. Por ejemplo, si selecciona el marco de trabajo Cocoon, el valor de URI de inicio es `index.xml`. Si se especifica una URI de inicio, los comandos Ejecutar web, Depurar web y Optimizar web aparecen como disponibles en el menú contextual de nodo WebApp del panel del proyecto.
- 8 Pulse Aceptar para cerrar el asistente y crear la WebApp.

Figura 3.1 Asistente para aplicaciones web



Puede utilizar también el Asistente para aplicaciones web con el fin de importar una aplicación web ya creada. Rellene el campo Nombre y utilice el campo Directorio para apuntar a la ubicación del directorio que contiene la aplicación web. Si la aplicación web es válida y los descriptores de distribución son correctos para el servidor web configurado, se puede ejecutar desde el IDE de JBuilder inmediatamente.

La WebApp y sus propiedades

Un servidor web con Java ubica una aplicación web mediante su `ServletContext`, que apunta hacia la WebApp. JBuilder IDE representa una WebApp mediante un nodo WebApp. Este es un nodo en el árbol del panel del proyecto que tiene el nombre de la WebApp.

Figura 3.2 panel del proyecto que muestra un nodo WebApp



El nodo WebApp tiene dos o tres nodos dependientes; un Directorio raíz para la aplicación, un nodo de Descriptores de distribución que representa el directorio `WEB-INF` para la WebApp y un nodo opcional de archivo WAR.

Los archivos de contenido web (tales como los archivos HTML, SHTML y JSP) se deben colocar en el directorio raíz de la WebApp o en uno de sus subdirectorios. Los archivos de contenido web son archivos a los que se puede acceder directamente mediante el navegador web del cliente. Los recursos de Java utilizados por la WebApp (tales como servlets o JavaBeans) deben tener sus archivos fuente en el directorio fuente del proyecto. El navegador web del cliente no puede acceder directamente a estos archivos, pero pueden ser llamados por otros componentes tales como un archivo JSP. El Asistente para servlets de JBuilder, el Asistente para JSP y el Asistente para el inicio de Web Start facilitan la creación de aplicaciones web que sigan estas reglas. Asegúrese de especificar una WebApp existente cuando utilice estos asistentes.

Directorio raíz

El directorio raíz define la ubicación base para la aplicación web. Cada parte de la aplicación web se situará en relación al directorio raíz. Los archivos de contenido web, tales como los archivos HTML, SHTML, JSP o de imagen, deben situarse en este directorio. Los archivos de contenido web son archivos a los que se puede acceder directamente mediante el navegador web del cliente.

Los archivos en el directorio raíz de la WebApp (y cualquier subdirectorio del directorio raíz) se muestran automáticamente en el nodo del Directorio raíz del panel del proyecto. Solamente se muestran los archivos del tipo que se especifique en la ficha WebApp del cuadro de diálogo Propiedades

de la WebApp. Los tipos de archivo por defecto son los únicos con los que se trabaja habitualmente en una aplicación web. Esto le permite trabajar con archivos HTLM o con archivos de imagen utilizando sus herramientas favoritas para trabajar con esos tipos de archivo. Guarde estos archivos en el directorio raíz de la WebApp o en uno de sus subdirectorios. Sólo le resta pulsar el botón Actualizar del panel del proyecto para ver los archivos actuales definidos en JBuilder.



Descriptores de distribución

Cada WebApp debe tener un directorio `WEB-INF`. Este directorio contiene la información que necesita el servidor web cuando se distribuye la aplicación. Esta información tiene la forma de archivos descriptores de distribución. Dichos archivos tienen extensiones `.xml`. Se muestran en el nodo Descriptores de distribución en el panel del proyecto. El directorio `WEB-INF` es, de hecho, un subdirectorio del directorio raíz de la WebApp. No se presenta bajo el nodo del Directorio raíz del panel del proyecto porque es un directorio protegido que no puede ser atendido por el servidor web. Puede cambiar esto utilizando la ficha Directorios del cuadro de diálogo Propiedades de WebApp.

El nodo de descriptores de distribución WebApp siempre contiene un archivo descriptor de distribución llamado `web.xml`. Este archivo es requerido por todos los servidores web habilitados para Java y lo crea JBuilder cuando crea la WebApp. Si está utilizando el marco de trabajo Struts en la WebApp, el archivo `struts-config.xml` también se encuentra en el nodo Descriptores de distribución.

Su servidor web también puede necesitar descriptores de distribución adicionales que sean únicos para ese servidor web en particular. Éstos pueden modificarse en JBuilder y se crean si se especifica su necesidad por el plug-in del servidor web actualmente configurado. Compruebe la documentación de su servidor web para averiguar qué descriptores de distribución requiere.

JBuilder proporciona un editor de descriptores de distribución para modificar el archivo `web.xml`. También puede modificar descriptores de distribución específicos del servidor en JBuilder. Se necesitan algunas correspondencias en el archivo `Web.xml` para que la WebApp trabaje como se desea en el IDE de JBuilder. Estas asignaciones se crearán automáticamente cuando utilice asistentes de JBuilder para crear las partes de su aplicación web. Pueden necesitarse otras asignaciones cuando se distribuya la aplicación. Para obtener más información acerca de los descriptores de distribución y del Editor DD para WebApp, consulte el apartado “[Modificación del descriptor de distribución](#)” en la página 11-4.

Propiedades de WebApp

El nodo de WebApp en el panel del proyecto tiene diferentes propiedades que se pueden editar.

Si desea editar las propiedades del nodo WebApp, haga clic con el botón derecho en el nodo y seleccione Propiedades en el menú. El cuadro de diálogo Propiedades de la WebApp tiene cinco fichas:

- WebApp
- Directorios
- Clases
- Dependencias
- Descriptor

La ficha WebApp

La ficha WebApp del cuadro de diálogo Propiedades de la WebApp indica el nombre de la WebApp, la ubicación del directorio de la WebApp y la ubicación del directorio del archivo WAR. Existe una lista desplegable que indica el momento en que se genera o actualiza el archivo WAR, y una casilla que indica si se comprime o no el archivo. La configuración para crear el archivo corresponde a la lista desplegable “Generar WAR”, situada en el Asistente para aplicaciones web. Debe desactivar la generación de WAR durante el desarrollo, y activarla solamente antes de generar el proyecto por última vez antes de su distribución.

La parte inferior de la ficha WebApp se divide en dos fichas con pestañas: Marcos de trabajo JSP/Servlet y Tipos de archivo incluidos.

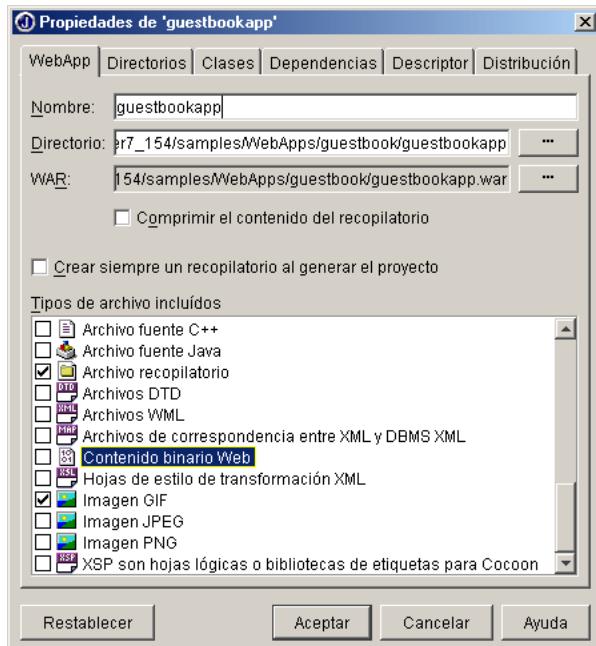
La ficha Marcos de trabajo JSP/Servlet permite seleccionar marcos de trabajo JSP/Servlet con el fin de utilizarlos en la WebApp. La ficha cuenta automáticamente con cuatro marcos de trabajo que se suministran con JBuilder: Cocoon, InternetBeans Express, JSTL (Biblioteca de etiquetas estándar para páginas JavaServer) y Struts. Si utiliza el cuadro de diálogo Configurar bibliotecas para añadir marcos de trabajo definidos por el usuario, estos marcos de trabajo están disponibles en la lista.

La ficha Tipos de archivo incluidos contiene también una lista de tipos que se incluyen tanto para su visualización como para la generación del archivo WAR. Sólo se incluirán los tipos de archivo que estén seleccionados, dependiendo de sus extensiones. Las extensiones de archivo asociadas con cada tipo de archivo pueden verse o cambiarse en el cuadro de diálogo de Opciones del IDE, disponible en el menú de herramientas.

La lista desplegable URI de inicio se utiliza para especificar una URI que se inicie al ejecutar la WebApp. Si una webapp dispone de una URI de

inicio específica, se puede Ejecutar/Depurar/Optimizar web directamente desde el menú contextual del nodo WebApp del panel del proyecto. Algunos marcos de trabajo, como Cocoon, disponen de una URI de inicio predilecta, que se establece automáticamente cuando se selecciona el marco de trabajo. Esta URI de inicio predilecta se puede redefinir. Si la URI que desea no aparece en la lista desplegable, búsquela pulsando el botón de puntos suspensivos.

Figura 3.3 Ficha WebApp del cuadro de diálogo Propiedades de WebApp



Consulte

- “Creación de una WebApp con el Asistente para aplicaciones web” en la página 3-3
- “Presentación de documentos XML” en la *Guía del desarrollador de aplicaciones XML*
- [Capítulo 7, “InternetBeans Express”](#)
- [Capítulo 8, “El marco de trabajo Struts en JBuilder”](#)
- “Utilización del cuadro de diálogo Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario” en la página 6-6

La ficha Directorios

La ficha Directorios del cuadro de diálogo Propiedades de WebApp le permite especificar subdirectorios personalizados y archivos concretos bajo el directorio WEB-INF con el fin de incluirlos en su proyecto y en sus recopilatorios. También le permite excluir directorios que nunca se deberían haber incluido, como el directorio CVS que aparece en todos los directorios que se encuentran bajo el control de CVS. La ficha Directorios proporciona las siguientes posibilidades:

- La capacidad de especificar una lista de directorios a excluir. Estas exclusiones se aplican después de las inclusiones para WEB-INF.
- La opción de tratar WEB-INF como cualquier otro directorio, de forma que aparece bajo la carpeta Directorio raíz en el panel del proyecto y muestra los mismos tipos de archivo que otros directorios en la WebApp.
- La opción de que directorios específicos bajo WEB-INF sean incluidos en sus proyectos y recopilatorios.

La razón de que los subdirectorios de WEB-INF deban ser añadidos explícitamente es que a menudo contienen subdirectorios que nunca deben incluirse en un archivo WAR. Observe que los directorios WEB-INF/classes y WEB-INF/lib son especiales y contienen archivos generados por JBuilder y aunque nunca se muestran en el panel del proyecto, son siempre incluidos en los recopilatorios WAR. No debe añadir archivos a WEB-INF/classes y WEB-INF/lib.

Si se trata a WEB-INF como un directorio normal, los descriptores de distribución aparecerán tanto en la carpeta Descriptores de distribución como bajo el WEB-INF de la carpeta Directorio raíz del panel del proyecto. Al hacer clic sobre un descriptor de distribución mostrado en cualquier lugar, se abrirá el Editor de Descriptores de distribución debido a que los nodos duplicados del panel del proyecto representan al mismo archivo.

Ambas listas en la ficha Directorios emparejan nombres de directorio mediante las mismas reglas:

- Las vías de acceso pueden contener uno o más elementos, separados por barras.
- Cada elemento de una vía de acceso se considera por separado.
- Los caracteres * y ? juegan su típico papel de comodines.
- Cuando se comparan las vías de acceso en disco, se consideran relativas al directorio raíz de la WebApp, o a su directorio WEB-INF.
- Las vías que no comienzan con una barra, se comprueban al revés, desde el final. El caso más común es un único elemento de vía sin barra delante, lo que significa cualquier vía que termina con dicho elemento.
- Las vías que comienzan con una barra se comprueban hacia adelante.

Figura 3.4 Ficha Directorios del cuadro de diálogo Propiedades de WebApp

La ficha Clases

La ficha Clases tiene opciones que controlan qué clases y recursos de Java se copian en el subdirectorio de `WEB-INF/classes` que está relacionado con el directorio raíz del proyecto en el disco (utilizado durante el desarrollo) y con el subdirectorio `WEB-INF/classes` del archivo WAR generado.

En la ficha Clases puede elegir qué parte del proyecto se incluye en el archivo WAR. También puede escoger archivos o clases adicionales.

Para indicar qué partes del proyecto desea incluir:

- 1** Seleccione una de estas opciones para las clases. Si elige Sólo las especificadas o Especificadas y dependientes, debe añadir las clases con el botón Añadir clases.
 - Sólo las especificadas.
 - Las especificadas y sus dependientes.
 - Todos.
- 2** Seleccione una de estas opciones para los recursos. Si selecciona Sólo los especificados, debe añadir los recursos mediante el botón Añadir archivos.

- Sólo los especificados
- Todos

Por ejemplo, si desea incluir todas las clases y recursos del proyecto en el recopilatorio, debe seleccionar Todos, en Clases y Recursos. Si selecciona Todos en Recursos, añade también todos los recursos de clases de la vía de acceso de origen del proyecto. Estos recursos de clases son distintos de los archivos de contenido web. Para ser considerado un recurso de clase, un archivo debe estar en la vía de acceso de origen del proyecto y como extensión debe asignársele Copiar en la pestaña Recurso de la ficha Generar de Propiedades de proyecto.

Los archivos de contenido web no deberían estar en la vía de acceso de origen del proyecto. El contenido web debería colocarse en el directorio raíz de la WebApp o en uno de sus subdirectorios. Se copia en la ubicación correspondiente en el archivo WAR cuando éste es generado. Los tipos de archivo considerados contenido web se seleccionan en la ficha WebApp del cuadro de diálogo Propiedades de WebApp.

Advertencia

Si se selecciona Todos, tanto para Clases como para Recursos, cada archivo de clase en la vía de salida se incluye en el archivo WAR. Esto significa que se incluirán archivos de clase y recursos que no son necesarios. Sea consciente de que generar un WAR con esta configuración podría tardar algún tiempo y dar un resultado de tamaño excesivo.

Si no desea incluir dependencias en el recopilatorio y sólo los recursos especificados, seleccione Sólo las especificadas, tanto en Clases como en Recursos y, a continuación, añada las clases y recursos que deseé mediante los botones Añadir clases y Añadir recursos. Para añadir las clases y los archivos, las clases deben encontrarse en la vía de salida del proyecto, y los archivos en la vía de acceso a archivos fuente.

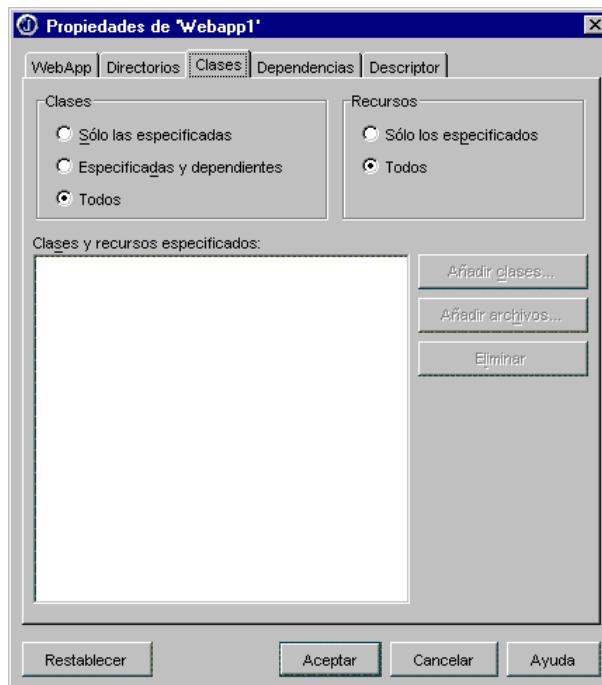
- 3** Seleccione el botón Añadir clases si en la categoría Clases seleccionó Sólo las especificadas o Especificadas y dependientes. A continuación, seleccione las clases que desea añadir a su recopilatorio. Las clases deben encontrarse en la vía de salida del proyecto. Si seleccionó Especificadas y dependientes, se incluyen en el recopilatorio las dependencias de clases adicionales.

Recuerde que las clases se copian en WEB-INF/classes o en sus subdirectorios. Por lo tanto, las clases que se seleccionen deben ser clases a las que pueda accederse desde el servidor, no clases a las que tenga que prestar servicio el servidor web. Por ejemplo, pueden seleccionarse las clases servlet, pero no las clases applet.

- 4** Seleccione el botón Añadir archivos si ha elegido Sólo las especificadas en la categoría Recursos. A continuación, seleccione los archivos que desea añadir a su recopilatorio. Los archivos deben estar en la vía de acceso a fuentes del proyecto. Se copiarán a su ubicación correspondiente en WEB-INF/classes. Por ejemplo, /myproject/src/com/whatever/whatever.properties se copia a /WEB-INF/classes/com/whatever/whatever.properties.

Nota El cuadro de diálogo Añadir archivos no permite examinar el interior de los archivos recopilatorios. Si un archivo o paquete que se necesita, se encuentra en un archivo recopilatorio, se debe extraer a la carpeta fuente y después incorporarlo pulsando el botón Añadir archivos.

Figura 3.5 Ficha Clases del cuadro de diálogo Propiedades de WebApp



La ficha Dependencias

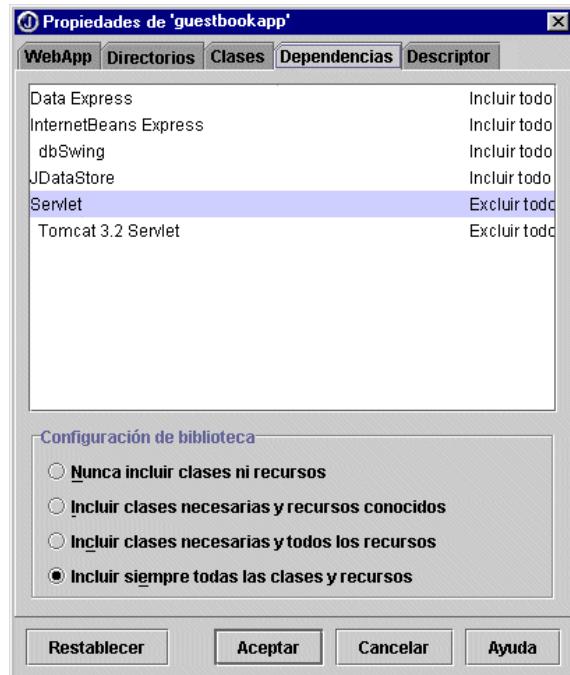
La ficha Dependencias permite especificar qué hacer con las dependencias de las bibliotecas para su WebApp. Generalmente, deberá incluir las bibliotecas necesarias. A diferencia de los archivos normales, los archivos JAR de bibliotecas se almacenan tal cual en el directorio WEB-INF/lib (cuando la Configuración de biblioteca en esta pestaña es Incluir siempre todas las clases y recursos - la configuración recomendada).

Las bibliotecas se configuran por defecto como Incluir todas, excepto la biblioteca Servlet, que no debería ser incluida en un archivo WAR (la proporciona el servidor web). Debería excluir también toda biblioteca

suministrada por el servidor y las que no sean utilizadas en su WebApp. Al asegurarse de que estas bibliotecas innecesarias son excluidas reducirá el tamaño de su archivo WAR y hará más rápida la generación de su WebApp.

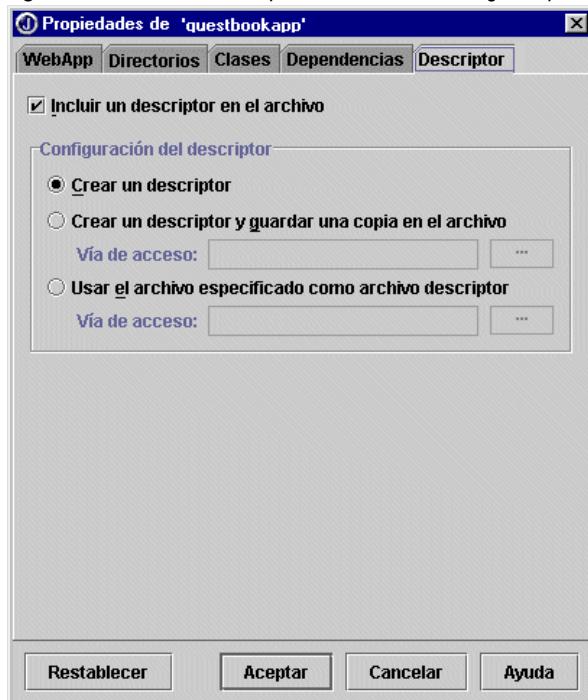
La ficha Dependencias del cuadro de diálogo Propiedades de WebApp tiene el mismo aspecto que la ficha Dependencias de cualquier otro tipo de archivo. Consulte el tema de Ayuda en línea “El creador de recopilatorios, Determinar cómo tratar las dependencias de las bibliotecas” para obtener más información.

Figura 3.6 Ficha Dependencias del cuadro de diálogo Propiedades de WebApp



La ficha Descriptor

La ficha Descriptor del cuadro de diálogo Propiedades de WebApp es igual que la ficha Descriptor de cualquier otro tipo de archivo. Consulte el tema de Ayuda en línea “Creador de recopilatorios, Definición de las opciones del descriptor de archivos” para obtener más información.

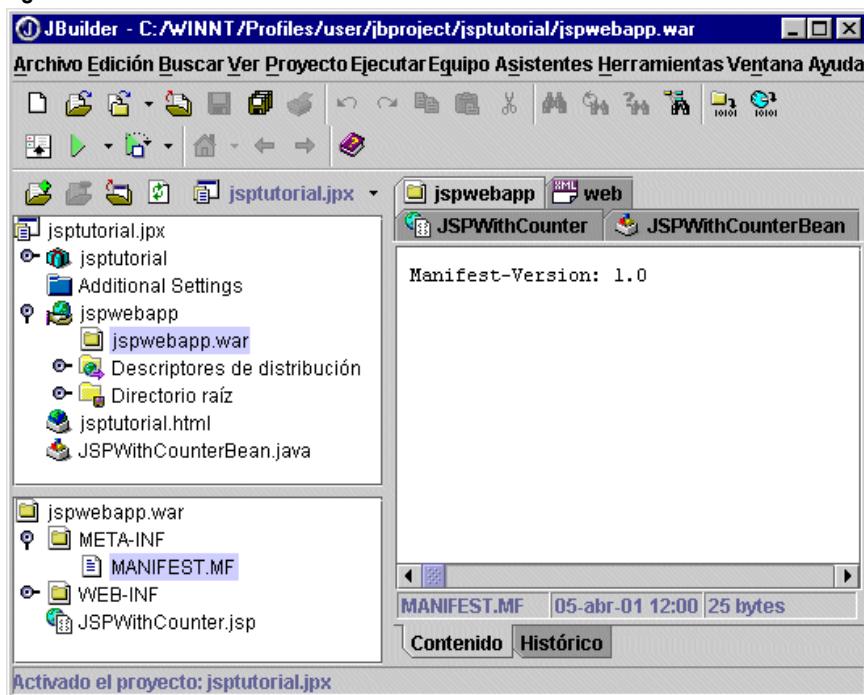
Figura 3.7 La ficha Descriptor del cuadro de diálogo Propiedades de WebApp

El archivo WAR

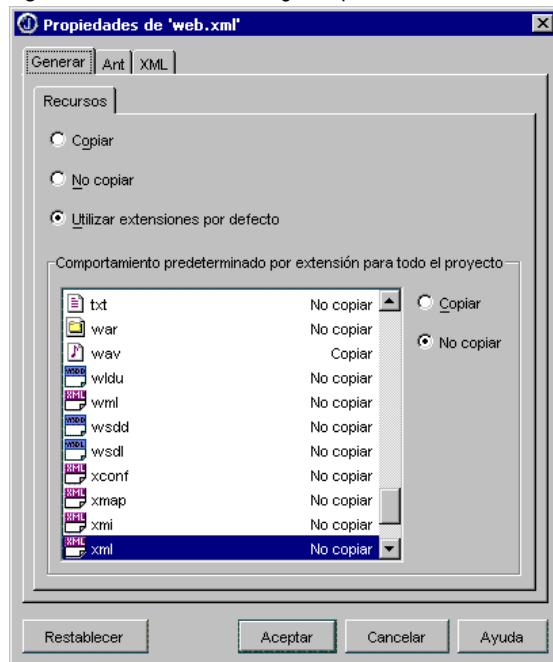
El archivo WAR es un recopilatorio de la aplicación web. Poniendo en el archivo WAR todos los archivos y recursos que se necesitan para la WebApp, se facilita su distribución. Asumiendo que todo está correctamente configurado y que todos los recursos necesarios se encuentran en el archivo WAR, todo lo que se necesita para la distribución es copiar el archivo WAR en la ubicación correcta del servidor web.

Si está presente un archivo WAR, aparece un nodo que lo representa debajo del nodo de la WebApp en el árbol del proyecto. Al abrir el nodo del archivo WAR, se muestran sus contenidos en el panel de estructura y también se muestran los visualizadores de archivo WAR en el Visualizador de aplicaciones. Los visualizadores del archivo WAR constan de las pestañas Contenido e Histórico.

Figura 3.8 El nodo del archivo WAR abierto en JBuilder IDE



El nodo del archivo WAR tiene un cuadro de diálogo de Propiedades, al que se accede haciendo clic con el botón derecho sobre el nodo y seleccionando Propiedades. En el cuadro de diálogo Propiedades se puede especificar si el archivo WAR se considera un recurso de Java. Esta es la misma ficha Recursos que está disponible en el cuadro de diálogo Generar propiedades de todos los tipos de archivos distintos de Java. Para obtener más información acerca de Propiedades de recursos, consulte el apartado “Recursos” del tema de ayuda en línea “Ficha Generar (Cuadro de diálogo Propiedades del proyecto)”. Las propiedades más importantes para el archivo WAR se encuentran en la ficha WebApp del cuadro de diálogo Propiedades de WebApp.

Figura 3.9 Cuadro de diálogo Propiedades del archivo WAR

Con el fin de conseguir que JBuilder cree un archivo WAR para su aplicación web, primero debe tener un nodo WebApp en su proyecto. Esta WebApp puede crearse con el Asistente para aplicaciones web, disponible en la galería de objetos o se puede utilizar la WebApp por defecto.

Puede indicar a JBuilder que cree o actualice automáticamente un archivo WAR siempre que se genere el proyecto, siempre que se genere la WebApp, en ambos casos o en ninguno de los dos. Para cambiar la configuración del archivo WAR de la WebApp:

- 1** Para hacer esto, haga clic con el botón derecho sobre el nodo WebApp en el panel del proyecto y seleccione Propiedades en el menú.
- 2** Seleccione la pestaña WebApp del cuadro de diálogo de Propiedades de WebApp.
- 3** Elija la opción deseada para Generar en la lista desplegable. Están disponibles las siguientes opciones de generación:
 - Al generar el proyecto o la webapp.
 - Sólo al generar la webapp.
 - Sólo al generar el proyecto.
 - Nunca.

- 4 Verifique que son correctos el nombre y la ubicación del archivo WAR que se desea.

Existe la opción de comprimir el archivo WAR, si así se desea.

Ahora tiene un archivo WAR asociado con su WebApp, recuerde que las diferentes piezas de su aplicación web deben estar asociadas con la WebApp que se añadirá al archivo WAR. Muchos de los asistentes disponen de una lista desplegable WebApp para que usted seleccione una WebApp cuando se crean estas partes.

Applets en el archivo WAR

Se puede añadir un applet al archivo WAR, pero se requiere algún trabajo adicional. Con el fin de que el applet funcione en la WebApp, las clases deben ser accesibles. Esto significa que el archivo HTML y las clases para la applet deben situarse debajo del directorio raíz de la WebApp o en uno de sus subdirectorios. No pueden estar debajo del directorio WEB-INF o de sus subdirectorios.

El Asistente para applets no admite que se coloquen los archivos de applet dentro de una WebApp; por lo tanto, tendrá que mover el archivo HTML de la applet y los archivos de clase compilados al directorio raíz de la WebApp (o a uno de sus subdirectorios). Dependiendo de a donde se muevan los archivos de clase, quizás deba modificar el atributo codebase de la etiqueta <applet> de forma que pueda encontrarlos.

Si se incluyen archivos de clase applet sueltos (sin archivar) en una WebApp, asegúrese de seleccionar Archivo De Clase Java como uno de los tipos de archivo incluidos en la ficha WebApp del cuadro de diálogo Propiedades de WebApp. De otra forma, los archivos de clase no se incluirán en el WAR.

También se puede poner un archivo JAR de applet en el archivo WAR. Se aplican las mismas reglas. El archivo JAR necesita estar accesible. Esto significa que va debajo del directorio raíz de la WebApp, o de uno de sus subdirectorios, pero no debajo del directorio WEB-INF.

4

Los servlets

El desarrollo web es una
función de JBuilder
Enterprise

Los servlets Java proporcionan un protocolo y un método independiente de las plataformas para crear aplicaciones web sin las limitaciones de rendimiento de los programas CGI. Un servlet se ejecuta dentro de un servidor web y, al contrario que un applet, no necesita una interfaz gráfica de usuario. Lo que hacen los servlets es interactuar, por medio de peticiones y respuestas, con el motor de servlets que funciona en el servidor web. Un programa cliente, que puede estar escrito en cualquier lenguaje de programación, accederá al servidor web y efectuará una solicitud. La solicitud es procesada entonces por el motor de servlets del servidor Web, que la pasa al servlet. Entonces, el servlet, a través del servidor web, devuelve una respuesta al cliente.

En la actualidad, los servlets se utilizan con frecuencia para crear aplicaciones web interactivas. Existe una amplia variedad de servidores web creados por otras compañías con extensiones de motores de servlets: Tomcat (la implementación de referencias de la API Servlet/JSP), iPlanet de Sun y otros. Los servidores web con motores para servlets, conocidos además como contenedores servlet, pueden también integrarse con servidores de aplicaciones preparados para web. JBuilder Enterprise incluye el Borland Enterprise Server y también es compatible con WebLogic, WebSphere e iPlanet de Sun.

Nota JBuilder Enterprise incorpora Tomcat como el servidor web por defecto. Una ventaja fundamental de la tecnología servlet es la velocidad. A diferencia de los programas CGI, los servlets se cargan en la memoria una vez y se ejecutan desde la memoria después de su carga inicial. Los servlets se generan como un hilo y son, por naturaleza, multihilo. Además, desde el momento en que están basados en lenguaje Java, son independientes de la plataforma.

La tecnología JSP (JavaServer Page - Página de servidor Java) es una ampliación de la tecnología de servlets, creada específicamente para aceptar el diseño de páginas HTML y XML. La tecnología JSP facilita la combinación de datos de plantillas fijas o estáticas y contenido dinámico. Incluso aunque los servlets sean suficientes de momento, existen buenas razones para considerar la posibilidad de utilizar la tecnología JSP como complemento. Para obtener más información sobre la escritura de páginas JavaServer, consulte el [Capítulo 6, “Desarrollo de Páginas JavaServer”](#).

Si desea obtener más información sobre los servlets, consulte las siguientes sedes web. Las direcciones y vínculos facilitados son válidos a la fecha de impresión de este manual. Las sedes web no pertenecen a Borland, por lo que no se puede garantizar su contenido ni su continuidad.

- [Java Servlet Technology en `http://java.sun.com/products/servlet/index.html`](http://java.sun.com/products/servlet/index.html)
- [Java Servlet Technology: Artículo de fondo en `http://java.sun.com/products/servlet/whitepaper.html`](http://java.sun.com/products/servlet/whitepaper.html)
- [La página Java Servlet Technical Resources en `http://java.sun.com/products/servlet/technical.html`](http://java.sun.com/products/servlet/technical.html)
- [La página Java Servlet Third-Party Resources en `http://java.sun.com/products/servlet/resources.html`](http://java.sun.com/products/servlet/resources.html)
- [El Servlet Trail del tutorial de Java en `http://java.sun.com/docs/books/tutorial/servlets/index.html`](http://java.sun.com/docs/books/tutorial/servlets/index.html)

También puede consultar los siguientes tutoriales con el fin de obtener más información sobre la creación de servlets en JBuilder:

- [Capítulo 16, “Tutorial: Creación de un servlet sencillo”](#)
- [Capítulo 17, “Tutorial: Creación de un servlet que actualiza un libro de visitas”](#)
- [Capítulo 19, “Tutorial: Creación de un servlet con InternetBeans Express”](#)

Servlets y JSP

Las tecnologías JSP y Servlets tienen distintas ventajas. ¿Cómo decidir cuál es la más adecuada para cada situación?

- Los *servlets* son una herramienta de programación adecuada para las funciones de aplicación de bajo nivel que no requieren elevadas prestaciones de presentación.

- Las páginas JSP constituyen una forma declarativa, centrada en las presentaciones, de enlazar contenido dinámico y lógica de programación. Las páginas JSP son adecuadas para gestionar la representación del código HTML que generan las páginas. Están escritas como páginas similares a las HTML, con una estructura y un contenido que no resultan ajenos a los proveedores de contenido web. Sin embargo, las páginas JSP son mucho más versátiles que las páginas HTML convencionales. Las páginas JSP pueden gestionar código de aplicaciones mediante el uso de componentes JavaBeans, Enterprise JavaBeans (EJB) y etiquetas personalizadas. También se pueden utilizar como componentes de presentaciones modulares y reutilizables, que pueden asociarse con un mecanismo de plantillas.

Las páginas JSP se compilan en servlets, por lo que en teoría se pueden escribir servlets que acepten las aplicaciones web. Sin embargo, la tecnología JSP se ha desarrollado con el objeto de simplificar el proceso de creación de páginas separando la presentación del contenido. En muchas aplicaciones, la respuesta que se envía al cliente es una combinación de datos de plantilla y datos generados dinámicamente. En estos casos resulta mucho más fácil trabajar con páginas JSP que hacerlo todo con servlets.

Servlets y servidores web

En 1999, Sun Microsystems entregó las entonces últimas versiones de las API de Servlet y JSP a la Apache Software Foundation. Apache, junto con Sun y otras muchas compañías, desarrolló e hizo pública la implementación oficial de referencia para JSP/Servlet, denominada Tomcat. Es la única implementación de referencia disponible. Tomcat está disponible de forma gratuita para cualquier compañía o desarrollador. Si desea obtener más información acerca de la Apache Software Foundation y Tomcat, diríjase a <http://jakarta.apache.org>. Para obtener más información sobre JBuilder y los servidores web, consulte el [Capítulo 9, "Configuración del servidor web"](#).

JBuilder Enterprise incluye Tomcat, que se puede utilizar como servidor web para desarrollar y comprobar las páginas JSP y los servlets dentro del entorno de desarrollo JBuilder. Existen otros muchos servidores web que admiten los servlets de Sun y las API de JSP. Si desea obtener una lista de estos productos, consulte el tema "Servers and Engines" en la página *Servlet Technology Industry Momentum* de Sun en <http://java.sun.com/products/servlet/industry.html>.

La API de servlet

La API de servlet está contenida en el paquete `javax.servlet`. Todos los servlets deben implementar, directa o indirectamente, la interfaz `javax.servlet.Servlet`. Esta interfaz permite al servlet ejecutarse en un motor servlet (una extensión de un navegador web). También define el ciclo de vida de los servlets. La API de servlet está incrustada en muchos servidores web, incluyendo Tomcat, el servidor web por defecto suministrado con JBuilder. La siguiente tabla ofrece una lista de las clases e interfaces utilizadas con más frecuencia en el paquete `Servlet` y suministra una breve descripción de cada una.

Tabla 4.1 Introducción general a la API de Servlet

Nombre	Clase o interfaz	Descripción
GenericServlet	Clase	Define un servlet genérico independiente del protocolo.
RequestDispatcher	Interfaz	Define un objeto que recibe solicitudes del cliente y las envía hacia cualquier recurso (tales como un servlet, un archivo HTML o un archivo JSP) en el servidor.
Servlet	Interfaz	Define métodos que deben implementar todos los servlets.
ServletConfig	Interfaz	Un objeto de configuración de servlet usado por un contenedor de servlet para pasar información a un servlet durante la inicialización.
ServletContext	Interfaz	Define un conjunto de métodos que utiliza un servlet para comunicarse con su contenedor de servlet, por ejemplo, para obtener el tipo MIME de un archivo, resolver solicitudes o escribir en un archivo histórico.
ServletException	Clase	Define una excepción general que puede lanzar un servlet cuando encuentre dificultades.
ServletInputStream	Clase	Suministra un flujo de entradas para leer datos binarios de una solicitud del cliente, incluyendo un eficiente método <code>readLine</code> para leer datos línea a línea.
ServletOutputStream	Clase	Proporciona un flujo de salida para enviar datos binarios al cliente.
ServletRequest	Interfaz	Define un objeto para proporcionar información acerca de las solicitudes del cliente a un servlet.
ServletResponse	Interfaz	Define un objeto para ayudar a un servlet a enviar una respuesta al cliente.
SingleThreadModel	Interfaz	Asegura que los servlets manejen solamente una solicitud a la vez.
UnavailableException	Clase	Define una excepción que un servlet lanza para indicar que no está disponible temporal o permanentemente.

El paquete servlet.HTT P

El paquete `javax.servlet.http` se utiliza para crear servlets que admiten el protocolo HTTP y la generación de HTML. El protocolo HTTP utiliza un conjunto de métodos de solicitudes y de respuestas basadas en texto (métodos HTTP), que incluyen:

- GET
- POST
- PUT
- DELETE
- HEAD
- TRACE
- CONNECT
- OPTIONS

La clase `HttpServlet` implementa estos métodos HTTP. Para empezar, simplemente se extiende `HttpServlet` y se redefinen los métodos `doGet()` o `doPost()`. Con el fin de obtener un mayor control, también se pueden redefinir los métodos `doPut()` y `doDelete()`. Si se crea un servlet con el asistente para Servlets de JBuilder, se pueden seleccionar los métodos que se quieren redefinir y JBuilder genera el esqueleto del código automáticamente.

La siguiente tabla presenta una lista de las clases e interfaces más habitualmente utilizadas en el paquete `javax.servlet.http` y suministra una breve descripción de cada una.

Tabla 4.2 Interfaces y paquetes de clases servlet utilizados habitualmente

Nombre	Clase o interfaz	Descripción
<code>Cookie</code>	Clase	Crea una cookie, un pequeño fragmento de información enviada por un servlet a un navegador web que es almacenado por el navegador y posteriormente devuelta al servidor.
<code>HttpServlet</code>	Clase	Proporciona una clase abstracta que se convierte en subclase para crear un servlet HTTP adecuado para una página web.
<code>HttpServletRequest</code>	Interfaz	Extiende la interfaz <code>ServletRequest</code> para proporcionar una solicitud de información para los servlets HTTP.
<code>HttpServletResponse</code>	Interfaz	Extiende la interfaz <code>ServletResponse</code> con el objeto de suministrar una funcionalidad específica de HTTP en el envío de una respuesta.

Tabla 4.2 Interfaces y paquetes de clases servlet utilizados habitualmente (continuación)

Nombre	Clase o interfaz	Descripción
HttpSession	Interfaz	Proporciona una manera de identificar a un usuario a través de más de una solicitud de página o de una visita a una sede web y para almacenar la información relativa al usuario.
HttpSessionBindingEvent	Clase	Envía un objeto que implementa HttpSessionBindingListener cuando el objeto se enlaza a una sesión o se desenlaza de la misma.
HttpSessionBindingListener	Interfaz	Provoca que se notifique a un objeto cuando se enlaza o se desenlaza de una sesión.

El ciclo de vida de los servlets

La interfaz `javax.servlet.Servlet` contiene los métodos del ciclo de vida del servlet. Estos métodos se implementan para:

- Construir el servlet e inicializarlo con el método `init()`.
- Gestionar las llamadas de los clientes al método `service()`.
- Dejar fuera de servicio el servlet, destruyéndolo con el método `destroy()` y realizar tareas de liberación de la memoria no utilizada.

Construcción e inicialización del servlet

Cuando se inicia el motor de servlets o cuando un servlet necesita responder a una solicitud, el motor de servlets llama al método `init()` para informar a un servlet que está siendo puesto en servicio. El motor de servlets llama a `init()` solamente una vez tras haber instanciado el servlet. Debe completarse el método `init()` antes de que el servlet pueda recibir solicitudes.

El parámetro `ServletConfig` del método `init()` es un objeto que contiene la configuración del servlet y sus parámetros de inicialización. (Después de que se haya inicializado el servlet, se puede utilizar `getServletConfig()` para recuperar esta información.)

Una vez que el servlet se ha cargado en memoria, puede residir en un sistema de archivos local, en un sistema de archivos remoto o en una red.

Gestión de las peticiones del cliente

El motor de servlets llama al método `service()` para permitir que el servlet responda a una solicitud. Los objetos de solicitud y de respuesta se pasan como parámetros al método `service()` cuando un cliente efectúa una solicitud.

El servlet también puede implementar las interfaces `ServletRequest` y/o `ServletResponse` para permitir que el servlet acceda a los parámetros de la solicitud y a los datos de la respuesta. Los parámetros de la solicitud incluyen datos o métodos de protocolo. Los datos de la respuesta incluyen las cabeceras de la respuesta y los códigos de estado.

Servlets y multihilo

Típicamente, los servlets son multihilo, lo que permite que un servlet sencillo gestione múltiples solicitudes de forma concurrente. Como desarrollador, debe hacer que cualquier recurso, tales como archivos, conexiones de red y variables de clase y de instancia del servlet, pueda ser gestionado mediante hilos con garantía. Para obtener más información sobre los hilos y su seguridad, consulte “Técnicas de hilos” en *Procedimientos iniciales con Java*.

Se puede crear un servlet con un hilo único, utilizando la interfaz `SingleThreadModel`. Esta interfaz permite que un servlet responda solamente a una solicitud a la vez. Esto no suele ser práctico para los servlets. Si un servlet está restringido a un solo hilo, el servidor web coloca las solicitudes en cola e inicia otra instancia del servlet para dar respuesta a la demanda.

Destrucción de un servlet

Un motor de servlets no mantiene cargado un servlet durante un cierto período de tiempo especificado o durante el tiempo de vida del servidor. Los motores de servlets pueden retirar los servlets en cualquier momento. A causa de ello, se deben programar los servlets de forma que no almacenen información de estado. Con el fin de liberar recursos, utilice el método `destroy()`.

Servlets con enlace a HTML

Los servlets pueden generar fácilmente texto con formato HTML, lo que permite utilizar los servlets para generar o modificar dinámicamente páginas HTML. Con la tecnología servlet, no se necesita usar lenguajes de script. Por ejemplo, se pueden utilizar los servlets para personalizar la utilización que un usuario hace de una página Web modificando continuamente dicha página HTML.

Si su servidor web cuenta con la funcionalidad SSI, puede utilizar la etiqueta del <servlet> para preprocesar páginas web. Esta etiqueta debe estar en un archivo con una extensión .shtml. La etiqueta <servlet> informa al servidor web de que un servlet preconfigurado debería cargarse e inicializarse con el conjunto dado de parámetros de configuración. La salida del servlet se incluye en una respuesta con formato HTML. Al igual que ocurre con la etiqueta <applet>, se pueden utilizar los atributos class y codebase para especificar las ubicaciones del servlet.

Un ejemplo de una etiqueta <servlet> es como sigue:

```
<servlet>
  codebase=""
  code="dbServlet.Servlet1.class"
  param1=in
  param2=out
</servlet>
```

El Asistente para servtles de JBuilder puede crear una etiqueta <servlet> en un archivo .shtml.

Servlets específicos para HTTP

Los servlets utilizados con protocolo HTTP (mediante la extensión de `HTTPServlet`), pueden admitir cualquier método HTTP. Pueden redireccionar las solicitudes hacia otras ubicaciones y enviar mensajes de error específicos para HTTP. Adicionalmente, pueden acceder a los parámetros que se hayan pasado a través de formularios HTML estándar, incluyendo el método HTTP a realizar y la URI que identifica el destino de la solicitud:

```
String method = request.getMethod();
String uri = request.getRequestURI();
String name = request.getParameter("name");
String phone = request.getParameter("phone");
String address = request.getParameter("address");
String city = request.getParameter("city");
```

En el caso de los servlets HTTP, los datos de solicitud y respuesta se suministran como datos con el formato MIME. El servlet especifica el tipo de datos y los escribe codificados en ese formato. Eso permite que el servlet reciba datos de entrada de un tipo determinado y devuelva datos con la forma apropiada para esa solicitud.

Cómo se utilizan los servlets

Debido a la robusta funcionalidad de la API de los servlets, éstos pueden utilizarse en una gran variedad de formas:

- Como parte de un sistema de introducción y procesamiento de órdenes, que trabajan con las bases de datos de clientes, productos e inventarios. Por ejemplo, un servlet podría procesar datos, como los de una tarjeta de crédito, y enviarlos (POST) sobre HTTPS utilizando una etiqueta <form> HTML.
- En conjunción con una applet, como parte de una intranet corporativa, para seguir la trayectoria de la información de los empleados o de los historiales de salarios.
- Como parte de un sistema colaborativo (p.e., un sistema de envío de mensajes) en donde uno o varios servlets manejan solicitudes múltiples de forma concurrente.
- Como parte de un proceso de equilibrado de carga, en donde los servlets envían solicitudes en cadena.
- Como un servlet de enlace a HTML, para insertar datos formateados en una página web desde una consulta de base de datos o desde una búsqueda en la web, o para crear banners publicitarios con destino personalizado.

Distribución de servlets

Los servlets pueden distribuirse autónomamente a un servidor web de producción. También pueden distribuirse como un módulo J2EE. Un módulo J2EE consta de uno o más componentes J2EE del mismo tipo (web, EJB, cliente, etc.) que comparten un descriptor de distribución.

Para obtener más información y sugerencias sobre distribución, consulte el [Capítulo 11, “Distribución de aplicaciones web”](#).

Los servlets en JBuilder

El desarrollo web es una
función de JBuilder
Enterprise

En JBuilder Enterprise, puede utilizar el Asistente para servlets con el fin de crear un archivo Java que derive de `javax.servlet.http.HttpServlet`. El Asistente para servlets de JBuilder crea servlets que generan los siguientes tipos de contenido:

- HTML (HyperText Markup Language, lenguaje de marcas de hipertexto).
- XHTML (redefinición de HTML 4.0 como aplicación de XML).
- WML (Wireless Markup Language, lenguaje de marcas para telefonía móvil).
- XML (eXtensible Markup Language, lenguaje de marcas ampliable).

Con el Asistente para servlets, también se pueden crear servlets estándar, servlets de filtro o servlets de monitor (si su servidor web admite las especificaciones Java Servlets v2.3).

Opciones del Asistente para servlets

El Asistente para servlets facilita la creación de servlets. Para iniciar el asistente, seleccione Archivo | Nuevo. Pulse la pestaña Web de la galería de objetos, seleccione Servlet y pulse Aceptar. (En primer lugar hay que activar un servidor para el proyecto.)

Ficha Seleccione el nombre y el tipo de servlet

En el primer paso del Asistente para servlets, se puede seleccionar el paquete al que pertenecerá el servlet. También se puede introducir el nombre del servlet en el campo Clase. Si se quieren reproducir los

comentarios de la cabecera añadidos a otras clases en el proyecto (consulte el tema de Ayuda en línea Asistente para proyectos), seleccione la opción Generar comentarios de cabecera.

La opción Modelo de hilo único implementa la interfaz SingleThreadModel. Si se selecciona esta opción, el servlet será un poco menos eficaz, ya que el servidor web pondrá en cola las solicitudes e iniciará otra instancia del servlet para atender la demanda.

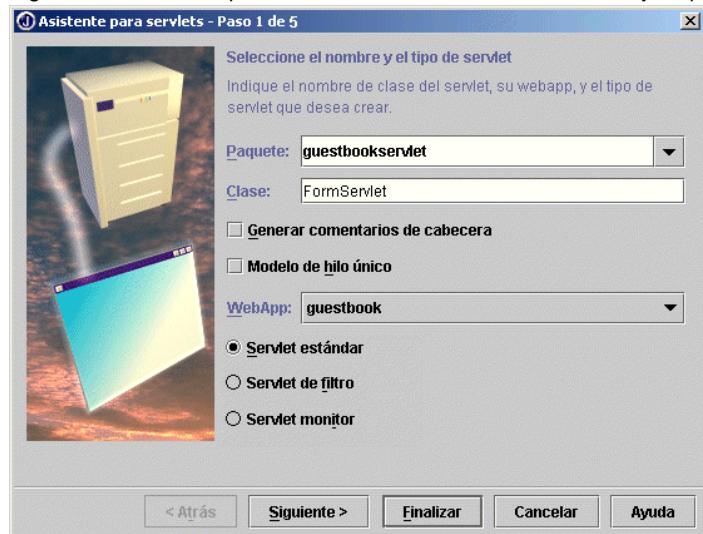
Seleccione de la lista desplegable de WebApp el nombre de la WebApp bajo la cual quiere que se ejecute este servlet. Todos los archivos de contenido web, como un archivo SHTML de un servlet, se colocan en el directorio WebApp.

Las opciones en la parte inferior del cuadro de diálogo permiten seleccionar el tipo de servlet que se está creando.

Nota Estas opciones están disponibles solamente si su servidor web admite las especificaciones de Java Servlets v2.3. (Tomcat 4.x es la implementación de referencia de estas especificaciones.) De otra forma, se creará por defecto un servlet estándar.

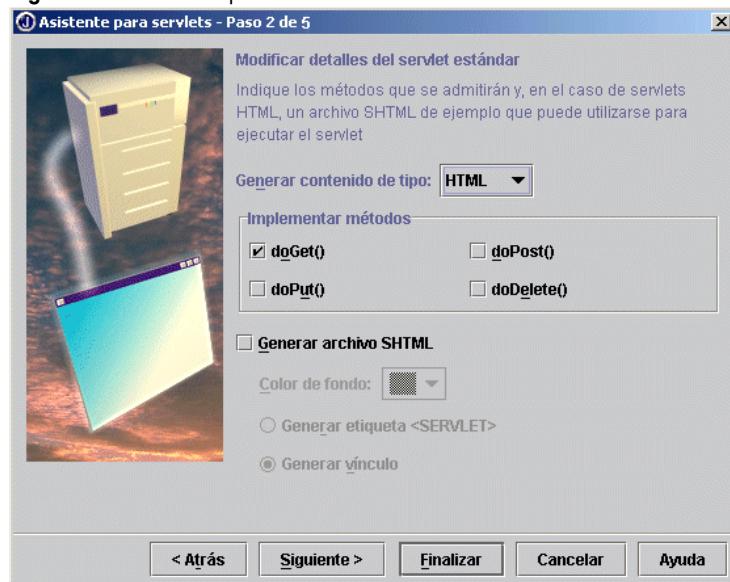
Tabla 5.1 Opciones de tipo de Servlet

Tipo de servlet	Descripción
Servlet estándar	Un servlet que no es de filtro ni monitor. La opción Servlet estándar está disponible siempre que se seleccione la WebApp <default> o que se nombre una WebApp. Cuando esta opción se encuentra seleccionada en la WebApp por defecto, el nombre del servlet es por defecto el nombre de la clase simple del servlet, en minúsculas (consulte el apartado “ Ficha Introduzca los detalles de webapp ” en la página 5-7). Si el WebApp ya tiene nombre, el servlet URL (también en la ficha Introduzca los detalles de webapp) tiene por defecto el modelo URL: /servletname (todo en minúsculas).
Servlet de filtro	Un servlet que actúa como un filtro para otro servlet o para la WebApp. Además del nombre, se debe seleccionar un modelo de URL para el otro servlet (que debe tener un nombre). Si esta opción está seleccionada, la ficha Introduzca los detalles de webapp, donde se da un nombre al servlet y se especifica la asignación del filtro, sería la única ficha disponible del asistente para servlets.
Servlet monitor	Un servlet que se añade a la lista de monitores de la WebApp. Si selecciona esta opción, la ficha Introduzca los detalles del servlet monitor es la única disponible en el Asistente para servlets.

Figura 5.1 Asistente para servlets — ficha Seleccione el nombre y el tipo de servlet

Ficha Modificar detalles del servlet estándar

Si se ha seleccionado Servlet estándar como tipo de servlet en la ficha Seleccione el nombre y el tipo de servlet del Asistente para servlets, la ficha Modificar detalles del servlet estándar es el Paso 2 del asistente. Esta ficha permite seleccionar el tipo de contenido del servlet, los métodos que se van a implementar y el archivo SHTML que se va a generar.

Figura 5.2 Asistente para servlets— ficha Modificar detalles del servlet estándar

Opción Generar contenido de tipo

Se utiliza la lista desplegable Generar contenido de tipo, para seleccionar el contenido del servlet. Entre las opciones están:

- HTML (HyperText Markup Language, lenguaje de marcas de hipertexto). Un lenguaje de marcas para documentos de hipertexto utilizados en Internet. HTML permite incrustar imágenes, sonidos, vídeos, campos de formulario y referencias a otros objetos mediante direcciones URL, así como un funciones básicas de formato de texto.
- XHTML: Una reformulación de HTML 4.0 como aplicación de XML. Las etiquetas y los atributos son casi idénticos a los de HTML, pero XHTML es más estricto y mejor estructurado. Por ejemplo, todas las etiquetas XHTML se escriben en minúsculas y todas las etiquetas de apertura deben tener una etiqueta de cierre. XHTML permite a los diseñadores Web evolucionar hacia una Web más modular y ampliable basada en XML, a la vez que se mantiene la compatibilidad con los navegadores HTML 4 actuales.

Se añade al archivo generado .java una línea de código que indica que DOC TYPE corresponde a XHTML. Su aspecto es el siguiente:

```
private static final String DOC_TYPE = "<!DOCTYPE html  
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN\"\n" +  
" \\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\\>";
```

Se añade un texto de las mismas características en la parte superior del archivo SHTML, si se ha generado.

- WML (Wireless Markup Language, lenguaje de marcas para telefonía móvil). Los archivos WML son archivos de texto normales con contenido XML. Debido a las restricciones de ancho de banda y capacidad de memoria de los dispositivos de telefonía móvil, el contenido debe compilarse en formato binario compacto para poder suministrarlo a los dispositivos compatibles con WAP (Protocolo de aplicaciones inalámbricas).

Para servlets WML, solamente se genera un archivo de clase .java. No existe la opción de generar archivos SHTML con servlets de tipo WML. El archivo de clase .java contiene una línea de código que indica que el TIPO DE DOCUMENTO es WML y el TIPO DE CONTENIDO es WAP. Tiene el siguiente aspecto:

```
private static final String CONTENT_TYPE = "text/vnd.wap.wml";  
private static final String DOC_TYPE = "<!DOCTYPE wml  
PUBLIC "-//WAPFORUM//DTD WML 1.2//EN\"\n" +  
" \\"http://www.wapforum.org/DTD/wml12.dtd\\>";
```

También se añade código que indica que la solicitud GET del servlet se debe ejecutar a petición del servlet WML. Su aspecto es el siguiente:

```
out.println("<?xml version=\"1.0\"?>");  
out.println(DOC_TYPE);  
out.println("<wml>");  
out.println("<card>");  
out.println("<p>The servlet has received a GET. This is the  
reply.</p>");  
out.println("</card></wml>");
```

- XML (eXtensible Markup Language, lenguaje de marcas ampliable). Lenguaje de marcas que permite definir las etiquetas (marcas) necesarias para identificar los datos y el texto de los documentos XML. Si desea más información sobre las funciones XML de JBuilder, consulte "Introducción" a la *Guía del desarrollador de aplicaciones XML*.

Para servlets XML, se genera una clase .java. La opción para crear un archivo SHTML se elimina. El archivo .java incluye código para identificar el servlet como un servlet-XML. Tiene el siguiente aspecto:

```
private static final String CONTENT_TYPE = "text/xml";
```

El método `doGet()` de la versión HTML del servlet se modifica con el código siguiente:

```
out.println("<?xml version=\"1.0\"?>");  
if (DOC_TYPE != null) {  
    out.println(DOC_TYPE);  
}
```

Opciones de Implementar métodos

Este área del Asistente para servlets proporciona opciones para la redefinición de los métodos estándar. `HttpServlet` proporciona una clase abstracta que se puede utilizar como subclase para crear un servlet HTTP, que recibe peticiones y envía respuestas a un cliente web. Cuando se utiliza `HttpServlet` como subclase, se debe modificar al menos un método, normalmente uno de los siguientes. Si desea obtener más información sobre los métodos, consulte la documentación sobre los servlets en <http://java.sun.com/products/servlet/index.html>.

Los siguientes métodos pueden generarse automáticamente en el servlet:

- `doGet()`: permite a un cliente leer información del servidor web mediante una cadena de consulta añadida a una dirección URL que indica al servidor qué información debe devolver. También se produce un GET cuando se escribe una dirección, se pulsa un enlace o se utiliza un marcador. Pase por alto este método si su servlet va a admitir solicitudes GET HTTP.

- `doPost()`: permite que el cliente envíe datos de longitud ilimitada al servidor web. También se produce un POST cuando se pulsa un botón Enviar en el navegador. Pase por alto este método si su servlet va a admitir solicitudes POST HTTP.
- `doPut()`: permite a un cliente colocar un archivo en el servidor, de forma similar al envío de un archivo mediante FTP. Pase por alto este método si el servlet admite peticiones HTTP PUT.
- `doDelete()`: permite a un cliente eliminar un documento o página web del servidor. Pase por alto este método si el servlet admite peticiones HTTP DELETE.

Opciones de los archivos SHTML

Las opciones Generar archivos SHTML se presentan si se seleccionó HTML o XHTML en la lista desplegable Generar contenido de tipo. Si desea llamar al servlet desde una página HTML, como se explica en “[Llamada a un servlet desde una página HTML](#)” en la página 5-13, seleccione la opción Generar archivo SHTML. Se añade al proyecto un archivo SHTML con el mismo nombre que el servlet. El archivo SHTML se sitúa en el directorio WebApp.

Si desea ejecutar el servlet directamente, como se explica en “[Llamada a un servlet desde una ventana de un visualizador](#)” en la página 5-12, no seleccione la opción Generar archivo SHTML.

Seleccione el color de fondo del archivo SHTML de la lista desplegable Color de fondo. Si desea conectar con el servlet desde el archivo SHTML, se puede utilizar bien una etiqueta `<servlet>` o bien una etiqueta de enlace `<a href>`.

- Si se selecciona la opción de etiqueta `<servlet>`, el archivo SHTML ejecuta el servlet usando una etiqueta `<servlet>`. La etiqueta contiene una propiedad `codebase` y `code`, similar a una applet. Por defecto, `codebase` se fija a la carpeta actual y `code` se fija al nombre de clase del servlet.
- Si elige ejecutar el servlet desde un enlace, se colocará una etiqueta `<a href>` en el archivo SHTML. La etiqueta se vincula con el patrón de URL del servlet. Es necesario ir a la siguiente ficha del asistente con el fin de que sea generado el mapeo del patrón de URL. Si pulsa Finalizar en esta ficha, se creará un enlace al nombre de la clase.

Ficha Introduzca los detalles de webapp

Esta ficha permite definir rápidamente correspondencias básicas de WebApps. (La WebApp seleccionada se utiliza para ejecutar o depurar el servlet.) El Asistente para servlets añade automáticamente las asignaciones de los servlets a las secciones Servlet o Filtros del archivo descriptor de distribución `web.xml`. Para obtener más información, consulte “[Servlets](#)” en la página 11-2 o “[Ficha Filtros](#)” en la página 12-5.

- Si ha seleccionado Servlet Estándar como tipo de servlet en la ficha Seleccione el nombre y el tipo de servlet, este es el Paso 3.
- Si se ha seleccionado Servlet de filtro, este es el Paso 2 y el paso final del Asistente para servlets.

Si desea obtener más información acerca de las asignaciones de nombres de servlets y patrones URL, consulte “[Cómo las URL ejecutan los servlets](#)” en la página 10-11.

El campo Nombre se aplica a servlets tanto estándar como de filtro. Este campo se presenta para los servlets estándar que se ejecutan tanto en las WebApps `<default>` como en las WebApps con nombre y para los servlets de filtro.

El asistente sugiere un nombre de servlet que es el mismo que el de la clase en minúsculas. Se sugiere el mismo nombre para la URI, precedido por una barra (todas las URI deben comenzar con una barra). Sin embargo, puede cambiar estos nombres por otros, siempre que sean válidos. Por ejemplo, podría tener el patrón de URL `/myhtmlfile.html` ejecutando el servlet llamado `custlist` que realmente es la clase `package.subpackage.CustomerList`.

La entrada en el campo Nombre es el nombre utilizado para identificar al servlet en el archivo `web.xml`. El nombre se muestra en el nodo `Servlets` en el panel de estructura para el archivo `web.xml`. También se utiliza para asignar el patrón de URL al servlet. El patrón de URL mapea explícitamente a un servlet mediante el nombre del mismo; no hay una asignación implícita entre el nombre del servlet y la URL. Por ejemplo, el siguiente URL:

`http://localhost:8080/guestbook/formservlet`

genera la siguiente URI (la URI permanece después de que el protocolo, el nombre de host y el número de puerto se han eliminados de la URL):

`/guestbook/formservlet`

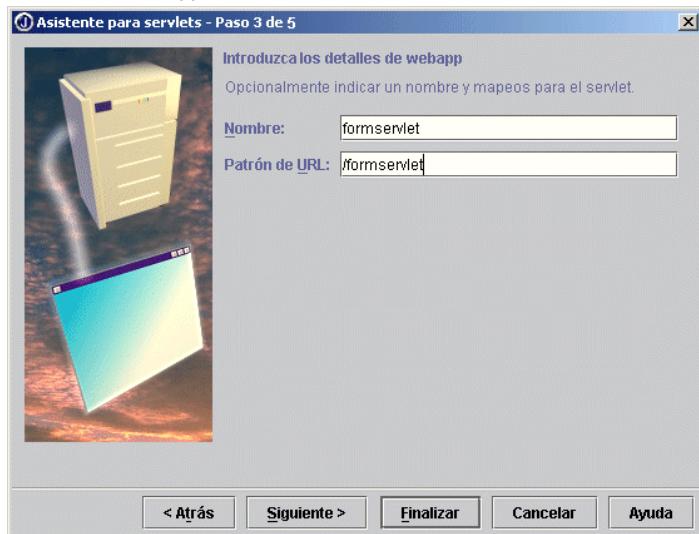
El principio de la URI se compara con todos los nombres de WebApp conocidos. Una coincidencia produce que la parte restante de la URI sea dirigida a esa particular WebApp/contexto.

Nota Si no hay coincidencia de WebApp/contexto, la URI se pasa tal cual a la WebApp por defecto.

La URI es entonces comparada con todos los patrones de URL conocidos. Estos patrones se encuentran en la ficha Servlets del Editor DD para WebApp. Un patrón determinado, tiene un correspondiente nombre de servlet. Ese nombre identifica al “servlet”, que podría ser también un archivo JSP, puesto que éstos se compilan en servlets.

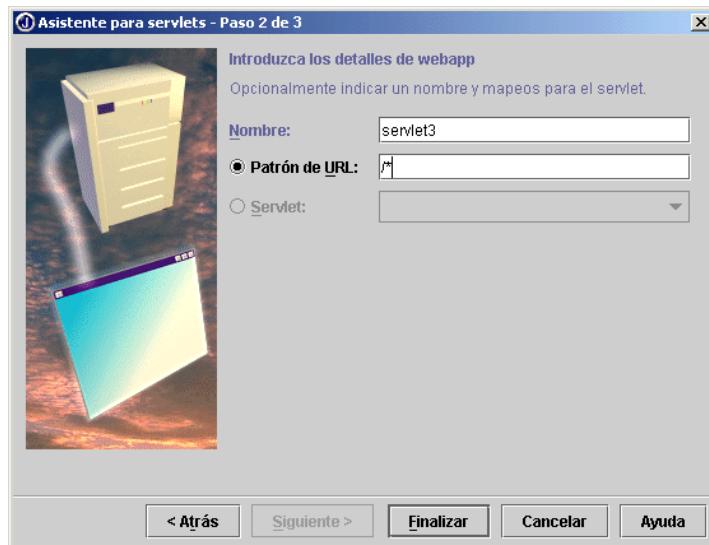
- Advertencia** Los mapeos URL no se generarán si pulsa el botón Finalizar en alguno de los pasos previos del asistente. Debe utilizar la ficha Introduzca los detalles de webapp para generar un mapeo URL.

Figura 5.3 Asistente para servlets – Servlet estándar, ficha Introduzca los detalles de webapp



Para los servlets de filtro, puede seleccionarse cómo asignar el servlet, bien con la opción Patrón de URL o con la opción Servlet.

La lista desplegable Servlet permite seleccionar otro servlet en el proyecto que será filtrado por este servlet. (Este campo contiene por defecto el nombre en minúsculas del primer servlet del proyecto, precedido por una barra. Por ejemplo, si su proyecto ya contiene `Servlet1.java` y `Servlet2.java`, ambos servlets estándar, el campo Servlet asignado contendría por defecto `/servlet1`.) La lista desplegable presenta todos los servlets del proyecto previamente asignados. Si no hay ninguno, la opción está desactivada.

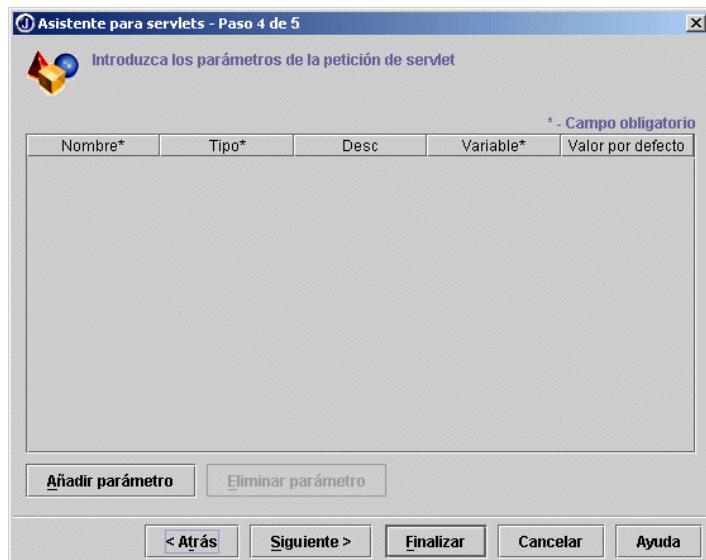
Figura 5.4 Asistente para servlets – Servlet de filtro, ficha Introduzca los detalles de webapp

Ficha Introduzca los parámetros de la petición del servlet

En esta ficha Introduzca los parámetros de la petición del servlet se especifican los parámetros del servlet. Los parámetros son valores que se pasan al servlet. Los valores podrían ser valores de entrada simples. No obstante, también podrían afectar a la lógica de ejecución del servlet. Por ejemplo, el valor introducido por el usuario podría determinar la tabla de la base de datos que se presentará o actualizará. Alternativamente, un valor introducido por el usuario podría determinar el color de fondo de una página HTML que genera el servlet.

El servlet descrito en el [Capítulo 16, “Tutorial: Creación de un servlet sencillo,”](#) utiliza un parámetro de tipo `String` para permitir la introducción del nombre de usuario. El nombre del parámetro en el archivo SHTML es `UserName`; la variable correspondiente, utilizada en el archivo de clase del servlet es `userName`.

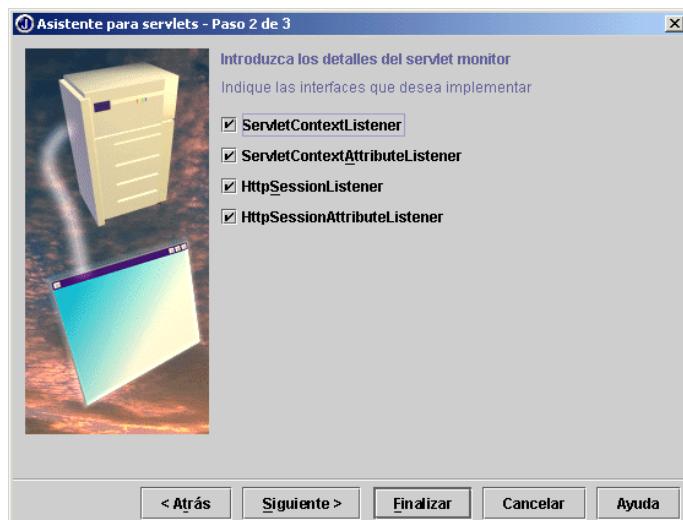
Figura 5.5 Asistente para servlets – ficha Introduzca los parámetros de la petición del servlet



Ficha Introduzca los detalles del servlet monitor

Esta ficha está disponible solamente si se ha seleccionado como tipo de servlet el de Servlet monitor en la ficha de denominación y tipo del Asistente para servlets. Esta ficha se utiliza para implementar una o más interfaces de servlet monitor. Los métodos correspondientes se añaden al servlet.

El Asistente para servlets añade automáticamente los monitores seleccionados a la sección Monitores del archivo descriptor de distribución `web.xml`. Para obtener más información, consulte "["Ficha Monitores"](#) en la página 12-8.

Figura 5.6 Asistente para servlets - ficha Introduzca los detalles del servlet monitor

Ficha Definir configuración de servlet

Esta ficha crea una configuración de ejecución para su servlet. Solamente necesita crear una configuración para el servlet que inicia el flujo de procesos en la aplicación, pero no para los servlets que llaman otros servlets.

Para crear la configuración, asegúrese de que está activada la opción Crear una configuración de ejecución. Escriba el nombre de la configuración en el campo Nombre. Puede utilizar una configuración existente como base; selecciónela en la lista desplegable Configuración base.

Si desea más información, consulte:

- “[Creación de una configuración de ejecución](#)” en la página 10-2
- “Definición de las configuraciones de ejecución” en el capítulo “Ejecución de programas en Java” de *Creación de aplicaciones con JBuilder*

Figura 5.7 Asistente para servlets – Ficha Definir configuración de servlet

Cómo llamar a los servlets

Llamada a un servlet desde una ventana de un visualizador

Se puede llamar a un servlet directamente escribiendo su URL dentro del campo de ubicación del navegador Web. No obstante, esto sólo funciona cuando el servidor web tiene un servlet llamador de servidor (un servlet cuyo único propósito es ejecutar otros servlets). Tomcat tiene un servlet llamador, pero muchos otros servidores no. El uso de un servlet llamador no se recomienda por no ser transportable.

Si decide utilizar un servlet llamador, la forma general del URL del servlet, donde *nombredeclasedeservlet* corresponde al nombre de clase del servlet, es:

`http://nombredeequipo:número de puerto/servlet/nombredeclasedeservlet`

Por ejemplo, cualquier URL que cumpliera con uno de los siguientes formatos podría servir para ejecutar un servlet:

- `http://localhost:8080/servlet/Servlet1` (que se ejecuta en el ordenador local)
- `http://www.borland.com/servlet/Servlet1` (ejecución desde su URL)
- `http://127.0.0.1/servlet/Servlet1` (que se ejecuta desde esta dirección IP)

Nota

Si se omite el número de puerto, el protocolo HTTP escoge por defecto el puerto 80. La primera URL en el ejemplo anterior se usaría en el IDE si se ha configurado como puerto de configuración de ejecución el 80 y no se está ejecutando ya un servidor Web en este puerto. Los otros ejemplos se utilizarían en situaciones reales, después de que la aplicación web haya sido distribuida en un servidor web.

Las direcciones URL de servlets pueden contener consultas para las peticiones HTTP GET. Por ejemplo, el servlet que se muestra en el [Capítulo 16, "Tutorial: Creación de un servlet sencillo,"](#) puede ejecutarse introduciendo la siguiente URL para el servlet:

```
http://localhost:8080/servlet/simplesservlet.Servlet1?userName=Mary
```

`simplesservlet.Servlet1` es el nombre de clase completo. El `?` indica que una cadena de consulta se añade a la URL. `userName=Mary` es el nombre y el valor del parámetro.

Si `simplesservlet.Servlet1` se asigna al patrón de URL `/firstservlet`, debería escribir:

```
http://localhost:8080/firstservlet?userName=Mary
```

Para obtener más información sobre cómo se ejecutan los servlets, consulte el tutorial ["Cómo las URL ejecutan los servlets" en la página 10-11.](#)

Llamada a un servlet desde una página HTML

Para llamar a un servlet desde una página HTML basta con utilizar la dirección URL del servlet dentro de la correspondiente etiqueta HTML. Entre las etiquetas que reciben direcciones URL se encuentran las que empiezan con anclas y formularios, y las etiquetas META. Las URL de servlets se pueden utilizar como etiquetas HTML en cualquier parte de una URL normal, como el destino de un ancla, una acción de un formulario y la ubicación que deberá utilizarse cuando una etiqueta meta indique que hay que actualizar una página. En este apartado se supone que el lector está familiarizado con HTML. Si no conoce HTML, puede aprenderlo consultando diversos libros o la *HTML 4.0 Reference Specification*, disponible en la dirección <http://www.w3.org/TR/html4/#minitoc>.

Por ejemplo, en el [Capítulo 16, "Tutorial: Creación de un servlet sencillo,"](#) la página SHTML contiene un ancla con un servlet como destino:

```
<a href="/servlet1">Click here to call Servlet: Servlet1</a><br>
```

Las páginas HTML suelen utilizar también las siguientes etiquetas para realizar llamadas a servlets:

- Una etiqueta <form>

```
<form action="http://localhost:8080/servlet1" method="post">
```

- Una etiqueta meta que usa una dirección URL de servlet como parte del valor del atributo http-equiv.

```
<meta http-equiv="refresh" content="4;url=http://localhost:8080/servlet1">
```

Tenga en cuenta que se puede sustituir el patrón de URL del servlet para el nombre completo. Por ejemplo, si el nombre de clase del servlet es `Servlet1`, puede utilizar la siguiente etiqueta <form> para ejecutarlo:

```
<form action="http://localhost:8080/simple servlet.Servlet1" method="post">
```

Internacionalización de servlets

Los servlets plantean un interesante problema de internacionalización. Puesto que el servlet envía al cliente código fuente HTML, si ese código HTML contiene caracteres no disponibles en el juego de caracteres que admite el servidor sobre el que se ejecuta el servlet, podría suceder que el visualizador del cliente fuese incapaz de leerlos. Por ejemplo, si el sistema de codificación utilizado en el servidor es ISO-8859-1, pero el código HTML generado por el servlet contiene caracteres de doble byte, esos caracteres no aparecerán correctamente en el visualizador del cliente, incluso aunque se haya configurado correctamente para visualizarlos.

Especificando un tipo de codificación en el servlet, podemos situar el servlet en cualquier servidor, sin necesidad de conocer el tipo de codificación que dicho servidor utiliza. Los servlets pueden responder también a los datos introducidos por el cliente, y generar código HTML en un determinado idioma.

He aquí un ejemplo que muestra cómo especificar el tipo de codificación dentro del servlet. En el código fuente Java generado por el Asistente para servlets, el método `doPost()` contiene la siguiente línea:

```
PrintWriter out = response.getWriter();
```

Esta línea puede sustituirse por lo siguiente:

```
OutputStreamWriter writer = new OutputStreamWriter(
    response.getOutputStream(), "encoding");
PrintWriter out = new PrintWriter(writer);
```

El segundo argumento del constructor `OutputStreamWriter` es una cadena que representa el tipo de codificación deseado. Esta cadena puede residir en un recurso, estar codificada directamente en el programa, o definirse mediante una variable. Al llamar a `System.getProperty("file.encoding")`, se devolverá una cadena que representa el tipo de codificación utilizado actualmente por el sistema.

Si en la llamada al constructor `OutputStreamWriter` sólo se especifica el primer argumento, se empleará el tipo de codificación actual.

Escritura de servlets enlazados a datos

Las tecnologías de desarrollo de JBuilder incluyen el InternetBeans Express API para simplificar la creación de servlets con enlace a datos. InternetBeans Express es un conjunto de componentes que leen fichas HTML y generan código HTML a partir de modelos de datos DataExpress, facilitando la creación de servlets y páginas JSP asociados a datos. Los componentes generan HTML y se utilizan con servlets y JSP para crear contenido dinámico. Las asociaciones y optimizaciones están ideadas para su uso con determinadas funciones de los componentes DataExpress, pero se pueden utilizar con modelos de datos Swing genéricos.

Si desea obtener más información acerca de InternetBeans Express, consulte el [Capítulo 7, “InternetBeans Express”](#). También puede consultar el [Capítulo 19, “Tutorial: Creación de un servlet con InternetBeans Express”](#).

6

Desarrollo de Páginas JavaServer

El desarrollo web es una función de JBuilder Enterprise

La tecnología de Páginas JavaServer (JSP) permite a los programadores y diseñadores de web desarrollar rápidamente y mantener con facilidad el contenido de páginas web dinámicas y ricas en información, que aprovechan los sistemas empresariales existentes. La tecnología JSP, que forma parte de la familia Java, permite desarrollar con rapidez aplicaciones web que se ejecutan en cualquier sistema informático.

En teoría, la tecnología de Páginas JavaServer separa la interfaz de usuario de la generación de los contenidos, lo que permite cambiar el diseño general de la página sin alterar el contenido dinámico subyacente. En la práctica, se necesita una pequeña planificación y algunos estándar de codificación para asegurarse de que el HTML se diferencie claramente del código Java en la JSP, ya que ambos residen en el mismo archivo. Los diseñadores web que manejan la parte de HTML deberán tener una mínima idea de qué etiquetas se refieren al código Java incrustado, con el fin de evitar problemas a la hora de diseñar la IU.

La tecnología de Páginas JavaServer utiliza etiquetas estilo XML y scriptlets escritos en el lenguaje de programación Java, para encapsular el código que genera el contenido de la página. Además, el código de la aplicación puede encontrarse en recursos situados en el servidor (como la arquitectura de componentes JavaBeans) a los que accede la página mediante estas etiquetas y scriptlets. Devuelve cualquier etiqueta de formato (HTML o XML) directamente a la página de respuesta. Al separar la lógica de la página de su diseño y al aceptar un diseño reutilizable basado en componentes, la tecnología JSP agiliza y facilita más que nunca la creación de aplicaciones web.

La tecnología JSP es una extensión de la API para servlets Java. La tecnología JSP suministra esencialmente una forma simplificada de escribir servlets. Los servlets son módulos de servidor que se ejecutan en cualquier sistema informático, escritos al 100% en Java, que encajan directamente en el entorno de servidor web y se pueden utilizar para ampliar sus funciones con un mínimo de gastos y mantenimiento. A diferencia de otros lenguajes de guión, los servlets no implican consideraciones o modificaciones específicas con la plataforma. Juntos, la tecnología JSP y los servlets proporcionan una atractiva alternativa a otros tipos de guión/programación de web dinámica. La tecnología JSP y los servlets ofrecen independencia de plataforma, prestaciones mejoradas, separación entre lógica y presentación, facilidad de administración, capacidad de ampliación en la empresa y, lo más importante, facilidad de uso.

El sistema JSP se parece mucho a las ASP (páginas del servidor Active) de Microsoft. La principal diferencia entre las JSP y las ASP es que los objetos manejados en el primero de los sistemas son JavaBeans, que son independientes de plataforma. ASP maneja objetos COM, por lo que depende completamente de las plataformas Microsoft.

Todo lo que se necesita para una JSP es una página basada en la tecnología JSP. Las páginas JSP incluyen etiquetas, declaraciones y en ocasiones scriptlets propios de JSP, en combinación con contenidos estáticos (HTML o XML). Una página JSP tiene como extensión .jsp; esto indica al servidor web que el motor compatible con tecnología JSP procesará elementos en esta página. Una JSP opcionalmente también puede utilizar una o más JavaBeans en archivos .java independientes.

Cuando se compila una JSP mediante el motor de JSP en el servidor web, se convierte en un servlet. Como desarrollador, habitualmente no verá el código del servlet generado. Esto significa que cuando se compilan JSP en JBuilder IDE, se pueden ver mensajes de error que se refieren directamente al servlet generado y sólo indirectamente al código de la JSP. Recuerde que si tiene mensajes de error cuando compile su JSP, podrían referirse a líneas de código del servlet generado. Será más fácil identificar los problemas en su JSP si tiene una clara noción de cómo las JSP se traducen en servlets. Para conseguirlo, se necesita conocer la API de JSP.

Cuando JBuilder compila las JSP, traslada las ubicaciones del servlet generado de nuevo a las ubicaciones de la JSP originaria. Lo que significa que hacer clic en un error en el panel de estructura o en la vista de mensajes, le llevará directamente a la ubicación correspondiente de la JSP. Cuando el servidor web compila la JSP, no obstante, no realiza esta traslación. Cuando vea un seguimiento de la pila en el navegador o en Vista web, o en la salida del servidor web, el nombre de archivo y los números de línea no están traducidos.

El [Capítulo 18, “Tutorial: Creación de una JSP mediante el Asistente para JSP,”](#) muestra cómo crear una JSP utilizando el Asistente para JSP como punto de partida.

Si desea visitar otras páginas que contienen más información sobre la tecnología Páginas JavaServer, consulte [“Recursos adicionales de JSP”](#) en la página 6-14.

Etiquetas JSP

Una JSP incluye habitualmente numerosas etiquetas especializadas que contienen código Java o fragmentos de código Java. Esta es una lista de algunas de las etiquetas JSP más importantes:

Tabla 6.1 Etiquetas JSP habituales

Sintaxis de etiquetas	Descripción
<code><% fragmento de código %></code>	Etiqueta Scriptlet. Contiene un fragmento de código, que contiene una o más líneas de código que pueden aparecer normalmente dentro del cuerpo de un método en una aplicación Java. No se necesita declarar ningún método, ya que estos fragmentos de código se convierten en parte del método <code>service()</code> del servlet cuando se compila la JSP.
<code><%! declaración %></code>	Declaración de métodos o de variables. Cuando se declara un método en esta etiqueta, el método completo debe estar contenido en la etiqueta. Se compila en una declaración de método o variable en el servlet.
<code><%-- comentario --%></code>	Comentario. Este es un comentario de estilo de JSP que no se pasa al navegador cliente. (También se pueden usar comentarios HTML, pero estos pasan al navegador cliente.)
<code><%= expresión %></code>	Expresión. Contiene una expresión válida de Java. El resultado se presenta en ese punto de la página.
<code><%@ page [atributos] %></code>	Directiva page. Especifica atributos de la página JSP. Directivas como ésta y como la directiva taglib deben estar en las primeras líneas de la JSP. Uno de los atributos más comunes que se especifican en la directiva page es una sentencia de importación. Ejemplo: <code><%@ page import="com.borland.internetbeans.*" %></code>
<code><%@ taglib uri="vía de acceso a la biblioteca de etiquetas" prefix="prefijo de etiqueta" %></code>	Directiva taglib. Hace que la biblioteca de etiquetas esté disponible para su uso en la JSP, especificando la ubicación de la biblioteca y el prefijo que se debe utilizar en sus etiquetas asociadas. Directivas como ésta y como la directiva page deben estar en las primeras líneas de la JSP.

La especificación JSP incluye etiquetas estándares para la utilización y manipulación de beans. La etiqueta `useBean` crea una instancia de una clase de JavaBeans específica. Si la instancia ya existe, se recupera. Si no existe, se crea. Las etiquetas `setProperty` y `getProperty` permiten manipular propiedades del bean determinado. Éstas y otras etiquetas se describen más detalladamente en la especificación JSP y la guía del usuario, que se encuentra en <http://java.sun.com/products/jsp/techinfo.html>.

Es importante tener en cuenta que la mayoría del código Java contenido en las etiquetas de la JSP se convierte en parte del método `service()` del servlet cuando la JSP se compila en un servlet. Esto no incluye el código contenido en las etiquetas de declaración, que se convierten en declaraciones de métodos o de variables por derecho propio. Se llama al método `service()` cada vez que el cliente hace un GET o un POST.

Marcos de trabajo y bibliotecas de etiquetas JSP

Los marcos de trabajo y las bibliotecas de etiquetas JSP facilitan el desarrollo de JSP al proporcionar funcionalidad y etiquetas JSP adicionales que no se encuentran disponibles en la API de JSP.

Una biblioteca de etiquetas es un conjunto de código Java al que se puede llamar desde las marcas de un archivo JSP. Un marco de trabajo define una disciplina para el desarrollo de aplicaciones. Si acepta el paradigma de desarrollo de un marco de trabajo, a cambio recibe ayuda en el desarrollo y una aplicación más limpia y estandarizada.

Una biblioteca de etiquetas JSP consiste en un archivo descriptor de bibliotecas de etiquetas (TLD) y clases manejadoras de etiquetas que implementan o llaman a las funciones que necesitan las etiquetas. Una biblioteca de etiquetas JSP también puede formar parte de un marco de trabajo. Un marco de trabajo puede contener una o más bibliotecas de etiquetas. También puede haber otros tipos de marcos de trabajo, algunos de los cuales no están diseñados para utilizar con JSP.

Existen tres marcos de trabajo más conocidos que se suministran e integran con JBuilder y que resultan útiles para el desarrollo de JSP. Se describen del siguiente modo.

- Struts — El marco de trabajo Struts ofrece un concepto de diseño de software Modelo 2 o Controlador Modelo-Vista.
- JSTL (Biblioteca de etiquetas estándar para páginas JavaServer) — JSTL ofrece un modo estándar de llevar a cabo las tareas más comunes de codificación mediante sencillas etiquetas. Puede encontrar más información en <http://java.sun.com/products/jsp/jstl>.

- InternetBeans Express — InternetBeans Express es una biblioteca de componentes Borland que facilita la creación de JSP y servlets enlazados a datos.

Consulte

- “[Las bibliotecas de etiquetas JSP y los marcos de trabajo en JBuilder](#)” en la página 6-5
- [Capítulo 8, “El marco de trabajo Struts en JBuilder”](#)
- [Capítulo 7, “InternetBeans Express”](#)

JSP en JBuilder

JBuilder ofrece un completo sistema de desarrollo para JSP. A continuación se incluye una lista de las funciones de desarrollo de JSP de JBuilder:

- Un Asistente para JSP, para crear una JSP que pueda admitir uno o más marcos de trabajo
- Compatibilidad con los marcos de trabajo y las bibliotecas de etiquetas JSP
- CodeInsight, para completar las etiquetas específicas de JSP
- Depuración dentro del archivo JSP
- Comprobación y ejecución de JSP en cualquier motor de servlets que admita JSP, desde el entorno de desarrollo de JBuilder

Nota No puede abrir una JSP en el diseñador de JBuilder, porque no es un archivo .java.

Las bibliotecas de etiquetas JSP y los marcos de trabajo en JBuilder

JBuilder admite la utilización de bibliotecas de etiquetas JSP y marcos de trabajo de las siguientes maneras:

- El Asistente para aplicaciones web permite seleccionar uno o más marcos de trabajo con el fin de utilizarlo en la WebApp. El asistente añade las bibliotecas del marco de trabajo seleccionado a la webapp, añade las bibliotecas de etiquetas necesarias, añade las entradas requeridas al archivo `web.xml` y copia los JAR del marco de trabajo a `WEB-INF/lib`. También es posible añadir los nodos dependientes al panel del proyecto e, incluso, añadir archivos de configuración si se necesitan. Todo esto contribuye a garantizar que el marco de trabajo se distribuya

correctamente y que se pueda utilizar al ejecutar su WebApp en el IDE de JBuilder.

- El Asistente para JSP permite seleccionar una o más bibliotecas de etiquetas con el fin de utilizarlas en la JSP que genera el asistente. El asistente añade las directivas `page` y `taglib` adecuadas a la JSP generada.
- El cuadro de diálogo Configurar bibliotecas permite gestionar los marcos de trabajo y las bibliotecas de etiquetas definidas por el usuario.

Consulte

- “[Creación de una WebApp con el Asistente para aplicaciones web](#)” en la página 3-3
- “[Creación de una JSP mediante el Asistente para JSP](#)” en la página 6-11
- “[Utilización del cuadro de diálogo Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario](#)” en la página 6-6

Utilización del cuadro de diálogo Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario

La pestaña Marco de trabajo del cuadro de diálogo Configurar bibliotecas (Herramientas | Configurar bibliotecas) permite configurar las bibliotecas de etiquetas JSP definidas por el usuario con el fin de utilizarlos con las JSP en JBuilder. Un marco de trabajo consiste en una biblioteca de etiquetas y elementos específicos de taglib contenidos en un archivo JAR.

Los siguientes marcos de trabajo se suministran con JBuilder, y no es necesario que se configuren:

- Struts
- JSTL
- InternetBeans Express
- Cocoon

Estos marcos de trabajo se encuentran disponibles directamente en los asistentes adecuados. Por ejemplo, si selecciona el Asistente para JSP, los marcos de trabajo InternetBeans, JSTL y Struts están disponibles en la lista Bibliotecas de etiqueta, de la ficha Introducir detalles del archivo JSP del asistente.

Si ha configurado correctamente la biblioteca de etiquetas JSP definida por el usuario mediante el cuadro de diálogo Configurar bibliotecas, la biblioteca de etiquetas se encuentra disponible en los Asistentes para aplicaciones web y para JSP.

Cómo añadir una biblioteca de etiquetas JSP definida por el usuario

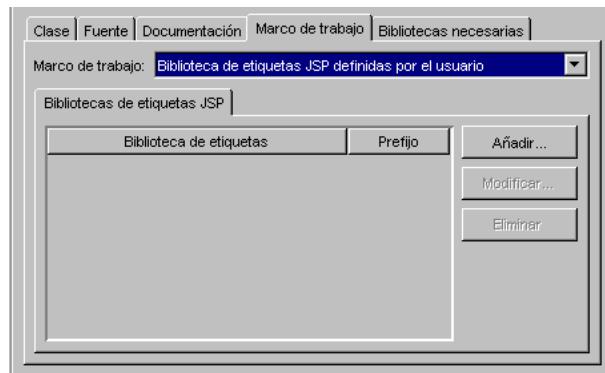
Al añadir una nueva biblioteca de etiquetas JSP definida por el usuario, esta biblioteca pasa a estar disponible en los Asistentes para JSP y para aplicaciones web. Para añadir una biblioteca de etiquetas:

- 1 Seleccione Herramientas | Configurar bibliotecas y se abrirá el cuadro de diálogo homónimo.
- 2 Pulse el botón Nuevo, abajo a la izquierda, para abrir el Asistente para bibliotecas.



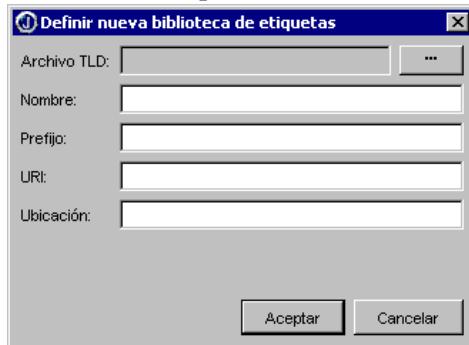
- 3 Escriba el nombre de la biblioteca de etiquetas en el campo Nombre. Elija su ubicación en la lista desplegable Ubicación.
- 4 Pulse Añadir para añadir los archivos JAR que contienen la biblioteca de etiquetas. Busque cada archivo JAR, selecciónelo y pulse Aceptar.
- 5 Pulse Aceptar con el fin de cerrar el Asistente para bibliotecas. Se vuelve a abrir el cuadro de diálogo Configurar bibliotecas. La nueva biblioteca aparece en la carpeta asignada, a la izquierda del cuadro de diálogo.
- 6 Pulse la pestaña Marco de trabajo del cuadro de diálogo Configurar bibliotecas.

- 7** Seleccione Biblioteca de etiquetas JSP definidas por el usuario, en la lista desplegable Marco de trabajo.



Aparece la pestaña Bibliotecas de etiquetas JSP.

- 8** Pulse el botón Añadir para mostrar el cuadro de diálogo Definir nueva biblioteca de etiquetas.



- 9** Pulse el botón de puntos suspensivos, junto al campo Archivo TLD, con el fin de buscar el archivo TLD que deseé definir en su biblioteca de etiquetas. El TLD puede ser un archivo externo distribuido con el JAR, dentro del JAR o ambos. El archivo TLD debe tener una extensión .TLD. Es necesario que las clases manejadoras de etiquetas se encuentren en un archivo JAR. Una vez elegido un archivo TLD válido, los campos Nombre, Prefijo, URI y Ubicación se llenan automáticamente a partir de la definición TLD.
- 10** Pulse Aceptar y se cerrará el cuadro de diálogo Definir nueva biblioteca de etiquetas.
- 11** Pulse Aceptar otra vez, con el fin de cerrar el cuadro de diálogo Configurar bibliotecas.

Nota Puede añadir todos los TLD que deseé a un marco de trabajo.

Modificación de las propiedades de una biblioteca de etiquetas en un marco de trabajo

Una vez que se haya definido el marco de trabajo, la tarea de edición más normal es cambiar el prefijo de una biblioteca de etiquetas. Tanto el prefijo como el URI se utilizan en la etiqueta `taglib` generada por el Asistente para JSP. Puede utilizar cualquier prefijo, siempre y cuando no contenga espacios o caracteres especiales. Es recomendable evitar utilizar el mismo prefijo para dos bibliotecas de etiquetas distintas. Si solamente desea modificar el prefijo, puede cambiarlo en la tabla sin necesidad de abrir el cuadro de diálogo Modificar biblioteca de etiquetas. No olvide pulsar `Intro` para guardar las modificaciones.

Para modificar las propiedades de una biblioteca de etiquetas en un marco de trabajo:

- 1** Seleccione Herramientas | Configurar bibliotecas y se abrirá el cuadro de diálogo homónimo.
- 2** A la izquierda del cuadro, elija el nombre de la biblioteca del marco de trabajo que contenga la biblioteca de etiquetas cuyas propiedades deseé modificar.
- 3** Pulse sobre la pestaña Marco de trabajo.
- 4** En la pestaña Bibliotecas de etiquetas JSP, seleccione la definición de biblioteca de etiquetas que deseé modificar.

Nota Si no aparece la pestaña Biblioteca de etiquetas JSP, significa que el marco de trabajo seleccionado no contiene bibliotecas de etiquetas JSP o que no se pueden modificar.

- 5** Pulse el botón Modificar. Aparece el cuadro de diálogo Modificar biblioteca de etiquetas.
- 6** Cambie el Archivo TLD, Nombre, Prefijo, URI o Ubicación y pulse Aceptar.
- 7** Pulse de nuevo Aceptar para guardar los cambios y cerrar el cuadro de diálogo Configurar bibliotecas.

Nota Puede añadir todos los TLD que deseé a un marco de trabajo.

Eliminación de una biblioteca de etiquetas de un marco de trabajo

Un buen motivo para eliminar una biblioteca de etiquetas de un marco de trabajo sería si nunca utilizará esa biblioteca. Para eliminar una biblioteca de etiquetas de un marco de trabajo:

- 1** Seleccione Herramientas | Configurar bibliotecas y se abrirá el cuadro de diálogo homónimo.
- 2** A la izquierda del cuadro, elija la biblioteca del marco de trabajo que contenga la biblioteca de etiquetas que deseé eliminar.

- 3** Pulse sobre la pestaña Marco de trabajo.
 - 4** En la pestaña Bibliotecas de etiquetas JSP, seleccione la definición de biblioteca de etiquetas que deseé eliminar.
- Nota** Si no aparece la pestaña Biblioteca de etiquetas JSP, significa que el marco de trabajo seleccionado no contiene bibliotecas de etiquetas JSP o que no se pueden modificar.
- 5** Pulse Eliminar.
 - 6** Si desea eliminar toda la biblioteca, elija la biblioteca a la izquierda del cuadro y pulse Borrar en la parte inferior izquierda.
 - 7** Haga clic en Aceptar para cerrar el cuadro de diálogo Configurar bibliotecas.

Desarrollo de las JSP

El Asistente para JSP es un punto de partida opcional para desarrollar una JSP. El editor de JBuilder resalta la sintaxis de las JSP. JBuilder también proporciona CodeInsight y ErrorInsight para el código Java incrustado en una página JSP.

El panel de estructura en el IDE de JBuilder muestra la estructura de las etiquetas dentro de la JSP, además de cualquier error de HTML y de Java en el código. Estos errores son muy útiles, por ejemplo, a menudo le recuerdan que tiene que acabar una etiqueta que estaba incompleta.

El Asistente para JSP

JBuilder proporciona un Asistente para JSP. Este es un punto de partida lógico cuando se desarrolla una JSP. Para acceder a este asistente, pulse en la pestaña Web de la galería de objetos. Debe haber seleccionado un servidor para el proyecto para que el Asistente para JSP se encuentre disponible.

El Asistente para JSP genera el esqueleto de una JSP. Crea los archivos básicos de su JSP, cuyos detalles se llenan más tarde.

El asistente para JSP crea los siguientes archivos:

- Una JSP. Una JSP incluye etiquetas específicas de la tecnología JSP, declaraciones y posiblemente scriptlets, junto a otras etiquetas estáticas (HTML o XML). El archivo JSP tiene la extensión .jsp. Este archivo se crea siempre con el asistente para JSP.
- Un archivo Java. Este archivo es optativo y lo puede crear el asistente activando “Generar Bean de ejemplo”. El archivo Java contiene uno o más JavaBeans usados por el archivo JSP.

- Una página JSP de errores. El asistente para Páginas JavaServer puede crear optativamente una página de error que muestra los errores de tiempo de ejecución, de una manera útil y ordenada.

En el Asistente para JSP, se puede especificar a qué WebApp pertenece su JSP, si configura la JSP para utilizar una o varias de las bibliotecas de etiquetas disponibles, y si su JSP utiliza alguna JavaBean. Puede también crear una configuración de ejecución para su JSP.

Consulte

- Asistente para JSP en la ayuda en pantalla
- [Capítulo 18, "Tutorial: Creación de una JSP mediante el Asistente para JSP"](#)

Creación de una JSP mediante el Asistente para JSP

El Asistente para JSP resulta valioso como punto de partida opcional para desarrollar una JSP. Si todavía no dispone de una webapp en el proyecto, debe crearla con ayuda del Asistente para aplicaciones web antes de usar el Asistente para JSP. Creación de una JSP mediante el Asistente para JSP:

- 1 Seleccione la galería de objetos seleccionando Archivo | Nuevo. Haga clic en la pestaña Web. (Antes, el proyecto debe tener un servidor activado.)
- 2 Seleccione Página JavaServer. Pulse Aceptar.
- 3 Seleccione una WebApp para que contenga la nueva JSP. Si sólo tiene una WebApp en su proyecto, se desactiva la lista desplegable WebApp.
- 4 Defina si desea crear un bean de ejemplo y una página de errores.
 - Si está marcado Generar bean de ejemplo, el asistente crea un JavaBean de ejemplo que puede utilizar la JSP. Si se marca esta opción, se añade un paso al asistente.
 - Si se marca Generar página de error, el asistente crea una página de error que muestra los errores de ejecución de un modo útil y ordenado. Si se marca esta opción, se añade un paso al asistente.

Pulse Siguiente para pasar al siguiente paso. También puede pulsar Finalizar para evitar los pasos restantes, cerrar el asistente y crear la JSP.

- 5 Seleccione el color de fondo para la JSP.
- 6 Marque Generar formulario de envío si desea que el Asistente para JSP genere una etiqueta HTML `<form>` en el archivo JSP.
- 7 Seleccione las bibliotecas de etiquetas que deseé que utilice la JSP, marcando las casillas adecuadas en la lista Bibliotecas de etiqueta. El asistente añade las directivas apropiadas `page` y `taglib` para las

bibliotecas de etiquetas seleccionadas en la JSP creada. Las bibliotecas de etiquetas para los marcos de trabajo InternetBeans Express, Struts y JSTL están disponibles automáticamente. También están disponibles todas las bibliotecas de etiquetas definidas por el usuario que haya añadido en el cuadro de diálogo Configurar bibliotecas.

Cuando termine de seleccionar bibliotecas de etiquetas, pulse Siguiente para dirigirse al siguiente paso. También puede pulsar Finalizar para evitar los pasos restantes, cerrar el asistente y crear la JSP.

- 8 Defina el Paquete y Nombre de clase para el bean de ejemplo. Marque Generar comentarios de cabecera, si desea crearlos en el bean de ejemplo. Si no ha marcado Generar bean de ejemplo en el primer paso del asistente, este paso no aparece.

Pulse Siguiente para pasar al siguiente paso. También puede pulsar Finalizar para evitar los pasos restantes, cerrar el asistente y crear la JSP.

- 9 Utilice los botones Añadir Bean y Eliminar Bean con el fin de determinar los JavaBeans que desea que utilice la JSP. El asistente genera una etiqueta `jsp:useBean` para cada JavaBean que especifique.

Cuando termine de especificar los JavaBeans, pulse Siguiente para dirigirse al siguiente paso. También puede pulsar Finalizar para evitar los pasos restantes, cerrar el asistente y crear la JSP. Si no ha marcado Generar bean de ejemplo, ni Generar página de error, en el primer paso del asistente, no hay más pasos y, por lo tanto, el botón Siguiente está desactivado.

- 10 Defina el Nombre y el color de fondo para la página de error. Si no ha marcado Generar página de error en el primer paso del asistente, este paso no aparece.

Pulse Siguiente para pasar al siguiente paso. También puede pulsar Finalizar para evitar los pasos restantes, cerrar el asistente y crear la JSP.

- 11 Marque Crear una configuración de ejecución si desea crear una de tipo Servidor. Si esta opción está marcada, especifique un Nombre. También puede seleccionar una configuración base para copiar, si ya existe otra configuración de ejecución de tipo Servidor en su proyecto.

Pulse Finalizar para cerrar el asistente y crear la JSP.

Consulte

- Asistente para JSP en la ayuda en pantalla
- [Capítulo 18, “Tutorial: Creación de una JSP mediante el Asistente para JSP”](#)

- “Utilización del cuadro de diálogo Configurar bibliotecas para gestionar los marcos de trabajo definidos por el usuario” en la página 6-6
- “Las bibliotecas de etiquetas JSP y los marcos de trabajo en JBuilder” en la página 6-5
- Capítulo 8, “El marco de trabajo Struts en JBuilder”
- Capítulo 7, “InternetBeans Express”

Compilación de las JSP

Las JSP son una extensión de la Servlet API y se compilan con los servlets antes de utilizarse. Esto requiere que el proceso de compilación convierta los nombres de los archivos JSP y los números de línea en sus equivalentes Java. En JBuilder, las JSP pueden compilarse durante la generación. Si desea activar esta función para todas las JSP del proyecto:

- 1 Seleccione Proyecto | Propiedades de proyecto.
- 2 Seleccione la ficha Generar.
- 3 Abra la ficha General de la ficha Generar.
- 4 Seleccione Comprobar errores en JSP durante la generación
- 5 Pulse Aceptar.

Puede configurar esta propiedad por cada archivo JSP de su proyecto mediante el menú contextual de las JSP en el panel del proyecto. Esto le permite excluir determinados archivos JSP de la compilación. Por ejemplo, las JSP que están pensadas para ser incluidas en otras JSP probablemente no se compilarán con éxito por sí mismas, por lo que deberían excluirse esos archivos.

Para compilar su proyecto, incluidas las JSP configuradas para compilarse en la generación, seleccione Proyecto | Ejecutar Make del proyecto o Proyecto | Generar el proyecto.

Si desea obtener más información acerca de la compilación de las JSP, consulte “[Compilación de servlets o JSP](#)” en la página 10-15.

Ejecución web de las JSP

Para ejecutar una JSP en el IDE de JBuilder, haga clic sobre ella con el botón derecho en el panel del proyecto y seleccione Ejecutar web en el menú contextual. Ejecutar web está habilitado si existe una configuración de ejecución adecuada y el servidor actualmente seleccionado admite el servicio JSP/Servlet.

Si desea obtener más información acerca de la ejecución de las JSP, consulte “[Ejecución web de un servlet o una JSP](#)” en la página 10-16.

Depuración web de las JSP

Para depurar una JSP en el IDE de JBuilder, haga clic sobre ella con el botón derecho en el panel del proyecto y seleccione Depurar web en el menú contextual. Depurar web está habilitado si existe una configuración de ejecución adecuada y el servidor actualmente seleccionado admite el servicio JSP/Servlet.

Si desea obtener información acerca de la depuración de las JSP, consulte “[Depuración web del servlet o de la JSP](#)” en la página 10-19.

Distribución de las JSP

Si desea obtener más información acerca de la distribución de las JSP, consulte el [Capítulo 11, “Distribución de aplicaciones web”](#).

Recursos adicionales de JSP

Si desea obtener ayuda para desarrollar páginas JSP en JBuilder, visite el grupo de noticias de Borland, en borland.public.jbuilder.servlets-jsp. Se puede acceder a todos los grupos de noticias sobre JBuilder desde la sede web de Borland, en <http://www.borland.com/newsgroups/#jbuilder>.

Si desea más información sobre el desarrollo de Páginas JavaServer, visite las siguientes sedes web. Las direcciones y vínculos facilitados son válidos a la fecha de impresión de este manual. Las sedes web no pertenecen a Borland, por lo que no se puede garantizar su contenido ni su continuidad.

- Es posible obtener una descripción de la sintaxis JSP en formato HTML en <http://java.sun.com/products/jsp/tags/tags.html>.
- En <http://java.sun.com/products/jsp/technical.html> puede encontrar una tarjeta de sintaxis de JSP en formato PDF, que se puede abrir con Adobe Acrobat Reader. Esta página contiene también otros recursos técnicos sobre la tecnología JSP.
- Consulte las FAQ sobre las JSP en java.sun.com. Dirija el navegador a <http://java.sun.com/products/jsp/faq.html>.
- La actual especificación JSP se puede leer en el sitio web java.sun.com, en <http://java.sun.com/products/jsp/download.html>.

- GNUJSP es una implementación gratuita de JavaServer Pages de Sun. Si desea más información sobre el compilador GNUJSP, visite <http://klomp.org/gnujsp>.
- Puede encontrar un servidor de lista sobre las JSP en JSP-INTEREST@JAVA.SUN.COM. Para poder suscribirse a la lista de correo, envíe un mensaje a listserv@java.sun.com con el siguiente contenido:

subscribe jsp-interest Nombre completo
o busque información en sitio web java.sun.com.

InternetBeans Express

El desarrollo web es una
función de JBuilder
Enterprise

La tecnología InternetBeans Express se integra en la tecnología servlet y JSP para ofrecer un valor añadido a sus aplicaciones y simplificar las tareas de desarrollo de servlet y JSP. InternetBeans Express es un conjunto de componentes y extensiones de etiquetas de páginas JavaServer (JSP) utilizados para la generación de presentaciones y la interacción con aplicaciones web. Toma páginas estáticas de plantilla, inserta contenido dinámico desde un modelo dinámico de datos y lo presenta al cliente; a continuación, escribe en el modelo los datos que envía el cliente. Esto facilita la creación de servlets y JSP enlazados a datos. Por ejemplo, se pueden utilizar componentes InternetBeans Express con el fin de crear un servlet que proporcione un formulario de registro para usuarios nuevos, para entrar en una página web o una página JSP que muestre el resultado de una búsqueda en una tabla.

InternetBeans Express contiene soporte incorporado para DataExpress DataSets y DataModules. También puede utilizarse con modelos genéricos de datos y EJB. Las clases e interfaces se pueden dividir en tres categorías:

- Los InternetBeans son componentes que se encargan directamente de la generación dinámica de marcas y la gestión de semántica de solicitud y respuesta HTTP.
- Los manejadores de etiquetas JSP, que llaman internamente a InternetBeans, y la infraestructura en la que se basan. Estos son utilizados por las extensiones de etiquetas de JSP en la biblioteca de etiquetas InternetBeans.
- Aceptación de modelo de datos.

También existe una biblioteca de etiquetas JSP que contiene extensiones de etiquetas JSP cuyo fin es admitir InternetBeans en una JSP.

Descripción general de las clases de InternetBeans Express

InternetBeans Express incluye las siguientes clases:

Tabla 7.1 Clases de InternetBeans Express

Tipo de componente	Descripción
	IxPageProducer Lee y analiza archivos HTML estáticos, de forma que más adelante pueda regenerarlos insertando contenido dinámico de otros componentes. La mayoría de los servlets utiliza un <code>IxPageProducer</code> , que les permite generar la totalidad de la respuesta de una página web prediseñada, insertando datos dinámicos como cuadros de texto o controles en un formulario.
	IxControl Control genérico que determina, en la ejecución, el tipo de control HTML al que sustituye y emula. Este componente puede utilizarse en lugar de cualquier otro InternetBean específico para el control. En un servlet, este componente debe utilizarse con un <code>IxPageProducer</code> . No se necesita un <code>IxPageProducer</code> para utilizar este componente en una JSP.
	IxTable Genera tablas HTML a partir de conjuntos de datos y modelos de tablas.
	IxImage Representa una imagen que simplemente se muestra o se utiliza como un enlace. Si se quiere utilizar una imagen para enviar un formulario, utilice <code>IxImageButton</code> .
	IxLink Representa un enlace. Si se necesita reescribir una URL, <code>IxLink</code> puede manejarla automáticamente efectuando una llamada interna al método <code>HttpServletResponse.encodeURL()</code> .
	IxSpan Sustituye el contenido de sólo lectura por un atributo ID, probablemente <code></code> , pero podría ser cualquier otro tipo de elemento. Para los controles de un formulario, utilice <code>IxControl</code> .
	IxCheckBox Representa una casilla de selección. XHTML: <code><input type="checkbox" /></code>
	IxComboBox Representa un cuadro combinado. XHTML: <code><select size="1" /></code>
	IxHidden Representa un valor oculto. XHTML: <code><input type="hidden" /></code>
	IxImageButton Representa una imagen que, cuando se pulsa, envía el formulario. No debe confundirse con una imagen que simplemente se muestra o se utiliza como un enlace; para eso, utilice <code>IxImage</code> . XHTML: <code><input type="image" /></code> Si la imagen que coincide con este componente es la imagen que envió el formulario, se activará el suceso <code>submitPerformed()</code> de <code>IxImageButton</code> .

Tabla 7.1 Clases de InternetBeans Express (continuación)

Tipo de componente	Descripción
IxListBox	Representa un cuadro de lista. XHTML: <select size="3" />
IxPassword	Representa un campo de contraseña. XHTML: <input type="password" />
IxPushButton	Representa un botón en el equipo del cliente. XHTML: <input type="button" />
IxRadioButton	Representa un botón de radio. XHTML: <input type="radio" />
IxSubmitButton	Representa un botón de envío en un formulario. XHTML: <input type="submit" /> Si el botón que coincide con este componente fue el botón que envió el formulario, se activará el evento submitPerformed() de IxSubmitButton.
IxTextArea	Representa un área de texto. XHTML: <textarea>
IxTextField	Representa un campo de introducción de datos. XHTML: <input type="text" />

Utilización de InternetBeans Express con servlets

Los componentes InternetBeans Express simplifican la presentación de datos dinámicos en páginas web y el envío de datos de páginas web a bases de datos u otros modelos de datos.

Visualización de páginas web dinámicas con servlets mediante InternetBeans Express

La mayoría de los servlets utilizará un componente `IxPageProducer`. Esto les permite generar la totalidad de la respuesta de una página web prediseñada, insertando en ella un formulario de datos dinámicos, como cuadros de texto y controles. Este método presenta algunas ventajas:

- Ya conoce el aspecto de la respuesta. La página puede contener datos de prueba, que serán sustituidos.
- El aspecto se puede cambiar cambiando la página, sin necesidad de tocar el código.

Por ejemplo, cuando se utiliza InternetBeans Express con un servlet, se puede abrir el servlet en el diseñador. Una Database y un QueryDataSet de la pestaña DataExpress de la paleta pueden suministrar los datos para el servlet. Se puede añadir un IxPageProducer desde la pestaña InternetBeans de la paleta. Asigne el nombre de archivo de la página Web prediseñada a la propiedad htmlFile del IxPageProducer. Cuando se ejecuta el servlet, el IxPageProducer llama al HtmlParser interno con el fin de localizar todos los controles HTML sustituibles.

La forma más simple de sustituir controles HTML con controles que contengan datos de generación dinámica es utilizar IxControls. Se debe añadir un IxControl por cada control HTML en la página de plantilla que contendrá los datos. Asigne las propiedades pageProducer de IxControl a IxPageProducer. Asigne a la propiedad controlName de IxControl el valor que coincide con el atributo name del control HTML apropiado. Con la asignación de valores a las propiedades dataSet y columnName de IxControl se completa la asociación a datos.

El método IxPageProducer.servletGet() es uno de los que se llama normalmente para generar la página que se va a mostrar. Este método debe ser llamado dentro del método doGet() del servlet. El cuerpo del método doGet() de un servlet puede ser tan sencillo como el siguiente:

```
ixPageProducer1.servletGet(this, request, response);
```

Esta única llamada asigna el tipo de contenido de la respuesta y convierte la página dinámica en el flujo de salida de la respuesta. Advierta que el tipo de contenido por defecto de la respuesta es HTML.

Situados en el interior, el método IxPageProducer.render() genera la versión dinámica de la página, sustituyendo los controles que tienen un IxControl asignado mediante el procedimiento de ordenar la representación del componente, lo que genera un código HTML equivalente con los valores de datos extraídos de la fila actual del conjunto de datos. Puede llamar al método render(), pero resulta más sencillo llamar al servleGet().

Algunas situaciones en las que no se debería utilizar una IxPageProducer en un servlet son:

- Cuando no se necesiten las características de administrar y enviar sesiones de bases de datos de la IxPageProducer y solamente se quiera el motor de plantillas de página, se puede utilizar PageProducer.
- Cuando se están utilizando componentes individuales específicos para convertir en HTML. Por ejemplo, se puede crear una IxComboBox que contenga una lista de países y utilizarla en un servlet con HTML codificado a mano.

Recuerde que cuando utiliza InternetBeans en un servlet, habitualmente deberá utilizar un IxPageProducer. Cuando se utiliza IxControl, se debe utilizar un IxPageProducer.

Envío de datos con servlets mediante InternetBeans Express

El procesamiento de una HTTP POST es sencillo mediante el método `IxPageProducer.servletPost()`:

```
ixPageProducer1.servletPost(this, request, response);
```

Debe efectuarse una llamada a este método desde el método `doPost()` del servlet, junto con cualquier otro código que deba ejecutarse durante la operación de envío.

En el momento del diseño, se debe añadir un `IxSubmitButton` por cada botón de envío del formulario. Añada un monitor `submitPerformed()` para cada `IxSubmitButtons`. El monitor deberá llamar al código que se tiene que ejecutar cuando se presiona el botón. Por ejemplo, un botón Siguiente debe ejecutar `dataSet.next()` y un botón Anterior debe ejecutar `dataSet.prior()`.

En el momento de la ejecución, cuando se llama al método `servletPost()`, escribe los nuevos valores del envío en los componentes de InternetBeans correspondientes y transmite esos valores desde el cliente al servidor. A continuación provoca el suceso `submitPerformed` adecuado para el botón que ha enviado el formulario. Para enviar y guardar realmente los cambios en el conjunto de datos subyacente, debe llamar a los métodos `post()` y `saveChanges()` del conjunto de datos dentro del método `submitPerformed()`. Ahora, el método `doPost()` del servlet puede llamar a `doGet()` o directamente a `IxPageProducer.servletGet()`, para representar la página nueva.

Análisis de páginas

A diferencia del analizador de XML, que es estricto, el de HTML es flexible. En concreto, los elementos de HTML (etiquetas) y los nombres de atributos no distinguen entre mayúsculas y minúsculas. Sin embargo, XHTML distingue entre mayúsculas y minúsculas, y los nombres estándar están en minúscula por definición.

Para acelerar el proceso, los nombres de elementos y atributos HTML se convierten a minúscula, la norma de XHTML para almacenamiento. Al buscar atributos, se debe utilizar la minúscula.

Cuando se emparejan los componentes InternetBeans Express con los controles HTML de la página, las propiedades definidas en el componente InternetBeans Express tienen preferencia. Cuando se asignan las propiedades en el diseñador, debe pensar si quiere redefinir realmente un atributo de HTML determinado, asignando su propiedad correspondiente en el componente. Por ejemplo, si la página web contiene una textarea de

un tamaño determinado, no es conveniente redefinirlo cuando se genere automáticamente el control.

Generación de tablas

La presentación de datos en una tabla con un formato determinado es una tarea muy habitual y compleja. Por ejemplo, las filas pueden tener determinadas agrupaciones de celdas y colores alternos.

El diseñador de páginas Web sólo necesita proporcionar suficientes filas de ejemplo para presentar el aspecto de la tabla (si hay colores alternos, son dos filas). Cuando el componente `IxTable` genera la tabla real, este aspecto se repite automáticamente.

Los visualizadores de celdas se pueden definir por clases, o asignar a cada columna su propio `IxTableCellRenderer` con el fin de permitir la personalización del contenido; por ejemplo, los valores negativos se pueden presentar en rojo (preferiblemente mediante la definición de hojas de estilo en cascada (CSS) adecuadas en lugar de cambiar el color desde el código).

Si desea leer un tutorial sobre el uso de InternetBeans en un servlet, consulte el [Capítulo 19, “Tutorial: Creación de un servlet con InternetBeans Express”](#).

Utilización de InternetBeans Express con JSP

La clave para utilizar InternetBeans Express con JSP está en la biblioteca de etiquetas de InternetBeans Express, definida en el archivo `internetbeans.tld`. Esta biblioteca contiene un conjunto de etiquetas InternetBeans que puede incluir en su archivo JSP siempre que desee utilizar un componente InternetBeans. Estas etiquetas requieren muy poca codificación, pero cuando se procesa la JSP dentro de un servlet, dan como resultado la introducción en el código de unos componentes InternetBeans perfectos.

Para usar InternetBeans Express en una JSP, siempre se debe tener una línea importante de código al principio de la JSP. Es una directiva `taglib`, que indica que las etiquetas de la biblioteca de etiquetas de InternetBeans Express se utilizarán en la JSP y especifica un prefijo para estas etiquetas. La directiva `taglib` para usar la biblioteca de etiquetas de InternetBeans tiene este aspecto:

```
<%@ taglib uri="/internetbeans.tld" prefix="ix" %>
```

Si quiere instanciar clases en sus scriptlets y no quiere escribir el nombre completo de la clase, se pueden importar archivos o paquetes a la JSP mediante una directiva `page`. Esta directiva `page` puede especificar que el

paquete com.borland.internetbeans debe importarse a la JSP. La directiva page debe tener este aspecto:

```
<%@ page import="com.borland.internetbeans.* ,com.borland.dx.dataset.* ,  
com.borland.dx.sql.dataset.*" %>
```

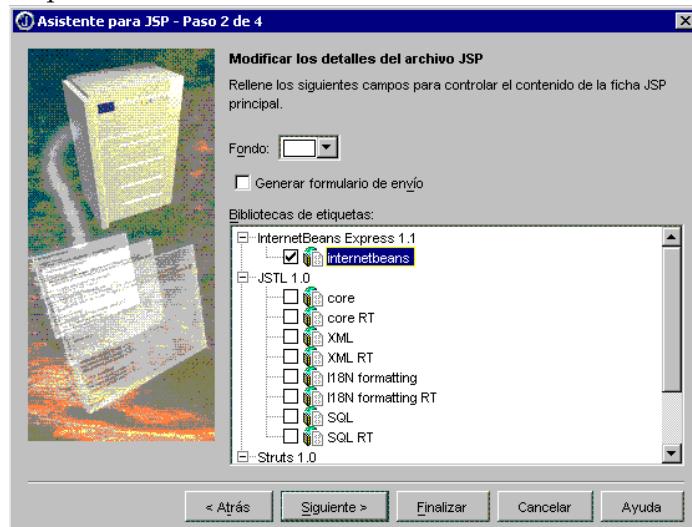
Recuerde que las directivas tales como la taglib y la page deben ser siempre las primeras líneas de su JSP.

El Asistente para JSP de JBuilder inserta una directiva taglib y una directiva page si se selecciona la biblioteca de etiquetas internetbeans en el paso Modificar los detalles del archivo JSP del asistente. El Asistente para JSP también completa el resto de pasos necesarios para configurar la JSP para que utilice InternetBeans Express. Estos pasos son:

- 1** Añade la biblioteca InternetBeans Express a su proyecto.
- 2** Configura las dependencias como Incluir todas, en su WebApp, para las bibliotecas InternetBeans Express, dbSwing y DataExpress. Esto quiere decir que los archivos jar necesarios se copian al directorio WEB-INF/lib de la WebApp cuando se compila el proyecto.
- 3** Añade una etiqueta de mapeo de bibliotecas entre internetbeans.tld y WEB-INF/lib/internetbeans.jar al archivo web.xml.

Necesita realizar estos pasos si está configurando su JSP para usar InternetBeans Express sin utilizar el Asistente para JSP.

A continuación puede ver el Asistente para JSP con la biblioteca de etiquetas internetbeans seleccionada:



Este es un ejemplo del aspecto que tiene una etiqueta de InternetBeans cuando se utiliza en su JSP:

```
<ix:database id="database1"
  driver="com.borland.datastore.jdbc.DataStoreDriver"
  url="jdbc:borland:dslocal...\\guestbook\\guestbook.jds"
  username="user">
</ix:database>
```

Este ejemplo utiliza la etiqueta `database`. Si realmente estuviera utilizando la etiqueta `database` en una JSP, en la mayoría de los casos será conveniente anidar otras etiquetas dentro de esta etiqueta, como la etiqueta `query`. Esto no es necesario, pero hace que el código de la JSP sea más legible.

Nota El prefijo `ix` puede ser cualquier texto. Todo depende del prefijo que se haya indicado en la directiva `taglib`. El Asistente para JSP utiliza el prefijo `ix` para la biblioteca de etiquetas `internetbeans`.

Para leer un tutorial sobre este tema, consulte el [Capítulo 20, “Tutorial: Creación de una página JSP con InternetBeans Express”](#).

Tabla de etiquetas de InternetBeans

Las etiquetas incluidas en la biblioteca de etiquetas de InternetBeans Express se describen en la tabla siguiente. Los atributos en negrita son obligatorios.

Tabla 7.2 Etiquetas InternetBeans Express

Nombre de la etiqueta	Descripción	Atributos
database	Define una base de datos DataExpress	<ul style="list-style-type: none"> • id—texto usado para identificar esta base de datos • driver—propiedad controlador de la base de datos • url—propiedad url de la base de datos • username — el nombre de usuario para la base de datos • password — la contraseña para la base de datos
query	Define un QueryDataSet de DataExpress	<ul style="list-style-type: none"> • id—texto usado para identificar esta consulta • database—identifica la base de datos a la que pertenece esta consulta. Esta no es necesaria ya que se simplifica si la etiqueta <code>query</code> está anidada dentro de la etiqueta <code>database</code>. Si la etiqueta <code>query</code> no está anidada dentro de la etiqueta <code>database</code>, es necesario especificar este atributo. • statement—la sentencia de SQL ejecutada por esta consulta.

Tabla 7.2 Etiquetas InternetBeans Express (continuación)

Nombre de la etiqueta	Descripción	Atributos
control	Define una <code>IxControl</code> de InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code>—texto usado para identificar este control • <code>tupleModel</code>—el <code>tupleModel</code> para este control • <code>dataSet</code>—identifica el conjunto de datos (consulta) al cual está conectado el control. No es posible tener ambos pero se necesita el <code>dataSet</code> o el <code>tupleModel</code>. • <code>columnName</code>—identifica el <code>columnName</code> al cual está conectado el control.
image	Define una <code>IxImage</code> de InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code>—texto usado para identificar esta imagen • <code>tupleModel</code>—el <code>tupleModel</code> para este control • <code>dataSet</code>—identifica el conjunto de datos (consulta) al cual está conectada esta imagen. No es posible tener ambos pero se necesita el <code>dataSet</code> o el <code>tupleModel</code>. • <code>columnName</code>—identifica el <code>columnName</code> al cual está conectada esta imagen
submit	Define una <code>IxSubmitButton</code> de InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code>—texto usado para identificar este botón de envío • <code>methodName</code>—nombre del método que se ejecutará cuando se presione este botón
table	Define una <code>IxTable</code> de InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code>—texto usado para identificar esta tabla • <code>dataSet</code>—identifica el conjunto de datos (consulta) al cual está conectada esta tabla • <code>tableModel</code>—el nombre del modelo de datos para esta tabla. Se necesita bien el <code>dataSet</code> o bien el <code>tableModel</code>, pero no es posible tener ambos

Sólo hay seis etiquetas en la biblioteca de etiquetas de InternetBeans Express, aunque hay diecisiete componentes de InternetBeans. Esto puede considerarse una gran limitación, aunque realmente no lo es. La etiqueta `control` se dirige hacia una `IxControl`, que delega en todos los otros InternetBeans específicos de controles. Las únicas InternetBeans que no están comprendidas en la biblioteca de etiquetas son `IxSpan` e `IxLink`. Ninguna de ellas resulta útil en una JSP, ya que podría realizar lo mismo de forma sencilla, mediante su propio scriptlet de expresiones JSP.

Por supuesto, también es posible usar InternetBeans directamente, al igual que cualquier otra clase bean o Java. Es mucho más conveniente utilizar la biblioteca de etiquetas, que además realiza algunas tareas adicionales (como mantener el estado de la sesión de entrada de datos).

Formato de internetbeans.tld

Es útil saber que siempre se puede ver el código fuente del archivo `internetbeans.tld` en busca de sugerencias acerca del uso de las diferentes etiquetas. Para hacer esto, ábralo en el editor de JBuilder. Este archivo no puede (ni debe) modificarse.

El archivo `internetbeans.tld` se encuentra disponible en `internetbeans.jar`. No necesita poder ver los contenidos de dicho archivo para utilizar sus etiquetas en su JSP, pero si desea verlo en el editor, necesita ese paso adicional de añadirlo a su proyecto. Para ello:

- 1** Pulse el botón Añadir Archivos/Paquetes, que se encuentra en la barra de herramientas encima del panel del proyecto.
- 2** En su directorio de proyecto, busque `internetbeans.jar`. Estará en el directorio `WEB-INF/lib` de su WebApp.
- 3** En el árbol de directorios, expanda el nodo `internetbeans.jar`.
- 4** Situado en `com.borland.internetbeans.taglib`, localice el archivo `internetbeans.tld` y selecciónelo.
- 5** Pulse Aceptar para añadir el archivo a su proyecto.

La información de la parte inicial del archivo `internetbeans.tld` tiene poco interés. La información que merece la pena conocer comienza con la primera etiqueta `<tag>` dentro del archivo. Cada etiqueta `<tag>` representa una definición de etiqueta de InternetBeans.

Al principio de cada definición de etiqueta, verá una etiqueta `<name>` que indica su nombre. La primera es la etiqueta `database`. Anidadas dentro de cada definición de etiqueta también verá etiquetas `<tagclass>`, `<info>` y `<attribute>`. Para ver un ejemplo del aspecto que tiene una definición de etiqueta de InternetBeans, consulte el fragmento del archivo `internetbeans.tld` que define la etiqueta `submit` a continuación:

```

<tag>
  <name>submit</name>
  <tagclass>com.borland.internetbeans.taglib.SubmitTag</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>Submit button or submit image control</info>
  <attribute>
    <name>id</name>
    <required>false</required>
    <rteprvalue>false</rteprvalue>
  </attribute>
  <attribute>
    <name>methodName</name>
    <required>true</required>
    <rteprvalue>false</rteprvalue>
  </attribute>
</tag>

```

La etiqueta <tagclass> indica el nombre de la clase dentro del paquete com.borland.internetbeans.taglib que es responsable de interpretar esta etiqueta de InternetBeans cuando se usa en una JSP. La etiqueta <info> suministra una descripción de la etiqueta de InternetBeans.

La etiqueta <attribute> describe un atributo para una etiqueta de InternetBeans. Hay una etiqueta <attribute> para cada atributo. Pueden ser consideradas como las propiedades del componente. Verá estas propiedades anidadas dentro de la etiqueta <attribute>. Cada propiedad tiene un nombre, un valor booleano, que indica si es o no una propiedad necesaria y un valor booleano que indica si su valor puede asignarse o no mediante una expresión de java. El nombre se encuentra dentro de la etiqueta <name>, la etiqueta <required> indica si se requiere o no la propiedad y la etiqueta <rteexprvalue> indica si la propiedad puede asignarse o no mediante una expresión de java. Aquellas propiedades que se pueden asignar mediante una expresión requieren un valor literal.

El marco de trabajo Struts en JBuilder

El desarrollo web es una función de JBuilder Enterprise

El marco de trabajo de código abierto Struts se conoce como el concepto de diseño de software Modelo 2 o Controlador Modelo-Vista. Este marco evolucionó a partir del diseño de Modelo 1, la tecnología de Páginas JavaServer. Esta tecnología ofrecía grandes avances a partir de servlets sencillos, mientras que la presentación HTML se codificaba mediante extensas sentencias `out.println` en métodos `doGet()` y `doPut()`. Las JSP ofrecían un modo de incluir HTML en código Java y viceversa. Sin embargo, estas JSP son difíciles de leer y de mantener. Es necesario que tanto diseñadores como desarrolladores trabajen en el mismo conjunto de archivos fuente.

El marco de trabajo Struts, desarrollado inicialmente en 2001, combina lo mejor de los servlets y las JSP. Este marco de trabajo consiste en un paradigma de diseño de tres niveles: Controlador, Modelo y Vista. Struts incorpora su propio componente Controlador y se integra con otras tecnologías con el fin de proporcionar el Modelo y la Vista.

La capa Controlador controla el flujo de la aplicación. Recibe solicitudes de un visualizador y decide cómo procesarlas. Con Struts, el controlador se implementa como un servlet de la clase `org.apache.struts.action.ActionServlet`. El archivo `struts-config.xml` configura el Controlador mediante elementos `<action-mapping>`.

Se establece una capa lógica empresarial entre el Controlador y el Modelo. Esta se implementa con una clase `Action`, un fino englobador alrededor de la lógica empresarial real. La clase `Action` recibe el envío de un formulario o la solicitud de una página, y delega en las clases lógicas empresariales. Se guarda toda la información necesaria para las páginas futuras. La clase `Action` separa la lógica empresarial de la presentación y decide qué se muestra a continuación.

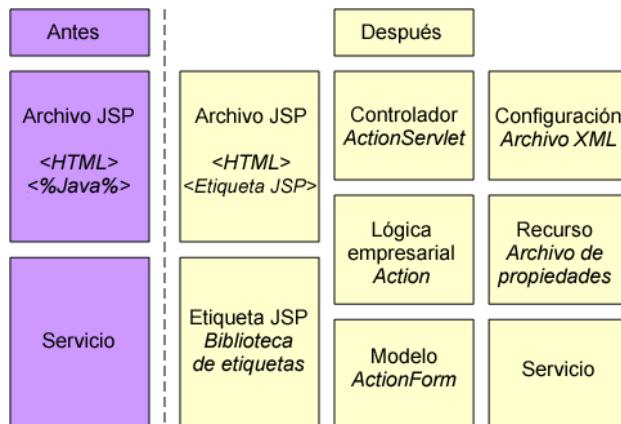
El Modelo representa el estado de la aplicación. Los objetos empresariales actualizan el estado de la aplicación. Un bean `ActionForm` representa el estado del modelo a nivel de petición de sesión. La clase `ActionForm` contiene métodos de obtención y definición para las propiedades y un método de validación. El archivo JSP (o la vista) lee la información de la clase `ActionForm` mediante etiquetas JSP. Debe haber una clase `ActionForm` para cada formulario de entrada.

La Vista es un archivo JSP. En la JSP no hay lógica de flujo, lógica empresarial o información de Modelo; sólo es una presentación. Struts ofrece un amplio conjunto de bibliotecas de etiquetas para utilizar en la capa Vista. Hay cuatro bibliotecas principales de etiquetas Struts: HTML, Bean, Logic y Template.

- Biblioteca de etiquetas HTML: Contiene etiquetas HTML que sustituyen a la mayoría de los elementos HTML comunes.
- Biblioteca de etiquetas Bean: Contiene etiquetas para mostrar datos de JavaBeans, como los que devuelve una consulta a una base de datos.
- Biblioteca de etiquetas Logic: Contiene etiquetas para realizar iteraciones y procesamiento condicional de datos, como la presentación de la información de una base de datos en formato de tabla.
- Biblioteca de etiquetas Template: Contiene etiquetas para importar fragmentos de código a páginas JSP; por ejemplo, el logotipo y cabecera de una empresa que se repiten en todas las páginas de una sede web.

La capa Vista puede utilizar un archivo de propiedades para guardar cadenas y facilitar, de este modo, la internacionalización.

En conjunto, la utilización del marco de trabajo Struts ha cambiado de forma significativa el diseño de aplicaciones web. En el siguiente diagrama se muestra la composición de una aplicación web antes y después de utilizar Struts. Observe cómo Struts separa limpiamente el código Java del Javascript y del código HTML.

Figura 8.1 Struts — antes y después

Si desea obtener más información, consulte “The Jakarta Project: Struts” en <http://jakarta.apache.org/struts/index.html>. Esta página contiene enlaces a la Struts Users Guide, a los artículos Struts, a programas de ejemplo y a los tutoriales.

Versiones beta Struts 1.0 y 1.1

JBuilder 8 oficialmente admite Struts 1.0. Aunque JBuilder 8 no admite oficialmente la versión beta de Struts 1.1. Sin embargo, si activa el archivo struts-config.xml para Struts 1.1 y descarga el archivo JAR de Struts 1.1 en la carpeta <jbuilder>/extras, se logra la compatibilidad con Struts 1.1. El Editor de configuración de Struts muestra elementos y atributos específicos de 1.1. Además, si su aplicación web Struts utiliza un archivo tileDefinitions.xml para la presentación, dispondrá de un editor visual para el archivo de configuración Tiles.

Herramientas de JBuilder para Struts

JBuilder proporciona un amplio conjunto de herramientas y asistentes para crear rápidamente una aplicación web preparada para Struts, incluido:

- Aceptación del marco de trabajo Struts
- Asistente para WebApp
- Asistente para JSP
- Asistente para Action

- Asistente para ActionForm
- Asistente para JSP a partir de ActionForm
- Asistente para conversión de Struts
- Editor de configuración de Struts

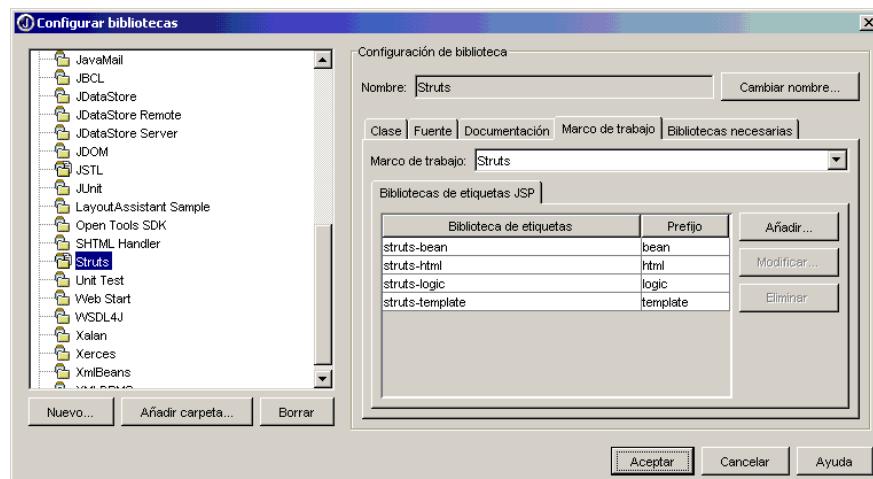
Aceptación del marco de trabajo Struts

La pestaña Marco de trabajo del cuadro de diálogo Configurar bibliotecas (Herramientas | Configurar bibliotecas) permite configurar marcos con el fin de utilizarlos en JBuilder. Un marco de trabajo consiste en una biblioteca de etiquetas y elementos específicos de taglib contenidos en un archivo JAR. Los siguientes marcos de trabajo se suministran junto con JBuilder, y están disponibles en la carpeta de JBuilder del lado izquierdo del cuadro de diálogo Configurar bibliotecas.

- Struts
- JSTL
- InternetBeans
- Cocoon

En el siguiente cuadro se muestra el marco de trabajo Struts que aparece en la pestaña del cuadro de diálogo Configurar bibliotecas.

Figura 8.2 Marco de trabajo Struts en Configurar bibliotecas



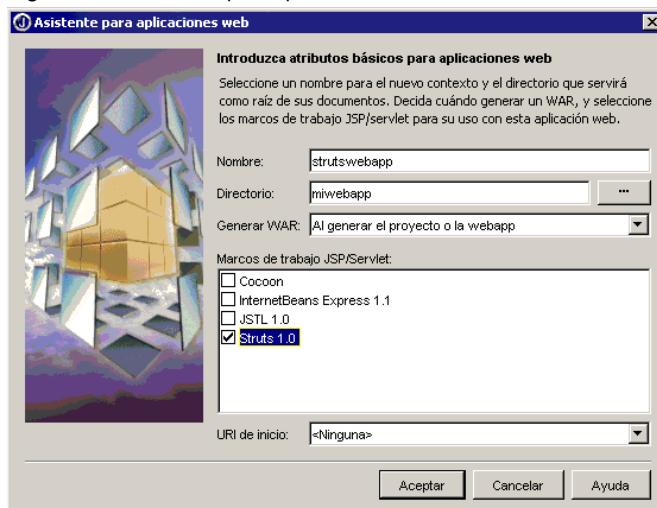
Observe cómo aparecen las bibliotecas de etiquetas Struts en la pestaña Bibliotecas de etiquetas JSP de la pestaña Marco de trabajo.

- Si desea obtener más información acerca de las bibliotecas de JBuilder y del cuadro de diálogo Configurar bibliotecas, consulte “Las bibliotecas” en el capítulo “Gestión de las vías de acceso” de *Creación de aplicaciones con JBuilder*.
- Para obtener más información sobre la creación de bibliotecas, consulte “[Las bibliotecas de etiquetas JSP y los marcos de trabajo en JBuilder](#)” en la página 6-5.

Asistente para aplicaciones web preparado para Struts

El Asistente para aplicaciones web está preparado para el marco de trabajo Struts. Con el fin de elegir el marco de trabajo Struts para su aplicación web, seleccione Struts 1.0 en la lista Marcos de trabajo JSP/Servlet. El asistente para aplicaciones web tendrá este aspecto:

Figura 8.3 Asistente para aplicaciones Web



Nota Se puede utilizar más de un marco de trabajo en una aplicación web. Por ejemplo, puede utilizar Cocoon, Struts y JSTL en una WebApp.

Una vez que haya elegido el marco de trabajo Struts para su WebApp, JBuilder añade correspondencias al archivo `web.xml` y copia o crea archivos si es necesario. En el caso del marco de trabajo Struts, los archivos `*.tld` se copian en el directorio `WEB-INF` y sus correspondencias se añaden al archivo `web.xml`. JBuilder crea el archivo `struts-config.xml` en el directorio `WEB-INF`.

Algunos marcos de trabajo web pueden declarar una página de bienvenida estándar que utiliza el URI. En este caso, el campo URI de inicio se actualiza automáticamente con el nombre de la página de bienvenida. Por ejemplo, si selecciona el marco de trabajo Cocoon, el valor de URI de inicio es /index.xml.

Una vez creada la aplicación web, puede cambiar las propiedades del nodo, incluida la selección de marcos de trabajo, si pulsa el nodo WebApp con el botón derecho del ratón en el panel del proyecto y selecciona Propiedades. Puede cambiar el marco de trabajo en la pestaña Marcos de trabajo JSP/Servlet de la ficha WebApp.

Importante

Para generar una aplicación web, lo primero que se ha de hacer es crear una webapp. Para obtener más información sobre la secuencia de pasos necesarios para crear una aplicación web Struts, consulte “[Creación de una aplicación web preparada para Struts en JBuilder](#)” en la página 8-18.

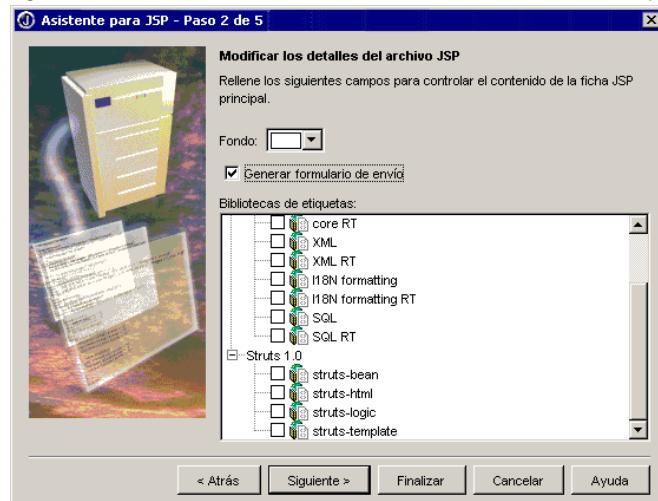
Asistente para JSP preparado para Struts

El Asistente para JSP también está preparado para el marco de trabajo Struts. En la ficha Modificar los detalles del archivo JSP del asistente se seleccionan las bibliotecas de etiquetas para utilizar con JSP. Para poder ver las bibliotecas de etiquetas Struts, es necesario que se desplace por la lista, tal y como se muestra en la siguiente ilustración.

Importante

Para obtener más información sobre la secuencia de pasos necesarios para crear una aplicación web Struts, consulte “[Creación de una aplicación web preparada para Struts en JBuilder](#)” en la página 8-18.

Figura 8.4 Ficha Modificar los detalles del archivo JSP – Asistente para JSP



Las bibliotecas de etiquetas se seleccionan marcando la casilla junto a la que desee utilizar. Puede elegir cualquier biblioteca de etiquetas, no es necesario que la que elija esté ya preparada para la aplicación web.

Asistente para ActionForm

Una clase `ActionForm` contiene el estado de sesión de una página web. Cada bean `ActionForm` constituye una subclase de la clase abstracta `ActionForm`. `ActionForm` contiene métodos de obtención y definición para cada uno de los campos de datos de los que `ActionForm` es responsable. Estos son normalmente todos los campos de un formulario en una página, pero también puede abarcar varias fichas relacionadas que actúan como un asistente. `ActionForm` también contiene un método `validate()` que valida los datos suministrados.

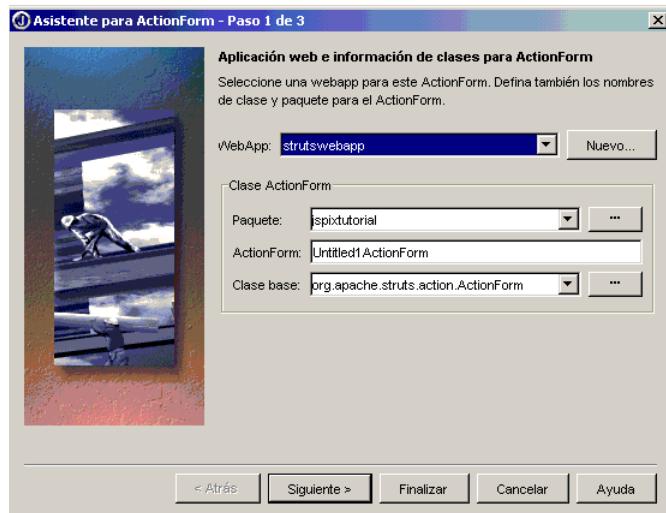
Importante Está creando una clase `ActionForm` para una JSP. Para poder utilizar el asistente, debe tener una JSP en el proyecto. Para obtener más información sobre la secuencia de pasos necesarios para crear una aplicación web Struts, consulte “[Creación de una aplicación web preparada para Struts en JBuilder](#)” en la página 8-18.

El Asistente para `ActionForm` permite crear rápida y fácilmente una `ActionForm`. Puede añadir campos mediante el asistente o rellenarlos previamente a partir de una página JSP ya creada. El asistente crea la clase `ActionForm` y la registra en el archivo `web.xml`.

Ficha Aplicación web e información de clases para ActionForm

En la ficha Aplicación web e información de clases para `ActionForm`, del asistente, se especifica la WebApp que se va a utilizar. Puede elegir una WebApp ya creada o crear una pulsando el botón Nuevo con el fin de abrir el Asistente para aplicaciones web. Esta ficha del asistente también se utiliza para seleccionar el paquete de la clase `ActionForm`, especificar el nombre de la clase y elegir la clase base.

Figura 8.5 Ficha Aplicación web e información de clases para ActionForm – Asistente para ActionForm



Ficha Definición de campos para ActionForm

La ficha Definición de campos para ActionForm permite definir los campos de los que ActionForm se hará responsable. Puede añadir los campos directamente a la lista o seleccionar una JSP desde la que importarlos. Los campos se derivan de las etiquetas de entrada descubiertas en la JSP.

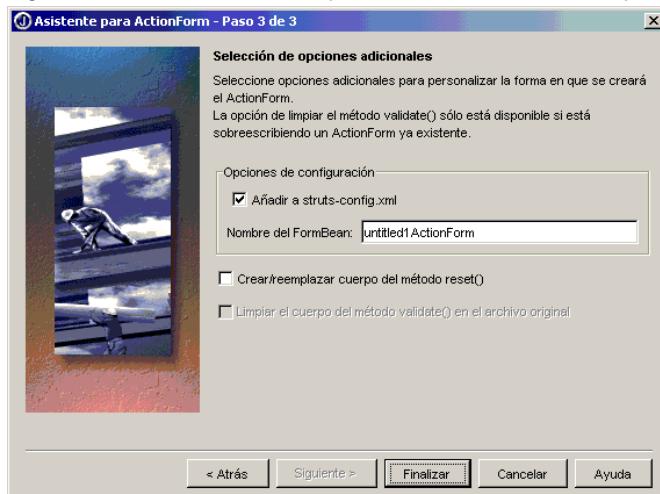
Figura 8.6 Ficha Definición de campos para ActionForm – Asistente para ActionForm



Ficha Selección de opciones adicionales

En la ficha Selección de opciones adicionales se eligen las opciones de configuración adicionales. Puede elegir si añade la correspondencia de ActionForm a struts-config.xml. También puede crear y sustituir el método reset() en ActionForm o borrar el método validate() existente.

Figura 8.7 Ficha Selección de opciones adicionales – Asistente para ActionForm



Nota

Si el nombre de la clase ActionForm que seleccione ya existe y es una ActionForm válida, se le solicita que importe los campos existentes de ActionForm. Esto resulta de gran utilidad para poder ampliar un formulario ya existente o para generar un formulario que ocupa varias JSP. Si elige no importar estos campos, el archivo original no se elimina, pero todos los métodos de obtención y definición serán suprimidos y sustituidos por los campos de la lista final. Se conservan los demás métodos adicionales. Si se eliminaran los campos, aparecería un mensaje de advertencia recordándole que tiene que reparar el método validate().

Después de introducir todos los campos, el asistente genera la ActionForm y la registra en el archivo struts-config.xml.

Asistente para Action

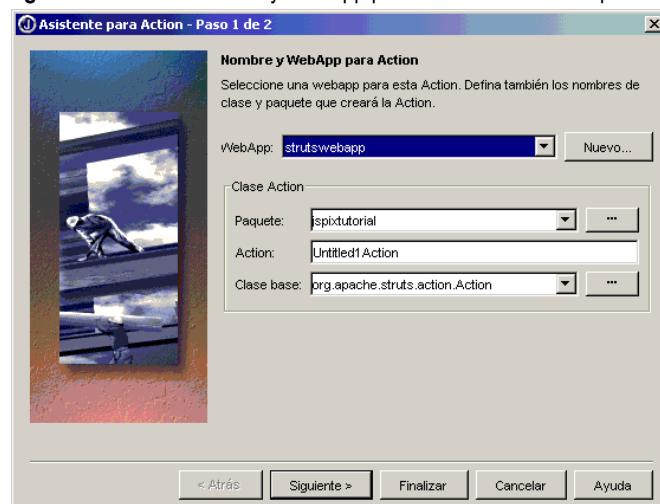
Una Action es la clase de control en el marco de trabajo Struts. Se llama al método Action.perform() para que ejecute la lógica empresarial. Se devuelve una clase ActionForward. ActionForward indica al servlet del controlador Struts la siguiente acción en la cadena de sucesos. El Asistente para Action crea rápidamente una clase Action y la registra en struts-config.xml.

Ficha Nombre y WebApp para Action

La ficha Nombre y WebApp para Action, del asistente, permite especificar la WebApp que se va a utilizar. Puede elegir una WebApp ya creada o crear una pulsando el botón Nuevo con el fin de abrir el Asistente para aplicaciones web. Esta ficha del asistente también se utiliza para introducir el paquete de la clase Action, especificar el nombre de la clase y seleccionar la clase antecesora.

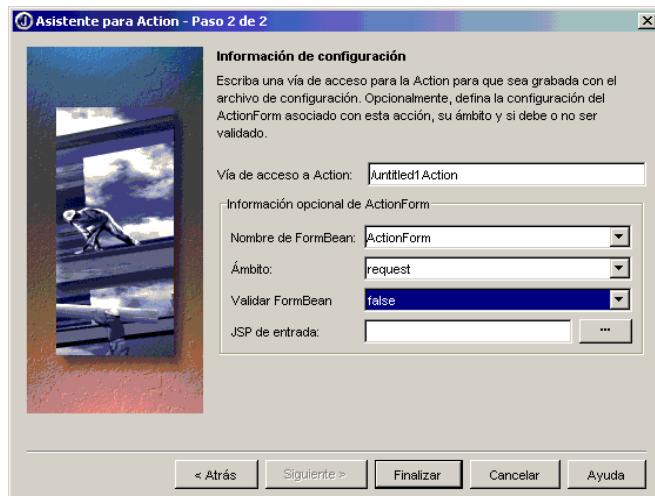
- Importante** Está creando una clase Action para una clase ActionForm. Para poder utilizar el asistente, debe tener una JSP y una clase ActionForm en el proyecto. Para obtener más información sobre la secuencia de pasos necesarios para crear una aplicación web Struts, consulte “[Creación de una aplicación web preparada para Struts en JBuilder](#)” en la página 8-18.

Figura 8.8 Ficha Nombre y WebApp para Action — Asistente para Action



Ficha Información de configuración

En la ficha Información de configuración del asistente se puede seleccionar la vía de acceso a Action y la configuración de ActionForm asociada con esta clase Action. Puede seleccionar una vía de acceso relativa al contexto para la clase Action, la ActionForm que se va a utilizar, el ámbito de Action, la validación de Action y la JSP de entrada. En caso de error de validación, el control se devuelve en la JSP de entrada. Los errores se muestran en la etiqueta `<html:errors />` de la JSP, que es normalmente la misma JSP que se envió al principio.

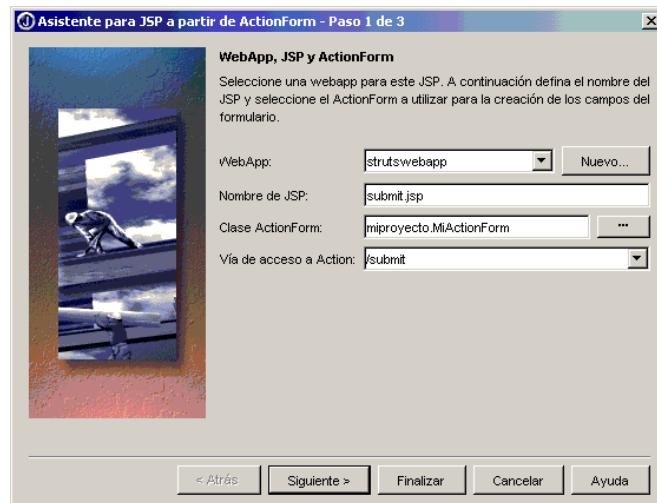
Figura 8.9 Ficha Información de configuración — Asistente para Action

Asistente para JSP a partir de ActionForm

El Asistente para JSP a partir de ActionForm toma la clase `ActionForm` y crea una página JSP. Este formulario se rellena con los campos seleccionados de `ActionForm`. Puede seleccionar los campos que añade a la JSP, así como el tipo de etiqueta que se utiliza en el campo, por ejemplo, `text`, `textarea`, `hidden` o `password`. Este asistente representa la inversa del Asistente para `ActionForm`.

Ficha WebApp, JSP y ActionForm

En la ficha WebApp, JSP y ActionForm, del asistente, se especifica la WebApp que se va a utilizar. Puede elegir una WebApp ya creada o crear una pulsando el botón Nuevo con el fin de abrir el Asistente para aplicaciones web. Esta ficha también se utiliza para elegir el nombre de la JSP, la clase `ActionForm` que se usa para crear los campos y la vía de acceso relativa al contexto de la acción.

Figura 8.10 Ficha WebApp, JSP y ActionForm — Asistente para JSP a partir de ActionForm**Ficha Tipos de etiquetas para campos ActionForm**

En la ficha Tipos de etiquetas para campos ActionForm en JSP se selecciona el tipo de etiqueta que debe utilizar cada campo ActionForm. El nombre del campo se deriva de los métodos de obtención y definición de la clase ActionForm, especificados en la ficha anterior del asistente. El valor por defecto para Tipo es text; sin embargo, puede cambiarlo a hidden, textarea o password. Pulse sobre el campo Tipo para abrir la lista desplegable. También puede elegir que no se utilice este campo.

Figura 8.11 Ficha Tipos de etiquetas para campos ActionForm en JSP — Asistente para JSP a partir de ActionForm

Ficha Indique las opciones para crear este JSP Struts

La ficha Indique las opciones para crear este JSP Struts permite seleccionar etiquetas y atributos adicionales. Puede elegir si utiliza la etiqueta Struts <html:base/>. También determina si se utilizan los atributos Struts locale o xhtml y cómo se importan las bibliotecas de etiquetas Struts.

Figura 8.12 Ficha Indique las opciones para crear este JSP Struts — Asistente para JSP a partir de ActionForm



Asistente para conversión de Struts

El Asistente para conversión de struts convierte las páginas JSP o HTML de modo que puedan utilizar etiquetas específicas de Struts. El asistente:

- Cambia el nombre de archivos .html por archivos .jsp si es necesario.
- Permite seleccionar las etiquetas convertibles Struts que se convierten realmente.
- Lee y analiza el HTML ya creado e identifica las etiquetas.
- Sustituye las etiquetas adecuadas por etiquetas Struts.
- Incluye las importaciones que requiera Struts.
- Escribe el archivo final, conservando al máximo la estructura y el contenido originales.

Advertencia

El archivo que crea el Asistente para conversión de Struts sustituye a su archivo original. Sin embargo, puede utilizar la vista del histórico de JBuilder para recuperarlo.

El Asistente para conversión de Struts se encuentra disponible en la ficha Web de la galería de objetos. También está disponible desde el menú

contextual del editor cuando se abre una JSP o archivo HTML. Si abre el asistente desde el menú contextual, el archivo a convertir se recoge en la lista Archivos JSP y HTML para su conversión en Struts, de la primera ficha del asistente.

Ficha Indique las fichas para convertir a Struts

En la ficha Indique las fichas para convertir a Struts se seleccionan las fichas que se convertirán. Los archivos deben ser o HTML o JSP. También se elige la WebApp de los archivos que crea el asistente. Puede especificar una WebApp ya creada u optar por una nueva si pulsa el botón Nuevo con el fin de abrir el Asistente para aplicaciones web.

Figura 8.13 Ficha Indique las fichas para convertir a Struts – Asistente para conversión a struts

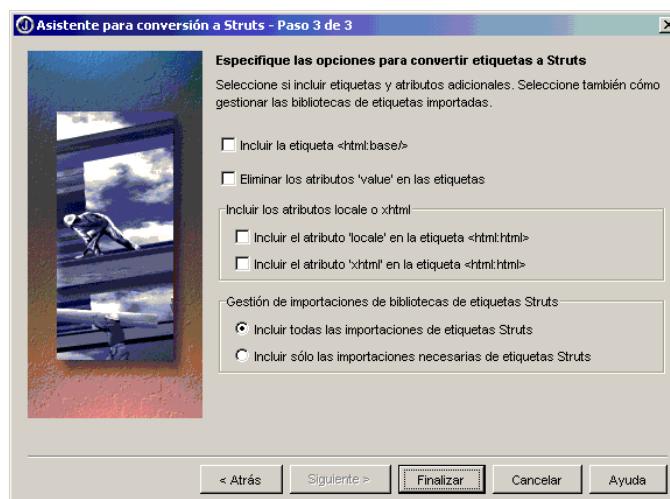


Ficha Etiquetas que se han de convertir

En la ficha Etiquetas que se han de convertir se seleccionan las etiquetas que se van a convertir a Struts. Esta ficha muestra todas las etiquetas disponibles para convertir. Puede seleccionar clases de etiquetas individuales para la conversión. El botón Seleccionar todo selecciona todas las etiquetas, mientras que el botón Ninguno deshace la selección de todas las etiquetas.

Figura 8.14 Ficha Etiquetas que se han de convertir — Asistente para conversión a struts**Ficha Especifique las opciones para convertir etiquetas a Struts**

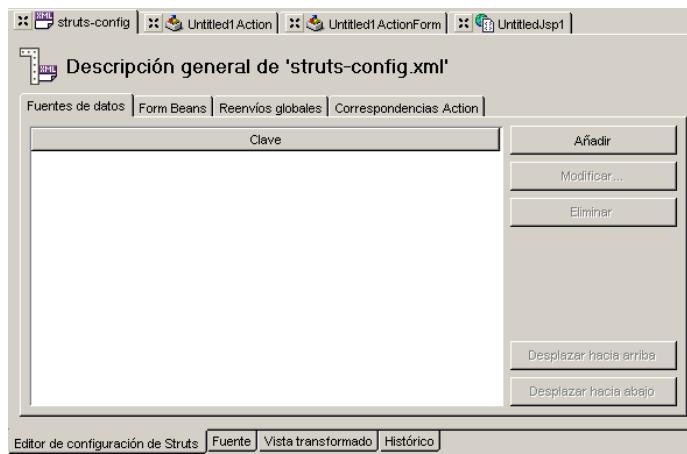
La ficha Especifique las opciones para convertir etiquetas a Struts permite seleccionar etiquetas y atributos adicionales. Puede elegir si utiliza la etiqueta Struts `<html:base>`. También puede elegir si elimina los atributos `value` en las etiquetas de entrada de texto, ya que Struts no los utiliza. Además, puede seleccionar los atributos `locale` o `xhtml` y elegir qué importaciones de bibliotecas de etiquetas Struts se utilizan.

Figura 8.15 Ficha Especifique las opciones para convertir etiquetas a Struts — Asistente para conversión a struts

Editor de configuración de Struts

El Editor de configuración de Struts es un editor visual que permite modificar el archivo `struts-config.xml`. Este archivo es un descriptor de distribución para su aplicación web Struts, y se debe ubicar en el directorio `WEB-INF`. Si utiliza los asistentes de JBuilder para crear su aplicación web preparada para Struts, se crea el archivo `struts-config.xml` y se coloca en la ubicación necesaria. El archivo `struts-config.xml` informa al servlet del controlador (`ActionServlet`) sobre sus correspondencias de aplicaciones.

Figura 8.16 Editor de configuración de Struts



Si desea obtener más información sobre el Editor de configuración de Struts, consulte el [Capítulo 13, “Modificación del archivo struts-config.xml”](#).

Implementaciones del marco de trabajo Struts en JBuilder

Mientras se crea una aplicación web preparada para Struts con las herramientas y asistentes de JBuilder, se va implementando el marco de trabajo Struts. JBuilder actualiza el archivo `web.xml`, crea un archivo `struts-config.xml`, copia los archivos Struts TLD en el directorio `WEB-INF` y añade las asignaciones TLD al archivo `web.xml`.

JBuilder:

- 1 Actualiza el archivo `web.xml`, el descriptor de distribución de la aplicación web, mediante:
 - La adición de una correspondencia de servlet para el servlet de control (`ActionServlet`) en `web.xml`.
 - La activación de la depuración a nivel 2.

- El establecimiento del valor por defecto para el archivo de configuración, struts-config.xml.
- La correspondencia del servlet mediante la correspondencia de extensiones *.do.

Para hacer esto, JBuilder añade el siguiente código en la parte superior del archivo web.xml:

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>2</param-value>
    </init-param>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Nota Si el archivo web.xml ya se corresponde con un servlet de nombre action, no se realiza ninguno de estos cambios.

- 2 Cree un WEB-INF/struts_config.xml si todavía no existe ninguno.
- 3 Copie los archivos TLD en el directorio WEB-INF.
- 4 Añada las correspondencias para los archivos TLD en web.xml, utilizando sus ubicaciones en el directorio WEB-INF. Las correspondencias tendrán este aspecto:

```
<taglib>
    <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
    <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
</taglib>
<taglib>
    <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
    <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>
<taglib>
    <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
    <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
</taglib>
<taglib>
    <taglib-uri>/WEB-INF/struts-template.tld</taglib-uri>
    <taglib-location>/WEB-INF/struts-template.tld</taglib-location>
</taglib>
```

Creación de una aplicación web preparada para Struts en JBuilder

La creación de una aplicación web preparada para Struts en JBuilder es un proceso que consta de varios pasos. Lo mejor es que primero cree su aplicación en papel. Determine el flujo lógico de su aplicación. ¿Qué vista se va a mostrar primero al usuario? ¿Qué acción va a seguir a esa vista? ¿Cómo se van a gestionar los errores? Existen varios tutoriales muy completos sobre Struts en la web que pueden ayudarle en la determinación del diseño y el flujo de la aplicación.

Consulte los enlaces de la página Jakarta Struts Resources Tutorials en <http://jakarta.apache.org/struts/resources/tutorials.html>.

Una vez que ya cuente con el diseño de su aplicación, puede empezar a utilizar las herramientas y asistentes de JBuilder para crearla. En los siguientes pasos se ofrece una breve descripción general de la creación de una aplicación web preparada para Struts en JBuilder.

- 1** Cree un proyecto.
- 2** Utilice el Asistente para aplicaciones web (Archivo | Nuevo | pestaña Web | Aplicaciones web) con el fin de crear su aplicación web preparada para Struts. Seleccione el marco de trabajo Struts.
- 3** Utilice el Asistente para JSP (Archivo | Nuevo | Web | Página JavaServer) con el fin de crear la página JSP de la vista introductoria de la aplicación. Esta es la capa de vista de la aplicación web Struts.
 - a** En la ficha Declarar JSP y componentes, del asistente, escriba el nombre de la JSP. No seleccione ninguna de las opciones.
 - b** En la ficha Modificar los detalles del archivo JSP, elija el color de fondo deseado. No genere un formulario de envío. Desplácese por la lista Bibliotecas de etiqueta hasta que aparezcan las bibliotecas de etiquetas Struts. Seleccione las bibliotecas de etiquetas Struts que vaya a utilizar con esta JSP:
 - struts-beans: Etiquetas para mostrar datos a partir de JavaBeans, como los devueltos por una consulta a una base de datos.
 - struts-html: Etiquetas para sustituir la mayoría de los elementos HTML.
 - struts-logic: Etiquetas para realizar iteraciones y procesamiento condicional de datos de vista, como la presentación de la información de una base de datos en formato de tabla.
 - struts-template: Etiquetas para importar fragmentos de código a páginas JSP; por ejemplo, el logotipo y cabecera de una empresa que aparecen en todas las páginas de una sede web.
 - c** Puede pulsar Finalizar en esta ficha, no es necesario que configure opciones en otras fichas.

- 4** Añada etiquetas Struts a la JSP para diseñar la vista. Si desea obtener más información acerca de la generación de la capa de vista en la aplicación web Struts, consulte "Building View Components" en *Struts User's Guide* en <http://jakarta.apache.org/struts/userGuide>. El documento también incluye información acerca de la creación de formularios con Struts y la internacionalización de la aplicación web Struts. También se discuten otras técnicas de presentación, como la inclusión, la organización en mosaico (una función de Struts 1.1), los componentes de visualización de imágenes y la visualización de texto.

Nota Si no utiliza etiquetas Struts en la JSP, puede utilizar el Asistente para conversión a Struts para convertir las etiquetas HTML de la JSP a Struts.

- 5** Utilice el Asistente para ActionForm (Archivo | Nuevo | pestaña Web | ActionForm) con el fin de diseñar el ActionForm para la capa de vista.
- a** En la ficha Aplicación web e información de clases para ActionForm, del asistente, seleccione la aplicación web y el paquete para ActionForm. Normalmente se trata del valor por defecto. Introduzca el nombre de la clase. Utilice un nombre parecido al nombre de la JSP.
 - b** En la ficha Definición de campos para ActionForm, elija Añadir desde JSP si ya ha añadido los campos a la JSP. Seleccione la JSP del cuadro de diálogo Añadir JSP. Los campos y sus tipos se rellenan automáticamente.
 - c** Configure las opciones por defecto en la ficha Selección de opciones adicionales. La opción por defecto crea automáticamente la correspondencia <form-bean> en struts-config.xml.
 - d** Pulse el botón Finalizar.
- 6** JBuilder crea automáticamente los métodos de obtención y definición en ActionForm, según los nombres de los campos. No es necesario que modifique este archivo. Si desea obtener más información acerca de la clase ActionForm, consulte "ActionForm classes" en *Struts User's Guide* de <http://jakarta.apache.org/struts/userGuide>.
- 7** Utilice el Asistente para Action (Archivo | Nuevo | pestaña Web | Action) con el fin de crear la clase Action para la capa de vista.
- a** En la ficha Nombre y WebApp para Action, acepte la aplicación web, paquete y clase base por defecto. Cambie el nombre de Action por el que desee. Debe ser un nombre parecido al nombre del archivo JSP.
 - b** En la ficha Información de configuración, asigne el valor por defecto a Vía de acceso a Action. Elija el form bean de la lista desplegable (es el nombre asignado a la clase ActionForm que creó con el Asistente para ActionForm). Establezca la JSP de entrada como la JSP a la que se devuelve el control si se produce un error de validación. No es necesario que configure las opciones Ámbito o Validar FormBean.

- c** Pulse Finalizar para crear la clase Action.
- 8** Abra la clase Action en el editor. Añada lógica empresarial al método `perform()`. Una vez añadido el código al método, puede borrar la última línea del método, la línea de gestión de errores. (Esta línea estará fuera de ámbito, una vez que haya implementado el método.) Si desea obtener más información acerca de la clase Action, consulte “Action classes” de *Struts User's Guide* en <http://jakarta.apache.org/struts/userGuide>.
- 9** Añada la correspondencia de acción a la JSP en el elemento `action` de la etiqueta `<html:form />`.
- 10** Seleccione Archivo | Guardar todo y, a continuación, elija Ejecutar | Ejecutar el proyecto.

Configuración del servidor web

El desarrollo web es una
función de JBuilder
Enterprise

Tanto los servlets de Java como las Páginas de JavaServer (JSP) se ejecutan dentro de servidores web. Tomcat, la implementación de referencia de las Páginas de JavaServer/ Servlets de Java, se incluye en JBuilder. Aunque pudiera ser diferente de su servidor web, Tomcat permite desarrollar y comprobar sus servlets y JSP dentro del entorno de desarrollo de JBuilder.

JBuilder Enterprise admite los servidores de aplicaciones Borland Enterprise Server, Sun iPlanet, IBM WebSphere y BEA WebLogic. Estos servidores de aplicaciones contienen servidores web. No obstante, para utilizar el servidor web incluido, se debe tener instalado y configurado el correspondiente servidor de aplicaciones. No se puede utilizar el servidor web fuera del servidor de aplicaciones que lo contiene. Después de que se haya configurado la aplicación web y el servidor web, se pueden compilar, ejecutar y depurar los servlets y las JSP. Para obtener más información, consulte el [Capítulo 10, “Las aplicaciones web en JBuilder”](#).

Nota No obstante, en JBuilder Enterprise se puede utilizar el servidor web Tomcat como servidor web autónomo.

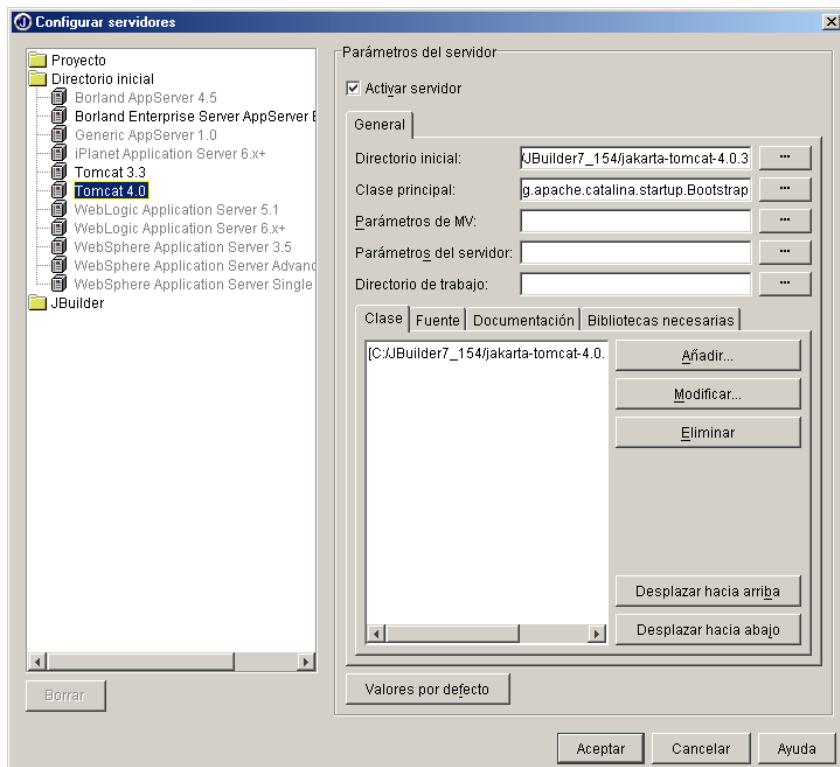
Visualización de las configuraciones de Tomcat

Cuando se instala JBuilder Enterprise, tres versiones de Tomcat —la 3.3, la 4.0 y la 4.1— se instalan automáticamente en el directorio de JBuilder. Por defecto, se configuran automáticamente. Se puede examinar dicha configuración con el cuadro de diálogo Configurar servidores.

Tomcat 4.0 y 4.1 forman JDK 1.4 Lightweight Editions. Estas ediciones no contienen un analizador XML (JDK 1.4 lleva uno incorporado) ni algunos componentes disponibles para el público. Puede descargarse la edición completa en <http://jakarta.apache.org/> y utilizar en cuadro de diálogo Configurar servidores para apuntar hacia ella.

Nota En JBuilder Enterprise, Tomcat 4.0 es el servidor por defecto. Si desea cambiar esta opción en su proyecto, consulte “[Elección de un servidor para su proyecto](#)” en la página 9-4.

- 1 Seleccione Herramientas | Configurar servidores. Se muestra el cuadro de diálogo Configurar servidores.



Nota En el árbol de la izquierda del cuadro de diálogo, las entradas en gris todavía no han sido configuradas. Las entradas en rojo no son válidas.

- 2 Escoja una de las instalaciones de Tomcat en la carpeta Directorio inicial. La opción Activar servidor del extremo superior derecho del cuadro de diálogo se marcará automáticamente. No necesita cambiar ningún parámetro.
- 3 Pulse Aceptar para cerrar el cuadro de diálogo.

Si ha instalado Tomcat en otro directorio, y desea ejecutarlo desde él y no desde el predeterminado, puede cambiar los parámetros de modo que se dirijan a dicho directorio. La siguiente tabla explica los parámetros de configuración.

Tabla 9.1 Parámetros del cuadro de diálogo Configurar servidores para Tomcat

Opción	Descripción
Directorio de inicio	Directorio de inicio de Tomcat. Si está reconfigurando una de las versiones de Tomcat instaladas con JBuilder, estará en el directorio <jbuilder>/thirdparty.
Clase principal	La clase principal para iniciar el servidor Web Tomcat.
Parámetros de MV	Los parámetros que hay que pasarle a la MV Java que ejecuta el servidor web.
Parámetros del servidor	Los parámetros que hay que pasarle al servidor web.
Directorio de trabajo	El directorio de trabajo.
Clase	La ubicación del archivo de clase de Tomcat.
Fuente	Ubicación de los archivos fuente de Tomcat.
Documentación	Ubicación de los archivos de documentación de Tomcat.
Bibliotecas necesarias	Bibliotecas que necesita Tomcat.

Si necesita más información acerca de Tomcat o si quiere ejecutarlo de forma autónoma, encontrará más información en el directorio de la documentación de instalación de Tomcat de JBuilder.

- Tomcat 3.3 — <jbuilder>/thirdparty/jakarta-tomcat-3.3.1/doc
- Tomcat 4.0 — <jbuilder>/thirdparty/jakarta-tomcat-4.0.6-LE-jdk14/webapps/tomcat-docs
- Tomcat 4.1 — <jbuilder>/thirdparty/jakarta-tomcat-4,10,12-LE-jdk14/webapps/tomcat-docs

Nota Tomcat 4.1 no admite la depuración de JSP.

Configuración de otros servidores web

JBuilder es compatible con otros servidores web, además de Tomcat:

- Borland Enterprise Server
- iPlanet de Sun
- Sybase Enterprise Application Server
- WebLogic de BEA
- WebSphere de IBM

Estos servidores web se configuran automáticamente cuando se configura el correspondiente servidor de aplicaciones en el cuadro de diálogo Configurar servidores (Herramientas | Configurar servidores). No se pueden utilizar estos servidores web fuera del servidor de aplicaciones que los contiene.

Si desea configurar los servidores de aplicaciones que contienen estos servidores web, consulte “Configuración del servidor de aplicaciones de destino” en la *Guía del desarrollador de Enterprise JavaBeans*.

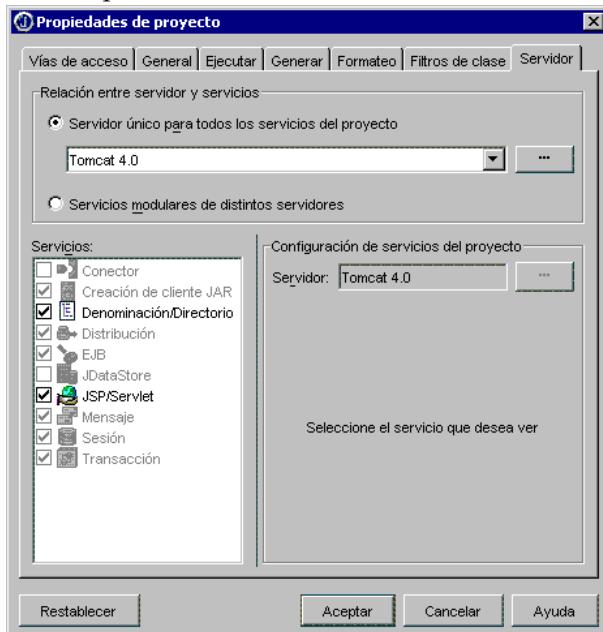
- Nota** El servidor de aplicaciones Borland Enterprise Server utiliza Tomcat internamente como su servidor web de producción. No se necesita configurar Tomcat por separado.

Elección de un servidor para su proyecto

JBuilder puede funcionar con varios servidores de aplicaciones, y sus servidores web incluidos, en un solo proyecto. Se puede elegir un único servidor de aplicaciones contenedor para todas las etapas del desarrollo de aplicaciones web y EJB. También se pueden elegir distintos servicios suministrados por distintos servidores para distintos aspectos del desarrollo. Por ejemplo, puede seleccionar el servicio EJB suministrado por Borland Enterprise Server para el desarrollo de enterprise beans y el servicio JSP/Servlet suministrado por Tomcat para el desarrollo de aplicaciones web.

- Sugerencia** Si no se tiene ningún servidor de aplicaciones instalado, JBuilder dirigirá automáticamente su proyecto a Tomcat 4.0. No obstante, si los iconos web de la ficha Web de la galería de objetos están desactivados, debería seleccionar un servidor para su proyecto; siga estos pasos:
- Importante** El único servicio disponible para Tomcat 3.3 es el servicio JSP/Servlet. Para Tomcat 4.0 y 4.1 están disponibles los servicios JSP/Servlet y Denominación/Directorio.
- Para seleccionar uno o más servidores de aplicaciones (y sus servidores web incluidos) con el fin de utilizarlos en su proyecto:
- 1 Seleccione Proyecto | Propiedades de proyecto.

- 2** Pulse la pestaña Servidor. Se muestra la ficha Servlet:

**Nota**

Para que un servidor de aplicaciones se muestre en la lista desplegable de la parte superior de este cuadro de diálogo, debe haber sido previamente configurado en el cuadro de diálogo Configurar servidores.

- 3** Si tiene instalados y configurados uno o más servidores de aplicaciones, decida si quiere utilizar un único servidor web y de aplicaciones para todos los aspectos del desarrollo o si desea distintos servidores de aplicaciones para gestionar diferentes áreas de desarrollo.
- Para utilizar un único servidor:
 - 1 Seleccione la opción Servidor único para todos los servicios del proyecto y elija el servidor en la lista desplegable. Este servidor puede ser un servidor de aplicaciones que contenga un servidor web o bien el servidor web Tomcat instalado con JBuilder.
 - 2 Si quiere evitar que se añadan a su proyecto librerías que no va a utilizar, en la lista Servicios desactive la casilla de selección junto al servicio o servicios que no necesita. Si desactiva servicios, se desactivarán las correspondientes características de JBuilder. Por ejemplo, si desactiva el servicio JSP/Servlet, se desactivarán la mayoría de los Asistentes web y la característica de compilación JSP.

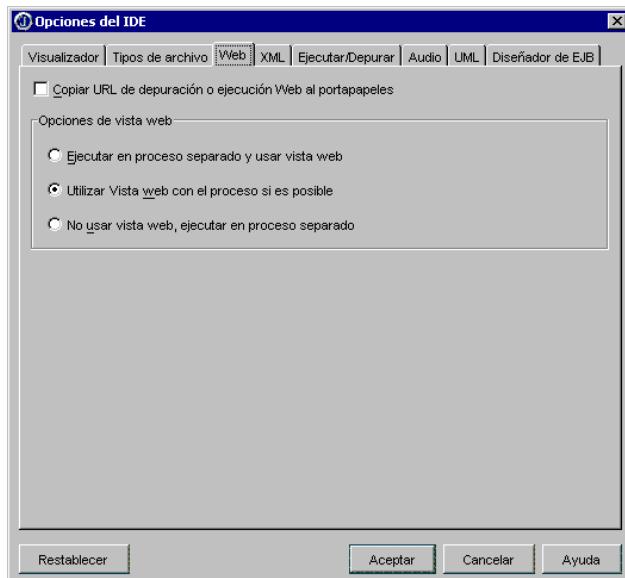
- 3 Si desea realizar cambios a los parámetros de configuración del servidor seleccionado, haga clic sobre el botón con puntos suspensivos, a la derecha del nombre del servidor, y modifique los parámetros deseados en las fichas General y Personalizada. Pulse Aceptar cuando haya terminado.
- Para utilizar diferentes servidores en diferentes servicios:
 - 1 Seleccione la opción Servicios modulares de distintos servidores.
 - 2 Si quiere evitar que se añadan a su proyecto librerías que no va a utilizar, en la lista Servicios desactive la casilla de selección junto al servicio o servicios que no necesita. Si desactiva servicios, se desactivarán las correspondientes características de JBuilder. Por ejemplo, si desactiva el servicio JSP/Servlet, se desactivarán la mayoría de los Asistentes web y la característica de compilación JSP.
 - 3 Actualice la configuración del servicio seleccionado en el lado derecho del cuadro de diálogo. Dependiendo del servidor/servicio seleccionado, esta información será configurable o de sólo lectura. Si desea más información, consulte “Selección de un servidor” en la *Guía del desarrollador de Enterprise JavaBeans*.
 - 4 Para utilizar un servidor web, haga clic en el servicio JSP/Servlet. En la lista desplegable Servidores del lado derecho del cuadro de diálogo, seleccione el servidor web que desea utilizar. Si quiere hacer cambios en los parámetros de configuración del servidor web, haga clic sobre el botón con puntos suspensivos y modifique los parámetros que desee en la ficha General. Pulse Aceptar cuando haya terminado.
- 4 Pulse Aceptar de nuevo para cerrar la ficha Servidor.

Configuración del IDE para ejecutar/depurar web

Una vez que haya configurado JBuilder con un servidor web, puede configurar opciones para el servidor web, incluidas las opciones de Ver web y cómo se inicia el servidor web en el IDE.

Para configurar el comportamiento del IDE cuando se intenta Ejecutar web o Depurar web:

- 1** Seleccione la pestaña Web en el cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE). La ficha Web tendrá este aspecto.



- 2** Seleccione la opción Copiar URL de depuración o ejecución Web al portapapeles, con el fin de copiar la URL utilizada para iniciar la aplicación web. Esto facilita la posibilidad de ir a la misma URL con un visualizador externo. Active esta opción si está creando una aplicación o un applet Java Web Start.
- 3** Seleccione Opciones de vista web. Estas opciones se usan en conjunción con la opción Buscar puerto disponible, en el cuadro de diálogo Configuración de ejecución, cuando el puerto especificado está siendo utilizado por un proceso ajeno a la web. (Consulte "["Creación de una configuración de ejecución"](#) en la página 10-2 para obtener más información.)
- Seleccione la opción Ejecutar en proceso separado y usar vista web, con el fin de utilizar tanto un visualizador web interno como uno externo. Esta opción muestra automáticamente su servlet o JSP en la ficha Vista web del panel de contenido.
 - Si desea utilizar el visualizador web interno para ver su página, seleccione la opción Utilizar vista web con el proceso si es posible. Esta opción muestra automáticamente su servlet o JSP en la ficha Vista web del panel de contenido. Si ya se está ejecutando un servidor web, JBuilder utiliza el mismo proceso en el puerto existente. Ésta es la opción por defecto.
 - Cuando inicie su aplicación web en un visualizador web externo, seleccione la opción No usar vista web, ejecutar en proceso separado.

10

Las aplicaciones web en JBuilder

El desarrollo web es una función de JBuilder Enterprise

JBuilder proporciona comandos en el menú contextual del panel del proyecto que facilitan la ejecución y depuración de servlets y JSP. Ejecutar web ejecuta su servlet o JSP mediante la configuración de ejecución del servidor seleccionado. Depurar web depura la JSP o los servlets mediante la configuración de ejecución del servidor seleccionado con opciones de depuración, permitiendo la ejecución paso a paso y la verificación del código de forma sencilla. Optimizar web ejecuta el servidor con un optimizador/analizador como OptimizeIt (necesita tener uno instalado). Los comandos web muestran las configuraciones de ejecución disponibles en el proyecto.

Mediante la selección de Ejecutar web o Depurar web se ejecuta o se depura el servlet o la JSP en su contexto WebApp. Si el archivo JSP, HTML o SHTML no está en una WebApp, no puede ser ejecutado o depurado en web y no aparecen los comandos Ejecutar web ni Depurar web en el menú contextual. Para obtener más información sobre los hilos, consulte “[La WebApp y sus propiedades](#)” en la página 3-6.

Nota

Si se establece una URI de inicio para una webapp (en el Asistente para aplicaciones web), se puede ejecutar la webapp desde esa ubicación específica, desde el menú contextual del nodo WebApp. De esta forma, si la aplicación tiene un punto de inicio obvio, se puede ejecutar, depurar y optimizar sin necesidad de profundizar en el contenido web.

Cuando se crea un servlet o una JSP utilizando el Asistente para servlets o JSP de JBuilder, los comandos Ejecutar web y Depurar web se activan automáticamente.

Importante Las applets no pueden ejecutarse o depurarse. Esto se debe a que no tienen una URL o un contexto web en el que ejecutarse. Además, las applets se ejecutan en el navegador de un cliente, en lugar de un servidor. Habitualmente, un applet se ejecuta en appletviewer, de Sun, o en AppletTestbed, de JBuilder. Para obtener más información, consulte “[El AppletTestBed de JBuilder y el appletviewer de Sun](#)” en la página 14-22.

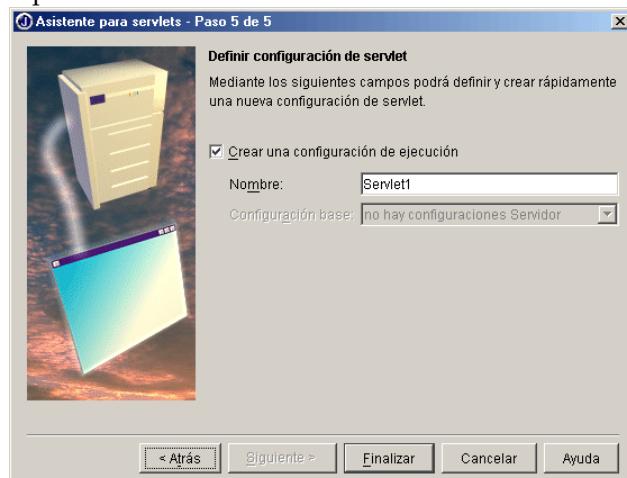
Antes de que pueda ejecutar su applet, servlet o JSP, necesita crear una configuración de ejecución. Para los servlets no creados con el asistente, también necesita configurar propiedades de ejecución. Después, una vez compile su servlet o JSP, ya podrá ejecutarlo.

Creación de una configuración de ejecución

Antes de poder ejecutar su applet, servlet o JSP en JBuilder, debe crear una configuración de ejecución. La configuración determina el tipo de ejecutor y qué se ejecuta. También se pueden definir opciones de depuración en la configuración. Hay dos formas de crear una configuración para un applet, JSP o servlet; puede utilizar el asistente para crear la configuración básica y después personalizarla si es necesario. Alternativamente, puede crear la configuración directamente mediante el comando Ejecutar | Configuraciones.

Creación de una configuración de ejecución con los asistentes

La ficha de configuración de ejecución de los asistentes para Applet, JSP y Servlet le permite crear rápidamente una configuración de ejecución. En el Asistente para servlet, la ficha de configuración de ejecución tiene este aspecto:



Cuando crea una configuración de ejecución del applet con el asistente, el ejecutor de applet se activa. El applet se ejecuta en el visualizador de applets de JBuilder, AppletTestbed, mediante el archivo de clase de applet. (Observe que no puede Ejecutar web o Depurar web un applet porque no inicia un servidor para ejecutarse, se ejecuta en un navegador. Además, las applets no tienen un contexto de aplicación web, o WebApp.)

Cuando crea una configuración de ejecución de servlet con el asistente, el ejecutor de servidor se activa. El servidor seleccionado en la ficha Servidor de Propiedades de Proyecto se convierte en el servidor web activo. La URI a ejecutar se toma de la entrada del campo Patrón de URL en la ficha Introduzca los detalles de webapp del Asistente para servlet y desde la WebApp del servlet.

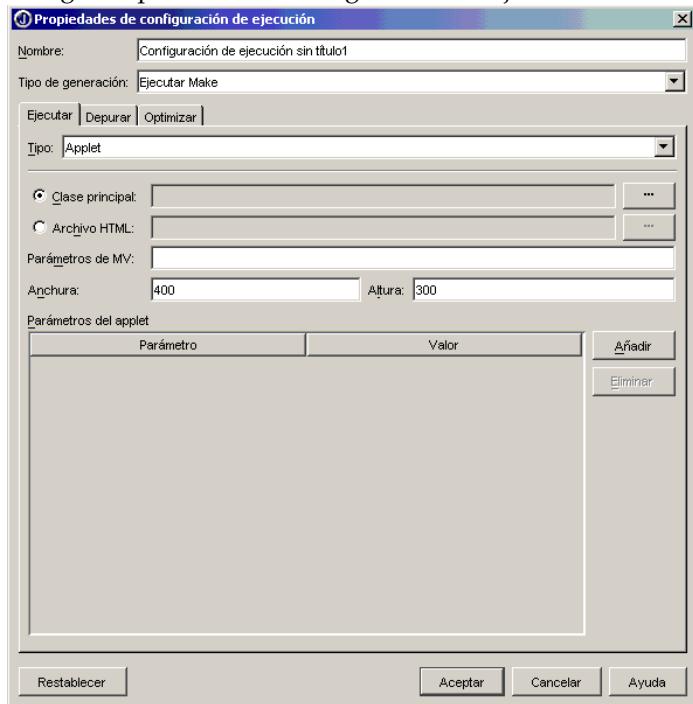
Cuando crea una configuración de ejecución de JSP con el asistente, el ejecutor de servidor se activa. El servidor seleccionado en la ficha Servidor de Propiedades de Proyecto se convierte en el servidor web activo. La URI a ejecutar se toma de la WebApp y del nombre de archivo JSP.

Creación de una configuración de ejecución de applets

Si ya tiene un programa, o elige no crear una configuración de ejecución con el asistente para Applet, puede crearla mediante el cuadro de diálogo Propiedades de configuración de ejecución. Para ello:

- 1 Seleccione Ejecutar | Configuraciones. Aparece la ficha Ejecutar de Propiedades de proyecto.

- 2** Pulse Nuevo para crear una configuración. Se muestra el cuadro de diálogo Propiedades de configuración de ejecución.



- 3** Introduzca el nombre de la configuración en el campo Nombre.
- 4** Elija el tipo de generación que desea ejecutar antes de iniciar. Si desea obtener más información, consulte “Tipos de generación” en el capítulo “Definición de las configuraciones de ejecución” de *Creación de aplicaciones con JBuilder*.
- 5** Seleccione Applet en la lista desplegable Tipo.
- 6** Elija Clase principal con el fin de ejecutar el applet en AppletTestbed de JBuilder. Cuando se crea un applet mediante el Asistente para applets, éste fija la clase principal automáticamente. La clase seleccionada debe contener un método `init()`. Busque la clase principal pulsando el botón con puntos suspensivos y seleccione la clase en el cuadro de diálogo Seleccionar clase principal del proyecto.
- 7** Elija la opción Archivo HTML con el fin de ejecutar el applet en el **appletviewer** de Sun. El archivo HTML debe contener la etiqueta `<applet>` y el atributo `code` debe contener el nombre completo de la clase. Busque el archivo HTML pulsando el botón con puntos suspensivos y selecciónelo en el cuadro de diálogo Seleccione el archivo HTML que se va a ejecutar.

- 8** En el campo Parámetros de la MV, introduzca cualquier parámetro que quiera pasar a la Máquina Virtual (MV) de Java. Si desea obtener más información sobre la MV de Java y las opciones que se le pueden pasar, consulte “Basic Tools: java - The launcher for Java technology applications”.
- 9** Si seleccionó la opción Clase principal para ejecutar el applet, introduzca el tamaño y cualquier otro parámetro inicial:
 - a** Introduzca el ancho y alto iniciales del applet en los campos Anchura y Altura.
 - b** Introduzca los parámetros del applet en la lista Parámetros de applet. Escriba el nombre del parámetro en la columna Nombre y su valor inicial en la columna Valor.

Nota Si ejecuta el applet desde la página HTML, no necesita declarar parámetros, porque ya están configurados en la página HTML.

- 10** Al terminar, pulse Aceptar con el fin de cerrar el cuadro de diálogo Propiedades de configuración de ejecución. A continuación se muestra la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto. Su nueva configuración está resaltada.
- 11** Si desea que esta configuración sea la configuración por defecto, marque la casilla de verificación en la columna Por defecto. (Sólo una configuración del proyecto puede ser la configuración por defecto.) Si desea añadir esta configuración al menú contextual del panel del proyecto, marque la casilla de la columna Menú contextual. Pulse Aceptar para cerrar el cuadro de diálogo.

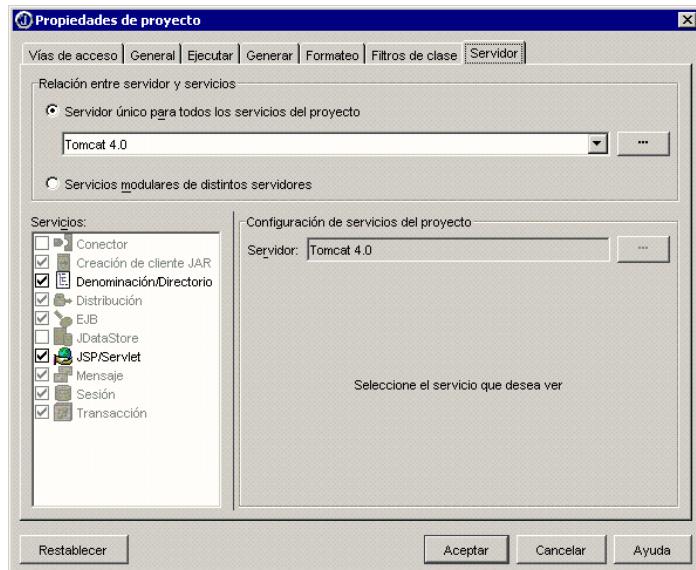
Las applets no pueden ejecutarse o depurarse. Esto se debe a que no tienen una URL o un contexto web en el que ejecutarse. Además, las applets se ejecutan en el navegador de un cliente, en lugar de un servidor. Habitualmente, un applet se ejecuta en appletviewer, de Sun, o en AppletTestbed, de JBuilder. Para obtener más información, consulte [“El AppletTestBed de JBuilder y el appletviewer de Sun” en la página 14-22](#).

Para obtener más información acerca de las configuraciones de ejecución, consulte el capítulo “Definición de las configuraciones de ejecución” de *Creación de aplicaciones con JBuilder*.

Creación de una configuración de ejecución del servidor

Antes de configurar el servidor, es necesario que verifique que hay un servidor activado para su proyecto. Para ello:

- 1 Abra el cuadro de diálogo Propiedades de proyecto y seleccione la pestaña Servidor.



- 2 Compruebe que hay un servidor seleccionado.

- Si la opción Servidor único para todos los servicios del proyecto está marcada en este proyecto, compruebe que el valor de la lista desplegable no es <Ninguno>. Asegúrese de que el servidor seleccionado admite el servicio JSP/Servlet. La casilla de verificación de ese servicio debe estar activada en el árbol del lado izquierdo de la ficha. Si el servidor no admite el servicio JSP/Servlet, ese servicio no estará disponible.
- Si se selecciona la opción Servicios modulares de distintos servidores, escoja el servicio JSP/Servlet en el lado izquierdo de la ficha del cuadro de diálogo. Elija un servidor en la lista desplegable del lado derecho de la ficha; asegúrese de que no está seleccionado <ninguno>.

Nota

Observe que el servidor seleccionado puede permitir que los servicios estén desactivados durante la ejecución. Por ejemplo, Borland Enterprise Server lo permite. Asegúrese de que el servicio JSP/Servlet no ha sido deshabilitado en la configuración de ejecución, puesto que si es así, los comandos web no estarán disponibles.

- 3 Pulse Aceptar para cerrar el cuadro de diálogo Propiedades de proyecto.

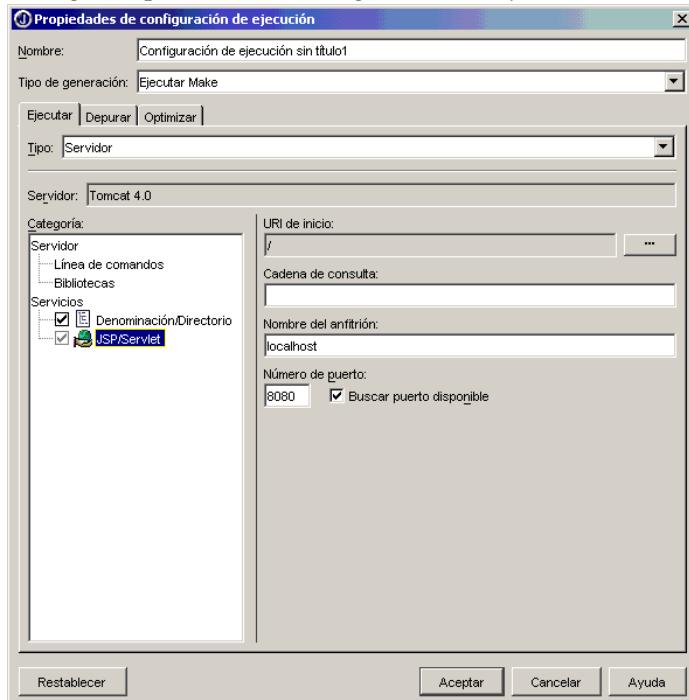
Si desea obtener más información sobre los servidores web, consulte el [Capítulo 9, "Configuración del servidor web"](#).

Una vez activado el servidor para su proyecto, puede modificar o crear una configuración de ejecución del servidor.

Importante Si cuenta con dos servidores diferentes en el mismo proyecto, necesitará dos configuraciones diferentes de ejecución, una para cada servidor.

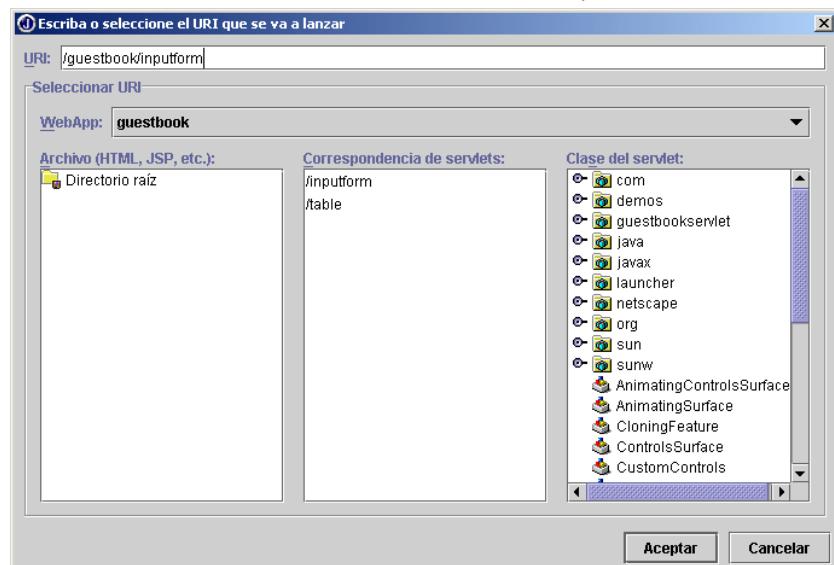
Para crear una configuración de ejecución del servidor:

- 1 Seleccione Ejecutar | Configuraciones. Aparece la ficha Ejecutar de Propiedades de proyecto.
- 2 Pulse Nueva para crear una configuración. Se muestra el cuadro de diálogo Propiedades de configuración de ejecución.



- 3 Introduzca el nombre de la configuración en el campo Nombre.
- 4 Elija el tipo de generación que deseé ejecutar antes de iniciar. Si desea obtener más información, consulte “Tipos de generación” en el capítulo “Definición de las configuraciones de ejecución” de *Creación de aplicaciones con JBuilder*.
- 5 Seleccione Servidor en la lista desplegable Tipo.
- 6 Elija el servicio JSP/Servlet en la lista Categoría de la parte izquierda del cuadro de diálogo.

- 7 Con el fin de elegir el servlet o la JSP que se desea iniciar, pulse el botón puntos suspensivos a la derecha del campo URI de inicio. Se presenta el cuadro de diálogo Escriba o seleccione el URI que se va a lanzar. El servlet o JSP se ejecuta a través de un URI, de modo que la atención cambie a la vista de web al utilizar el comando Ejecutar web.



Se puede introducir directamente el URI (Universal Resource Identifier - Identificador uniforme de recursos) en el campo de texto URI en la parte superior del cuadro de diálogo o seleccionar la WebApp y el URI de los árboles en la parte inferior del cuadro de diálogo:

- a** Seleccione la WebApp del URI que quiere ejecutar de la lista desplegable WebApp. La lista muestra todas las WebApps definidas en el proyecto.
- b** Seleccione el URI directamente de uno de los tres árboles que se encuentran en la parte inferior del cuadro de diálogo URI de inicio. Estos árboles corresponden a los distintos tipos de correspondencias de servlets. El árbol Archivo, o contenido web, en la izquierda, crea los URI que o bien coincidirían probablemente con las correspondencias de extensiones (como *.jsp) o bien no coinciden con nada y son atendidas por el servlet por defecto. El árbol Correspondencia de servlets contiene en el medio patrones de URL para coincidencias exactas. El árbol de Clases de servlet, que se muestra sólo si el servidor tiene un invocador de servlets, crea URI que coincidan con el invocador de servlets. (Tomcat 3.3 y 4.0 cuentan con un invocador de servlets, pero Tomcat 4.1 no.)

Para obtener más información sobre los puntos de interrupción, consulte “[Cómo las URL ejecutan los servlets](#)” en la página 10-11.

Tabla 10.1 Árboles URI

Nombre de árbol	Descripción	Ejemplo del URI resultante
Archivo (HTML, JSP, etc.)	Todos los archivo de tipo HTML en la WebApp seleccionada.	/selectedwebapp/hello.html o bien /selectedwebapp/login.jsp
Correspondencia de servlets	Todos los valores de patrón de URL que no contienen comodines. Permite que un usuario llame a un servlet o a una JSP por su nombre. Este valor fue introducido en los campos Nombre/Modelo de URL en el Asistente para servlets — ficha Introduzca los detalles de webapp.	/selectedwebapp/form o bien /selectedwebapp/table
Clase del servlet	Todas las clases en el proyecto abierto; no obstante, se asume que las clases presentadas son servlets y se ejecutan utilizando el invocador de servlets del servidor Web.	/selectedwebapp/servlet/com.test.MyServlet

El URI se añade al `hostname:port` durante la ejecución, por ejemplo:

```
http://localhost:8080/selectedwebapp/hello.html
http://localhost:8080/selectedwebapp/login.jsp
http://localhost:8080/selectedwebapp/form
http://localhost:8080/selectedwebapp/table
http://localhost:8080/selectedwebapp/servlet/com.test.MyServlet
```

- 8 Pulse Aceptar para cerrar el cuadro de diálogo Escriba o seleccione el URI que se va a lanzar.
- 9 Introduzca los parámetros de usuario en el campo Cadena de consulta, del cuadro de diálogo Propiedades de configuración de ejecución. Los parámetros de usuario son una serie de parejas nombre/valor separadas por un símbolo de unión, por ejemplo, `a=1&b=2`. Puede introducir también una cadena de consulta si el cliente está utilizando el método `doGet()` para leer información del servlet o la JSP. Esta cadena se añade a cualquier URL generado para una ejecución de web y, habitualmente, incluye parámetros para el servlet o la JSP. Por ejemplo, su aplicación web puede contener tanto un servlet (`SalesHistory.java`, dado el nombre del servlet `saleshistory`) como una JSP (`showproduct.jsp`) que utilicen un número de ID de producto. Cuando se ejecuta la aplicación web, el comando Ejecutar web genera las siguientes URL:

```
http://localhost:8080/showproduct.jsp
http://localhost:8080/saleshistory
```

Si se especifica “`product=1234`” en el campo Cadena de consulta, la consulta se aplicará al final de la URL:

<http://localhost:8080/showproduct.jsp?product=1234>
<http://localhost:8080/saleshistory?product=1234>

El signo de interrogación (?) separa el nombre de la JSP o del servlet a ejecutar de la cadena de consulta. Esto permite al servlet y a la JSP pedir el parámetro product y devolver 1234. Tenga en cuenta que los valores de parámetro son siempre cadenas. Los diferentes parámetros se separan con el signo (&), p.e. producto=1234&cliente=6789.

- 10 En el campo Nombre de host, escriba el nombre que el servidor web debe tener. No escoja un nombre que ya esté en uso en su sub-red. localhost es el nombre por defecto.
- 11 Introduzca el número de puerto al que debe atender el servidor web en el campo Número de puerto. Por lo general, se debe utilizar el número de puerto por defecto, 8080. No cambie este valor a no ser que el valor por defecto ya se esté usando.
- 12 Seleccione la opción Buscar puerto disponible con el fin de indicar a JBuilder que elija otro puerto si el especificado no está disponible. Resulta útil seleccionar esta opción cuando se ejecuta más de un servlet o una JSP, pues de otra forma se podría recibir un mensaje indicando que el puerto está ocupado. También es útil activar esta opción en el caso de que un problema de usuario provoque la caída del servidor web. Con esta opción seleccionada, está protegido en el caso de que el servidor web no se cierre de manera adecuada. Esta opción se usa en conjunción con las opciones de inicio de la ficha Opciones del IDE cuando el puerto especificado está siendo utilizado por un proceso ajeno a la web. (Consulte “[Configuración del IDE para ejecutar/depurar web](#)” en la página 9-6 para obtener más información.)

Para crear una configuración de ejecución con opciones de depuración:

- 1 Pulse la pestaña Depurar del cuadro de diálogo Propiedades de configuración de ejecución.
- 2 Si desea información sobre las opciones, consulte “Configuración de opciones de depuración” en “Depuración de programas en Java” de *Creación de aplicaciones con JBuilder*.

Cuando haya terminado, pulse Aceptar para cerrar el cuadro de diálogo Propiedades de configuración de ejecución. Pulse Aceptar de nuevo para cerrar la ficha Ejecutar de Propiedades del proyecto y guardar los cambios hechos a su configuración de ejecución.

Cómo las URL ejecutan los servlets

La URL (Uniform Resource Locator - Localizador uniforme de recursos) se utiliza para ejecutar un servlet. Un URI (Uniform Resource Identifier - Identificador uniforme de recursos) es un concepto más amplio que incluye tanto a la URL como a las *vías de acceso a URI de solicitud*. La vía de acceso a URI de solicitud viene a continuación del nombre del servidor y el número de puerto optativo. Comienza con una barra.

Por ejemplo, en la URL:

`http://localhost:8080/filename.html`

`/nombre_de_archivo.html` es la vía de acceso al URI de solicitud.

En un servidor web que no gestione contenedores de servlets como IIS o Apache sin Tomcat, el manejo básico de la vía de acceso a URI de solicitud es simple. El contenido web está establecido en un directorio particular, de forma que el servidor web pueda resolver esa vía de acceso utilizando la barra invertida inicial para indicar el directorio raíz de la web. Entonces puede devolver el archivo correspondiente, si está allí.

Los contenedores de Servlets como Tomcat y WebLogic son más complejos, pero más flexibles. Estos contenedores permiten contextos y redefiniciones, de forma que su aplicación web pueda tener cualquier número de contextos denominados. Cada contexto es asignado a su propio directorio raíz.

El trabajo principal de un contenedor de servlets en lo que respecta a la evaluación de la vía de acceso a URI de solicitud consiste en ver si la primera parte de la vía coincide con un nombre de contexto. En JBuilder, estos son los nombres de las WebApps. Cuando se selecciona Ejecutar web, esos nombres se asocian con los directorios raíces de las WebApps. (Si desea obtener más información sobre las clases consulte “[La WebApp en la página 3-1](#)”.)

Si coincide, la primera parte de la vía de acceso a URI de solicitud se convierte en la vía de acceso de contexto. La parte restante de la vía de acceso, que empieza con la barra, se convierte en la *vía de acceso a la URL a asignar*. Si no hay coincidencia, la vía de acceso de contexto es una cadena vacía y la vía completa de acceso a URI de solicitud se considera la vía de acceso a la URL que se asignará.

Por ejemplo, en un proyecto con una WebApp simple denominada `myprojectwebapp`, la vía de acceso a URI de solicitud `/myprojectwebapp/subpackage/somename.jsp` se evaluaría como sigue:

- La vía de acceso de contexto sería `/myprojectwebapp`.
- La vía de acceso a la URL a asignar sería `/subpackage/somename.jsp`.

No obstante, la vía de acceso a URI de solicitud /test/subpackage/somename.jsp no contendría ninguna vía de acceso de contexto en la evaluación, dado que la única WebApp existente es myprojectwebapp. En este caso, la vía de acceso de contexto estaría vacía y la vía de acceso a la URL a asignar sería el URI completo: /test/subpackage/somename.jsp

Advierta que la configuración del contexto se hace de forma específica para el servidor. Sin embargo, la coincidencia de la vía de acceso a la URL a asignar se hace por medio de las entradas que direccionan el servlet en cada descriptor de distribución de WebApp estándar del contexto, el archivo web.xml: Cada correspondencia de servlet tiene dos partes: un patrón de URL y un nombre de servlet.

Existen tres tipos especiales de patrones de URL:

Tabla 10.2 Patrones de URL

Tipo	Descripción
Asignación de la vía de acceso	Comienza con / y acaba con /*
Asignación de extensión	Empieza con *.
Asignación por defecto	Sólo incluye /

Nota Hay tres árboles en la parte inferior del cuadro de diálogo Introducir o seleccionar el URI a ejecutar que se corresponden aproximadamente con los tres tipos diferentes de asignaciones. Para detalles más concretos acerca de cómo se utilizan estos direccionamientos en el cuadro de diálogo Escriba o seleccione el URI, consulte “[Creación de una configuración de ejecución del servidor](#)” en la página 10-5.

Las demás cadenas de patrón de URL se utilizan solamente para coincidencias exactas. Cuando se busca la coincidencia de la vía de acceso a la URL que se asigna, primero se intenta una coincidencia exacta. Por ejemplo, si la WebApp somewebapp incluye un patrón de URL /test/jspname.jsp, se utilizaría el servlet correspondiente.

Si no hay coincidencia exacta, se intenta una coincidencia de la vía de acceso, empezando por la vía de acceso más larga. En el contexto por defecto, el patrón de URL /test/* sería la primera coincidencia.

Si no hay coincidencia de la vía de acceso, entonces se intenta una coincidencia de la extensión. El patrón de URL *.jsp coincidiría en ambos casos con las dos vías de acceso a la URI de solicitud siguientes: /testwebapp/subpackage/jspname.jsp y /myprojectwebapp/anyfolder/myjsp.jsp.

Finalmente, si no hay coincidencia en la extensión, se utiliza el servlet asignado a / (el servlet por defecto).

La mayoría de los servidores web ya tienen algunas asignaciones por defecto, efectuadas de nuevo de forma específica para el servidor. Por ejemplo, *.jsp puede ser asignado a un servlet:

- 1 Tomando la vía de acceso que coincide (p.e. /sub/somename.jsp o /test/subpackage/somename.jsp).
- 2 Buscando el archivo correspondiente en relación con el directorio raíz de la web del contexto.
- 3 Convertiéndolo en un servlet si todavía no se ha hecho.
- 4 Ejecutando ese servlet.

Otro direccionamiento típico por defecto es /servlet/*, que es un *servlet invocador*. Un servlet invocador:

- 1 Toma lo que sigue a /servlet/ como un nombre de clase.
- 2 Trata de ejecutar esa clase como un servlet.

El servlet por defecto (el que está asignado al patrón de URL /):

- 1 Toma la vía de acceso a la URL a asignar (que no tiene ningún direccionamiento).
- 2 Busca el archivo correspondiente con relación al directorio raíz de la web de contexto.
- 3 Lo envía al visualizador.

Cuando se crea un servlet estándar con el Asistente para servlets, no hay la más mínima necesidad de crear una correspondencia. Por ejemplo, con el nombre myproject.MyServlet, el Asistente para servlets:

- 1 Crea un servlet con el nombre de servlet myservlet asociado con la clase del servlet myproject.MyServlet.
- 2 Crea el patrón de URL /myservlet y lo asigna al nombre de servlet myservlet.

Si se hizo eso para la WebApp test, entonces /test/myservlet podría ejecutar la clase del servlet myproject.MyServlet dentro del contexto de la WebApp test. Para obtener más información acerca de cómo el Asistente para servlets crea un direccionamiento, consulte “[Ficha Introduzca los detalles de webapp](#)” en la página 5-7.

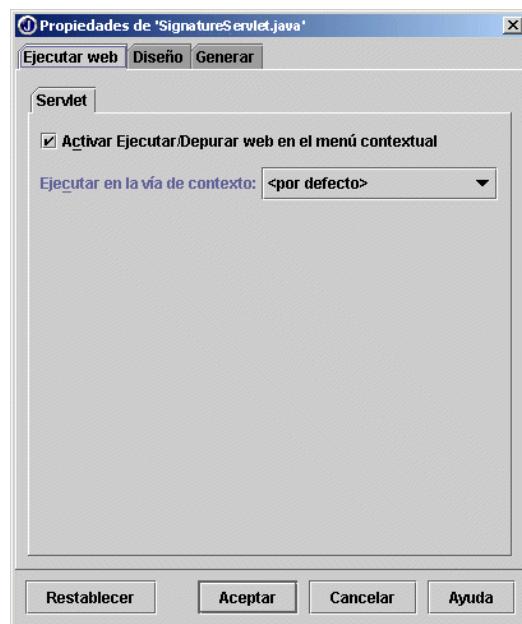
Cuando ejecuta un archivo servlet .java (o .class), se utiliza el invocador del servlet si no se encuentra un patrón de URL exacto para esa clase de servlet. En el ejemplo anterior, una ejecución bajo la test de WebApp ejecutaría /test/myservlet/myproject.MyServlet.

Configuración de propiedades de ejecución

Si ha creado un servlet mediante codificación, sin utilizar el Asistente para servlets, necesita activar los comandos Ejecutar web y Depurar web y elegir la WebApp en la cual ejecutarlo. Si se está usando una JSP o un servlet creados con los asistentes para JSP o Servlets, no se necesita seguir estos pasos. JBuilder ya ha realizado esta configuración.

Para habilitar los comandos web y designar la WebApp:

- 1 Haga clic con el botón derecho del ratón en el archivo .java en el panel del proyecto.
- 2 Seleccione Propiedades con el fin de que aparezca el cuadro de diálogo homónimo.

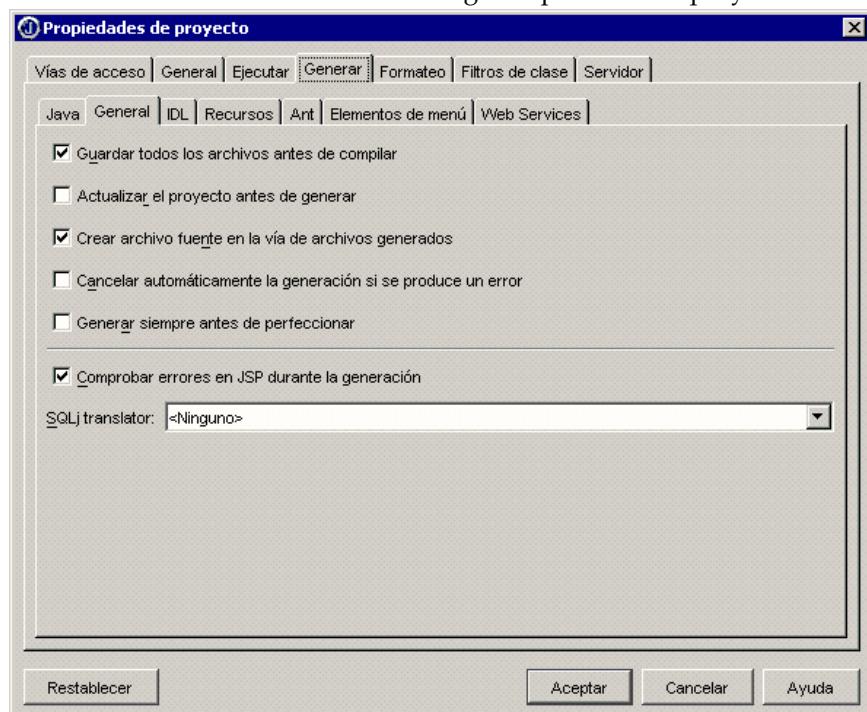


- 3 Seleccione la opción Activar Ejecutar/Depurar web en el menú contextual, de la pestaña Servlet de la ficha Ejecutar web. Esta opción muestra los comandos Ejecutar web y Depurar web cuando se hace clic con el botón derecho en el archivo de servlet en el panel del proyecto. (Si se ha creado el archivo utilizando los asistentes para servlet o para JSP, esta opción está activada por defecto.)
- 4 Seleccione la WebApp a ejecutar de la lista desplegable Ejecutar en la vía de contexto. Esta es la vía de contexto que se utiliza al ejecutar el comando Ejecutar web. Se configura en el Asistente para aplicaciones web.
- 5 Haga clic en Aceptar para cerrar el cuadro de diálogo.

Compilación de servlets o JSP

Al igual que con cualquier programa Java, antes de ejecutarse, debe ser compilado. Se puede compilar el proyecto completo con los comandos Proyecto | Ejecutar Make o Proyecto | Generar de nuevo. También se puede compilar un archivo individual de servlet o JSP si hace clic con el botón derecho sobre el archivo, en el panel del proyecto, y selecciona Limpiar, Ejecutar Make o Generar de nuevo. (Tenga en cuenta que para las JSP, sólo estará disponible el comando Ejecutar Make.) Los errores del compilador se muestran en la pestaña Generar del panel de mensajes. Si desea obtener más información acerca del compilador, consulte el capítulo “Compilación de programas en Java” de *Creación de aplicaciones con JBuilder*.

Las JSP son una extensión de la Servlet API y se compilan con los servlets antes de utilizarse. Esto requiere que el proceso de compilación convierta los nombres de los archivos JSP y los números de línea en sus equivalentes Java. Las JSP pueden compilarse durante la generación. Si desea activar esta característica para todas las JSP del proyecto, seleccione la opción Comprobar errores en JSP durante la generación, en la pestaña General, de la ficha Generar del cuadro de diálogo Propiedades de proyecto.



Se puede configurar esta propiedad para cada archivo JSP del proyecto, de manera que se puedan excluir algunos archivos de la compilación. Por ejemplo, las JSP que están pensadas para ser incluidas en otras JSP probablemente no se compilarán con éxito por sí mismas, por lo que deberían excluirse esos archivos.

Ejecución web de un servlet o una JSP

Si desea realizar una ejecución web de un servlet o una JSP en JBuilder, haga clic con el botón derecho sobre el archivo de servlet o la JSP, en el panel del proyecto, y seleccione Ejecutar web. El comando Ejecutar web utiliza su configuración en el servidor web seleccionado sin depurarla. Si el servlet se ejecuta desde un archivo HTML o SHTML, haga clic con el botón derecho sobre ese archivo y seleccione Ejecutar web.

Si el comando Ejecutar web o Depurar web está deshabilitado en el menú contextual para un servlet o JSP, compruebe la configuración de ejecución. Con el fin de ejecutar una aplicación web, necesita crear una configuración de ejecución con el tipo de servidor seleccionado como ejecutor actual. Dicho servidor debe admitir el servicio JSP/Servlet incluido en la ficha Servidor de Propiedades de proyecto, en el árbol del lado izquierdo del cuadro de diálogo. Para crear una configuración de ejecución, consulte [“Creación de una configuración de ejecución” en la página 10-2](#).

Nota Las applets no pueden ejecutarse o depurarse. Esto se debe a que no tienen una URL o un contexto web en el que ejecutarse. Además, las applets se ejecutan en el navegador de un cliente, en lugar de un servidor. Habitualmente, un applet se ejecuta en appletviewer, de Sun, o en AppletTestbed, de JBuilder. Para obtener más información, consulte [“El AppletTestBed de JBuilder y el appletviewer de Sun” en la página 14-22](#).

Inicio del servidor web

Cuando se selecciona Ejecutar web, JBuilder inicia su servidor web, mediante:

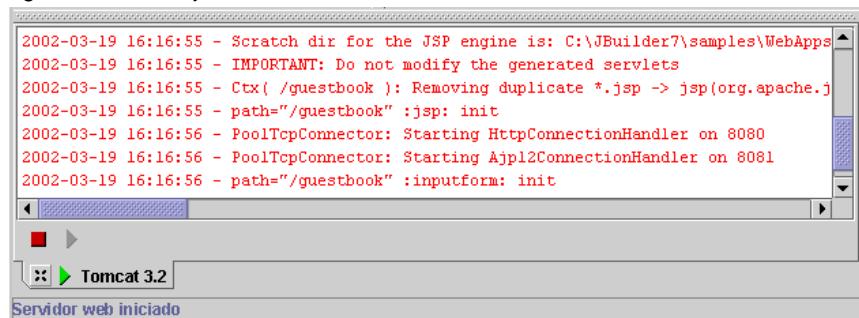
- La configuración de ejecución (consulte [“Creación de una configuración de ejecución” en la página 10-2](#) para obtener más información).
- Las opciones configuradas en la ficha Web del cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE). Consulte [“Configuración del IDE para ejecutar/depurar web” en la página 9-6](#) para obtener más información.

Los mensajes se registran en la pestaña Servidor web, mostrada en el panel de mensajes al final del Visualizador de aplicaciones. Los comandos HTTP y los valores de los parámetros se envían a este panel. El nombre de la pestaña reflejará el del servidor web, por ejemplo, Tomcat en el caso del servidor web Tomcat.

- Nota** Cuando ejecute un servlet o JSP utilizando el servidor web iPlanet, verá dos nuevas pestañas en el panel de mensajes. Una es para la fase de preparación de iPlanet y muestra la línea de comando utilizada para configurar el servidor web. La otra es para la salida del propio servidor. Ambas pestañas se llaman iPlanet.

A continuación se muestra un ejemplo de salida del panel de mensajes del servidor web Tomcat:

Figura 10.1 Mensajes Tomcat



```

2002-03-19 16:16:55 - Scratch dir for the JSP engine is: C:\JBuilder7\samples\WebApps
2002-03-19 16:16:55 - IMPORTANT: Do not modify the generated servlets
2002-03-19 16:16:55 - Ctx( /guestbook ): Removing duplicate *.jsp -> jsp(org.apache.jsp)
2002-03-19 16:16:55 - path="/guestbook" :jsp: init
2002-03-19 16:16:56 - PoolTcpConnector: Starting HttpConnectionHandler on 8080
2002-03-19 16:16:56 - PoolTcpConnector: Starting Ajp12ConnectionHandler on 8081
2002-03-19 16:16:56 - path="/guestbook" :inputform: init

```

Cuando se inicia la Ejecución web, aparecen dos nuevas pestañas en el panel de contenido: Vista web y Ver código fuente web. Haga clic en la pestaña para abrir Vista web o Ver código fuente web.

- Nota** Las pestañas Ver web y Ver código fuente web no se mostrarán si ha seleccionado la opción No usar vista web en la ficha Web del cuadro de diálogo Opciones del IDE. Para obtener más información, consulte “[Configuración del IDE para ejecutar/depurar web](#)” en la página 9-6.

Vista web

La salida con formato puede verse en el panel Vista web del panel de contenido. La URL generada aparece en el campo Ubicación, en la parte superior de la ficha Vista web.

La Vista web muestra el servlet después de que haya sido procesado por el motor de servlets. Puede haber algún retraso entre el momento en que se modifica el archivo de servlet o JSP y el momento en que se muestra ese cambio en Vista web. Con el fin de ver los cambios más recientes, pulse el botón Actualizar, en la parte superior de la Vista web.

La Vista web es simplemente un navegador y no tiene relación con otras funciones de JBuilder. Por ejemplo, puede ver su salida formateada en un navegador externo a JBuilder. Las opciones de la ficha Web del cuadro de

diálogo Opciones del IDE agiliza el uso reiterado de esta función. Consulte “[Configuración del IDE para ejecutar/depurar web](#)” en la página 9-6.

Figura 10.2 Salida de Ver web



Ver código fuente web

La salida en bruto del servidor web aparece en el panel Ver código fuente web.

Figura 10.3 Ver código fuente web

The screenshot shows a code editor window titled 'FormServlet'. The code editor displays the generated HTML code for the 'Sign the guestbook' form:

```
<h1>Sign the guestbook</h1>
<strong>Enter your name and comment in the input fields below.</strong>
<br><br>
<form action=guestbookservlet.DBServlet method=POST>
Name:  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<input type=text name=UserName value= &nbsp; size=20 maxlength=150>
<br><br>
Comment:&nbsp;
<input type=text name=UserComment value= &nbsp; size=50 maxlength=150><br><br><br>
Save in guestbook:&nbsp;&nbsp;&nbsp;&nbsp;
<input type=submit value=Submit></form>
```

The code editor has tabs for 'Fuente', 'Ver web', 'Ver código fuente web', 'Diseño', 'Bean', 'Doc', and 'Histórico'. The 'Ver código fuente web' tab is currently selected. The status bar at the bottom shows the file name 'FormServlet.java', line number '1:1', and options 'Sólo lectura' and 'Insertar'.

Detención del servidor web

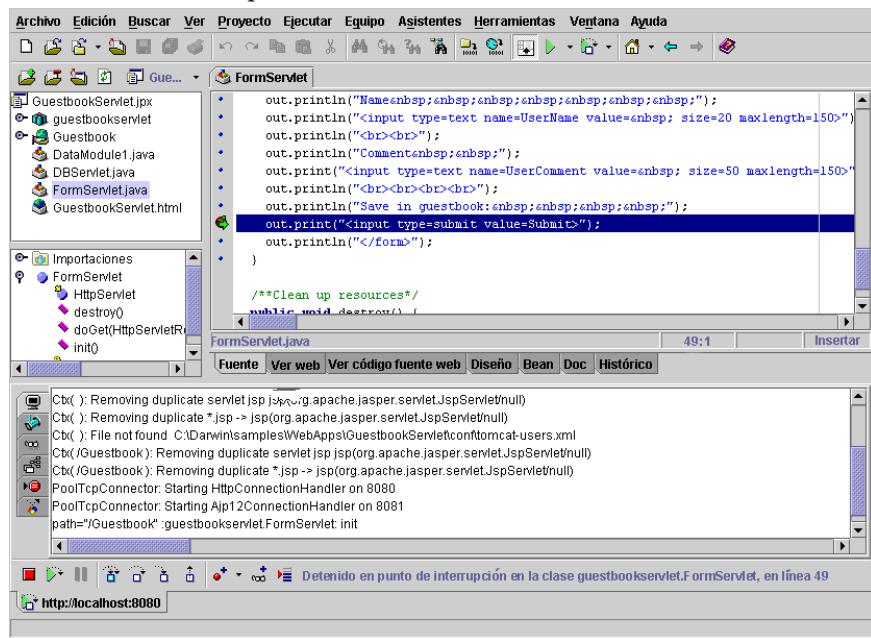
- Si desea detener el servidor web, pulse el botón Terminar el programa en la pestaña del servidor web. Si desea iniciar el servidor web de nuevo y volver a ejecutar su aplicación web, pulse el botón Reiniciar el programa.
 - ▶ Cuando se hagan cambios en el código fuente, se recompile y se vuelva a ejecutar, lo más probable será que se sigan estos pasos. No se necesita cerrar el panel de servidor Web cada vez que se inicia el servidor Web.
- Nota** La primera vez que se pulsa el botón Terminar el programa, simplemente envía un comando al servidor para que se apague. Más concretamente, si se intenta depurar los manejadores de sucesos del ciclo de vida de la webapp, se realiza una llamada a los métodos `Servlet.destroy()` y `ServletContextListener.contextDestroyed()`. Si no está depurando, el servidor, por lo general, se apaga por sí solo (Tomcat lo hace en unos segundos; a otros servidores les lleva más tiempo). En cualquier caso, si se pulsa el botón Terminar programa una segunda vez, el proceso que esté realizando el servidor se termina inmediatamente.

Depuración web del servlet o de la JSP

Para depurar web, inicie el servidor en modo depuración con el comando Depurar web. Podrá entonces depurar cualquier servlet o JSP que el servidor procese. Si desea que el depurador se detenga automáticamente en una línea de código específica, defina un punto de interrupción en el servlet o en la JSP.

- Nota** JBuilder proporciona la posibilidad de depuración del código fuente para las JSP. Esto permite efectuar directamente un seguimiento del código de la JSP, dado que no se necesita efectuar un seguimiento del servlet en el que se compila la JSP ni tratar de hacer coincidir los números de línea para encontrar los errores. Tenga en cuenta que Tomcat 4.1 no admite la depuración de código fuente de páginas JSP.

El comando Depurar web presenta el depurador en el panel de servidor web. Los mensajes tanto del servidor como del depurador se escriben en la vista Consola del depurador.



Nota Si desea obtener más información acerca de la depuración, consulte los siguientes temas:

- “Depuración de programas en Java” en *Creación de aplicaciones con JBuilder*
- “Tutorial de compilación, ejecución y depuración” en *Creación de aplicaciones con JBuilder*
- “Depuración remota” en *Creación de aplicaciones con JBuilder*

11

Distribución de aplicaciones web

El desarrollo web es una función de JBuilder Enterprise

La distribución de una aplicación web es el proceso de trasladar la aplicación web al servidor web, colocándola en el lugar correcto y completando cualquier otro paso necesario, con el fin de que el servidor reconozca correctamente la aplicación. Los distintos servidores web requieren pasos diferentes para la correcta distribución de la aplicación web. Deberá consultar la documentación del servidor Web para informarse sobre aspectos específicos de ese servidor.

Aspectos generales

Los siguientes apartados analizan algunos temas a tener en cuenta cuando se distribuye a servidores web.

Archivos recopilatorios

La reunión de los archivos de la aplicación Web en un recopilatorio puede simplificar la distribución. Algunos servidores requieren que una WebApp esté empaquetada en un archivo WAR (Web Archive – Recopilatorio web). Un archivo WAR organiza en la jerarquía correcta los archivos que necesita para su aplicación web. Entonces se puede copiar el archivo recopilatorio en la ubicación apropiada del servidor web. Esto elimina la necesidad de copiar cada archivo individualmente y asegura que son copiados en la ubicación adecuada.

Un archivo WAR es un recopilatorio web. Contiene la aplicación web completa en una estructura apropiada, lo que facilita la copia de la estructura en el servidor web. Los archivos WAR se estudian con más profundidad en el [Capítulo 3, “Las WebApps y los archivos WAR”](#).

Si la aplicación web contiene applets, debería colocarlos en un archivo JAR. Para obtener más información acerca del uso de los archivos JAR que contienen applets, consulte el [Capítulo 14, “Las applets”](#).

Su aplicación web, archivo WAR o archivo JAR también pueden empaquetarse en un archivo EAR. Consulte la ayuda en línea del “Asistente para EA”.

Descriptores de distribución

Los descriptores de distribución son archivos XML que contienen la información relativa a la WebApp necesaria para el servidor web.

Probablemente se utilizarán uno o más descriptores de distribución si la aplicación web contiene servlets o JSP. Compruebe la documentación de su servidor para obtener más información acerca de qué descriptores de distribución requiere.

Applets

Consulte [“Distribución de las applets” en la página 14-14](#) para obtener más información acerca de la distribución de un applet.

Servlets

La distribución de servlets puede ser delicada, ya que si no se hace correctamente, el servidor Web podría fallar en el reconocimiento de su servlet. Con el fin de simplificar la distribución, debería incluir el servlet en un archivo WAR. Esto hace posible reunir todos los archivos y recursos necesarios para el servlet en una jerarquía correctamente organizada antes de la distribución. Entonces todo lo que necesita es distribuir el archivo WAR en el servidor web.

Al margen del servidor web, la distribución satisfactoria del servlet requiere que cierta información esté presente en el descriptor de distribución `web.xml`. Su servidor web podría tener también exigencias adicionales. Como mínimo, antes de distribuir un servlet estándar, necesita especificar lo siguiente en un elemento `servlet` del archivo `web.xml`:

- `servlet-name` - un nombre para el servlet
- `servlet-class` - el nombre completo de la clase del servlet

Cada servlet estándar debe especificar también una asignación de servlets en el web.xml. En un elemento de asignación de servlets tiene que especificar lo siguiente:

- servlet-name - el nombre utilizado en la etiqueta del servlet
- url-pattern - el patrón de URL al cual está direccionado el servlet

Los servlets de filtro y los servlets de monitor requieren diferentes etiquetas. Consulte los apartados “[Ficha Filtros](#)” en la página 12-5 y “[Ficha Monitores](#)” en la página 12-8 para obtener más información sobre las etiquetas requeridas para estos tipos de servlets.

Si para generar el servlet utiliza el Asistente para servlets de JBuilder, el asistente insertará la información requerida por el servlet en el web.xml. Esto es verdad para un servlet estándar, uno de filtro o uno monitor.

Los servlets deben compilarse antes de ser distribuidos en el servidor web.

JSP

Las JSP se distribuyen más fácilmente que los servlets. Debería incluirlas en un archivo WAR para facilitar aún más la distribución.

Las JSP son asignadas por un servidor web de la misma manera que los archivos HTML; el servidor reconoce la extensión del archivo. El servidor web también gestiona la compilación. Recuerde, el servidor web compila las JSP en servlets antes de su ejecución.

Algunas JSP utilizan bibliotecas de etiquetas JSP. Esta bibliotecas también deben distribuirse en el servidor web, que necesita saber cómo encontrarlas. Por esta razón, si tiene una JSP que utilice una biblioteca de etiquetas, el descriptor de distribución web.xml necesitará un elemento taglib que indique qué biblioteca de etiquetas hay que utilizar y dónde puede encontrarse. La siguiente información se encuentra en el elemento taglib:

- taglib-uri - el URI usado en la JSP para identificar la biblioteca de etiquetas. Coincide con la directiva taglib (<%@ taglib uri="whatever" prefix="whatever" %>) de la JSP.
- taglib-location - la ubicación real de la biblioteca de etiquetas.

Si utiliza el Asistente para JSP de JBuilder con el fin de crear una JSP que utilice una de las bibliotecas de etiquetas reconocidas por el asistente, la información de la biblioteca de etiquetas se añade al archivo web.xml.

Comprobación de su aplicación web

Después de haber distribuido la aplicación web en el servidor web, debe comprobarla para asegurarse de que está correctamente distribuida. Deberá intentar acceder a todas las páginas, servlets, JSP y applets de la aplicación y asegurarse de que funcionan como se espera. Esto debe hacerse desde un visualizador en otro equipo, de forma que pueda asegurarse de que la aplicación web es accesible desde la web y no sólo localmente. También debería efectuar la comprobación con diferentes tipos de visualizadores, ya que podría variar la forma en que aparece la aplicación en visualizadores diferentes, especialmente cuando se utilizan applets.

Modificación del descriptor de distribución

Los descriptores de distribución son archivos XML que contienen la información sobre la WebApp necesaria para el servidor web. Todos los servidores web aptos para Java esperan un descriptor de distribución estándar llamado `web.xml`.

Si utiliza el marco de trabajo Struts en la WebApp, se requiere el descriptor de distribución de nombre `struts-config.xml`. Algunos otros marcos de trabajo requieren sus propios descriptores de distribución.

Algunos servidores pueden tener también descriptores de distribución específicos de fabricante, que utilizan adicionalmente a `web.xml`. Por ejemplo, BEA WebLogic utiliza `weblogic.xml`. Consulte la documentación de su servidor e infórmese de los archivos descriptores que utiliza. Tomcat, el servidor web que viene con JBuilder, necesita solamente `web.xml`.

JBuilder proporciona un editor de descriptor de distribución para `web.xml`. Se llama Editor DD para WebApp. Este proporciona una interfaz gráfica de usuario para editar la información más comúnmente utilizada en el archivo `web.xml`.

Cuando se abre el archivo `web.xml` en el IDE de JBuilder, su contenido se presenta en el panel de estructura y el Visualizador de aplicaciones muestra el Editor DD de WebApp y el editor de código fuente. El archivo `web.xml` se puede modificar en el Editor DD de WebApp o en el editor de código fuente. Los cambios efectuados en el Editor DD de WebApp se reflejarán en el código fuente y los cambios en el código fuente se reflejarán en el WebApp DD Editor.

Advertencia

Si introduce comentarios en el archivo `web.xml` utilizando el editor original, se borrarán si posteriormente abre el archivo en el Editor DD de WebApp.

JBuilder también proporciona el Editor de configuración de Struts para modificar el descriptor de distribución `struts-config.xml` que requiere el marco de trabajo Struts.

Edición de descriptores de distribución específicos de un fabricante

También se pueden editar los descriptores de distribución específicos de un fabricante en el IDE de JBuilder. JBuilder reconoce y muestra el archivo descriptor de distribución en el proyecto si el correspondiente plugin del servidor está configurado apropiadamente.

Los fabricantes que ofrecen plugins para otros servidores web se empeñan en suministrar editores de interfaz gráfica de usuario para sus descriptores de distribución específicos. Por ejemplo, el plug-in Sybase proporciona un editor GUI para el archivo `sybase-easerver_config.html`.

Si el fabricante no proporciona un editor de interfaz gráfica de usuario para sus descriptores de distribución, existe la posibilidad de editar el código fuente de los descriptores de distribución con el editor de código fuente XML de JBuilder. Para hacerlo, abra el descriptor de distribución específico del fabricante y pulse la pestaña Fuente en el panel de contenido.

Cuando se abre un descriptor de distribución específico de un fabricante en el IDE de JBuilder, su contenido se muestra en el panel de estructura y el Visualizador de aplicaciones presenta el editor de código fuente y la vista Histórico, junto con cualquier editor de interfaz gráfica de usuario específica del fabricante que pudiera aportar el plugin del servidor web.

Más información sobre descriptores de distribución

Si desea obtener más información sobre los descriptores de distribución y sobre el descriptor de distribución `web.xml` en particular, consulte la Java Servlet Specification, que puede descargarse desde <http://java.sun.com/aboutJava/communityprocess/first/jsr053/index.html>.

12

Modificación del archivo web.xml

El Editor DD de WebApp está activo cuando se abre el archivo `web.xml` en el panel de contenido. Al mismo tiempo, el panel de estructura mostrará un esquema del contenido del archivo. Pulsando sobre los diferentes nodos dentro del panel de estructura se muestran las diferentes fichas del editor. Hay 15 nodos principales y algunos de ellos tienen nodos dependientes. Los nodos principales son:

- Descriptor de distribución para la WebApp
- Parámetros de contexto
- Filtros (especificación de Servlet 2.3)
- Monitores (especificación de Servlet 2.3)
- Servlets
- Bibliotecas de etiquetas
- Tipos de MIME
- Páginas de error
- Entradas de entorno
- Referencias EJB
- Referencias EJB locales (especificación de Servlet 2.3)
- Referencias de la factoría de conexión del gestor de recursos
- Referencias de variables del entorno (especificación de Servlet 2.3)
- Conexión
- Seguridad

Cada uno de estos nodos contiene etiquetas que se pueden modificar en el Editor DD de WebApp. El Editor DD de WebApp cubre todas las etiquetas de descriptores de distribución del `web.xml` en la especificación de Servlet 2.3. También puede modificar el código fuente del archivo `web.xml`.

- Advertencia** Si introduce comentarios en el archivo `web.xml` utilizando el editor original, se borrarán si posteriormente abre el archivo en el Editor DD de WebApp. Las etiquetas contenidas en cada uno de los nodos del Editor DD de WebApp se explican en las siguientes secciones.

Menú contextual del Editor DD de WebApp

Si hace clic con el botón derecho en cualquiera de los nodos principales del Editor DD de WebApp aparece un menú contextual que permite la adición de un nodo de filtro, un nodo de servlet o un nodo de restricción de seguridad. La adición de un nodo de filtro está disponible sólo si el servidor web admite la especificación de Servlet 2.3, ya que los servlets de filtro son nuevos para esta versión de especificación de servlet.

Si hace clic con el botón derecho en un nodo de restricción de seguridad, aparece un menú contextual que permite la adición de un nodo de colección de recursos web a ese o a otro nodo de restricción de seguridad. Debe haber al menos un nodo de restricción de seguridad antes de que pueda añadirse un nodo de colección de recursos web.

El menú contextual de un nodo de servlet, de filtro o de colección de recursos web también contiene opciones para renombrar o eliminar el nodo actual. El menú contextual de un nodo de restricción de seguridad existente, contiene la opción de borrar el nodo actual, y cuando se modifica un archivo Servlet 2.3 `web.xml` también contiene la opción de renombrar la restricción de seguridad. Renombrar o borrar cualquier nodo extiende la modificación en cascada a todas las partes relacionadas del archivo `web.xml`.

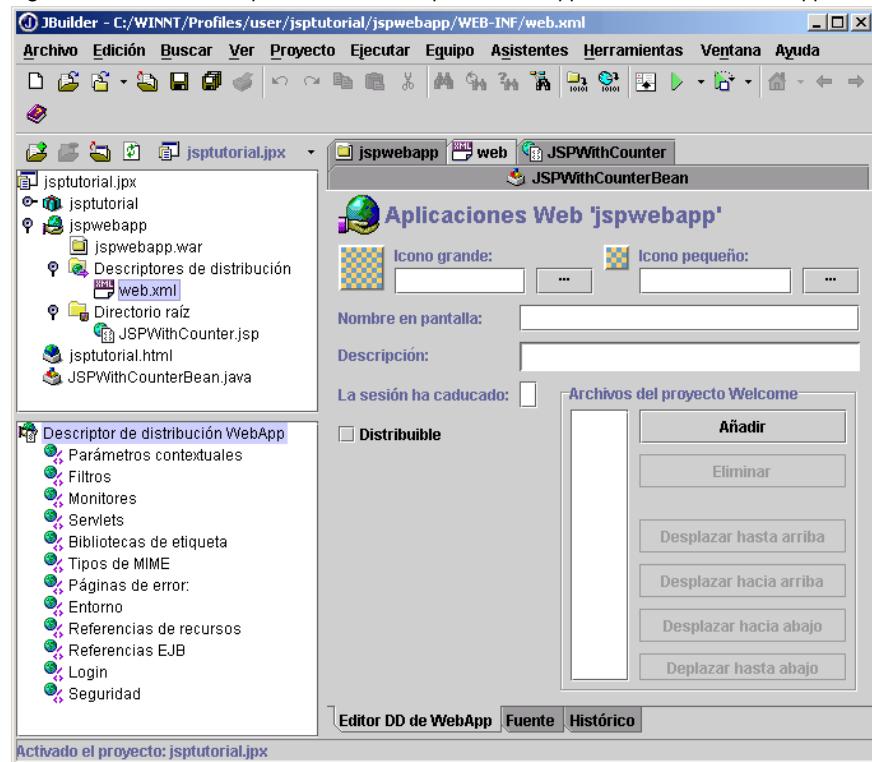
Ficha Descriptor de distribución para la WebApp

La ficha principal del Editor DD de WebApp contiene información de identificación básica de la WebApp. La información que se puede editar en esta ficha es:

Tabla 12.1 Ficha Descriptor de distribución para la WebApp del Editor DD de WebApp

Elementos	Descripción
Icono grande	Señala la ubicación de un ícono grande para la WebApp (32 x 32 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Icono pequeño	Señala la ubicación de un ícono pequeño para la WebApp (16 x 16 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Nombre en pantalla	Nombre que se muestra. Las herramientas de administración de motores servlet pueden utilizarlo.
Descripción	Descripción de la WebApp. Las herramientas de administración de motores servlet pueden utilizarlo.
Duración de la sesión	Número entero de minutos que pueden pasar antes de que termine la sesión.
Distribuible	Si la aplicación es distribuible en un contenedor de servlet distribuido (multi MV).
Archivos del proyecto Welcome	El archivo o archivos que van a ser mostrados cuando la URL apunte hacia un directorio, por ejemplo: <code>index.html</code>

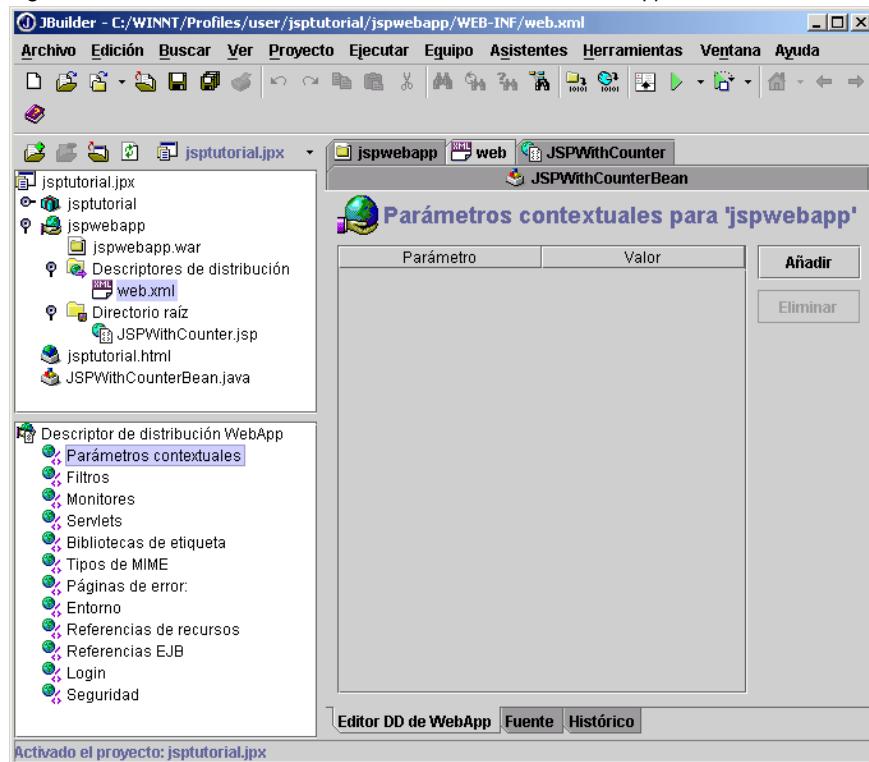
Figura 12.1 Ficha Descriptor de distribución para la WebApp del Editor DD de WebApp



Ficha Parámetros contextuales

La ficha Parámetros contextuales contiene una rejilla de parámetros de inicialización para todo el `ServletContext` de la WebApp y los valores de esos parámetros. A estos parámetros se accede mediante los métodos `ServletContext.getInitParameterNames` y `ServletContext.getInitParameter`.

Figura 12.2 Ficha Parámetros contextuales del Editor DD de WebApp



Ficha Filtros

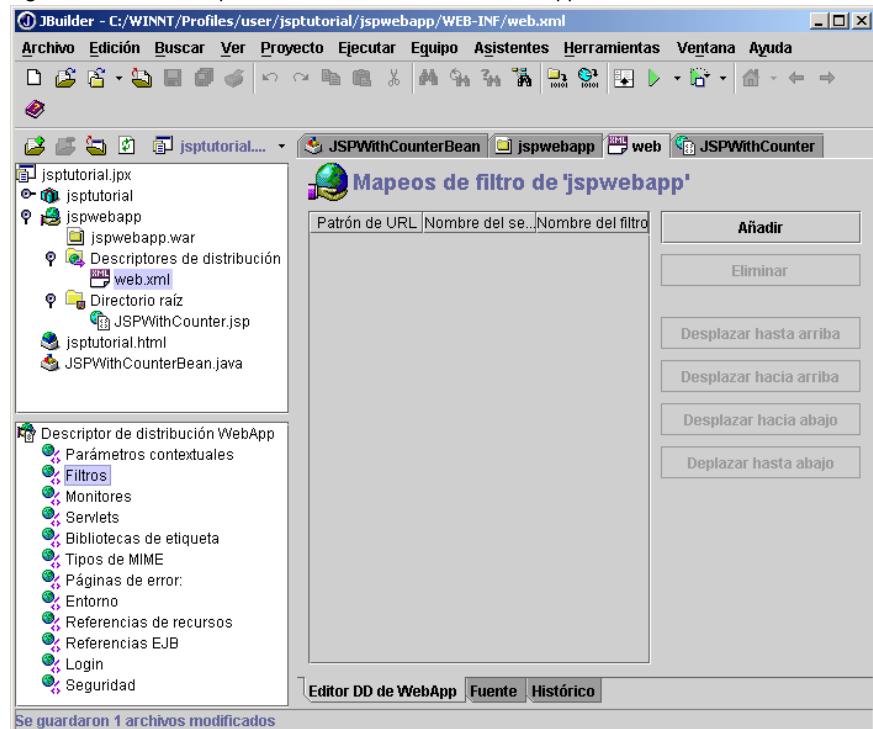
La ficha Filtros será visible solamente si su servidor web admite la especificación de Servlet 2.3. Esta ficha contiene una rejilla para asignar los filtros (por `filter-name`) bien a un patrón de URL o a un nombre de servlet (pero no a ambos). El orden de los filtros es importante ya que es el orden en el cual se aplicarán. Esta ficha permite cambiar el orden en el cual se aplican los filtros. La información que se presenta en la ficha de filtros es:

Tabla 12.2 Ficha Filtros del Editor DD para Webapp

Elementos	Descripción
Patrón de URL	El patrón de url para la ubicación del filtro. Se necesita bien éste o el nombre de servlet cuando se distribuye un servlet de filtro.
Nombre de Servlet	El nombre de servlet que se utiliza para asignar el filtro. Se necesita bien éste o el patrón de url cuando se distribuye un servlet de filtro.
Nombre de Filtro	El nombre de filtro que se utiliza para asignar el filtro. Se requiere cuando se distribuye un servlet de filtro.

Si para crear un servlet de filtro utiliza el Asistente para servlets de JBuilder, el asistente añadirá automáticamente la asignación de filtro necesaria.

Figura 12.3 Ficha Mapeos de filtro del Editor DD de Webapp



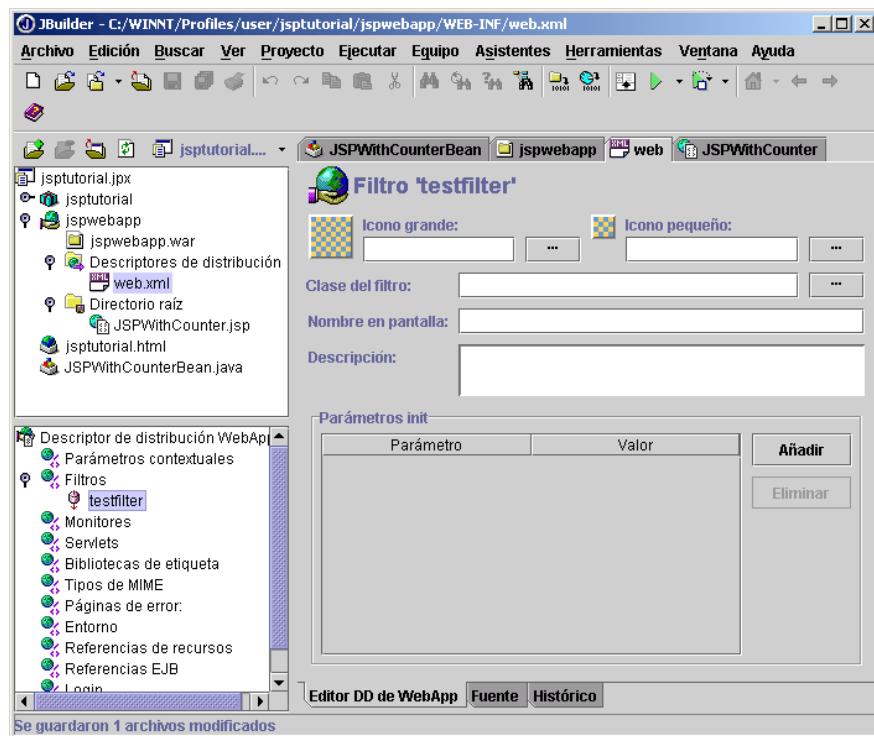
Cada filtro individual se lista en el panel de estructura como un nodo dependiente separado del nodo Filtros. El nombre de filtro del filtro se presenta en el árbol. Se puede renombrar o eliminar un filtro si hace clic con el botón derecho en el nodo de un filtro determinado y selecciona Renombrar o Eliminar en el menú contextual. Si se renombra o se elimina un filtro, este cambio se extenderá en cascada a todas las partes relacionadas del descriptor de distribución.

Cuando se abre un filtro individual, el Editor DD de WebApp muestra una ficha para ese filtro específico. Esta ficha contiene la siguiente información de identificación para el filtro:

Tabla 12.3 Ficha Filtro individual del Editor DD para Webapp

Elementos	Descripción
Icono grande	Indica la ubicación de un ícono grande para el filtro (32 x 32 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Icono pequeño	Indica la ubicación de un ícono pequeño para el filtro (16 x 16 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Clase del filtro	Nombre completo de la clase del filtro. Esta información es necesaria cuando se distribuye un servlet filtro.
Nombre en pantalla	Nombre que se presenta para el filtro. Las herramientas de administración de motores servlet pueden utilizarlo.
Descripción	Descripción del filtro. Las herramientas de administración de motores servlet pueden utilizarlo.
Parámetros init	Parámetros de inicialización del filtro.

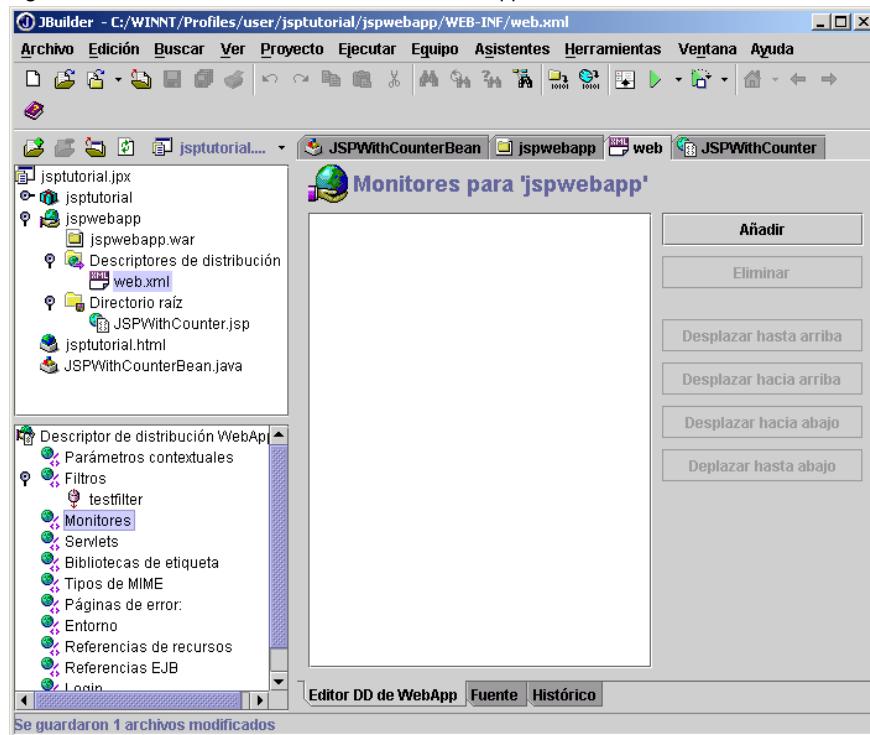
Figura 12.4 Nodo de filtro individual en el Editor DD de Webapp



Ficha Monitores

La ficha Monitores sólo será visible si el servidor web admite la especificación de Servlet 2.3. La ficha Monitores tiene un cuadro de lista de las clases de monitores de una aplicación web. El orden de los monitores es importante ya que es el orden en el cual se aplicarán. La ficha Monitores permite cambiar el orden en el cual se aplican. La información de la ficha Monitores es necesaria cuando se distribuye un servlet monitor. Si para crear un servlet monitor utiliza el Asistente para servlets de JBuilder, la clase de servlet se añadirá a la lista automáticamente.

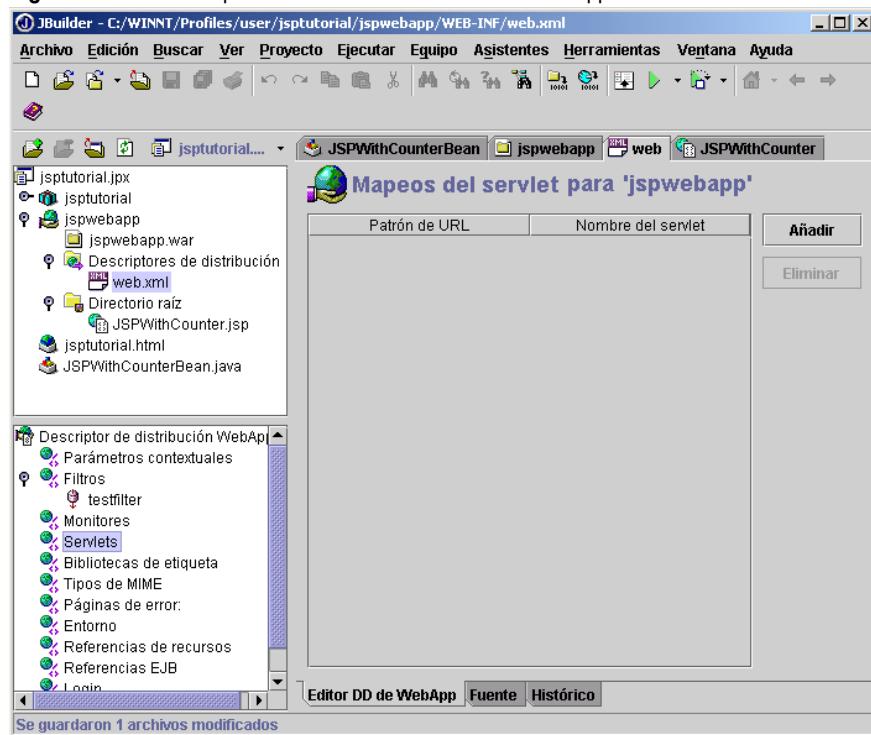
Figura 12.5 Ficha Monitores del Editor DD de Webapp



Ficha Servlets

La ficha Servlets tiene una rejilla que mapea los patrones de URL hacia un nombre de servlet. Un servlet puede estar asignado a más de un patrón de URL. Se recomienda al menos una asignación por cada servlet. Si para crear un servlet estándar utiliza el Asistente para servlets de JBuilder, se creará la asignación automáticamente.

Figura 12.6 Ficha Mapeos del servlet del Editor DD de Webapp



Si se han definido servlets, se encontrarán nodos dependientes que representan servlets individuales debajo del nodo de la ficha Servlets en el panel de estructura. El nombre de servlet del servlet se muestra en el árbol. Se puede renombrar o eliminar un servlet si hace clic con el botón derecho en el nodo del servlet individual y selecciona Renombrar o Eliminar en el menú contextual. Si se renombra o se elimina un servlet, este cambio se extenderá en cascada a todas las partes relacionadas del descriptor de distribución. Por ejemplo, la eliminación de un servlet también produce la eliminación de sus asignaciones de URL y de los filtros.

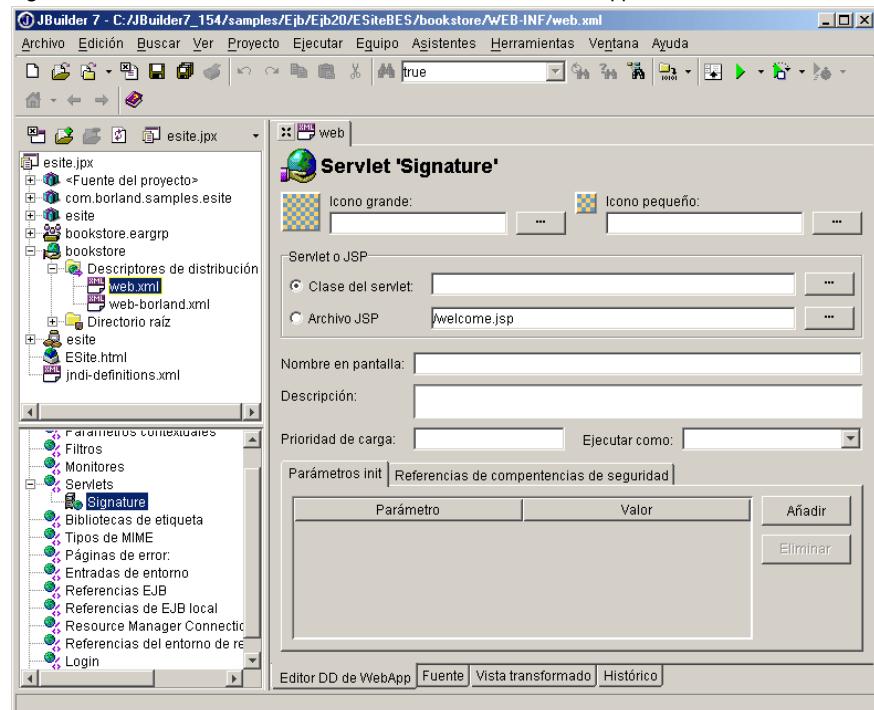
Recuerde que también se pueden asignar patrones de URL a JSP. Esto significa que un nodo de un servlet individual en el panel de estructura puede representar una JSP o un servlet.

Cuando se abre un nodo de un servlet individual, el Editor DD de WebApp presenta una ficha para ese servlet específico. Esta ficha contiene información de identificación del servlet:

Tabla 12.4 Ficha Servlet individual del Editor DD de Webapp

Elementos	Descripción
Icono grande	Señala la ubicación de un ícono grande para el servlet (32 x 32 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Icono pequeño	Señala la ubicación de un ícono pequeño para el servlet (16 x 16 píxeles), que debe estar contenido dentro del árbol de directorio de la WebApp.
Clase del servlet	Seleccione este botón de radio para un servlet. Introduzca el nombre completo de la clase del servlet en el campo de texto.
Archivo JSP	Seleccione este botón de radio para una JSP. Introduzca la vía de acceso al archivo de JSP en el campo de texto.
Nombre en pantalla	El nombre del servlet que se mostrará. Las herramientas de administración de motores servlet pueden utilizarlo.
Descripción	Descripción del servlet. Las herramientas de administración de motores servlet pueden utilizarlo.
Prioridad de carga	La prioridad de carga en el inicio para el servlet, expresada como un entero positivo. Los servlets con los enteros más bajos se cargan antes de aquellos que tienen enteros más altos.
Ejecutar como	Los servlets con especificación Servlet 2.3 se pueden Ejecutar como otro usuario, como uno de los Nombres de competencia definidos en la ficha Seguridad.
Parámetros Init	Esta ficha contiene los parámetros de inicialización del servlet.
Referencias de competencias de seguridad	Esta ficha contiene una rejilla con los cambios de nombres de competencias de seguridad. Esta rejilla se utiliza para asignar un nombre alternativo codificado en el servlet hacia los nombres de competencias reales en el descriptor de distribución.

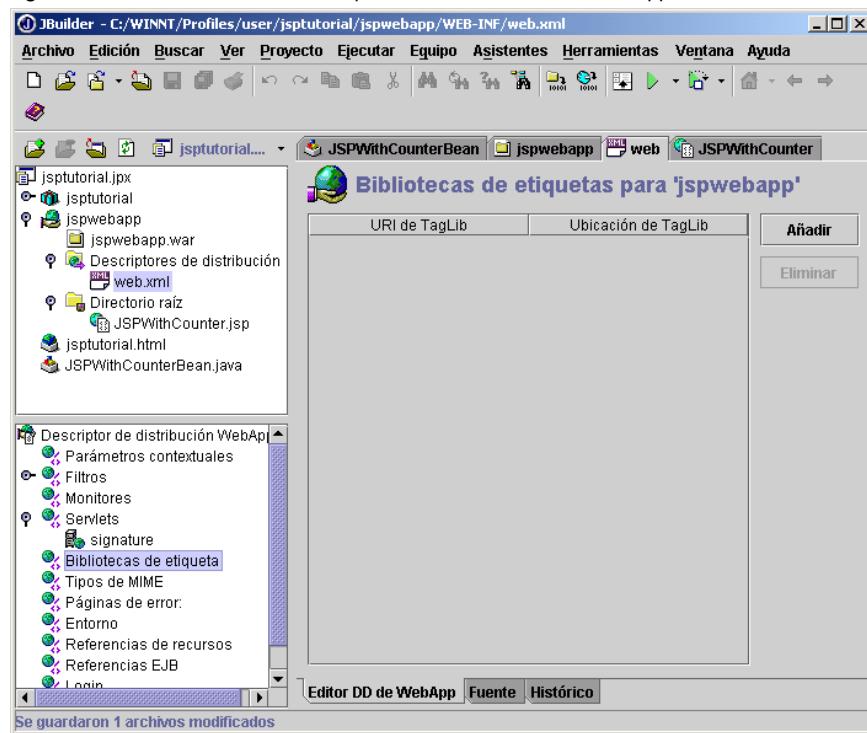
Figura 12.7 Nodo de servlet individual en el Editor DD de WebApp



Ficha Bibliotecas de etiquetas

La ficha Bibliotecas de etiquetas tiene una rejilla para asignar los URI utilizados en una JSP hacia las ubicaciones reales de los archivos de Definición de bibliotecas de etiquetas (.tld). Esta información es necesaria para la distribución de una JSP que utilice una biblioteca de etiquetas. Si utiliza el Asistente para JSP de JBuilder para crear una JSP que utilice una biblioteca de etiquetas reconocida por el asistente, se rellena la información de la biblioteca de etiquetas.

Figura 12.8 Ficha Bibliotecas de etiquetas en el Editor DD de WebApp



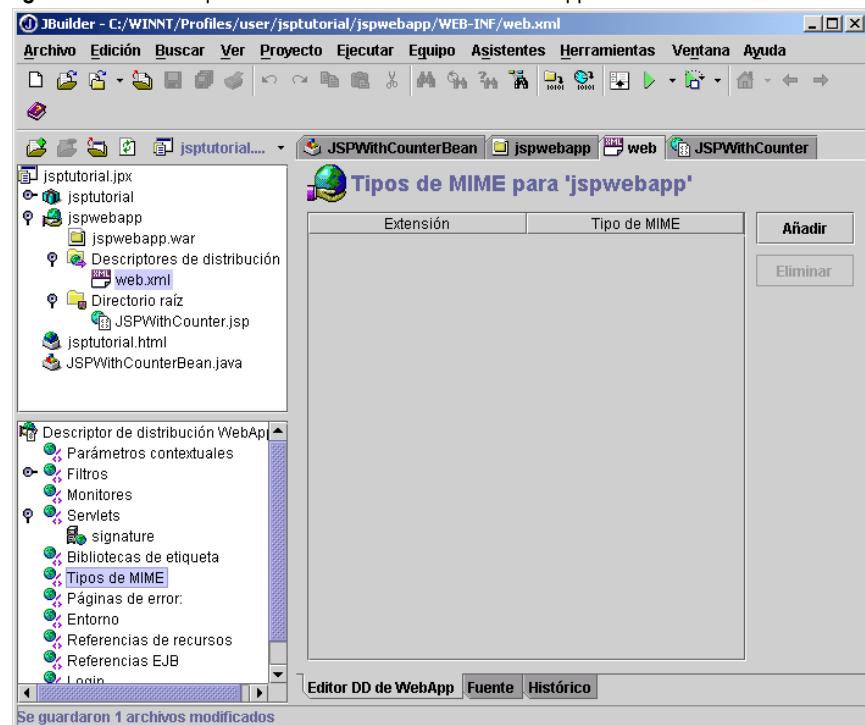
Ficha Tipo de MIME

La ficha Tipo de MIME tiene una rejilla que asigna las extensiones a nombres de tipos. Utilice la ficha Tipo de MIME para añadir tipos personalizados. He aquí algunos ejemplos de tipos de MIME incluidos con muchos servidores web:

- txt - texto/plano
- html - texto/html
- xml - texto/xml
- pdf - aplicación/pdf
- zip - aplicación/x-zip-comprimida
- jpeg - imagen/jpeg
- gif - imagen/gif

Ficha Tipo de MIME

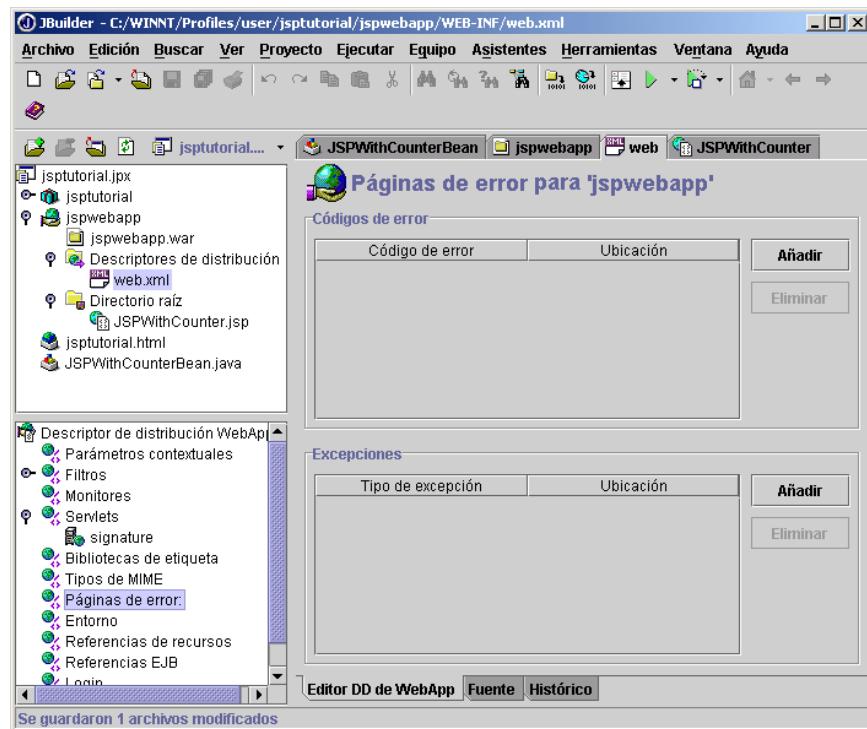
Figura 12.9 Ficha Tipos de MIME en el Editor DD de WebApp



Ficha Páginas de error

La ficha Páginas de error tiene dos rejillas, una para los números de código y una para los nombres de clase de excepción, que están direccionadas hacia las ubicaciones de las páginas que se deben mostrar en el caso de que se produzcan condiciones de error o de excepción.

Figura 12.10 Ficha Páginas de error en el Editor DD de WebApp



Ficha Entradas de entorno

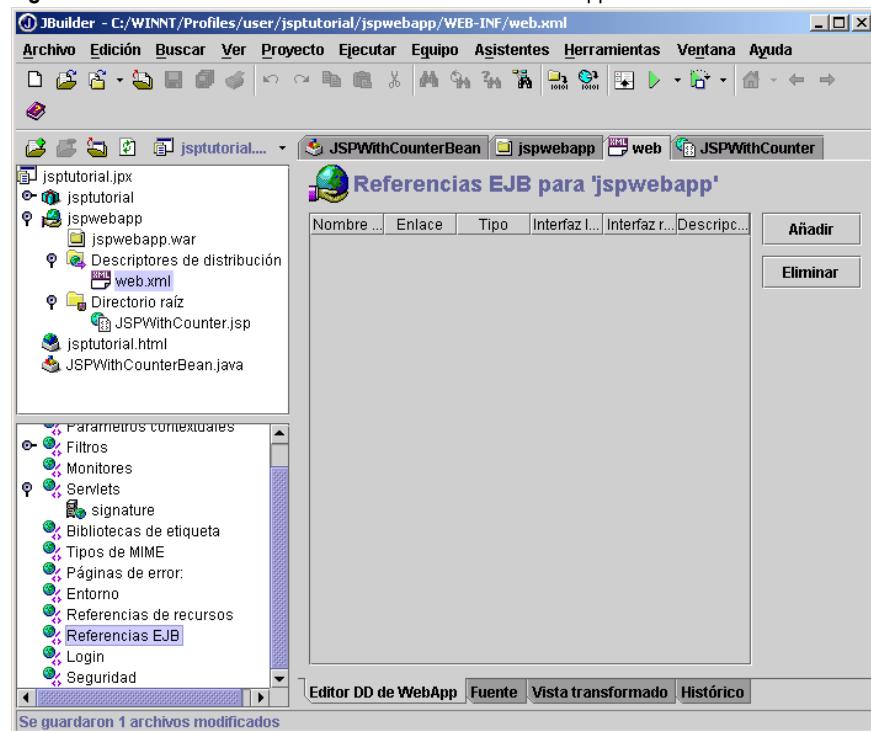
La ficha Entradas de entorno tiene una rejilla con nombres de entradas de entorno, sus valores y sus tipos, más un campo de texto descriptivo de la entrada actualmente seleccionada.

Figura 12.11 Ficha Entorno en el Editor DD de WebApp

Ficha Referencias EJB

La ficha Referencias EJB tiene una rejilla con nombres EJB, sus tipos, interfaces local y remoto, ejb-link optativo y una descripción. Esta es similar a la ficha Referencias EJB del Editor DD de EJB.

Figura 12.12 Ficha Referencias EJB en el Editor DD de WebApp



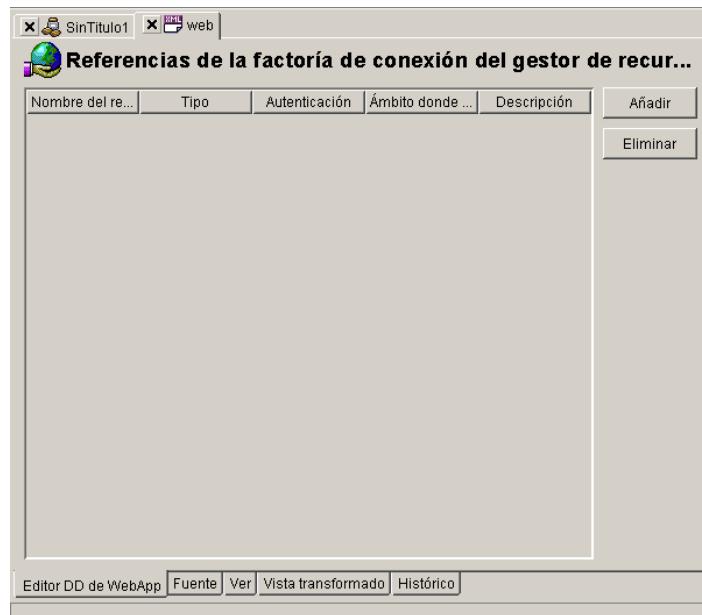
Ficha Referencias EJB locales

La ficha Referencias EJB locales está presente en el panel de estructura sólo cuando se está modificando un archivo Servlet 2.3 web.xml. Sus contenidos son similares a los de la ficha Referencias EJB.

Ficha Referencias de la factoría de conexión del gestor de recursos

La ficha Referencias de la factoría de conexión del gestor de recursos tiene una rejilla con nombres de recursos, sus tipos, si utilizan autorización de Contenedor o de Servlet y su ámbito donde compartir, más un campo de texto descriptivo de la entrada actualmente seleccionada.

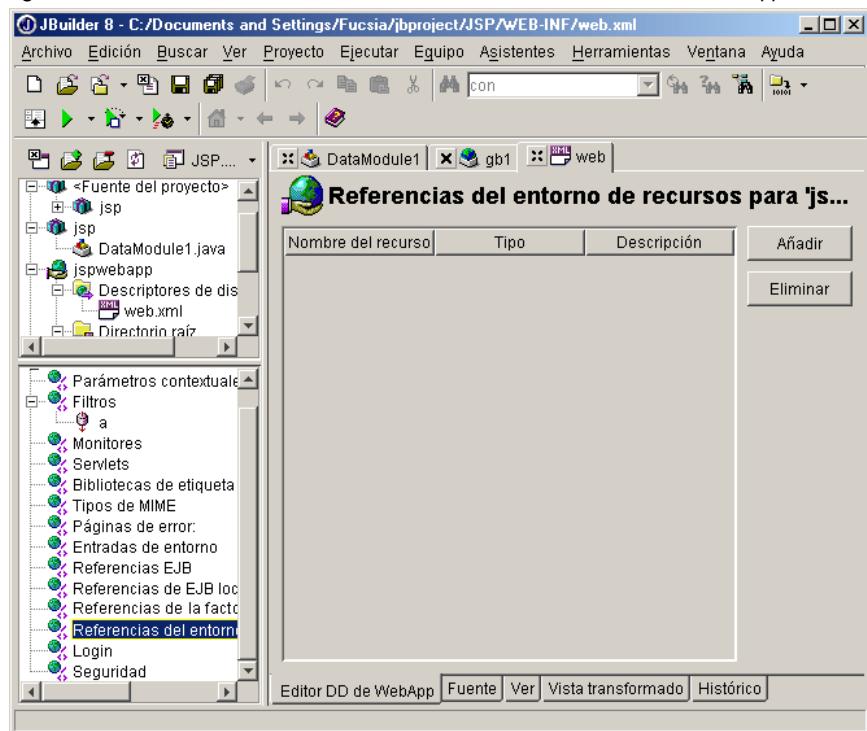
Figura 12.13 Ficha Referencias de la factoría de conexión del gestor de recursos en el Editor DD de WebApp



Referencias del entorno de recursos

La ficha Referencias del entorno de recursos tiene una rejilla con nombres de recurso, sus tipos y un campo de texto descriptivo para la entrada seleccionada.

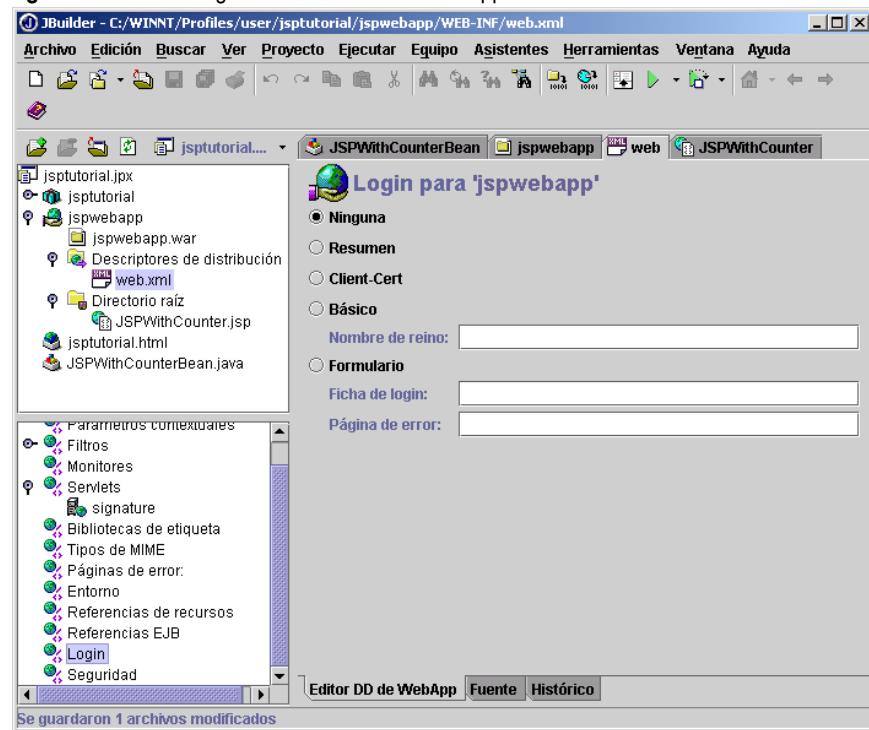
Figura 12.14 Ficha Referencias del entorno de recursos en el Editor DD de WebApp



Ficha Conexión

La ficha Conexión muestra un conjunto de botones de radio para elegir el método de autentificación de la WebApp. El valor por defecto es Ninguna. Otras opciones son Resumen, Client Cert, Básico y Formulario. Para la opción Básico, se puede especificar un nombre de Reino. Para Formulario, se puede especificar una ficha de conexión y una Página de error (de conexión).

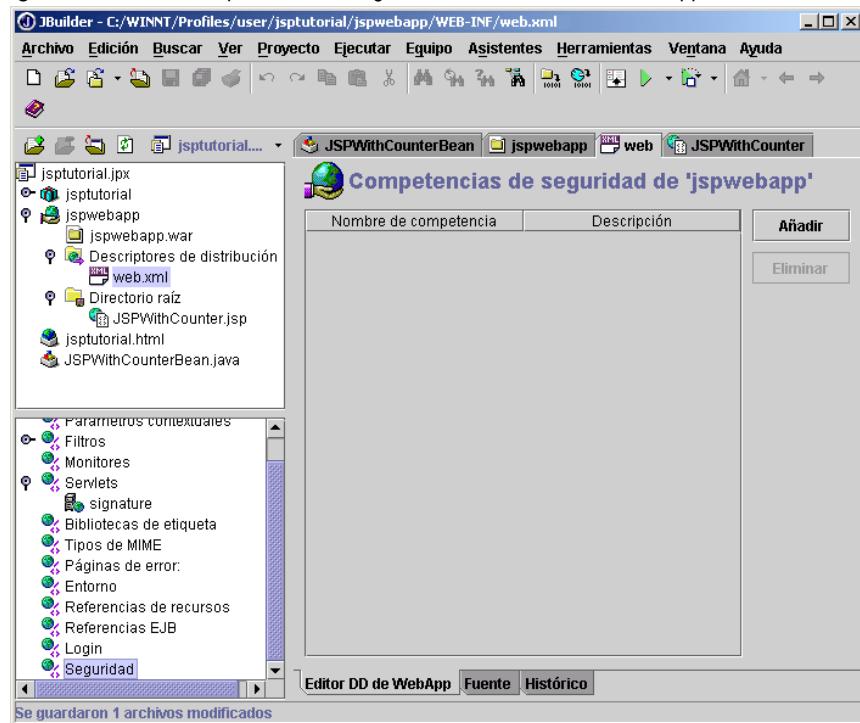
Figura 12.15 Ficha Login en el Editor DD de WebApp



Ficha Seguridad

La ficha Seguridad tiene una rejilla con nombres de competencias y sus descripciones.

Figura 12.16 Ficha Competencias de seguridad en el Editor DD de WebApp



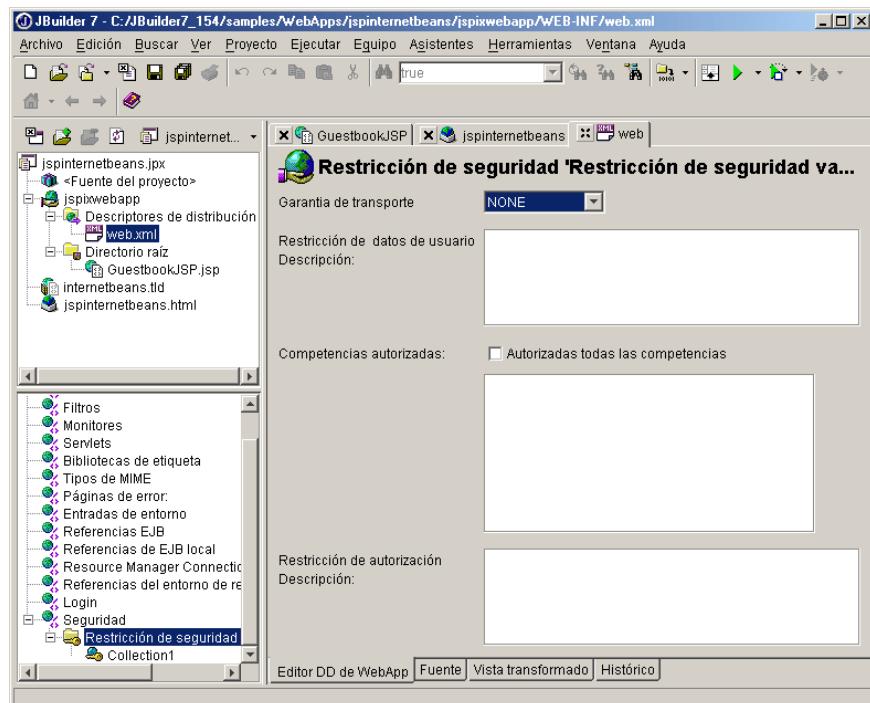
Restricciones de seguridad

Cada restricción de seguridad se lista como un nodo dependiente separado del nodo Seguridad. Una restricción de seguridad puede borrarse si hace clic con el botón derecho en el nodo de la restricción individual y selecciona Eliminar en el menú contextual. Si se borra una restricción, este cambio se extenderá en cascada a todas las partes relacionadas del descriptor de distribución.

Cuando se abre un nodo de restricción de seguridad individual, el Editor DD de WebApp muestra la información de esa restricción de seguridad: la garantía de transporte, los nombres de competencias autorizadas y descripciones de ambos.

Adicionalmente a la especificación por separado de autorizaciones, el nombre de competencia reservado * indica todas las competencias. Si se utiliza esta designación junto a nombres de competencias individuales, estas últimas no se tienen en cuenta. La ficha Restricciones de seguridad del Editor DD de WebApp incluye una casilla de selección Autorizadas todas las competencias, visible sólo cuando se modifica un archivo Servlet 2.3 web.xml.

Figura 12.17 Restricción de seguridad en el Editor DD de WebApp



Colección de recursos web

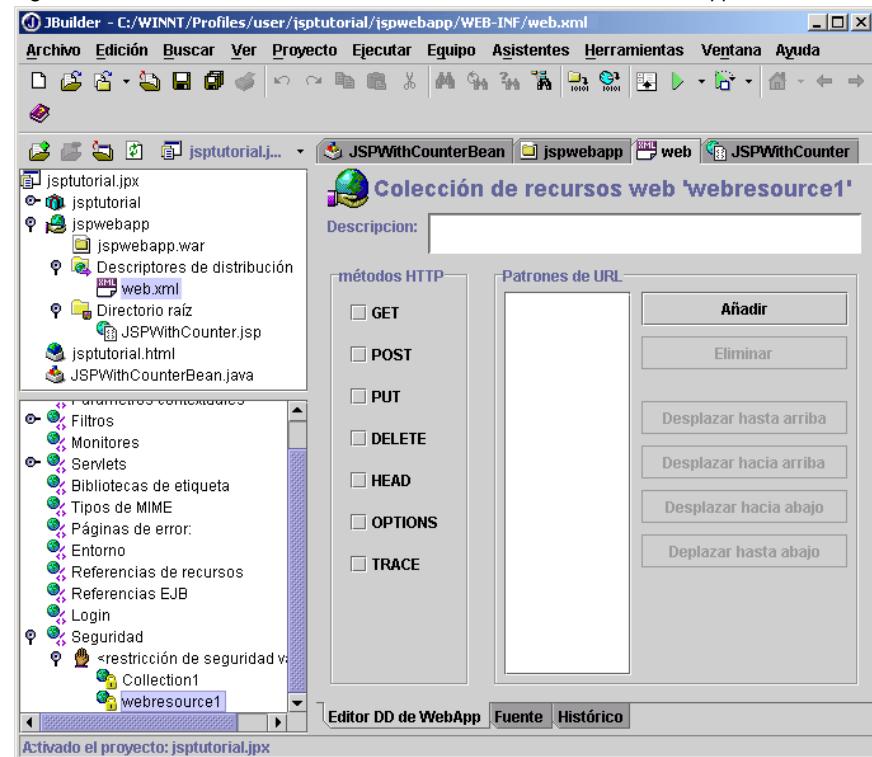
Cada nombre de recurso de web de la colección de recursos Web se lista por separado como un nodo dependiente de una o más restricciones de seguridad aplicables. No se puede añadir una colección de recursos web a menos que ya se tenga por lo menos una restricción de seguridad. Cada restricción de seguridad debe tener al menos una colección de recursos web. Se puede renombrar o eliminar una colección de recursos web si hace clic con el botón derecho en el nodo de la colección individual y selecciona Renombrar o Eliminar en el menú contextual. Si se renombra o se elimina una colección de recursos web, este cambio se extenderá en cascada a todas las partes relacionadas del descriptor de distribución. Si elimina la última colección de recursos web de una restricción, también se elimina la restricción.

Cuando se abre un nodo de colección de recursos web, el Editor DD de WebApp contiene la información de identificación de la colección de recursos web:

Tabla 12.5 Ficha Colección de recursos web del Editor DD de WebApp

Elementos	Descripción
Descripción	Descripción de la colección de recursos web.
Métodos HTTP	Un conjunto de casillas de verificación para los métodos HTTP (como GET y POST) que se aplican a los patrones de URL. Si no selecciona ninguna de estas casillas de verificación tiene el mismo efecto que seleccionarlas todas.
Patrones de URL	Un cuadro de lista de los patrones de URL de esa colección de recursos web.

Figura 12.18 Nodo de colección de recursos Web en el Editor DD de WebApp



13

Modificación del archivo struts-config.xml

El desarrollo web es una función de JBuilder Enterprise

El archivo struts-config.xml es el descriptor de distribución de una aplicación web preparada para Struts. Debe ajustarse a un formato determinado, como se describe en el DTD (Definición de tipo de documento) que se encuentra en http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd. Si desea obtener más detalles acerca de los elementos más importantes del DTD, consulte “The Struts Configuration File” en *Struts User’s Guide* en <http://jakarta.apache.org/struts/userGuide>.

Se puede utilizar el Editor de configuración de Struts visual de JBuilder con el fin de modificar y mantener el archivo struts-config.xml. Si abre el archivo struts-config.xml en el IDE de JBuilder, el contenido aparece en el panel de estructura, y el Visualizador de aplicaciones abre el Editor de configuración de Struts. Al pulsar un nodo en el panel de estructura aparecen las fichas correspondientes del editor. Existen cuatro fichas principales:

- Fuentes de datos
- Form Beans
- Reenvíos globales
- Correspondencias Action

Cada una de estas fichas configura elementos y atributos del descriptor de distribución. El Editor de configuración de Struts abarca todos los elementos del descriptor de distribución struts-config.xml de la especificación Struts 1.0.

Puede modificar el archivo struts-config.xml en el Editor de configuración de Struts o en el editor de fuentes. Los cambios efectuados en el Editor de configuración de Struts se reflejarán en el código fuente y viceversa. Si introduce comentarios en el archivo struts-config.xml utilizando el editor de código fuente, se borrarán si posteriormente abre el archivo en el Editor de configuración de Struts.

Nota Si utiliza los asistentes de JBuilder para crear los fragmentos de su aplicación web Struts, puede que no necesite modificar el archivo struts-config.xml.

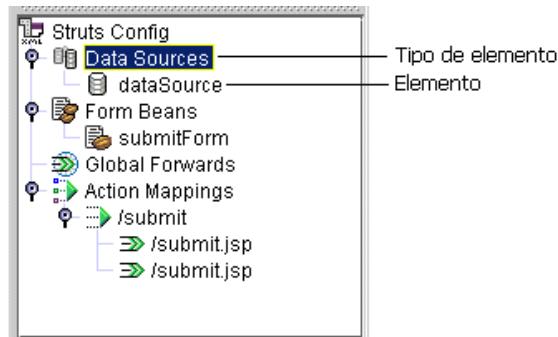
Mientras utiliza JBuilder para crear las piezas de su aplicación web, el archivo del descriptor de distribución web.xml se actualiza automáticamente para Struts. Para obtener más información, consulte ["Implementaciones del marco de trabajo Struts en JBuilder"](#) en la página 8-16.

Importante JBuilder 8 admite Struts 1.0 y no admite oficialmente la versión beta de Struts 1.1. Sin embargo, si activa el archivo struts-config.xml para Struts 1.1 y descarga el archivo JAR de Struts 1.1 en la carpeta <jbuilder>/extras, se logra la compatibilidad con Struts 1.1. El Editor de configuración de Struts muestra elementos y atributos específicos de 1.1.

Selección de una ficha del Editor de configuración de Struts

En el panel de estructura se seleccionan las fichas del Editor de configuración de Struts. Para seleccionar una ficha:

- 1 Haga doble clic en el archivo struts-config.xml, en el panel del proyecto. Se abre la ficha de visión general del editor en el panel de contenido, y su estructura aparece en el panel de estructura.
- 2 En el panel de estructura, pulse el nodo que desee modificar. Para dirigirse a un elemento ya creado, amplíe el nodo en el panel de estructura y, a continuación, haga clic sobre el elemento. El elemento y sus atributos se muestran en el Editor de configuración de Struts.



El menú contextual del Editor de configuración de Struts

Si abre el Editor de configuración de Struts en el panel de contenido, puede pulsar con el botón derecho del ratón cualquiera de los nodos del panel de estructura con el fin de acceder al menú contextual. Las opciones de menú varían ligeramente de acuerdo con el nodo seleccionado. Los comandos de menú permiten la adición de elementos, la eliminación de otros y la apertura de asistentes para crear clases.

Ficha Fuentes de datos

La ficha Fuentes de datos se utiliza para crear el elemento `<data-sources>` en el archivo `struts-config.xml`. Este elemento describe un conjunto de objetos de fuentes de datos JDBC 2.0 Standard Extension, que se configuran según los elementos anidados `<data-source>`.

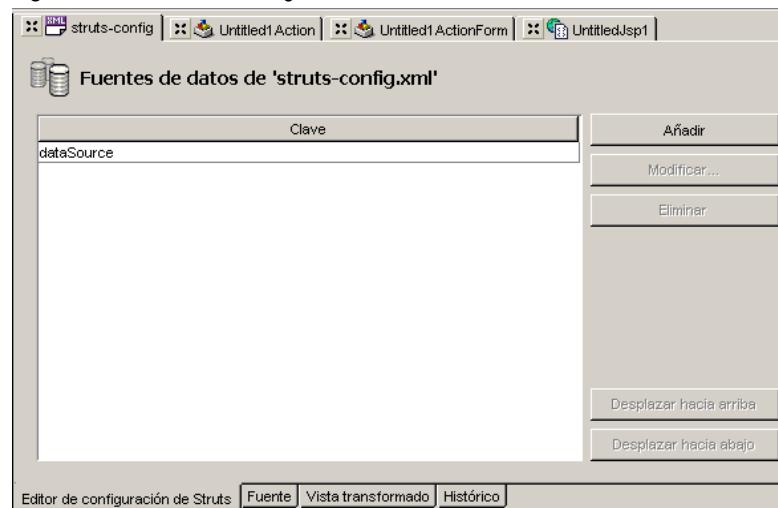
Esta es la apariencia de una entrada de ejemplo en `struts-config.xml`:

```
<data-sources>
  <data-source>
    <set-property property="autoCommit" value="true"/>
    <set-property property="description" value="Sample JDataStore Configuration"/>
    <set-property property="maxCount" value="4"/>
    <set-property property="minCount" value="2"/>
    <set-property property="password" value="mypasssword"/>
    <set-property property="url"
      value="jdbc:borland:dslocal:
      C:/JBuilder/samples/WebApps/guestbook/guestbook.jds"/>
    <set-property property="user" value="myusername"/>
  </data-source>
</data-sources>
```

Para que se abra la ficha de visión general de Fuentes de datos, del Editor de configuración de Struts, seleccione el nodo Fuentes de datos en el panel de estructura. La ficha de visión general de Fuentes de datos muestra los elementos `<data_source>` ya creados. Esta ficha se utiliza para añadir y eliminar elementos.

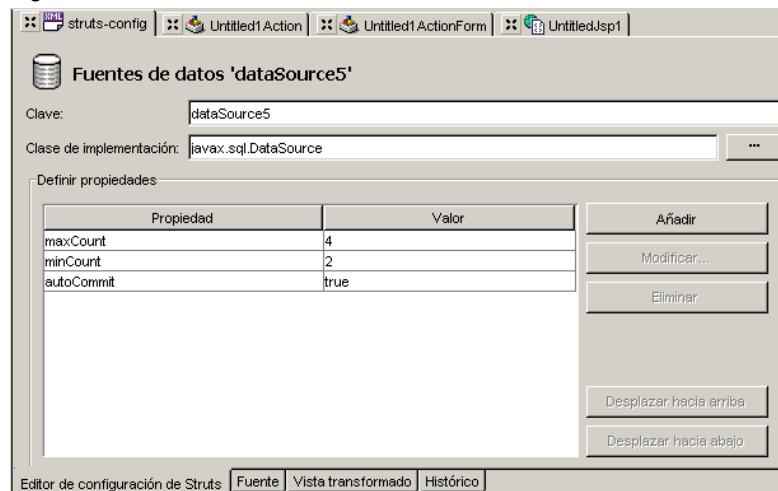
- Si desea añadir un `<data-source>`, pulse el botón Añadir. El elemento nuevo se añade con el nombre por defecto `dataSource`.
- Para eliminar un elemento, selecciónelo y pulse Eliminar.
- Para modificar un elemento recién añadido o ya presente, selecciónelo y pulse Modificar.

Figura 13.1 Ficha de visión general de Fuentes de datos



La ficha de atributos de Fuentes de datos se utiliza para modificar los atributos de un elemento <data-source>. La ficha de atributos tendrá este aspecto:

Figura 13.2 Ficha de atributos de Fuentes de datos



En la siguiente tabla se describen los atributos de la ficha.

Tabla 13.1 Atributos de Fuente de datos

Nombre de campo	Nombre del atributo	Descripción
Clave	key	Clave de atributos de contexto de Servlet bajo la que se guarda esta fuente de datos. El valor por defecto lo especifica la cadena constante Action.DATA_SOURCE_KEY.
Clase de implementación	type	El nombre de la clase Java del controlador JDBC que se utiliza.
Propiedad	property-name, por ejemplo autoCommit, loginTimeout, maxCount, minCount, password, readonly, url, y user. Observe que password, readonly, url y user son necesarios.	Una propiedad para esta conexión de fuentes de datos.
Valor	El valor de la propiedad, como por ejemplo, true, false, un nombre de usuario o contraseña o la URL de la fuente de datos.	El valor por defecto de la propiedad.

Menú contextual de la ficha Fuentes de datos

Para poder abrir el menú contextual de la ficha Fuentes de datos, haga clic con el botón derecho del ratón sobre el nodo Fuentes de datos del panel de estructura o amplíe el nodo y haga clic con el botón derecho del ratón sobre un elemento. Los comandos de menú disponibles de menú varían ligeramente de acuerdo con el nodo seleccionado. En la siguiente tabla se describen los comandos del menú contextual:

Tabla 13.2 Menú contextual de la ficha Fuentes de datos

Nodo	Comando Menú	Descripción
Nodo Fuentes de datos	Añadir fuente de datos	Abre la ficha de atributos Fuentes de datos del editor, en la que se añade un nuevo elemento <data-source>.
Elemento Fuentes de datos	Añadir fuente de datos	Añade un elemento de <data-source>.
Elemento Fuentes de datos	Borrar	Elimina el elemento seleccionado.

Configuración de los atributos de las propiedades

Los botones Añadir, Modificar y Eliminar se utilizan con el fin de configurar y modificar los atributos de propiedades del elemento Fuente de datos.

- Si desea añadir un atributo de propiedad, pulse el botón Añadir. Se añade automáticamente una propiedad con los valores por defecto `property` y `value`.
- Para modificar el atributo de una propiedad, haga clic dos veces en la columna Propiedad e introduzca un nuevo valor. También puede pulsar el botón Modificar para que se abra el cuadro de diálogo Modificar/Configurar propiedad, en el que se pueden cambiar la propiedad y el valor.
- Para eliminar una propiedad, selecciónela y pulse Borrar.

Ficha Form Beans

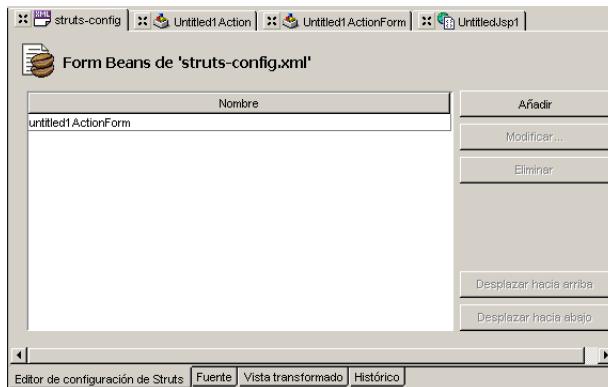
La ficha Form Beans se utiliza para crear el elemento `<form-beans>` del archivo struts-config.xml. El elemento `<form-beans>` es la raíz del conjunto de descriptores form bean de su aplicación web Struts. Los elementos anidados `<form-bean>` describen un form bean en concreto, que es un JavaBean que implementa el `org.apache.struts.action.ActionForm`.

Esta es la apariencia de una entrada de ejemplo en struts-config.xml:

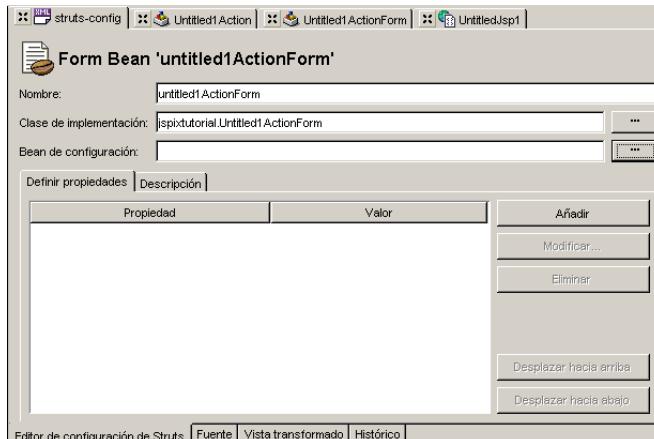
```
<form-beans>
    <form-bean name="submitForm" type="myproject.SubmitForm" />
</form-beans>
```

Con el fin de abrir la ficha de visión general de Form Bean del Editor de configuración de Struts, seleccione el nodo Form Beans en el panel de estructura. La ficha de visión general de Form Beans muestra los elementos `<form-bean>` existentes. Esta ficha se utiliza para añadir y eliminar elementos.

- Si desea añadir un `<form-bean>`, pulse el botón Añadir. El elemento nuevo se añade con el nombre por defecto `formBean`.
- Para eliminar un elemento, selecciónelo y pulse Eliminar.
- Para modificar un elemento recién añadido o ya presente, selecciónelo y pulse Modificar.

Figura 13.3 Ficha de visión general de Form Beans

La ficha de atributos de Form Bean se utiliza para configurar los atributos de un elemento <form-bean>. La ficha de atributos tendrá este aspecto:

Figura 13.4 Ficha de atributos de Form Bean

En la siguiente tabla se describen los atributos de la ficha.

Tabla 13.3 Atributos de Form Bean

Nombre del campo del Editor de configuración de Struts	Nombre del atributo	Descripción
Nombre	name	Identificador único del bean, que se usa para referirse a él en las correspondencias Action.
Clase de implementación	type	Nombre completo de la clase Java de la clase de implementación que se utiliza o genera.

Tabla 13.3 Atributos de Form Bean (continuación)

Nombre del campo del Editor de configuración de Struts	Nombre del atributo	Descripción
Bean de configuración	className	Nombre completo de la clase Java de la clase de implementación <code>ActionFormBean</code> que se utiliza. Toma por defecto el valor configurado como el parámetro de inicialización <code>formBean</code> del servlet del controlador Struts.
Propiedad (pestaña Definir propiedades)	property-name, por ejemplo <code>id</code> . No es necesaria una propiedad.	Una propiedad para este form bean.
Valor (pestaña Definir propiedades)	El valor por defecto de la propiedad, por ejemplo <code>ID</code> para la propiedad <code>id</code> .	El valor por defecto.
Icono grande (pestaña Descripción)	large-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de un ícono grande (32x32 píxeles), adecuada para utilizarla en herramientas GUI.
Icono pequeño (pestaña Descripción)	small-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de ícono pequeño (16x16 pixeles), adecuada para utilizar en herramientas GUI.
Nombre en pantalla (pestaña Descripción)	display-name	Contiene una breve descripción (una línea) del elemento, adecuado para utilizar en herramientas GUI.
Descripción (pestaña Descripción)	description	Contiene un texto descriptivo (un párrafo) sobre el elemento, adecuado para utilizar en herramientas GUI.

Menú contextual de la ficha Form Beans

Para poder abrir el menú contextual de la ficha Form Beans, haga clic con el botón derecho del ratón sobre el nodo Form Beans del panel de estructura o amplíe el nodo y haga clic con el botón derecho del ratón sobre un elemento. Los comandos de menú disponibles de menú varían ligeramente de acuerdo con el nodo seleccionado. En la siguiente tabla se describen los comandos del menú contextual:

Tabla 13.4 Menú contextual de la ficha Form Beans

Nodo	Comando Menú	Descripción
Nodo Form Beans	Añade form bean	Abre la ficha de atributos Form Bean, en la que se añade un elemento <form-bean>.
Elemento Form Beans	Añade form bean	Añade un elemento <form-bean>.
Elemento Form Beans	Asistente para ActionForm	Abre el Asistente para ActionForm, en el que se crea una clase ActionForm.
Elemento Form Beans	Borrar	Elimina el elemento seleccionado.

Configuración de los atributos de las propiedades

Los botones Añadir, Modificar y Eliminar se utilizan para configurar y modificar los atributos de propiedades en la ficha Definir propiedades.

- Si desea añadir un atributo de propiedad, pulse el botón Añadir. Se añade automáticamente una propiedad con los valores por defecto `property` y `value`.
- Para modificar un atributo de una propiedad, haga clic dos veces en las columnas Propiedad o Valor e introduzca un nuevo valor. También puede pulsar el botón Modificar para que se abra el cuadro de diálogo Modificar/Configurar propiedad, en el que se pueden cambiar la propiedad y el valor.
- Para eliminar una propiedad, selecciónela y pulse Borrar.

Ficha Reenvíos globales

La ficha Reenvíos globales se utiliza para crear el elemento `<global-forwards>` del archivo `struts-config.xml`. El elemento `<global-forwards>` configura las correspondencias globales de los nombres lógicos (utilizados en la aplicación) con los recursos correspondientes (identificados por las vías de acceso URI relativas al contexto). Los distintos elementos `<forward>` del elemento `<global-forwards>` describen una correspondencia de un nombre lógico (utilizado en la aplicación) con los recursos

correspondientes identificados por las vías de acceso URI relativas al contexto.

Nota Un elemento <global-forwards> con un nombre en concreto se puede modificar de forma local si se define un elemento <forward> con el mismo nombre dentro de un elemento <action>.

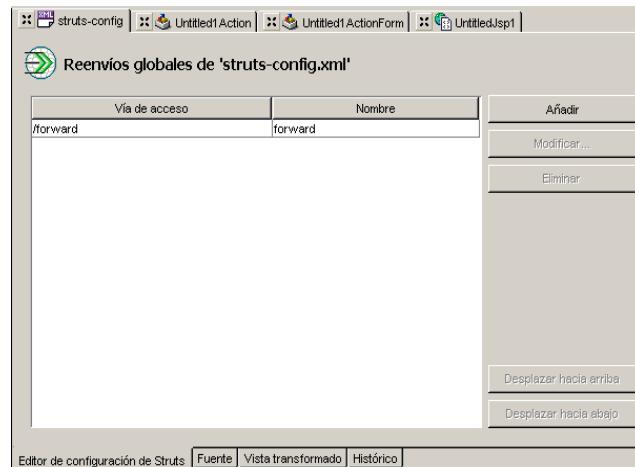
Esta es la apariencia de una entrada de ejemplo en struts-config.xml:

```
<global-forwards>
    <forward name="logon" path="/logon.jsp" redirect="false" />
</global-forwards>
```

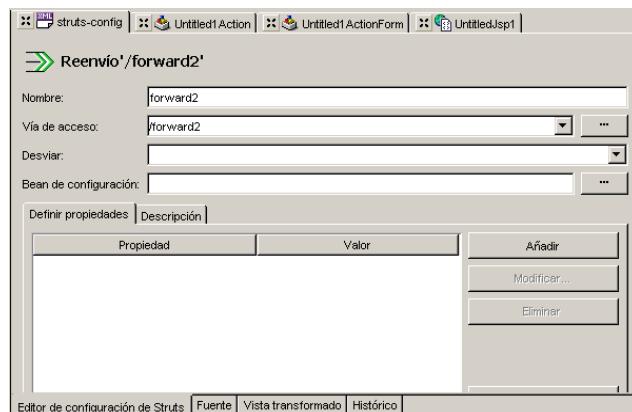
Para que se abra la ficha de visión general de Reenvíos globales del Editor de configuración de Struts, seleccione el nodo Reenvíos globales en el panel de estructura. La ficha de visión general de Reenvíos globales muestra los elementos <forward> existentes. Esta ficha se utiliza para añadir y eliminar elementos.

- Si desea añadir un elemento <forward>, pulse el botón Añadir. El elemento nuevo se añade con el nombre por defecto forward.
- Para eliminar un elemento, selecciónelo y pulse Eliminar.
- Para modificar un elemento recién añadido o ya presente, selecciónelo y pulse Modificar.

Figura 13.5 Ficha de visión general de Reenvíos globales



La ficha de atributos de Reenvío se utiliza para configurar los atributos de un elemento. La ficha de atributos tendrá este aspecto:

Figura 13.6 Ficha de atributos de Reenvío

En la siguiente tabla se describen los atributos de la ficha.

Tabla 13.5 Atributos de Reenvío

Nombre de campo	Nombre del atributo	Descripción
Nombre	name	Identificador único de forward, utilizado para referirse a él en las clases action de una aplicación.
Vía de acceso	path	La vía de acceso relativa al contexto del recurso relacionado.
Desviar	redirect	Asigne el valor <code>true</code> si se utiliza <code>sendRedirect()</code> para reenviar este recurso, o como <code>false</code> si se utiliza <code>RequestDispatcher.forward()</code> .
Bean de configuración	className	Nombre completo de la clase Java de la clase de implementación <code>ActionForward</code> que se utiliza. Toma por defecto el valor configurado como el parámetro de inicialización <code>forward</code> del servlet del controlador Struts.
Propiedad (pestaña Definir propiedades)	property-name, por ejemplo <code>id</code> . No es necesaria una propiedad.	Una propiedad para este reenvío.
Valor (pestaña Definir propiedades)	El valor por defecto de la propiedad, por ejemplo <code>ID</code> para la propiedad <code>id</code> .	El valor por defecto.

Tabla 13.5 Atributos de Reenvío (continuación)

Nombre de campo	Nombre del atributo	Descripción
Icono grande (pestaña Descripción)	large-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de un ícono grande (32x32 píxeles), adecuada para utilizarla en herramientas GUI.
Icono pequeño (pestaña Descripción)	small-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de ícono pequeño (16x16 píxeles), adecuada para utilizar en herramientas GUI.
Nombre en pantalla (pestaña Descripción)	display-name	Contiene una breve descripción (una línea) del elemento, adecuado para utilizar en herramientas GUI.
Descripción (pestaña Descripción)	description	Contiene un texto descriptivo (un párrafo) sobre el elemento, adecuado para utilizar en herramientas GUI.

Menú contextual de la ficha Reenvíos globales

Para poder abrir el menú contextual de la ficha Reenvíos globales, haga clic con el botón derecho del ratón sobre el nodo Reenvíos globales del panel de estructura o amplíe el nodo y haga clic con el botón derecho del ratón sobre un elemento. Los comandos de menú disponibles de menú varían ligeramente de acuerdo con el nodo seleccionado. En la siguiente tabla se describen los comandos del menú contextual:

Tabla 13.6 Menú contextual de la ficha Reenvíos globales

Nodo	Comando Menú	Descripción
Nodo Reenvíos globales	Añade un reenvío	Abre la ficha de atributos de Reenvío, en la que se añade un elemento <global-forward>.
Elemento Reenvíos globales	Añade un reenvío	Añade un elemento <global-forward>.
Elemento Reenvíos globales	Borrar	Elimina el elemento seleccionado.

Configuración de los atributos de las propiedades

Los botones Añadir, Modificar y Eliminar se utilizan para configurar y modificar los atributos de propiedades en la ficha Definir propiedades.

- Si desea añadir un atributo de propiedad, pulse el botón Añadir. Se añade automáticamente una propiedad con los valores por defecto `property` y `value`.
- Para modificar un atributo de una propiedad, haga clic dos veces en las columnas Propiedad o Valor e introduzca un nuevo valor. También puede pulsar el botón Modificar para que se abra el cuadro de diálogo Modificar/Configurar propiedad, en el que se pueden cambiar la propiedad y el valor.
- Para eliminar una propiedad, selecciónela y pulse Borrar.

Ficha Correspondencias Action

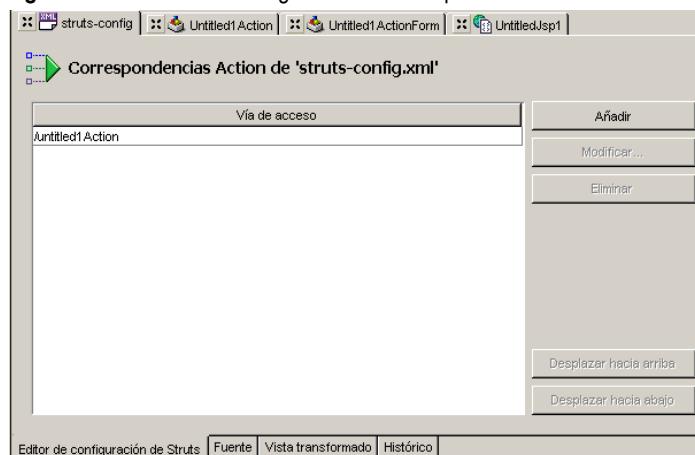
La ficha Correspondencias Action se utiliza para crear el elemento `<action-mappings>` de `struts-config.xml`. El elemento `<action-mappings>` configura las correspondencias de las vías de acceso a las peticiones enviadas con las clases `Action` que se deben utilizar para procesar estas solicitudes.

Esta es la apariencia de una entrada de ejemplo en `struts-config.xml`:

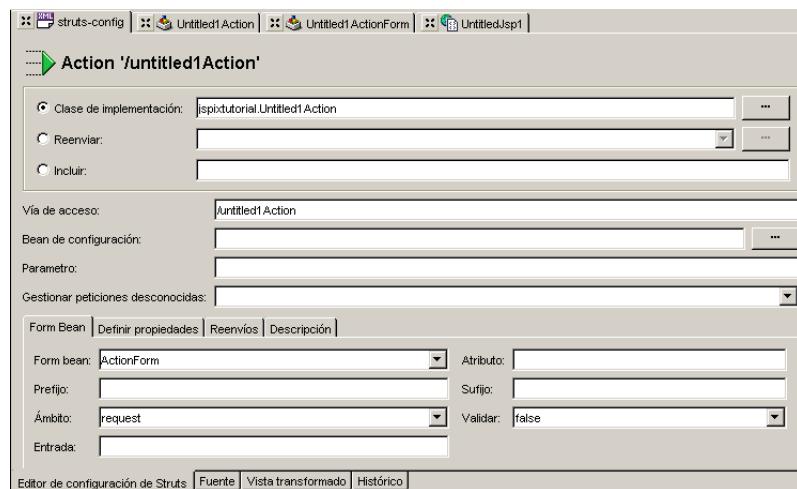
```
<action-mappings>
  <action name="submitForm" type="myproject.SubmitAction" input=
    "/submit.jsp" scope="request" path="/submit">
    <forward name="success" path="/submit.jsp" />
    <forward name="failure" path="/submit.jsp" />
  </action>
</action-mappings>
```

Para que se abra la ficha de visión general de Correspondencias Action del Editor de configuración de Struts, seleccione el nodo Correspondencias Action en el panel de estructura. La ficha de visión general de Correspondencias Action muestra los elementos `<action>` existentes. Esta ficha se utiliza para añadir y eliminar elementos.

- Si desea añadir un elemento `<action>`, pulse el botón Añadir. El elemento nuevo se añade con el nombre por defecto `/action`.
- Para eliminar un elemento, selecciónelo y pulse Eliminar.
- Para modificar un elemento recién añadido o ya presente, selecciónelo y pulse Modificar.

Figura 13.7 Ficha de visión general de Correspondencias Action

La ficha de atributos de Action se utiliza para configurar los atributos de un elemento. La ficha de atributos tendrá este aspecto:

Figura 13.8 Ficha de atributos de Action

En la siguiente tabla se describen los atributos de la ficha:

Tabla 13.7 Atributos de Action

Nombre de campo	Nombre del atributo	Descripción
Clase de implementación	type	Nombre completo de la clase Java de la clase Action que se utiliza para procesar las solicitudes de esta correspondencia. Si se selecciona este campo, los campos Reenvío e Incluir no están disponibles.
Reenvío	forward	Vía de acceso relativa al contexto del recurso servlet o JSP que procesa esta solicitud, en lugar de instanciar y llamar a la clase Action especificada en el campo Clase de implementación. Si se selecciona este campo, los campos Clase de implementación e Incluir no están disponibles.
Incluir	include	Vía de acceso relativa al contexto del recurso servlet o JSP que procesa esta solicitud, en lugar de instanciar y llamar a la clase Action especificada en el campo Clase de implementación. Si se selecciona este campo, los campos Clase de implementación y Reenvío no están disponibles.
Vía de acceso	path	La vía de acceso relativa al contexto de la solicitud enviada, que comienza con una barra diagonal ("/"), y sin la extensión de archivo si se ha utilizado la correspondencia de extensiones.
Bean de configuración	className	Nombre completo de la clase Java de la clase de implementación ActionMapping que se utiliza. Toma por defecto el valor configurado como parámetro de inicialización mapping en el servlet del controlador Struts.
Parámetro	parameter	Un parámetro de configuración general que se puede utilizar para pasar información adicional a la Action seleccionada por esta correspondencia.
Gestionar peticiones desconocidas	unknown	Se establece en true si esta acción se configura por defecto para esta aplicación, con el fin de gestionar todas las solicitudes que no gestiona ninguna otra acción. Sólo se puede definir una acción por defecto dentro de una aplicación.
Form Bean (pestaña Form Bean)	name	Nombre del form bean, si lo hay, asociado con esta acción.
Prefijo (pestaña Form Bean)	prefix	Prefijo utilizado para realizar correspondencias entre nombres de parámetros de solicitud y nombres de propiedades form bean, si los hay. Es optativo si se utiliza el campo Form Bean; si no, no está disponible.

Tabla 13.7 Atributos de Action (continuación)

Nombre de campo	Nombre del atributo	Descripción
Ámbito (pestaña Form Bean)	scope	Identificador del ámbito (<code>request</code> o <code>session</code>) en el que se accede al form bean, si lo hay. Es optativo si se utiliza el campo Form Bean; si no, no está disponible.
Entrada (pestaña Form Bean)	input	Vía de acceso relativa al contexto del formulario de entrada al que se devuelve el control si se encuentra un error de validación. Es necesario si se utiliza el campo Form Bean y el bean de entrada devuelve errores de validación. Es optativo si se utiliza el campo Form Bean y el bean de entrada no devuelve errores de validación. No se admite si no se utiliza el campo Form Bean.
Atributo (pestaña Form Bean)	attribute	Nombre del atributo de ámbito de petición o ámbito de sesión bajo el que se accede a este form bean, si es distinto del nombre que aparece en el campo Form Bean. Es optativo si se utiliza el campo Form Bean; si no, no está disponible.
Sufijo (pestaña Form Bean)	suffix	Se utiliza para asociar nombres de parámetros de peticiones con nombres de propiedades de form bean, si los hay. Es optativo si se utiliza el campo Form Bean; si no, no está disponible.
Validar (pestaña Form Bean)	validate	Se configura en <code>true</code> si se llama al método <code>validate()</code> del form bean antes de llamar a esta acción; se configura en <code>false</code> si no desea realizar la validación.
Propiedad (pestaña Definir propiedades)	property	Especifica el nombre y valor de una propiedad adicional de configuración JavaBeans cuyo método de definición llama el objeto que representa al elemento en cuestión. Esto resulta de gran utilidad cuando se configura una clase de implementación ampliada (con propiedades adicionales) en los elementos <code><global-forwards></code> o <code><action-mappings></code> .
Valor (pestaña Definir propiedades)	value	Representación en cadena del valor que se asignará a esta propiedad, después de realizar la conversión adecuada.
Vía de acceso (ficha Reenvíos)	path	La vía de acceso relativa al contexto del recurso relacionado. Modifica de forma lógica un <code>forward</code> global con el mismo nombre.
Nombre (pestaña Reenvíos)	name	El identificador único de este <code>forward</code> . Modifica de forma lógica un <code>forward</code> global con el mismo nombre.
Icono grande (pestaña Descripción)	large-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de un ícono grande (32x32 píxeles), adecuada para utilizarla en herramientas GUI.

Tabla 13.7 Atributos de Action (continuación)

Nombre de campo	Nombre del atributo	Descripción
Icono pequeño (pestaña Descripción)	small-icon	Especifica la ubicación, relativa al archivo de configuración Struts, de un recurso que contiene una imagen de ícono pequeño (16x16 píxeles), adecuada para utilizar en herramientas GUI.
Nombre en pantalla (pestaña Descripción)	display-name	Contiene una breve descripción (una línea) del elemento, adecuado para utilizar en herramientas GUI.
Descripción (pestaña Descripción)	description	Contiene un texto descriptivo (un párrafo) sobre el elemento, adecuado para utilizar en herramientas GUI.

Menú contextual de la ficha Correspondencias Action

Para poder abrir el menú contextual de la ficha Correspondencias Action, haga clic con el botón derecho del ratón sobre el nodo Correspondencias Action del panel de estructura o amplíe el nodo y haga clic con el botón derecho del ratón sobre un elemento. Los comandos de menú disponibles de menú varían ligeramente de acuerdo con el nodo seleccionado. En la siguiente tabla se describen los comandos del menú contextual:

Tabla 13.8 Menú contextual de la ficha Correspondencias Action

Nodo	Comando Menú	Descripción
Nodo Correspondencias Action	Añade una acción	Abre la ficha de atributos Action, en la que se añade un elemento <action>.
Nodo Correspondencias Action	Asistente para Action	Abre el Asistente para Action, en el que se crea una clase Action.
Elemento Correspondencias Action	Añade una acción	Añade un elemento <action>.
Elemento Correspondencias Action	Añade un reenvío	Añade un elemento <global-forward> al action seleccionado.
Elemento Correspondencias Action	Borrar	Elimina el elemento seleccionado.

Configuración de los atributos de las propiedades

Los botones Añadir, Modificar y Eliminar se utilizan para configurar y modificar los atributos de propiedades en la ficha Definir propiedades.

- Si desea añadir un atributo de propiedad, pulse el botón Añadir. Se añade automáticamente una propiedad con los valores por defecto `property` y `value`.
- Para modificar un atributo de una propiedad, haga clic dos veces en las columnas Propiedad o Valor e introduzca un nuevo valor. También puede pulsar el botón Modificar para que se abra el cuadro de diálogo Modificar/Configurar propiedad, en el que se pueden cambiar la propiedad y el valor.
- Para eliminar una propiedad, selecciónela y pulse Borrar.

14

Las applets

El desarrollo de applets
es una característica de
todas las ediciones de
JBuilder

Si ya ha intentado escribir applets y distribuirlos en una página Web, habrá comprobado que, comparados con las aplicaciones, las applets exigen que se resuelvan problemas adicionales para que funcionen. Las dificultades empiezan normalmente desde el momento que se distribuyen y comprueban las applets fuera del entorno IDE (Entorno integrado de desarrollo) y, si no se manejan de forma correcta algunas cuestiones básicas de las applets, éstas no funcionarán correctamente. Por ello es importante que sepa cómo funcionan las applets y cómo encajan todas las piezas, especialmente cuando se tenga intención de distribuirlas y cargarlas en una sede web externa.

Consulte

- “Tutorial: Creación de un applet” en *Introducción a JBuilder*
- El tutorial de Java en “Writing Applets” en <http://java.sun.com/docs/books/tutorial/applet/index.html>
- La página web de Sun en <http://java.sun.com/applets/index.html>
- Las Java FAQ en <http://www.afu.com/javafaq.html>
- La obra de Charlie Calvert, “Java Madness, Part II: en <http://homepages.borland.com/ccalvert/JavaCourse/index.htm>
- La página Web Java Curmudgeon de Rich Wilkman en <http://formlessvoid.com/jc/applets/>
- La página web de John Moore, “Applet design and deployment”, en http://www.microps.com/mps/p_appletdesign.html

Cómo funcionan

Las applets son programas de Java que se guardan en un servidor de Internet/intranet. Una vez transferidas a las diferentes plataformas cliente, se ejecutan en una máquina virtual (MV) de Java suministrada por el navegador instalado en el ordenador cliente. Esta distribución y ejecución se llevan a cabo bajo la supervisión de un administrador de seguridad. Un administrador de seguridad puede evitar que las applets realicen tareas no deseadas, como el formateo del disco duro o la apertura de conexiones a equipos “no certificados”.

Cuando el navegador encuentra una página web con un applet, arranca la MV de Java y le suministra la información de la etiqueta `<applet>`. El cargador de clases interno de la MV de Java controla qué clases se necesitan para el applet. Como parte del proceso de carga de clases, los archivos de clase se ejecutan a través de un verificador que comprueba que son válidos y no código dañino. Una vez verificado, el applet se ejecuta. Por supuesto, ésta es una visión simplificada de todo el proceso.

Este mecanismo de transferencia es el principal valor de las applets. No obstante, deben tenerse en cuenta algunas cuestiones propias del desarrollo de un applet con el fin de garantizar una correcta implementación.

La etiqueta `<applet>`

Todo lo que necesita saber un applet durante la ejecución queda determinado por el contenido de la etiqueta `<applet>` del archivo HTML del applet. Los atributos de la etiqueta le indican qué clase ejecutar, qué recopilatorios (si procede) debe buscar para las clases y la ubicación de estos recopilatorios y/o clases.

A diferencia de las aplicaciones Java, que utilizan una variable de entorno llamada `CLASSPATH` para buscar clases, las applets utilizan solamente el atributo `codebase` en la etiqueta `<applet>` para especificar dónde buscar las clases necesarias para el applet.

La clave para ejecutar un applet consiste precisamente en su capacidad para encontrar las clases que necesita.

Etiqueta de ejemplo `<applet>`

A continuación se presenta una sencilla etiqueta `<applet>` que utiliza los atributos más comunes.

```
<applet  
codebase = ". "  
archive = "test.jar"
```

```

code = "test.Applet1.class"
name   = "TestApplet"
width   = 400
height  = 300
vspace  = 0
hspace  = 0
>
<param name = "param1"  value = "xyz">
</applet>

```

Atributos de las etiquetas <applet>

La siguiente tabla describe los parámetros y atributos más comunes de la etiqueta <applet>. Observe que algunos elementos son necesarios.

Tabla 14.1 Atributos de las etiquetas <applet>

Elementos	Descripción
codebase	<p>Permite especificar para las clases o archivos recopilatorios una ubicación diferente a la del archivo HTML que contiene la etiqueta <applet>. Esta vía de acceso corresponde a la ubicación del archivo HTML y habitualmente se limita a subdirectorios de la ubicación del archivo HTML. El codebase se parece a una CLASSPATH en que le dice al navegador dónde buscar las clases.</p> <p>Por ejemplo, si las clases del applet se almacenan en un subdirectorio llamado <code>classes</code>, el atributo codebase es <code>codebase = "classes"</code>.</p> <p>El valor <code>..</code> especifica el mismo directorio del archivo HTML que ejecuta el applet.</p> <p>Importante: El atributo es necesario si la clase o los archivos de revisiones se encuentran en un directorio diferente que el archivo HTML del applet.</p>
archive	<p>Utilizado para identificar uno o más archivos (ZIP, JAR o CAB) que contengan las clases que necesita el applet. Los archivos recopilatorios deben colocarse en el directorio especificado por codebase. Se pueden tener varios archivos recopilatorios, separados por comas en el listado: <code>archive="archivo1.jar, archivo2.jar"</code>.</p> <p>Importante: Este atributo es necesario si el applet se distribuye en archivos recopilatorios.</p> <p>Nota: Algunos navegadores antiguos sólo admiten archivos ZIP.</p>
código (requerido)	<p>Nombre completo de la clase applet que contiene el método <code>init()</code>. Por ejemplo, si la clase <code>Applet1.class</code> forma parte del paquete llamado <code>test</code>, entonces code sería <code>code="test.Applet1.class"</code>.</p>
name (opcional)	Una cadena que describe el applet. Esta cadena se lee en la barra de estado del navegador.
width/height (necesario)	Define el tamaño en píxeles asignado al applet en el navegador. Esta información también se pasa al gestor de diseño del applet.

Tabla 14.1 Atributos de las etiquetas <applet> (continuación)

Elementos	Descripción
hspace/vspace (opcional)	Representa el relleno horizontal o vertical en píxeles alrededor del applet. Resulta útil si en la página web tiene texto que rodea al applet o si tiene más de un applet en la página.
param (opcional)	Permite especificar parámetros que puede leer la etiqueta <applet>. Son similares a la lista de argumentos utilizada por main() en las aplicaciones.
	Los parámetros tienen dos partes, nombre y valor, ambas cadenas entrecomilladas. nombre se usa en el applet cuando se quiere solicitar un valor. El código del applet debe gestionar las posibles conversiones de String a otro tipo de dato. Asegúrese de que las mayúsculas y minúsculas coinciden en el parámetro HTML y el código applet que lo solicita.

Errores habituales en la etiqueta <applet>

La mayoría de los problemas que se producen con la etiqueta <applet> se deben a los siguientes errores:

- Ausencia de una etiqueta </applet>

Si recibe un mensaje de error indicando que falta una etiqueta </applet> en la página HTML, compruebe si hay una etiqueta de cierre.

- Olvidar que Java diferencia entre mayúsculas y minúsculas

La distinción entre mayúsculas y minúsculas es extremadamente importante. Por lo general, las clases combinan mayúsculas y minúsculas mientras que los directorios y los archivos recopilatorios se escriben en minúscula. Los nombres de clases, archivos recopilatorios y directorios en la etiqueta <applet> deben ser una imagen exacta de los nombres que aparecen en el servidor web. Si se detecta la mínima diferencia, aparecerán mensajes de error del tipo “no se encuentra xyz.class”.

Si la clase se encuentra en el paquete com.mypackage, todos los directorios destinados a admitir esta clase deben crearse y señalarse en minúsculas.

Si se desplazan archivos entre Windows 95/98 y Windows NT e Internet, aumenta la probabilidad de errores de mayúsculas y minúsculas.

Una vez haya enviado los archivos a la Web, asegúrese de que los archivos, directorios y archivos de revisiones tienen los mismos valores de mayúsculas y minúsculas que la máquina local. Además, si ha recopilado todos sus archivos de clase, asegúrese de que la herramienta de recopilación ha conservado las mayúsculas y minúsculas.

- Usar el “nombre abreviado” de la clase en el atributo code

Debe utilizar el nombre completo porque es el nombre real de la clase. Si el nombre de la clase es `Applet1.class` y está en el paquete `test`, entonces debe referenciarlo en el atributo `code` como `test.Applet1.class`.

- Un valor `codebase` incorrecto

El `codebase` se resuelve contra el directorio que contiene el archivo HTML, formando de manera efectiva una entrada `classpath`. El valor `codebase` no es una URL ni otro tipo de referencia. En el sencillo ejemplo de applet presentado anteriormente, se utilizó `codebase = ". ."`. Esto especifica que los archivos que necesita el applet se encuentran en el mismo directorio del archivo HTML o, si no se ha especificado el archivo recopilatorio, los archivos se encuentran en subdirectorios de paquetes apropiados, con respecto al que contiene el archivo HTML.

- Ausencia de un atributo `archive`

Si el applet se distribuye en uno o más archivos JAR o ZIP, debe añadir el atributo `archive` al archivo HTML.

- Ausencia de comillas en los valores utilizados para `code`, `codebase`, `archive`, `name`, etc.

Con el fin de obtener una mejor compatibilidad con XHTML, se recomienda usar comillas también con los números.

Los navegadores

Una de las principales acciones para que las applets funcionen durante la ejecución consiste en resolver los problemas que plantean los navegadores que albergan las applets. El grado de compatibilidad de los navegadores en relación con el kit de desarrollo de Java (JDK) es muy variable, además de que no todos implementan Java del mismo modo. Las cuestiones más frecuentes que encontrará con respecto a los navegadores incluyen permitir la utilización de Java, obtener el navegador preferido por el usuario final, admitir múltiples navegadores y las diferencias en la implementación de Java.

Compatibilidad con Java

Uno de los problemas más frecuentes con las applets es que las versiones del JDK no coinciden plenamente en el applet y el navegador que lo ejecuta. Son muchos los usuarios que no trabajan con la última versión de su navegador. Estos navegadores quizás no sean compatibles con los nuevos JDK de Sun. Swing, introducido en JDK 1.1.7, quizás no sea admitido por muchos navegadores. El error que aparecerá con más frecuencia en alguna clase de los paquetes `java.*` es `NoClassDefFoundError`,

porque el Cargador de clases determinó la necesidad de una clase o método Java que no admite el JDK del navegador.

Los JDK 1.2, 1.3 y 1.4 todavía no son plenamente compatibles con todos los navegadores. Aunque prácticamente todas las últimas versiones de los navegadores admiten Java, conviene que se informe sobre cuál es la versión específica de Java compatible con el navegador en el que se ejecutará el applet. Esto es importante para que tome las medidas necesarias con el fin de garantizar la compatibilidad entre la versión del JDK utilizada en el applet y el JDK admitido por los navegadores.

Para saber qué versión de JDK admite el navegador, compruebe la Consola Java en el navegador o la página Web del navegador.

Para abrir la Consola Java, lleve a cabo una de las acciones siguientes:

- Seleccione Communicator | Herramientas | Consola de Java en las versiones antiguas de Netscape o, bien, Tareas | Herramientas | Consola de Java en las versiones más recientes.
- Seleccione Ver | Consola Java en Internet Explorer.

Con el Plug-in de Java™ de Sun se puede recuperar la versión necesaria de JDK para que los ordenadores de todos los usuarios finales utilicen la misma versión.

Obtención del navegador apropiado

En principio, los clientes deben instalar una versión mínima del navegador. Si ya están acostumbrados a un determinado navegador, deberá convencerlos de que los trastornos que pueda crear conseguir las actualizaciones se verán compensados por el valor de su applet. Cada vez que se actualiza el navegador es necesario descargar la actualización del navegador y/o instalar un parche.

Compatibilidad con varios navegadores

Si opta por que su applet sea compatible con más de un navegador, deberá brindar algún tipo de asistencia. Esto significa que si los usuarios tienen problemas relacionados con el navegador que utiliza su applet, llamarán para solicitarla.

Diferencias en la implementación de Java

Existe una serie de pautas y especificaciones sobre las prestaciones que deben suministrar los navegadores, pero la implementación de éstos y sus extensiones queda en manos del fabricante del navegador.

Una de las diferencias que existen entre los navegadores es la implementación del Administrador de seguridad y los niveles de seguridad. Lo que en un navegador se permite con una seguridad “media” podría no estar permitido en otro. El ajuste de la seguridad se realiza en el equipo cliente y cada navegador lo resuelve de forma diferente. Por ejemplo, el mecanismo para rebajar un poco la seguridad en un applet es la autenticación. Autenticar un applet, y lograr que el navegador acepte dicha firma, le otorga más flexibilidad al applet. El mecanismo apropiado se incluye en Java con la utilidad **Javakey**. Lamentablemente, los navegadores no están obligados a utilizar este mecanismo. De hecho, algunos utilizan sus propios mecanismos de autenticación que sólo funcionan en su navegador.

Otra diferencia es que los fabricantes de navegadores modifican a veces algunas funciones básicas de Java o incluso excluyen algunas de ellas. Esto puede dar lugar a comportamientos inesperados durante la ejecución. Por ejemplo, podría tener llamadas a la función `pintar` que no se producen con lo que el applet aparecería de modo extraño o incluso ni se vería. Podría tratar de corregir el código para cada situación específica, pero esto no siempre es posible.

Además, en función de la plataforma en la que se implemente Java, su utilización puede variar notablemente aunque se trabaje con el mismo navegador. Por ejemplo, un applet multihilo puede ejecutarse en una plataforma, pero debido a problemas del modelo de proceso de conexión en una plataforma diferente, es posible que los procesos se ejecuten secuencialmente en una plataforma diferente. A menudo, los navegadores no se actualizan para todas las plataformas a la vez, por lo que las actualizaciones de las plataformas menos utilizadas pueden aparecer más tarde.

Estas diferencias incrementan el tiempo que debe invertir en tareas de comprobación y depuración con un navegador determinado.

Soluciones a los problemas con los navegadores

En esta sección se ofrecen soluciones a los problemas que podría encontrar al ejecutar su applet en un navegador.

- Utilice el Plug-in de Java

La mayoría de los problemas con los navegadores se puede resolver con el Plug-in de Java, que se encuentra en la página <http://java.sun.com/products/plugin/>. Este plug-in proporciona archivos recopilatorios actualizados del JDK, lo que resulta de gran utilidad si sus applets utilizan Swing (JFC).

Deberá modificar los archivos HTML para que el applet utilice el plug-in. Sun proporciona una utilidad que realiza la conversión de forma rápida y sencilla. Si desea obtener más información acerca del

Convertidor HTML, consulte http://java.sun.com/j2se/1.4/docs/guide/plugin/developer_guide/html_converter.html.

El uso del Plug-in de Java es la única manera de disponer del mismo JDK en los diferentes navegadores. Además, generalmente el JDK que se obtiene es más reciente que el JDK del navegador. Algunas extensiones incorporadas por los fabricantes de navegadores pueden no funcionar del mismo modo con el plug-in.

La primera vez que un usuario final encuentre una página que interpreta el Plug-in de Java, se le solicitará que instale el plug-in. Una vez instalado, el equipo cliente dispondrá localmente de una MV oficial de Sun y del JDK más reciente (incluido Swing). Esto sólo se utilizará en las páginas que interpreten el “plug-in”, pero reduce la sobrecarga que representa la transferencia al navegador de la tecnología más reciente del JDK.

Importante

Existen varias versiones de plug-in: JDK 1.1.x, 1.2x, 1.3x y 1.4. Debe utilizar el que corresponda a la versión de JDK utilizada por su applet. Además, la versión del Convertidor HTML debe coincidir con la versión del plug-in. Observe también que sólo puede tener una versión del Plug-in en la máquina cliente.

- Utilice la misma versión del JDK admitido en los navegadores
- Evitará muchos problemas al escribir applets utilizando el mismo JDK que admiten los navegadores. JBuilder SE y Enterprise le permiten cambiar la versión de JDK utilizada para el desarrollo. Aunque JBuilder Personal no admite el cambio de JDK, sí permite editar el JDK en uso. Consulte Herramientas | Configurar JDK y “Modificación del JDK” en el capítulo “Creación y gestión de proyectos” de *Creación de aplicaciones con JBuilder*.
- Instale en el ordenador cliente las clases que no pertenecen al grupo principal de Java

Cada navegador tiene un directorio que Java utiliza en las búsquedas basadas en `CLASSPATH` por lo que, si se colocan allí los grandes archivos de revisiones que utiliza localmente el applet, el cliente no tendrá que descargarlos. Esto es posible sólo en un entorno Intranet, o cuando desde Internet trabaje sólo con clientes conocidos (como en un servicio basado en suscripciones). Sin embargo, esto dificulta las futuras actualizaciones del código.

- Seleccione un navegador

Los usuarios deben utilizar sólo un navegador para minimizar la necesidad de código y mantenimiento especiales. Normalmente esto sólo es posible en una intranet en la que el administrador del sistema puede establecer la política interna.

- Use Java Web Start

Java Web Start es una tecnología de distribución de Sun Microsystems. Permite ejecutar applets (y aplicaciones) Java en el navegador web desde un enlace en una página web. Java Web Start resuelve el problema de la falta de coincidencia de la versión del JDK. Si el navegador tiene el plug-in de Java Web Start, puede descargar automáticamente el JDK apropiado para ejecutar el applet.

Para obtener más información sobre Web Start, consulte la página Java Web Start en el [Capítulo 15, “Ejecución de aplicaciones web con Java Web Start,”](#) y visite la página de Java Web Start en <http://java.sun.com/products/javawebstart/>.

Consejos adicionales para que funcionen las applets

A continuación se incluyen algunos consejos adicionales para evitar problemas durante el desarrollo y distribución de las applets:

- Sitúe los archivos en la ubicación correcta

Un problema común con un applet distribuido es colocar los archivos en la ubicación equivocada (por ejemplo, “no se puede encontrar xyz.class”). Todo está relacionado con el archivo HTML que abre el applet. Si no existe un atributo `CLASSPATH` que ayude a encontrar las clases del applet, el applet se basa en los atributos `codebase` y `code` del archivo HTML para localizar las clases que necesita en la ejecución. Por lo tanto, antes de comenzar la distribución, es importante situar los archivos en la ubicación correcta.

El proceso de distribución le puede resultar más fácil si configura un entorno de trabajo que refleje las condiciones reales que tendrá el applet una vez distribuido, con lo que el desarrollo se acerca un poco más a la distribución instantánea. Por ejemplo, cuando el valor de `codebase` es “.”, el navegador busca la clase applet y los archivos recopilatorios en el mismo directorio que el archivo HTML. También, si la clase está en un paquete, el navegador construye una vía de acceso al directorio subordinado al directorio del archivo HTML de acuerdo con la estructura del paquete. Si utiliza el Asistente para applets de JBuilder, éste crea automáticamente la estructura de este archivo. Cuando el applet está generado y en ejecución en esta posición, se pueden recopilar todos los elementos en un archivo JAR o ZIP. Luego, salga de JBuilder y pruebe el applet con los navegadores que vaya a utilizar.

Sugerencia

Es mejor utilizar una copia de la página HTML que se encuentra en el servidor, en vez de emplear una página experimental, más sencilla. Esto aumenta el realismo de la prueba externa. Cuando esté satisfecho, envíe por todo el directorio a la página Web.

- Utilice paquetes

Siempre debe utilizar paquetes cuando escriba código en Java con el fin de organizar las clases similares. Esto facilita su localización y simplifica la distribución. Los paquetes también contribuyen a evitar conflictos con los nombres de clases ya que el nombre del paquete precede al nombre de la clase en Java.

- Utilice el navegador más actual

Conozca los navegadores en los que se ejecutará su applet. Si utiliza herramientas actuales como JBuilder, probablemente genere código JDK 1.3x o posterior y le harán falta los navegadores más actuales para ejecutar sus applets. Asegúrese de proporcionar el plug-in más reciente para que coincidan la versión de JDK del navegador y la del applet. Consulte el Plug-in de Java que se encuentra en la dirección <http://java.sun.com/products/plugin/>.

- Haga coincidir las mayúsculas/minúsculas

Recuerde que Java distingue mayúsculas y minúsculas. Compruebe que todas las mayúsculas y minúsculas de los nombres de clases, paquetes, archivos recopilatorios y directorios de la etiqueta <applet> coinciden exactamente con los nombres del servidor.

- Utilice recopilatorios

Los recopilatorios simplifican la distribución porque sólo hay que cargar el archivo HTML y el archivo recopilatorio. También aceleran el proceso de descarga del cliente; se descarga un archivo comprimido en lugar de los diferentes archivos de clase no comprimidos. Compruebe siempre la precisión de la estructura de directorios y de las mayúsculas y minúsculas de los nombres de archivo en el archivo recopilatorio.

Si desea obtener más información acerca de la creación de recopilatorios, consulte “[Distribución de las applets](#)” en la página 14-14 y “[Distribución de applets en JBuilder](#)” en la página 14-26.

Importante

Las versiones más antiguas de algunos navegadores no admiten varios archivos de revisiones y sólo admiten archivos ZIP no comprimidos.

- Distribuya todo

A menos que escriba algo que sólo utilice clases básicas de Java, debe distribuir en la web todos los archivos que necesite su applet. Por supuesto, debe distribuirlo todo en la ubicación adecuada. Consulte “[Distribución de las applets](#)” en la página 14-14 para obtener más información.

- Vuelva a comprobar el applet después de distribuirlo en el servidor

No es suficiente con probarlo localmente. También debe probarlo en varios navegadores después de distribuirlo en el servidor. Esto es esencial si desea asegurarse de que todas las clases necesarias para el applet se distribuyan adecuadamente en el servidor y que la etiqueta `<applet>` coincida con la distribuida en el servidor, además de encontrar posibles problemas relacionados con el navegador.

Para obtener más información, consulte “[Comprobación de las applets](#)” en la página 14-15.

La seguridad y el Administrador de seguridad

La seguridad es un aspecto importante cuando se ejecutan programas en Internet. Los usuarios se muestran extremadamente cautelosos a la hora de descargar y ejecutar en su equipo programas desconocidos, sin una garantía de seguridad que impida sucesos como la destrucción de archivos por parte de estos programas o la transferencia de información personal desde el ordenador.

La seguridad representa una ventaja para el desarrollador porque, sin ella, muchos de los usuarios finales no permitirían que las applets se ejecutaran en sus equipos. No obstante, también le aporta varias desventajas, especialmente si el desarrollador no está al tanto de las restricciones al comenzar el proyecto. Muchas actividades que se dan por hechas en una aplicación, se denegarán en un applet.

El espacio de contención

Applets aborda este problema ejecutando todo el código no certificado en un entorno restringido llamado *sandbox* (espacio de contención). En este lugar, las applets permanecen bajo la atenta mirada del *Administrador de seguridad*. El Administrador de seguridad protege el ordenador cliente de los programas dañinos que podrían eliminar archivos, leer información protegida o hacer algo que constituya una violación del modelo de seguridad utilizado. A su vez, las restricciones impuestas por el Administrador de seguridad limitan lo que puede hacer un applet.

Ahorrárá tiempo y dinero si comprende estas limitaciones antes de escribir applets. Las applets, al igual que cualquier herramienta, realizan determinadas tareas muy bien. Sin embargo, si se intenta utilizar un applet en el contexto equivocado, sólo habrá problemas.

Restricciones de las applets

Por defecto, todas las applets son “no fiables” y se impide que realicen determinadas actividades, como:

- Leer y escribir en el disco local

No se pueden leer archivos, comprobar su existencia, escribir archivos, cambiar sus nombres, etc.

- Conectarse con un ordenador que no sea el de procedencia del applet

Esto dificulta acceder a datos de una base que no resida en el servidor web.

Existen otras actividades que los desarrolladores de aplicaciones dan por descontadas (como la ejecución de programas del sistema local, para ahorrar tiempo) que no se permiten con las applets. Un buen libro sobre Java incluirá todas las restricciones que tienen las applets.

Si percibe que está intentando evadir el control del Administrador de seguridad en algún punto de su planificación, es preferible que considere escribir una aplicación en lugar de un applet. Sun se ha esforzado mucho para definir su modelo de seguridad e identificar los bits de información clave, o tipos de datos, de los que dependen conjuntos completos de la funcionalidad. Por lo tanto, resulta muy difícil engañar al modelo de seguridad.

Soluciones a problemas de seguridad

Para gestionar problemas de seguridad:

- Autentique el applet

Esto le permitirá mitigar algunas de las restricciones planteadas por el Administrador de seguridad. Tiene algunas desventajas, como es la carencia de un mecanismo de autenticación estándar que funcione en todos los navegadores. Si desea obtener más información acerca de la autenticación de archivos recopilatorios de applets, visite la sede web de su navegador.

Consulte “Code signing for Java applets” en http://www.suitable.com/Doc_CodeSigning.shtml para obtener mas información sobre la firma de applets.

- Logre que los usuarios cambien el modelo de seguridad en su ordenador

Los navegadores tienen parámetros que pueden utilizarse para ajustar el modelo de seguridad. Estos parámetros incluyen desde opciones muy simples que pueden ser “elevada”, “media” y “baja” hasta estructuras bastante flexibles. Lamentablemente, estos valores se

aplican por igual a todas las applets. Aunque los usuarios pueden confiar en que su applet no resultará dañina con un nivel de seguridad bajo, es difícil convencerlos de que piensen lo mismo sobre todas las applets que puedan encontrarse.

- Utilice otras tecnologías para evitar al Administrador de seguridad

Por ejemplo, si se necesita simplemente devolver datos al servidor, escriba aplicaciones Java que se ejecuten en el servidor, como los servlets, con las que se comunique su applet. Este código en el servidor no tiene las limitaciones del applet y puede ser bastante potente. Las aplicaciones en Java para el servidor requieren que tenga acceso al equipo servidor o trabaje con un PSI (Proveedor de servicios Internet) flexible y abierto.

- Reconozca las tareas no adecuadas para las applets

La recolección de datos de formularios en la web y la recogida de esos datos probablemente es una tarea más adecuada para los servlets o las Páginas JavaServer (JSP). Las operaciones complejas, como la utilización de bases de datos, requieren que en el servidor esté disponible software adicional como JDBC. Es una solución más compleja, pero sin ella, las acciones “normales” de una base de datos, como el envío de nuevos datos, no son posibles con las applets. Si se está escribiendo un servlet o una JSP que utilice JDBC para conectar con una base de datos, se debería considerar la posibilidad de utilizar InternetBeans Express para crear un servlet o una JSP con enlace a datos. Consulte los capítulos restantes en este manual si desea obtener más información sobre estos temas.

- Considere la posibilidad de escribir una aplicación, un servlet o una JSP en lugar de un applet

Las ventajas de las aplicaciones Java, incluido el acceso completo al lenguaje Java y la ausencia de problemas de seguridad, pueden superar las ventajas de las applets en determinadas situaciones. Los servlets y las JSP también tienen ventajas sobre los applets. No dependen del JDK del navegador del cliente, ya que Java se ejecuta en el servidor.

También, las servlets y las JSP son a menudo mas rápidas porque no necesitan descargarse en el navegador del cliente, como les ocurre a las applets.

Utilización de bibliotecas de terceros

Las bibliotecas de terceros son una gran ventaja para el desarrollador. A menudo, estas bibliotecas contienen componentes diversos que ahorran tiempo y dinero. Sin embargo, es importante sopesar las ventajas de las bibliotecas de componentes en relación con las desventajas que suponen para el usuario final.

Con las applets, todas las clases referenciadas deben enviarse al ordenador cliente a través de la red. Si la biblioteca de terceros tiene una arquitectura de entrelazado grande, puede incrementar sustancialmente el tamaño del archivo del applet. Además, debe tener en cuenta la licencia de redistribución de la biblioteca. Algunas bibliotecas exigen que se las redistribuya por completo (no sólo las partes que utilizará).

Recuerde Cuanto más grande sea el applet, mayor será el tiempo de transferencia y ejecución.

Distribución de las applets

La distribución es el proceso de reunir las clases de applet y sus dependientes en la jerarquía correcta para su publicación en un servidor. JBuilder SE y Enterprise incluyen el Creador de recopilatorios para la distribución en el IDE de JBuilder después del desarrollo del applet. Sun proporciona la herramienta **jar** en el JDK.

Existen varias medidas que se pueden tomar con el fin de simplificar la distribución:

- Desarrolle el applet en un entorno local que reproduzca exactamente la estructura de directorios de su sede web.
- Utilice archivos recopilatorios.
- Utilice el Creador de recopilatorios en JBuilder SE y Enterprise o la herramienta **jar** de JDK para crear los archivos recopilatorios.

Importante Si se escriben applets JDK 1.1.x con componentes Swing, lo que no se recomienda debido a las limitaciones de algunos navegadores, también será necesario descargar y repartir las clases JFC (Swing), `swingall.jar`, ya que no fueron suministradas como una parte del JDK.

Consulte

- “Distribución de applets en JBuilder” en la página 14-26
- “Depuración de programas en Java”, en *Creación de aplicaciones con JBuilder*
- **jar** en la documentación del JDK.
- “Java Web Start y JBuilder” en la página 15-5

Comprobación de las applets

El principal objetivo de un IDE es permitir el desarrollo y la comprobación local del código. Cuando se ejecuta una aplicación o un applet en JBuilder, si el programa necesita una clase determinada, JBuilder la busca donde haga falta. Esto permite que el usuario se pueda concentrar en el desarrollo de la aplicación o applet y se asegure de su funcionamiento.

Sin embargo, no le permite saber cómo se comportarán exactamente las applets durante la ejecución. Con el fin de comprobar correctamente un applet, debe efectuar las pruebas en el servidor web, es decir, en su entorno real de ejecución. Ésta es la única manera de garantizar que todas las clases que necesita el applet están disponibles en el servidor que lo ejecuta.

Cuando se ejecuta el applet en un navegador, el applet necesita disponer de información suficiente dentro de la etiqueta `<applet>` del archivo HTML, de modo que el cargador del applet encuentre todas las clases, imágenes y demás archivos necesarios. Éste es el “entorno real” del applet y es el único lugar en el que el applet obtiene la información necesaria para encontrar los archivos.

En las primeras pruebas del applet distribuido puede utilizar el “navegador” de prueba del JDK denominado **appletviewer**. Esta herramienta de línea de comandos incluida en JDK proporciona útiles pantallas de mensajes de error y seguimiento de la pila. Como se demuestra en los siguientes pasos, elimine `CLASSPATH` en la sesión de prueba.

Pasos básicos de la comprobación

Los pasos básicos para probar su applet son:

- 1 Distribuya los archivos necesarios del applet en el servidor.

Compruebe lo siguiente:

- Estos archivos se transfieren al servidor web en modo binario cuando se utiliza un cliente FTP.

- Los archivos se encuentran en las ubicaciones correctas *en relación con el archivo HTML* que ejecuta el applet y el valor de codebase es correcto.
- Las mayúsculas y minúsculas de los nombres en el servidor coinciden con las de los nombres originales.
- La etiqueta <applet> contiene el atributo archive con los nombres de los archivos de revisiones si el applet se distribuye en un archivo JAR o ZIP.
- La estructura de directorios coincide con la estructura de los paquetes.

La distribución se analiza en profundidad en “Distribución de programas Java” en *Creación de aplicaciones con JBuilder*.

- 2 Abra una ventana de línea de comandos.
- 3 Borre cualquier CLASSPATH que haya asignado a esa sesión. El **appletviewer** sabe dónde encontrar los archivos Java básicos.
- 4 Cambie al directorio que contiene el archivo HTML y el archivo JAR (o archivos de clase).
- 5 Ejecute el **appletviewer**.

```
<jbuilder>/<jdk>/bin/appletviewer TestApplet.html
```

en donde <jbuilder> es el nombre del directorio de JBuilder y <jdk> es el nombre del directorio de JDK. Por ejemplo, JBuilder8/jdk1.4/.

Nota Si JBuilder está en otra unidad, incluya la letra de la unidad. En Windows, se utiliza la barra invertida (\).

Si es la primera vez que ejecuta **appletviewer**, aparece una pantalla con el acuerdo de licencia. Pulse Sí.

Si todo va bien, se podrá ver el applet ejecutándose en el **appletviewer**. Si no, lea los mensajes de error que aparecen en la consola de **appletviewer**.

- Si aparece el error “no se encuentra la clase”, verifique la distribución.
- Si hay una excepción en tiempo de ejecución, tal como NullPointerException, depure el código para encontrar el error.

Comprobación en los navegadores

Pruebe siempre el applet en los navegadores después de distribuirlo en el servidor. Esto resulta esencial si desea asegurarse de que todas las clases necesarias para el applet están bien distribuidas en el servidor y que la etiqueta <applet> coincide con la distribuida en el servidor.

A continuación se incluyen algunas sugerencias para la prueba en el navegador:

- Asegúrese de que el navegador tiene activado Java. Consulte la ayuda en línea del navegador para obtener más detalles.
- No utilice los botones Recargar o Actualizar después de recompilar y distribuir el applet revisado. El navegador podría continuar cargando la versión anterior del applet desde la memoria caché.
- Utilice la Consola Java para leer los mensajes de error del navegador. Para abrir la Consola Java, lleve a cabo una de las acciones siguientes:
 - Seleccione Communicator | Herramientas | Consola de Java en las versiones antiguas de Netscape o, bien, Tareas | Herramientas | Consola de Java en las versiones más recientes.
 - Seleccione Ver | Consola Java en Internet Explorer.

Si desea obtener más información, consulte “Solving Common Applet Problems” en el Tutorial de Java “Writing Applets”

JBuilder y las applets

JBuilder proporciona una gran variedad de herramientas para distribuir las applets:

- Un Asistente para applets para crear rápidamente un applet.
- Un diseñador para diseñar visualmente una interfaz de usuario de un applet.
- Applet TestBed de JBuilder para ejecutar y depurar applets.
- **Appletviewer** de Sun para ejecutar y depurar applets.

Si desea obtener más información sobre la creación de applets en JBuilder, consulte

- “Tutorial: Creación de un applet” en *Introducción a JBuilder*
- “Introducción al diseñador” en *Diseño de aplicaciones con JBuilder*

Creación de applets con el Asistente para applets

JBuilder suministra el Asistente para applets con el fin de generar el código básico de un applet. Complete los siguientes pasos para crear un applet mediante el Asistente para applets.

Advertencia

Si crea una applet para la Web, consulte “[Los navegadores](#)” en la [página 14-5](#) para información acerca del navegador y la compatibilidad con JDK antes de ponerse a diseñarla.

- 1** Seleccione Archivo | Nuevo proyecto si debe crear un proyecto para su applet. Aparece el Asistente para proyectos, el cual sugiere un directorio y un nombre de proyecto por omisión. Cambie los nombres del archivo y del directorio del proyecto si quiere darles un nombre y ubicación determinados. Pulse la opción Generar archivo de notas del proyecto, con el fin de crear un archivo HTML para las notas del proyecto.

Nota	Las vías de acceso para los archivos de proyecto están previamente asignadas en las propiedades del proyecto por defecto. Es recomendable que sólo los usuarios de Java experimentados modifiquen este parámetro. Si desea obtener más información sobre las propiedades por defecto para proyectos, consulte “Creación y gestión de proyectos” en <i>Creación de aplicaciones con JBuilder</i> .
	El nombre del paquete del proyecto se deriva del nombre de archivo, y se muestra en el Asistente para applets. Por ejemplo, si crea un proyecto denominado <home>/jbproject/AppletProject, el Asistente para applets le propondrá utilizar el nombre de paquete appletproject.
	Si desea más información acerca de los paquetes, consulte el apartado “Paquetes” en el capítulo “Creación y gestión de proyectos” de <i>Creación de aplicaciones con JBuilder</i> .
a	Pulse Siguiente para pasar a la ficha Vías de acceso del proyecto del Asistente para proyectos. Las vías de acceso son las siguientes:
Vía de acceso a archivos generados	Donde se guardan los archivos de clase y el archivo HTML del applet.
Vía de acceso a las copias de seguridad	Donde se almacenan las copias de seguridad.
Directorio de trabajo	El directorio de trabajo del proyecto.
Vía de acceso a archivos fuente	Donde se guardan los archivos fuente y de prueba—Por defecto para los archivos fuente y Test para los archivos de prueba de la unidad.
Vía de acceso a la documentación	Donde se guardan los archivos de documentos.
Las vías de acceso a archivos generados, copias de seguridad y directorio de trabajo, así como la vía de acceso al JDK, se pueden modificar pulsando el botón puntos suspensivos. Las vías de acceso a archivos fuente, de prueba y documentación se pueden modificar seleccionando la vía de acceso y, a continuación, pulsando el botón Modificar. También es posible añadir las bibliotecas necesarias.	

Consulte

- “Configuración del JDK” en *Creación de aplicaciones con JBuilder*
 - “Cómo construye JBuilder las vías de acceso” en *Creación de aplicaciones con JBuilder*
- b** Haga clic sobre Finalizar para completar el Asistente para proyectos.
- El Asistente para proyectos crea un archivo de proyecto (.jpx). Si seleccionó la opción Generar archivo de notas del proyecto en la primera ficha del asistente, se tratará de un archivo HTML cuyo nombre coincide con el del proyecto y que contiene la información por defecto del proyecto. Puede modificarlo y guardar la información pertinente del proyecto que desee ver.
- 2** Despu s, seleccione Archivo | Nuevo y pulse la pesta a Web de la galer a de objetos.
- 3** Haga doble clic en el icono Applet de la galer a de objetos, lo que abrir  el Asistente para applets.
- a** No cambie el nombre del paquete que se sugiere. Cambie el nombre de la clase del applet, si lo desea.
 - b** Seleccione la clase b sica que desea ampliar en el applet:
java.applet.Applet (clase de AWT) o java.swing.JApplet (clase Swing de JFC).
 - c** Si lo desea, active las dem s opciones:
- | | |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Generar comentarios de cabecera | Utiliza informaci n del archivo del proyecto como comentarios de cabecera, en la parte superior del archivo de clase applet. |
| Puede ejecutarse aut nomamente | Crea una funci n main() que permite comprobar el applet sin necesidad de abrirla desde una p gina HTML. |
| Generar m todos est ndar | Crea stubs para los m todos est ndar del applet: start(), stop(), destroy(), getAppletInfo() y getParameterInfo(). |
- d** Pulse Siguiente para pasar a la ficha Par metros del applet. En este paso, puedeadir par metros. Partiendo de esta informaci n, el asistente genera etiquetas <param> dentro de la etiqueta <applet> del archivo HTML del applet, y el c digo de tratamiento de par metros en el nuevo archivo java del applet.
- Rellene una fila por cada par metro que dese .
- Si deseaadir un par metro, pulse el bot n A adir par metro.
 - Para seleccionar una celda, haga clic en ella o despl cese por medio de las teclas de desplazamiento.

- Para introducir un valor en una celda, escríbalo o selecciónelo en la lista desplegable, si la hay.
- Si desea quitar un parámetro, haga clic en una celda de su fila y pulse el botón Eliminar parámetro.

Las definiciones de los campos de parámetros son:

Nombre	El nombre del parámetro. se usará para proporcionar el atributo <code>name</code> en la etiqueta <code><param></code> en el archivo HTML y para proporcionar el parámetro <code>name</code> de la llamada correspondiente <code>getParameter()</code> en el código fuente de Java.
Tipo	Tipo de variable que se inserta en el código fuente Java del applet para recoger el valor del parámetro procedente de la página HTML.
Desc	Breve descripción del parámetro. Se utiliza para describir el parámetro cuando, mediante herramientas externas, se hace una consulta sobre los parámetros que acepta el applet. Una de estas herramientas, por ejemplo, está en Applet Info browser del appletviewer .
Variable	Nombre de la variable que se inserta en el código fuente Java del applet para recoger el valor del parámetro procedente de la página HTML.
Valor por defecto	El valor por defecto del parámetro. Éste es el valor que el código fuente Java de esta applet utiliza si un futuro archivo HTML que use el applet no dispone de una etiqueta <code><param></code> para este parámetro. Los archivos HTML no facilitan este parámetro si el atributo <code>name</code> de la etiqueta <code><param></code> no coincide exactamente con lo introducido en la columna Nombre. Para que haya coincidencia, las mayúsculas y las minúsculas también deben coincidir.

- e** Pulse Siguiente para pasar a la ficha Detalles HTML. Si no desea generar la página HTML automáticamente, puede desactivar la opción Generar página HTML. De lo contrario, deje esta opción activada e introduzca información sobre la página, como su nombre, el del applet, el tamaño del applet en la página y la posición de codebase. Por defecto, el Asistente para applets guarda el archivo HTML en el directorio de salida con las clases compiladas.

- f** Pulse Siguiente para pasar a la ficha Definir configuración del applet.
 - En JBuilder SE y Enterprise, active la opción Crear una configuración de ejecución y escriba un nombre para la configuración en el campo Nombre. Para obtener más información, consulte “[Creación de una configuración de ejecución de applets](#)” en la página 10-3.
 - En JBuilder Personal, asegúrese de que está activada la opción Sobreescribir la configuración existente. Escriba el nombre de la configuración en el campo Nombre.
 - g** Pulse Finalizar para generar los archivos .java y HTML del applet.
- El Asistente para applets crea dos archivos si se selecciona la opción Generar página HTML en la ficha Detalles HTML:
- Un archivo HTML que contiene una etiqueta <applet> con una referencia a la clase applet. Éste es el archivo que debe seleccionar para ejecutar o depurar el applet.
 - Una clase Java, que se deriva de Applet o JApplet y se puede diseñar por medio del diseñador de interfaces de usuario.

Nota

En JBuilder SE y Enterprise, aparece también un nodo de paquetes de código fuente en el panel de proyecto si está seleccionada la opción Activar la localización y compilación de paquetes fuente en la ficha General del cuadro de diálogo Propiedades de Proyecto (Proyecto | Propiedades de proyecto).

Consulte

- “[Creación de applets con el Asistente para applets](#)” en la página 14-17
- “Tutorial: Creación de un applet” en *Introducción a JBuilder*

Ejecución de applets

Para ejecutar un applet en un proyecto,

- 1** Guarde todos los archivos.
- 2** Compile el proyecto.
- 3** Realice una de las operaciones siguientes:



- Seleccione Ejecutar | Ejecutar proyecto o pulse el botón Ejecutar de la barra de herramientas para ejecutar el applet desde la clase principal en el AppletTestbed de JBuilder. (En JBuilder SE y Enterprise, elija la configuración de ejecución, si hay varias.)

- Haga clic con el botón derecho del ratón en el archivo HTML del applet en el panel de proyecto. Este archivo debe tener una etiqueta <applet>. Seleccione Ejecutar para ejecutar el applet en el **appletviewer** de Sun. (En JBuilder SE y Enterprise, elija la configuración de ejecución, si hay varias.)

Nota También se puede seleccionar el archivo .java del applet si tiene un método `main()` y selecciona Ejecutar. En la ficha Introduzca información sobre la clase del applet, del Asistente para applets, se puede crear un applet con un método `main()`, si selecciona la opción Puede ejecutarse autónomamente.

El applet se compila y, si no se producen errores, se ejecuta. El progreso de la generación se muestra en un cuadro de diálogo, mientras que el panel de mensajes muestra los errores de compilación. Si el programa se compila correctamente, se muestra la vía de acceso a clases en el panel de mensajes y el programa se ejecuta.

El AppletTestBed de JBuilder y el appletviewer de Sun

Hay dos formas de ejecutar un applet en JBuilder:

- Con el AppletTestbed de JBuilder
- y con el **appletviewer** de Sun

El comportamiento por defecto es como sigue, si se ha creado el applet con el Asistente para applets:

- 
- Seleccione Ejecutar | Ejecutar proyecto o pulse el botón Ejecutar para ejecutar el applet desde la clase principal en el visualizador de applets de JBuilder, AppletTestbed. (En JBuilder SE y Enterprise, elija la configuración de ejecución, si hay varias.)
 - Haga clic con el botón derecho en el archivo HTML del applet y seleccione Ejecutar para ejecutar el applet en el **appletviewer** (visualizador de aplicaciones) de Sun. (En JBuilder SE y Enterprise, elija la configuración de ejecución, si hay varias.)

Puede cambiar el comportamiento por defecto de Ejecutar | Ejecutar proyecto y del botón Ejecutar si personaliza la configuración de ejecución del applet. Hay dos posibilidades para ejecutar un applet en JBuilder:

- Clase principal—utiliza el AppletTestbed de JBuilder
- Archivo HTML—utiliza el **appletviewer** de Sun

Clase principal

Cuando se selecciona un clase principal para ejecutar el applet, se ejecuta en el visualizador de applets de JBuilder, AppletTestbed. Cuando se crea un applet usando el Asistente para applets, éste fija la clase principal automáticamente. La clase seleccionada debe contener un método `init()`.

Archivo HTML

Cuando se selecciona un archivo HTML para ejecutar el applet, se ejecuta en el **appletviewer** de Sun. El archivo HTML debe contener la etiqueta `<applet>` y el atributo `code` debe contener el nombre completo de la clase. Si el archivo de clase no se encuentra en el mismo directorio que el archivo HTML, el atributo `codebase` debe indicar la posición del archivo de clase en relación con el archivo HTML.

Ejecución de applets JDK 1.1.x en JBuilder

Si se ejecuta un applet JDK 1.1.x desde su clase principal en JBuilder, lo hace utilizando el AppletTestbed de JBuilder, que necesita JFC (Swing) para abrirse. Debido a que JDK 1.1.x **no** incluía las clases JFC, necesita descargar la versión específica para JDK 1.1.x de JFC (Swing), `swingall.jar`, desde [http://java.sun.com.products](http://java.sun.com/products). Acto seguido, cree una biblioteca para las clases de JFC y añádala al proyecto (Herramientas | Configurar bibliotecas).

Ejecución de applets JDK 1.2 en JBuilder

Para ejecutar un applet en Solaris o Linux desde JBuilder, debe añadir la biblioteca Open Tools SDK al proyecto. Si no se añade esta biblioteca, puede producirse una excepción del tipo

`NoClassDefFoundError:AppletTestbed`. Este problema afecta a algunos de los ejemplos de applet, incluido el Primes Swing.

Depuración de applets

Se pueden depurar applets en el IDE de JBuilder de igual forma que procede con los programas Java. Al igual que hay dos formas de ejecutar los applets en JBuilder, también se pueden depurar de dos maneras: El AppletTestbed de JBuilder y el **appletviewer** de Sun.

Para depurar con AppletTestbed de JBuilder, se debe ejecutar la clase principal del applet que contiene el método `init()`:

- 1 Defina la clase principal del applet, la cual contiene el método `init()`, como clase ejecutable (utilice una configuración de ejecución).
- 2 Compile el proyecto.
- 3 Defina un punto de interrupción en el archivo applet.
- 4  Elija Ejecutar | Depurar proyecto o pulse el botón Depurar de la barra de herramientas. El depurador toma el control.

Nota

Si se está desarrollando un applet usando un JDK anterior, para depurar necesitará cambiar a un JDK más reciente (JDK 1.2.2 o superior). Los JDK más antiguos no admiten la API de depuración de JPDA (Arquitectura del depurador para plataforma Java) que utiliza JBuilder. Consulte "Configuración del JDK" en *Creación de aplicaciones con JBuilder*.

Para obtener instrucciones más detalladas sobre la depuración en JBuilder, consulte "Depuración de programas Java" en *Creación de aplicaciones con JBuilder*.

Para depurar con el **appletviewer** de Sun, debe ejecutar el archivo HTML utilizando uno de estos métodos:

- Desde el panel del proyecto, siga estos pasos:
 - a Compile el proyecto.
 - b Defina un punto de interrupción en el archivo applet.
 - c Haga clic con el botón derecho en el archivo HTML del applet, en el panel del proyecto, y elija Depurar. El depurador toma el control.
- Desde el menú Ejecutar de JBuilder, siga estos pasos:
 - a Configure el archivo HTML del applet como el archivo ejecutable, modificando la configuración de ejecución.
 - b Compile el proyecto.
 - c Defina un punto de interrupción en el archivo applet.
 - d Elija Ejecutar | Depurar proyecto o pulse el botón Depurar de la barra de herramientas. El depurador se carga en el panel de mensajes y el applet se ejecuta en el **appletviewer** de Sun.



Depuración de applets en el Plug-in de Java

También se pueden depurar las applets desde Internet Explorer 5 o Netscape Navigator 6 y superior mediante el Plug-in de Java y el depurador de JBuilder.

Para depurar usando JDK 1.4:

- 1 Descargue e instale el plugin para JDK 1.4 desde la sede web de Sun:
<http://java.sun.com/products/plugin/>.
- 2 Configure los siguientes parámetros de ejecución en el campo Java Runtime Parameters, de la pestaña Advanced, del Control Panel del Plug-in de Java:

```
-Xdebug -Xnoagent -Djava.compiler=NONE  
-Xrunjdwp:transport=dt_socket,server=y,address=3999,suspend=n
```

Si desea obtener más información acerca de la depuración del plug-in, consulte: http://java.sun.com/j2se/1.4/docs/guide/plugin/developer_guide/debugger.html#how.

3 Inicie JBuilder y haga lo siguiente:

- a** Cree un proyecto.
- b** Utilice el Asistente para applets con el fin de crear una applet sencilla que se ejecute desde un archivo HTML. Asegúrese de que la applet utiliza Swing para su clase base (`javax.swing.JApplet`). Añada un JPanel a la applet y configure la presentación en `GridLayout`. Añada un botón JButton al panel, y un suceso al botón que cambie el texto del botón.
- c** Modifique la configuración de ejecución del applet y defina las siguientes opciones de la ficha Depurar, en el cuadro de diálogo Propiedades de ejecución:
 - 1** Active la casilla de selección Activar depuración remota.
 - 2** Seleccione la opción Acoplarse.
 - 3** Compruebe que el campo Nombre del host está configurado en localhost.
 - 4** Asegúrese de que Tipo de transporte está configurado en dt_socket y que la Dirección es 3999.

Si desea información sobre las opciones de depuración remota, consulte “Depuración remota” en *Creación de aplicaciones con JBuilder*.

4 Compile el proyecto.

- 5 Ejecute el Convertidor HTML y convierta el archivo HTML del applet (en el directorio classes de su proyecto) al formato requerido por el Plug-in de Java. Si desea obtener más información acerca del Convertidor HTML y el modo de ejecutarlo, consulte la sede web de Sun en http://java.sun.com/j2se/1.4/docs/guide/plugin/developer_guide/html_converter.html.**

Nota La versión del convertidor de HTML debe coincidir con la versión del Plug-in.

- 6 Inicie Netscape o Internet Explorer y abra el archivo HTML. Asegúrese de que el navegador tiene activado Java. Consulte la ayuda en línea del navegador para obtener más información.**
- 7 Vuelva a JBuilder y configure un punto de interrupción en el archivo del applet. Comience a depurar el proyecto. El depurador debería iniciarse correctamente. Vuelva al visualizador y actualice la página. En JBuilder, el depurador se debería haber detenido en el punto de interrupción del applet.**

Distribución de applets en JBuilder

JBuilder SE y Enterprise tienen un Creador de recopilatorios que crea recopilatorios ZIP y JAR. También puede crear archivos JAR a mano mediante la herramienta **jar** de Sun, incluida en el JDK. Existen varias herramientas ZIP que crean archivos ZIP, pero asegúrese de utilizar una versión que acepte nombres de archivo largos.

Si se tiene una aplicación web grande con servlets, JSP, applets y otro contenido web, debería utilizar un archivo WAR (un archivo recopilatorio de aplicaciones web). Su aplicación web, archivo WAR o archivo JAR también pueden empaquetarse en un archivo EAR.

Consulte

- “Utilización del Creador de recopilatorios en *Creación de aplicaciones con JBuilder*”
- “Depuración de programas en Java”, en *Creación de aplicaciones con JBuilder*
- Herramienta **jar** en <http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#basic>
- [Capítulo 3, “Las WebApps y los archivos WAR”](#)

15

Ejecución de aplicaciones web con Java Web Start

El desarrollo web es una función de JBuilder Enterprise

Java Web Start es una tecnología de distribución de Sun Microsystems. Permite iniciar cualquier aplicación o applet de Java desde un vínculo en una página web en su navegador web. Si la aplicación no está presente en su ordenador, Java Web Start descarga todos los archivos necesarios y los almacena localmente en el caché de forma que el programa pueda volverse a iniciar desde un ícono en su escritorio o desde el vínculo de la página web.

Java Web Start es la implementación de referencia de la tecnología del Protocolo Java de inicio desde la red (JNLP). Esta tecnología define un formato de archivo estándar que describe cómo iniciar una aplicación JNLP. El Asistente para el inicio de Web Start de JBuilder genera automáticamente un archivo JNLP, así como un página de inicio HTML para la aplicación.

Si desea obtener más información sobre Web Start, visite la página Java Web Start en http://java.sun.com/products/javaWeb_Start/. También puede consultar en:

- The Java Web Start *Developer's Guide* en http://java.sun.com/products/javaWeb_Start/docs/developersguide.html
- “Frequently Asked Questions” en http://java.sun.com/products/javaWeb_Start/faq.html

Consideraciones sobre las aplicaciones Java Web Start

Generalmente, desarrollar una aplicación para su distribución con Java Web Start es lo mismo que desarrollar una aplicación autónoma. El punto de entrada de la aplicación es el método `main()` y la aplicación debe suministrarse como un archivo JAR o como un conjunto de archivos JAR. No obstante, todos los recursos de la aplicación deben llamarse usando el mecanismo `getResource`. Para obtener más información, consulte “Application Development Considerations” en la *Java Web Start Developer’s Guide* en http://java.sun.com/products/javaWeb_Start/docs/developersguide.html#dev.

Una aspecto clave para la ejecución de aplicaciones en Internet es la seguridad. Los usuarios son cautos a la hora de descargar y ejecutar programas en sus ordenadores sin una garantía de seguridad que evite que los programas borren archivos o sustraigan información personal.

Java Web Start aborda este problema ejecutando todo el código no certificado en un entorno restringido llamado *sandbox* (espacio de contención). Mientras la aplicación está en el sandbox, Java Web Start puede asegurar que la aplicación no comprometerá la seguridad de los archivos locales o de los archivos de la red.

Java Web Start también admite firma de codificación digital. Esta tecnología permite que Java Web Start verifique que el contenido del archivo JAR no haya sido modificado desde que se firmó. Si esta verificación falla, Java Web Start no ejecutará la aplicación. Para obtener más información acerca de Java Web Start y la seguridad, consulte “Security And Code Signing” en el apartado “Application Development Considerations” de la *Developer’s Guide* en Java Web Start http://java.sun.com/products/javaWeb_Start/docs/developersguide.html#dev.

La API del JNLP suministra servicios adicionales de manipulación de archivos para la ejecución en entornos de ejecución restringidos. Estas clases sustituyen a las operaciones ordinarias de descarga, apertura, escritura y grabación de los archivos. Por ejemplo, se utilizarían los métodos de `javax.jnlp.FileOpenService` para importar archivos del disco local, incluso para aplicaciones que se ejecuten en el sandbox.

Tabla 15.1 Perspectiva general de la API del JNLP

Nombre	Métodos para
BasicService	Consultas e interacciones con el entorno.
ClipboardService	Acceder al portapapeles del sistema.
DownloadService	Permitir que una aplicación controle cómo se almacenarán los recursos en la caché.
FileOpenService	Importar archivos desde el disco local.
FileSavService	Exportar archivos desde el disco local.
PrintService	Acceder a la impresora.

Tabla 15.1 Perspectiva general de la API del JNLP (continuación)

Nombre	Métodos para
PersistenceService	Almacenar localmente los datos.
FileContents	Encapsular el nombre y el contenido de un archivo.

Para obtener más información, consulte “JNLP API Examples” en la Java Web Start *Developer’s Guide* en <http://java.sun.com/products/javaWebStart/docs/developersguide.html#api>.

Instalación de Java Web Start

La instalación del JDK 1.4 de JBuilder incluye un ejecutable que instala Web Start. Ejecute el ejecutable con el fin de completar la instalación de Web Start.

En las plataformas Windows, siga estos pasos:

- 1 Desplácese al directorio `jdk1.4\jre` de su instalación de JBuilder.
- 2 Ejecute `javaws-1_2-windows-i586-i.exe`.
- 3 Acepte el acuerdo de licencia.
- 4 Se le solicita que designe un directorio de instalación. El valor por defecto es el directorio que utilizó por última vez para Web Start o \ Program Files\Java Web Start. Si acepta el valor por defecto, tendrá que modificar la definición de bibliotecas Web Start de JBuilder, tal y como se recoge en “[Modificación de la definición de bibliotecas Web Start de JBuilder](#)” en la página 15-4. Es más sencillo cambiar el directorio de instalación a `<jbuilder>\jdk1.4\jre\javaws`, en el que `<jbuilder>` es el directorio en el que instaló JBuilder.
- 5 Quite la marca de la opción Create Shortcuts For All Users si lo desea.
- 6 Seleccione Siguiente para crear los archivos Web Start en la ubicación especificada.
- 7 El programa de instalación le solicita que cierre su navegador de web, de modo que pueda configurarse y admita Web Start.
- 8 La instalación crea un grupo de programas Web Start y añade un ícono de forma optativa en el escritorio.

En Linux o Solaris, siga estos pasos:

- 1 Desplácese hasta el directorio `jdk1.4/jre` bajo el directorio de instalación de JBuilder.
- 2 Descomprima el archivo recopilatorio para crear `install.sh` y los archivos de texto relacionados.
- 3 Ejecute `install.sh`.

- 4 Acepte el acuerdo de licencia.
- 5 Cuando se le solicite la ubicación de su JDK, introduzca <jbuilder>/jdk1.4/jre/javaws en el que <jbuilder> es el directorio de instalación de JBuilder.
- 6 Si selecciona Siguiente, se generará un directorio javaws en la ubicación especificada y los archivos Web Start se crean en ese directorio.

Modificación de la definición de bibliotecas Web Start de JBuilder

Si ha instalado Web Start tal y como se recomienda anteriormente, no será necesario que modifique la definición de bibliotecas. Si no es así:

- 1 Seleccione Herramientas | Configurar bibliotecas con el fin de que se abra el cuadro de diálogo Configurar bibliotecas.
- 2 En el árbol de la izquierda, seleccione la biblioteca Web Start.
- 3 Seleccione la pestaña Clases a la derecha y pulse el botón Añadir.
- 4 Busque la ubicación en la que instaló javaws.jar, selecciónela y pulse Aceptar.
- 5 Pulse Aceptar para terminar de definir la biblioteca.

Nota Esta biblioteca no es necesaria en todas las aplicaciones o applets preparadas para Web Start. Sólo tiene que añadir la biblioteca al proyecto cuando esté utilizando la API JNLP. En este caso, debe también conseguir la descarga del desarrollador, con los archivos JAR y la documentación adicional, en la sede web de Web Start en <http://java.sun.com/products/javaWebStart/>.

Importante Es necesario que lleve a cabo alguna configuración adicional si está utilizando Netscape 6. Si desea obtener más información, diríjase al tema “Using Java Web Start Technology with NetscapeTM 6 Web Browsers” de la *Java Web Start Installation Guide* en <http://java.sun.com/products/javaWebStart/docs/installguide.html>.

Web Start y JDK 1.3 ó 1.2

Si su proyecto utiliza JDK 1.3 ó 1.2, es necesario que descargue e instale Java Web Start por separado. Para ello, diríjase a la página Java Web Start en <http://java.sun.com/products/javaWebStart/> y pulse el botón “Get Java Web Start Now!”. (Tenga en cuenta que la instalación depende de cada plataforma.) Una vez instalado Java Web Start en su equipo, ya está listo para ejecutarse con su navegador web por defecto. No es necesario que configure Java Web Start, JBuilder ni su navegador web: los tres se ejecutarán de forma transparente.

Java Web Start y JBuilder

JBuilder suministra un número de características que pueden convertir su aplicación autónoma o applet en una aplicación o applet Web Start. Para lograrlo, deben seguirse los siguientes pasos:

- 1** Configure un servidor web, por ejemplo, Tomcat 4.0, para su proyecto. Si desea obtener más información sobre los servidores web, consulte el [Capítulo 9, “Configuración del servidor web”](#).
- 2** Cree una WebApp con el Asistente para aplicaciones web. Si desea obtener más información sobre las clases, consulte el [Capítulo 3, “Las WebApps y los archivos WAR”](#).
- 3** Cree el archivo JAR de la aplicación o el applet con el Creador de recopilatorios, utilizando las siguientes opciones en los pasos 1 y 2 del asistente. Las opciones en los pasos restantes pueden dejarse con los valores por defecto.

Table 15.2 Opciones del Creador de recopilatorios

Creador de recopilatorios	Opción
Paso 1	Tipo de recopilatorio — Applet o aplicación Web Start
Paso 2	WebApp — La WebApp en la que poner el archivo JAR Nombre — El nombre del nodo del recopilatorio. Archivo — El nombre del archivo JAR a colocar en la estructura de directorios de la WebApp. El archivo por defecto está en el directorio raíz de la WebApp.

Para obtener más información sobre la creación de archivos JAR, consulte “El Creador de recopilatorios” en *Creación de aplicaciones con JBuilder*.

- 4** Genere el proyecto para crear el archivo JAR.

- 5** Utilice el Asistente para el inicio de Web Start (Archivo | Nuevo | Página web) para crear el archivo JNLP y la página principal del applet o la aplicación. Introduzca las siguientes opciones:

Table 15.3 Opciones del asistente para el inicio de Web Start

Inicio de Web Start	Opción
Paso 1 — Introduzca la información solicitada	<p>Nombre — El nombre de la aplicación o applet Web Start.</p> <p>WebApp — El nombre de la WebApp que alberga al archivo JAR</p> <p>Archivo JAR — Nombre y vía de acceso al archivo JAR.</p> <p>Clase principal — Para applets, la clase del mismo. Para aplicaciones, la clase que contiene el método <code>main()</code>.</p> <p>Crear Página principal — Crea una página principal. Déjela activada.</p>
Paso 2 — Introduzca información del applet (sólo se muestra con las applets)	<p>Nombre del applet — Su nombre.</p> <p>Raíz de documentos — Raíz del applet. Para obtener más información, consulte “Directorio raíz en la página 3-6”.</p> <p>Anchura — El ancho del applet (en píxeles).</p> <p>Altura — La altura del applet (en píxeles).</p>
Paso 3 — Introduzca información descriptiva	<p>Título — Nombre de la aplicación.</p> <p>Fabricante — Nombre de la empresa.</p> <p>Descripción — Descripción de la aplicación.</p> <p>Permitir uso fuera de línea — Selecciónela para iniciar desde el escritorio.</p>

- 6** Cree una configuración de ejecución de servidor. Consulte “[Creación de una configuración de ejecución del servidor](#)” en la [página 10-5](#). Para URI de inicio, seleccione el archivo HTML creado por el Inicio de Web Start. Este archivo se encuentra en el directorio raíz de su aplicación web.
- 7** Haga clic con el botón derecho en el archivo HTML de la aplicación (creado por el Asistente para el inicio de Web Start) y seleccionar Ejecutar web “Nombre de la configuración de servidor”. El archivo HTML se encuentra en la carpeta Directorio raíz del nodo WebApp en el panel del proyecto.
- 8** Copie la URL de la aplicación que está en el campo URL, en la parte superior de la ficha Ver web. Esta opción se puede establecer automáticamente en la ficha Web del cuadro de diálogo Opciones del IDE (Herramientas | Opciones del IDE).
- 9** Pegue la URL en el navegador externo.
- 10** Pulse en el enlace a la aplicación en la página web.
- Cada uno de estos pasos se describe con mayor detalle en el [Capítulo 21, “Tutorial: Ejecución de la aplicación de ejemplo CheckBoxControl con Java Web Start”](#).

Nota El uso de Java Web Start con applets resuelve los problemas de incompatibilidad del navegador. Siempre que el navegador tenga el plugin Java Web Start, podrá descargar automáticamente los JDK adecuados para ejecutar el applet. La clase principal de un applet Java Web Start es la clase real del applet a ejecutar. La ficha de información de applet recoge unos cuantos valores necesarios para simular que el applet está ejecutándose en una página web, que es donde un applet espera ejecutarse (cuando de hecho está siendo alojado por el plugin Java Web Start).

El archivo JAR de la aplicación

El Creador de recopilatorios de JBuilder permite seleccionar un tipo de recopilatorio JAR de un applet Web Start o de una aplicación Web Start. El Creador de recopilatorios coloca el archivo JAR resultante en el directorio de aplicaciones web seleccionado (WebApp), de forma que pueda ser distribuido por el servidor web.

El archivo JNLP y la página de inicio de la aplicación

El Asistente para el inicio de Web Start de JBuilder crea el archivo JNLP y la página de inicio HTML de la aplicación. El asistente también permite que se especifique el nombre de la aplicación, el de la empresa que la creó y una descripción. Esta información se presenta cuando Java Web Start inicia su aplicación. Adicionalmente, se puede utilizar el asistente para permitir que la aplicación se inicie fuera de línea, desde un ícono en el escritorio.

Nota El Asistente para el inicio de Web Start supone que ya ha creado y generado el archivo JAR de su aplicación.

Advertencia El asistente para el inicio de Web Start otorga el mismo nombre a los archivos JNLP y HTML. Si el nombre introducido en el campo Nombre en el Paso 1 del asistente coincide con un archivo HTML o JNLP existente en el proyecto, se le pregunta si quiere sobreescribirlo.

El archivo JNLP es un documento XML. Los elementos del archivo describen las características de la aplicación, como el nombre de la aplicación, el fabricante y la página de inicio; así como las características JNLP. Para obtener más información, consulte "JNLP File Syntax" en la *Java Web Start Developer's Guide* en <http://java.sun.com/products/javaWebStart/docs/developersguide.html>.

Nota Antes de distribuir la aplicación de inicio Web, se debe cambiar el atributo codebase en el archivo JNLP. Este atributo se genera automáticamente como localhost:8080. Necesitará cambiarlo para que coincida con su servidor Web. Como parte del kit de desarrollo, ahora Java Web Start 1.2 incluye un servlet que le permite generar automáticamente la base de

código apropiada. Consulte http://java.sun.com/products/javaWeb_Start/1.2/docs/download servletguide.html para obtener más información.

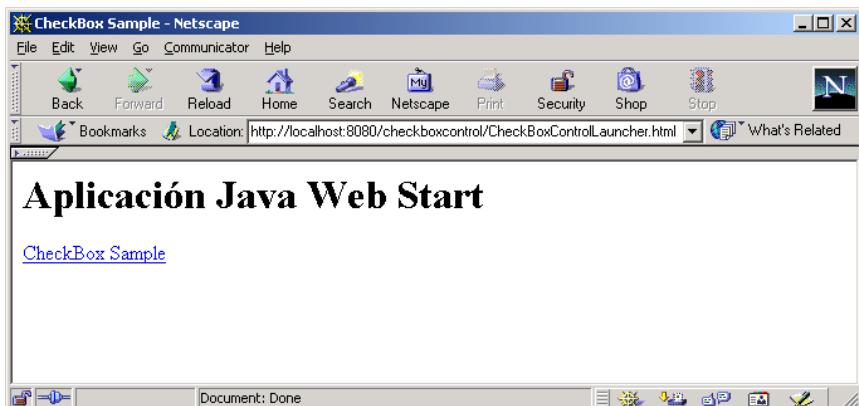
La página de inicio de su aplicación es un archivo HTML que contiene código tanto de JavaScript como de VBScript y etiquetado HTML. El script determina si se está ejecutando el archivo HTML en JBuilder o desde un navegador web externo. Si usted está en JBuilder, verá un mensaje en la vista Web que informa sobre la necesidad de Java Web Start para ejecutar esta aplicación. La vista web tiene este aspecto:

Figura 15.1 Vista web de Java Web Start



Si usted está en un navegador externo, verá un enlace a su aplicación (si ya ha instalado Web Start). Haga clic en el enlace para iniciar Java Web Start y ejecutar su aplicación. El navegador externo tendrá este aspecto.

Figura 15.2 Navegador externo para Java Web Start



Importante Las aplicaciones Java Web Start no pueden iniciarse desde JBuilder. Con el fin de iniciar su aplicación, debe pegar la URL de la aplicación en su navegador externo.

16

Tutorial: Creación de un servlet sencillo

El desarrollo web es una función de JBuilder Enterprise

Este tutorial muestra cómo desarrollar y probar un servlet en JBuilder, mediante el motor servlet Tomcat. Dicho servlet acepta entradas del usuario y cuenta el número de visitantes a una sede web. El servlet, generado con el Asistente para servlets, muestra un formulario para remitir un nombre de usuario. Cuando se envía el formulario, el servlet cambia para mostrar el nombre del usuario y un contador del número de visitantes.

Todos los servlets se generan ampliando una clase `Servlet` básica y definiendo métodos Java para atender las conexiones entrantes. El servlet de este ejemplo amplía la clase `HttpServlet` que interpreta el protocolo HTTP y realiza la mayoría de las operaciones subyacentes que una aplicación web necesita.

Si desea obtener más información acerca de los servlets, consulte los siguientes capítulos:

- [Capítulo 4, “Los servlets”](#)
- [Capítulo 5, “Los servlets en JBuilder”](#)

Para obtener más información sobre las aplicaciones web, consulte el [Capítulo 3, “Las WebApps y los archivos WAR”](#).

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

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

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte “[Convenciones de la documentación](#)” en la página [1-4](#).

Paso 1: Creación de un proyecto

Para desarrollar en JBuilder el servlet “Hola a Todos” de nuestro ejemplo, primero debe crear un proyecto. Para ello:

- 1 Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
- 2 Escriba SimpleServlet en el campo Nombre.
- 3 Asegúrese de que la opción Generar archivo de notas del proyecto se encuentra seleccionada.
- 4 Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.

El archivo de proyecto SimpleServlet.jpx y el archivo HTML del proyecto aparecen en el panel del proyecto.

Paso 2: Selección de un servidor

En este paso, se seleccionará el servidor Tomcat 4.0 como servidor de este proyecto.

- 1 Seleccione Proyecto | Propiedades de proyecto. Aparece el cuadro de diálogo Propiedades de proyecto.
- 2 Pulse la pestaña Servidor.
- 3 Asegúrese de que el botón de radio Servidor único para todos los servicios del proyecto está seleccionado.
- 4 Asegúrese de que Tomcat 4.0 está seleccionado en la lista desplegable de servidores.
- 5 Pulse Aceptar para cerrar el cuadro de diálogo.

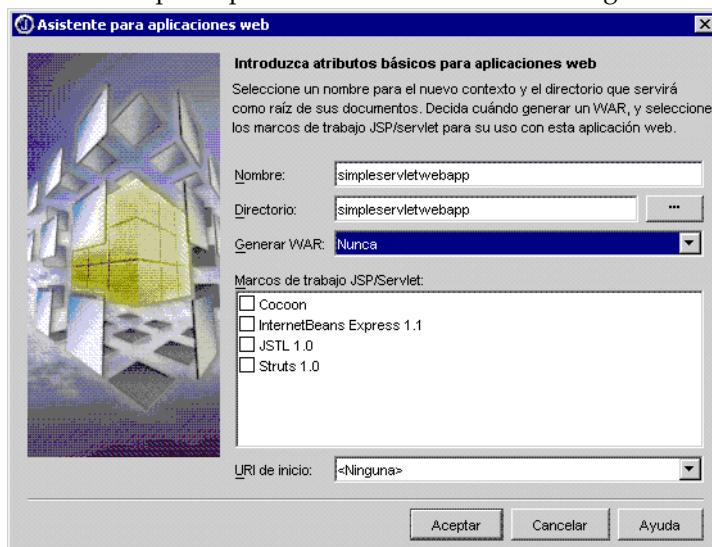
En el siguiente paso, creará una WebApp para su servlet. Aunque no se va a distribuir este proyecto, hay que tener en cuenta que, en una situación real, lo más probable es que se desee crear una WebApp.

Paso 3: Creación de una WebApp

Cuando se desarrollan aplicaciones Web, uno de los primeros pasos es crear una WebApp, la colección de los archivos de contenido de la Web de la aplicación Web. Para crear una WebApp:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo. Haga clic en la pestaña Web y seleccione Aplicaciones web. Pulse Aceptar.
- Se abre el Asistente para aplicaciones web.
- 2 Escriba simpleservletwebapp en el campo Nombre. El campo Directorio se rellena mientras escribe.
- 3 Cambie el valor de la opción Generar WAR a Nunca.
- 4 No seleccione ningún marco de trabajo JSP/Servlet.

El Asistente para aplicaciones Web debe tener el siguiente aspecto.



- 5 Pulse Aceptar para cerrar el asistente y crear la WebApp.

La simpleservletwebapp se muestra en el panel del proyecto como un nodo. Despliegue el nodo para ver el Descriptor de distribución y los nodos del Directorio raíz.



Si desea obtener más información sobre las clases, consulte el [Capítulo 3, "Las WebApps y los archivos WAR"](#).

Paso 4: Creación del servlet con el Asistente para servlets

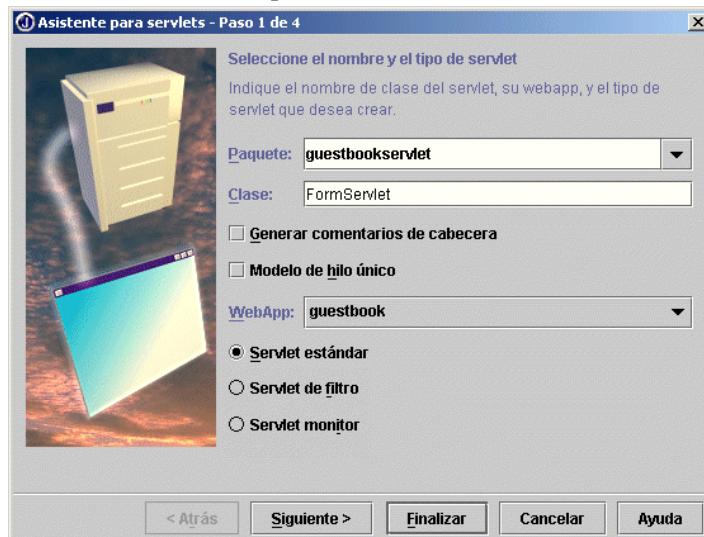
En este paso, se creará el servlet utilizando el Asistente para servlets. El asistente se utilizará para:

- Introducir el nombre de la clase del servlet.
- Escoger el tipo de servlet y su tipo de contenido.
- Escoger los métodos HTTP que hay que implementar.
- Crear un archivo SHTML para ejecutar el servlet.
- Crear parámetros para el servlet.

Si desea obtener más información sobre las opciones, pulse el botón Ayuda del asistente.

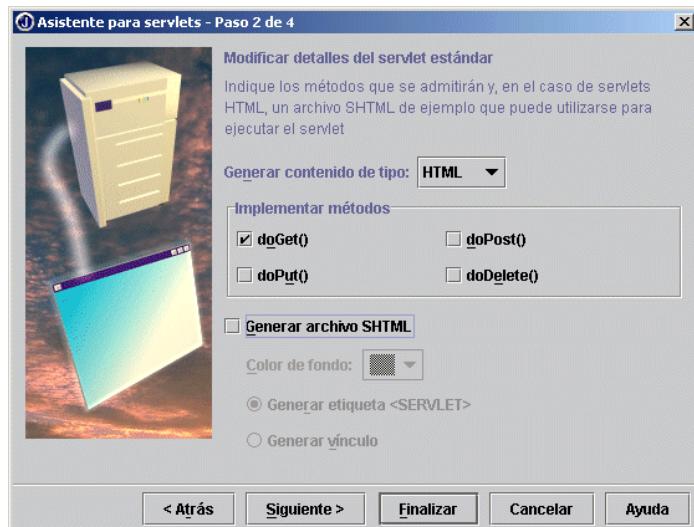
Para crear esta aplicación:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web y seleccione Servlet. Pulse Aceptar. Aparece la ficha Seleccione el nombre y el tipo de servlet del Asistente para servlets.
- 3 Acepte los valores por defecto. La ficha Seleccione el nombre y el tipo de servlet tiene este aspecto:

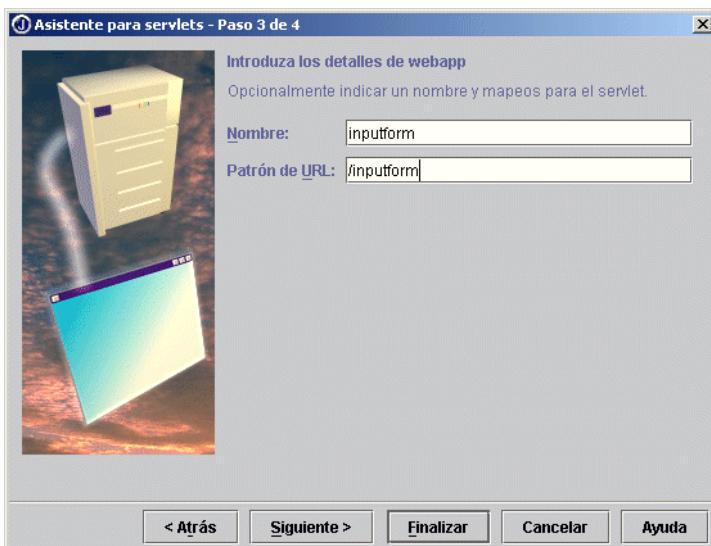


- 4 Haga clic en Siguiente para ir a la ficha Modificar detalles del servlet estándar.

- 5 Una vez en dicha ficha, asegúrese de que en la lista desplegable Generar contenido de tipo está seleccionado HTML. En la sección Implementar métodos, seleccione el método `doPost()`. Asegúrese de que está seleccionado el método `doGet()`. Seleccione la opción Generar archivo SHTML y la opción Generar etiqueta `<SERVLET>`. La ficha Modificar detalles del servlet estándar debería tener este aspecto:



- 6 Pulse Siguiente para pasar a la ficha Introduzca los detalles de webapp.
- 7 Acepte el nombre y el patrón de URL por defecto. La ficha Introduzca los detalles de webapp tiene este aspecto:



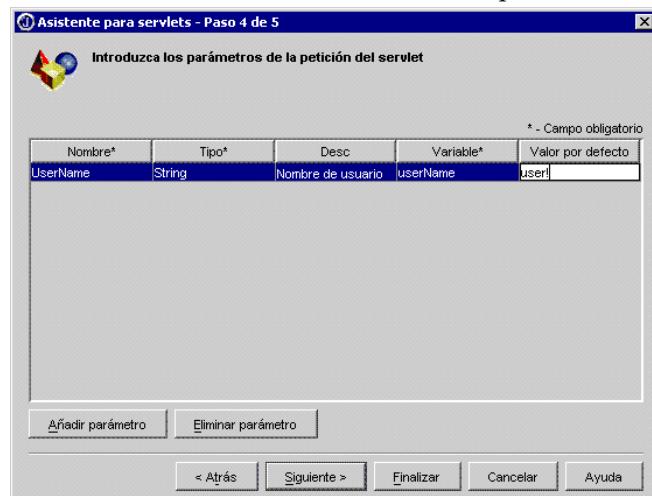
- 8 Haga clic en Siguiente para ir a la ficha Introduzca los parámetros de la petición del servlet.
- 9 Pulse el botón Añadir parámetro con el fin de crear un parámetro de servlet. Este parámetro contiene el nombre introducido en el campo de entrada de texto del servlet.

La siguiente tabla describe los campos y valores a utilizar por este tutorial. Introduzca los valores que están en la columna Valor de la tabla siguiente.

Table 16.1 Opciones de parámetros del Asistente para servlets

Nombre del parámetro	Valor	Descripción
Nombre*	UserName	El parámetro utilizado en la etiqueta <form> del archivo SHTML. Contiene la Cadena que introduce el usuario en el campo de entrada de texto del formulario.
Tipo*	String	El tipo de la variable en lenguaje Java. (Es la opción por defecto y ya está seleccionada.)
Desc	Nombre de usuario	El comentario que se añade al código fuente del servlet.
Variable*	userName	El nombre de la variable usada en <code>Servlet1.java</code> que contiene el nombre del usuario pasado a la misma por el archivo SHTML.
Valor por defecto	User!	El valor por defecto de la variable <code>userName</code> .

Cuando haya acabado, la ficha Introduzca los parámetros de la petición del servlet del asistente debe tener este aspecto:



- 10 Pulse Finalizar para crear el servlet.

Los archivos `Servlet1.java` y `Servlet1.shtml` se añaden al proyecto. Observe que se añade `Servlet1.shtml` al nodo del directorio raíz de la WebApp `simpleservletwebapp`. La biblioteca `Servlet` se añade a la lista de Bibliotecas necesarias en la ficha Vías de acceso del cuadro de diálogo Propiedades de proyecto (Proyecto | Propiedades de Proyecto). El Asistente para servlets también crea una configuración de ejecución de modo que el servlet pueda ejecutarse en el IDE. Para obtener más información, consulte “[Creación de una configuración de ejecución](#)” en la página 10-2.

- 11** Seleccione Archivo | Guardar todo y se guardará el trabajo.

Paso 5: Adición de código al servlet

En este paso, se añadirá código al `Servlet1.java`. Este código crea un contador para el número de veces que ha sido visitada la página y muestra el total.

- 1** Abra `Servlet1.java` en el editor. Busque el comentario:

```
//Inicializar variables globales
```

al principio del archivo. Inmediatamente antes de esa línea, introduzca la siguiente línea de código:

```
int connections = 0;
```

Esta línea de código crea la variable `connections` y la inicializa a cero.

- 2** Busque la línea de código que contiene la cadena:

```
The servlet has received a POST. This is the reply.
```

Inmediatamente después de esa línea de código añada el código siguiente:

```
out.println("<p>Thanks for visiting, ");
out.println(userName);
out.println("<p>");
out.println("Hello World - my first Java servlet program!");
out.println("<p>You are visitor number ");
out.println(Integer.toString(++connections));
```

Esas líneas de código capturan el parámetro `UserName` y lo muestran en una sentencia `out.println`. A continuación, el código incrementa el número de visitantes y lo muestra.

- 3** Seleccione Archivo | Guardar todo y se guardará el trabajo.

Paso 6: Compilación y ejecución del servlet

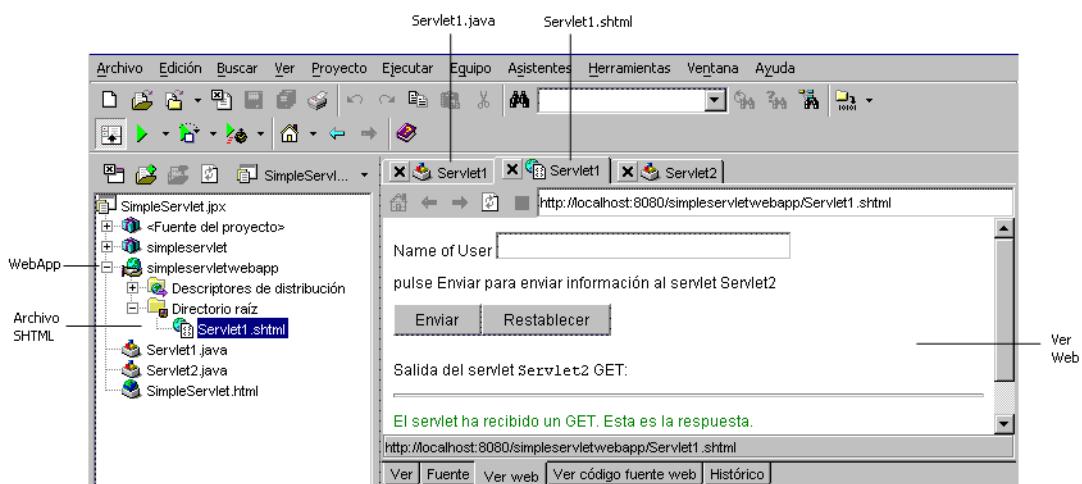
Para compilar y ejecutar el servlet:

- 1 Seleccione Proyecto | Ejecutar Make del proyecto "SimpleServlet.jpx".
- 2 Haga clic con el botón derecho en `Servlet1.shtml` en el panel del proyecto. (Está en el nodo Directorio raíz del nodo `simpleservletwebapp`).
- 3 Seleccione el comando Ejecutar web.

Nota También es posible ejecutar los servlets directamente haciendo clic en el archivo `java` con el botón derecho del ratón, en el panel de proyecto, para pulsar a continuación Ejecutar en web. En este ejemplo se ejecuta el servlet desde el archivo SHTML porque es aquí donde se encuentra el código correspondiente a los campos de introducción de parámetros y el botón de envío, según lo establecido en el Asistente para servlets.

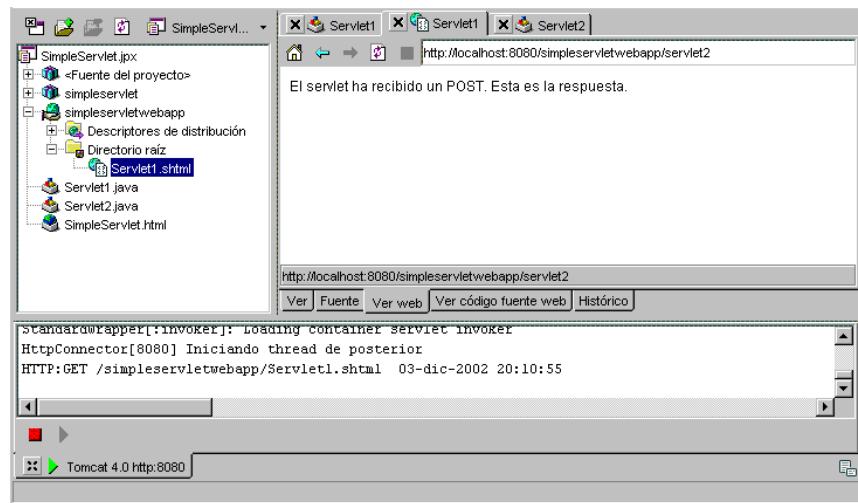
La ejecución del archivo SHTML inicia Tomcat, el servidor web por defecto de JBuilder. La salida de Tomcat se presenta en el panel de mensajes. Los comandos HTTP y los valores de parámetros también se reflejan en el panel de salida. Aparecen dos nuevas pestañas en el panel de contenido del servlet: Vista web y Ver código fuente web. El servlet en ejecución aparece en la Vista web. Tiene el siguiente aspecto:

Figura 16.1 Servlet que se ejecuta en la vista Web



Para ejecutar el servlet, escriba un nombre, como `User` en el campo de texto y pulse Enviar. Se llama al método `doPost()` y la respuesta se muestra en el visualizador web, como se ve a continuación.

Figura 16.2 Servlet en ejecución después del envío del nombre



Para ejecutar de nuevo el servlet y ver el incremento del número de conexiones, pulse flecha atrás en la parte superior de la vista web. Introduzca otro nombre de usuario y pulse el botón Enviar. Cuando aparezca la respuesta del método `doPost()`, verá que se ha incrementado el número de conexiones.

- Para detener el servidor web, pulse el botón Terminar el programa directamente encima de la pestaña del servidor web. Si se efectúan cambios en el código fuente, se debe detener el servidor web antes de su recompilación y nueva ejecución.

Ya ha completado su primer programa servlet. Si desea más información acerca de un servlet más avanzado que se conecte a una base de datos, consulte el [Capítulo 17, "Tutorial: Creación de un servlet que actualiza un libro de visitas"](#).

Tutorial: Creación de un servlet que actualiza un libro de visitas

El desarrollo web es una función de JBuilder Enterprise

Este tutorial muestra cómo crear un servlet que acepta una entrada de usuario y guarda los datos en una base de datos JDataStore.

Cuando acabe este tutorial, su proyecto contendrá las siguientes clases:

- `FormServlet.java`—Esta es la clase ejecutable del programa. Su método `doGet()` muestra un formulario a llenar que utiliza una etiqueta `<form>` de HTML. El servlet envía los valores introducidos por el usuario (mediante parámetros) al `DBServlet`.
- `DBServlet.java`—Este servlet pasa los valores de los parámetros (en su método `doPost()`) a `DataModule1`. El código en el método `doGet()` convierte el JDataStore del libro de visitas en una tabla HTML.
- `DataModule1.java`—El módulo de datos que contiene la lógica empresarial del programa. El código del módulo de datos se conecta con el JDataStore del libro de visitas, lo actualiza y guarda allí los cambios.

Este tutorial presupone que ya está familiarizado con los servlets, con las tecnologías el JDataStore y DataExpress de JBuilder. Si desea obtener más información sobre la depuración, consulte:

- [Capítulo 4, “Los servlets”](#)
- [Capítulo 5, “Los servlets en JBuilder”](#)

Para empezar, también puede trabajar con un servlet menos complejo, “Hola a todos” — consulte el [Capítulo 16, “Tutorial: Creación de un servlet sencillo”](#). Si desea obtener más información acerca de la funcionalidad de la base de datos de JBuilder, consulte la *Guía del desarrollador de aplicaciones de bases de datos*. Consulte la *Guía del desarrollador de JDataStore*, para obtener más información acerca de JDataStore.

Las clases necesarias para este tutorial se encuentran en el directorio samples/WebApps/GuestbookServlet de la instalación de JBuilder.

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

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte [“Convenciones de la documentación” en la página 1-4](#).

Paso 1: Creación de un proyecto

Para desarrollar el servlet de ejemplo Libro de visitas en JBuilder, primero se necesita crear un proyecto. Para ello:

- 1 Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
- 2 Escriba GuestbookServlet en el campo Nombre.
- 3 Asegúrese de que la opción Generar archivo de notas del proyecto se encuentra seleccionada.
- 4 Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.

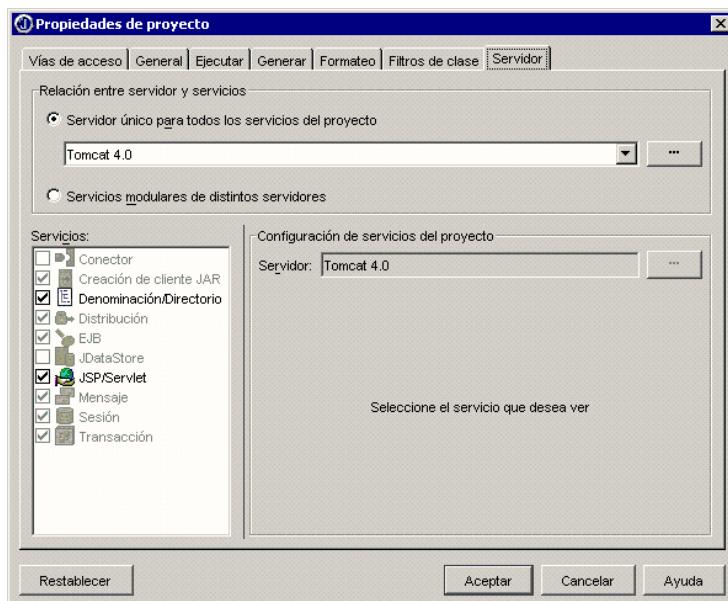
El archivo de proyecto GuestbookServlet.jpx y el archivo HTML del proyecto se presentan en el panel del proyecto.

Paso 2: Selección de un servidor

En este paso, debe comprobar que el servidor Tomcat 4.0 está seleccionado como servidor de este proyecto.

- 1 Seleccione Proyecto | Propiedades de proyecto. Se abre el cuadro de diálogo de Propiedades de proyecto.
- 2 Pulse la pestaña Servidor.

- 3 Asegúrese de que el botón de radio Servidor único para todos los servicios del proyecto está seleccionado.
- 4 Asegúrese de que Tomcat 4.0 está seleccionado en la lista desplegable de servidores. La pestaña Servidor presenta un aspecto similar al siguiente:



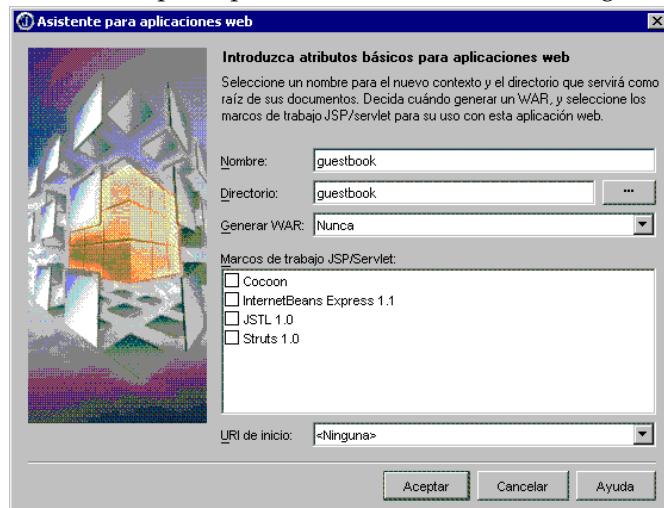
- 5 Pulse Aceptar para cerrar el cuadro de diálogo.

Paso 3: Creación de una WebApp

Cuando se desarrollan aplicaciones Web, uno de los primeros pasos es crear una WebApp, la colección de los archivos de contenido de la Web de la aplicación web. Para crear una WebApp:

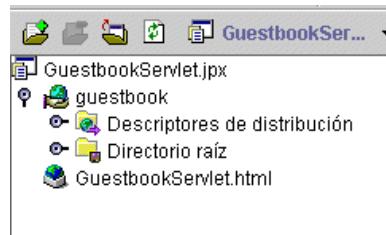
- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo. Haga clic en la pestaña Web y seleccione Aplicaciones web. Pulse Aceptar.
Se abre el Asistente para aplicaciones web.
- 2 Escriba `guestbook` en el campo Nombre. El campo Directorio se rellena automáticamente mientras escribe.
- 3 Cambie el valor de la opción Generar WAR a Nunca. No seleccione ningún marco de trabajo JSP/Servlet.

El Asistente para aplicaciones web debe tener el siguiente aspecto.



- 4 Pulse Aceptar para cerrar el asistente.

La WebApp guestbook se presenta en el panel del proyecto como un nodo. Despliegue el nodo para ver el Descriptor de distribución y los nodos del directorio raíz.



Si desea obtener más información sobre las clases, consulte el [Capítulo 3, "Las WebApps y los archivos WAR"](#).

Paso 4: Creación de los servlets

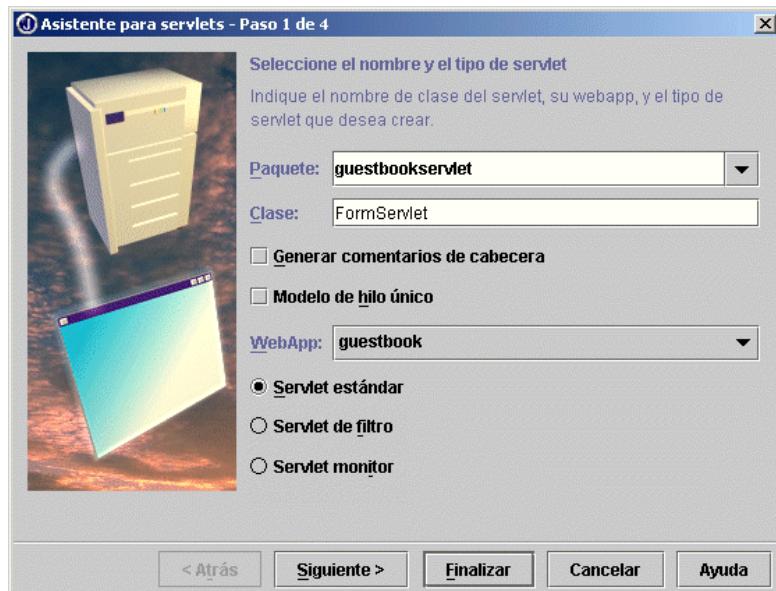
En este paso, se crearán las dos servlets del proyecto:

- `FormServlet.java`—Esta es la clase ejecutable del programa. Su método `doGet()` muestra un formulario a llenar que utiliza una etiqueta `<form>` de HTML. El servlet envía los valores introducidos por el usuario (mediante parámetros) al `DBServlet`.
- `DBServlet.java`—Este servlet pasa los valores de los parámetros (en su método `doPost()`) a `DataModule1`. El código en el método `doGet()` convierte el `JDataStore` del libro de visitas en una tabla HTML.

Para crear FormServlet.java:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo. Haga clic en la pestaña Web y seleccione Servlet. Pulse Aceptar.
Aparece la ficha Seleccione el nombre y el tipo de servlet del Asistente para servlets.
- 2 Asigne el nombre guestbookservlet al Paquete. Cambie el campo Clase a FormServlet.
- 3 Asegúrese de que no están seleccionadas las opciones Generar comentarios de cabecera y Modelo de hilo único. Compruebe que guestbook está seleccionado en la lista desplegable de la WebApp. (Acaba de crear esta WebApp en el paso previo). Deje seleccionada la opción Servlet estándar.

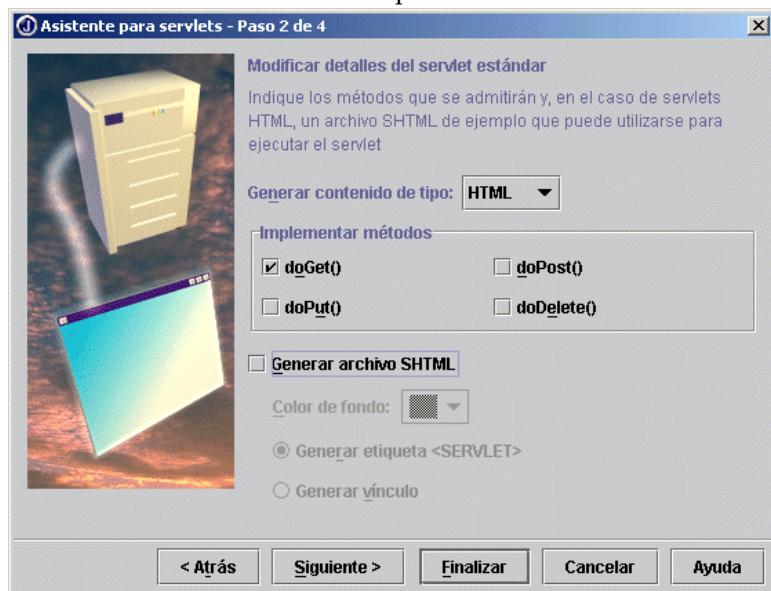
La ficha Seleccione el nombre y el tipo de servlet del asistente debe tener el siguiente aspecto:



- 4 Haga clic en Siguiente para ir a la ficha Modificar detalles del servlet estándar del asistente.

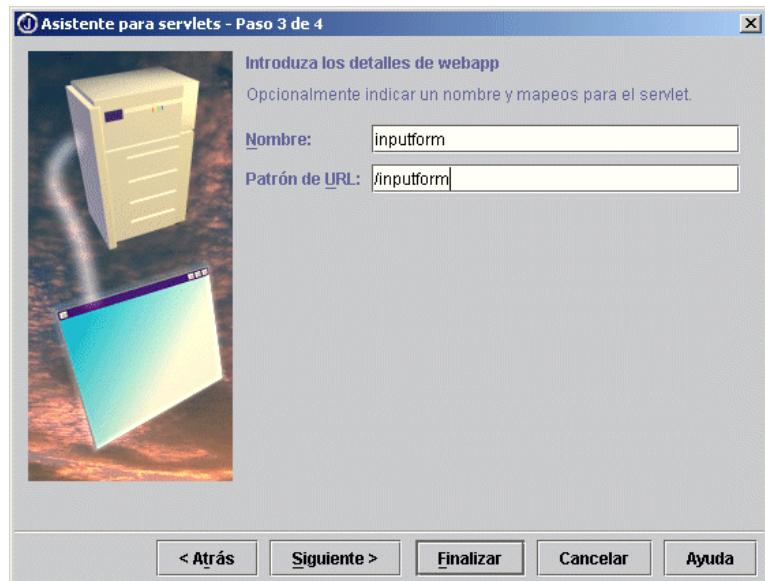
- 5 Asegúrese de que la opción Generar contenido de tipo tiene el valor HTML y que está seleccionado el método `doGet()`. Desactive la opción Generar archivo SHTML.

Cuando haya finalizado, la ficha Modificar detalles del servlet estándar del asistente debería tener este aspecto:



- 6 Haga clic en Siguiente para ir a la ficha Introduzca los detalles de webapp del asistente.
- 7 Cambie el valor por defecto del campo Nombre por `inputform`. Cambie el valor del campo Patrón de URL por `/inputform`. Asegúrese de que todas las entradas están en minúsculas. La entrada del Patrón de URL se utiliza para ejecutar el servlet en lugar del nombre de clase.

La ficha Introduzca los detalles de webapp debería tener este aspecto:



- 8** Pulse Finalizar para crear el servlet. No es necesario que configure otras opciones.

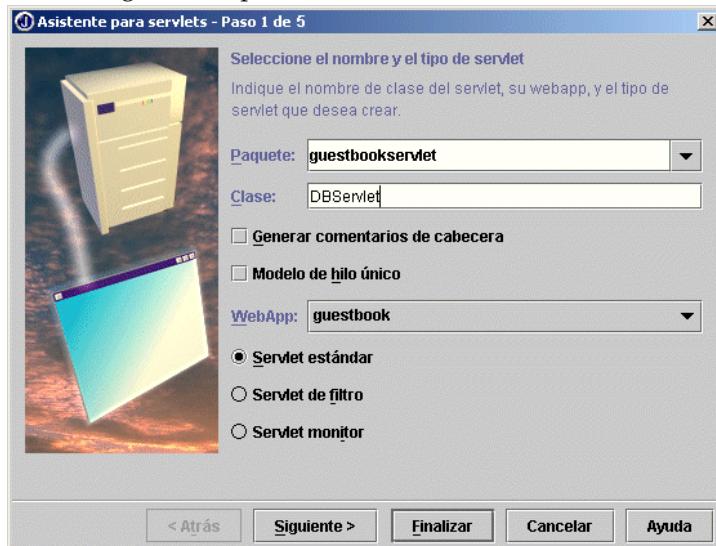


- 9** Seleccione Archivo | Guardar todo o pulse en la barra de herramientas para guardar su trabajo.

Para crear DBServlet.java:

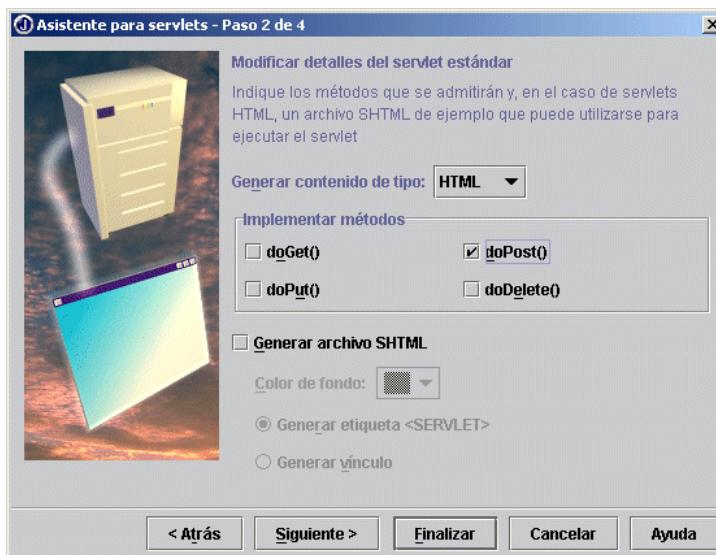
- 1** Para mostrar la galería de objetos, seleccione Archivo | Nuevo. Haga clic en la pestaña Web y seleccione Servlet. Pulse Aceptar.
Aparece la ficha Seleccione el nombre y el tipo de servlet del Asistente para servlets.
- 2** Asigne el nombre guestbookservlet al Paquete. Cambie el campo Clase a DBServlet.
- 3** Asegúrese de que no están seleccionadas las opciones Generar comentarios de cabecera y Modelo de hilo único. Debe seleccionar la guestbook en la lista desplegable de la WebApp. Deje seleccionada la opción Servlet estándar.

La ficha Seleccione el nombre y el tipo de servlet del asistente debe tener el siguiente aspecto:



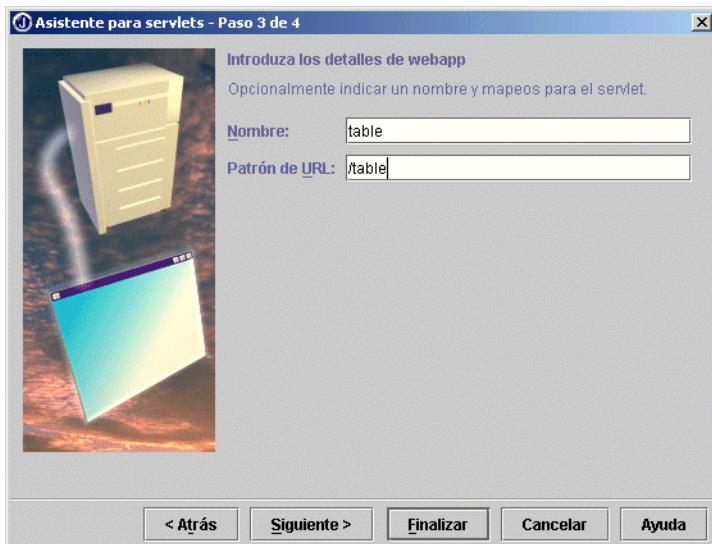
- 4 Haga clic en Siguiente para ir a la ficha Modificar detalles del servlet estándar del asistente.
- 5 Asegúrese de que en Generar contenido de tipo figura HTML. Elimine la selección del método `doGet()`. En su lugar, seleccione el método `doPost()`. No seleccione la opción Generar archivo SHTML.

Cuando termine, éste es el aspecto que debe tener la ficha Modificar detalles del servlet estándar:



- 6 Haga clic en Siguiente para ir a la ficha Introduzca los detalles de webapp del asistente.
- 7 Cambie el valor por defecto del campo Nombre por `table`. Cambie el valor del campo Patrón de URL por `/table`. Asegúrese de que todas las entradas están en minúsculas. Se utilizará el nombre `table` en vez del nombre de clase `DBServlet` en la etiqueta `<form>` de HTML de FormServlet.

La ficha Introduzca los detalles de webapp debería tener este aspecto:



- 8 Pulse Finalizar para crear el servlet. No es necesario que configure otras opciones.
- 9 Seleccione Archivo | Guardar todo o pulse en la barra de herramientas para guardar su trabajo.

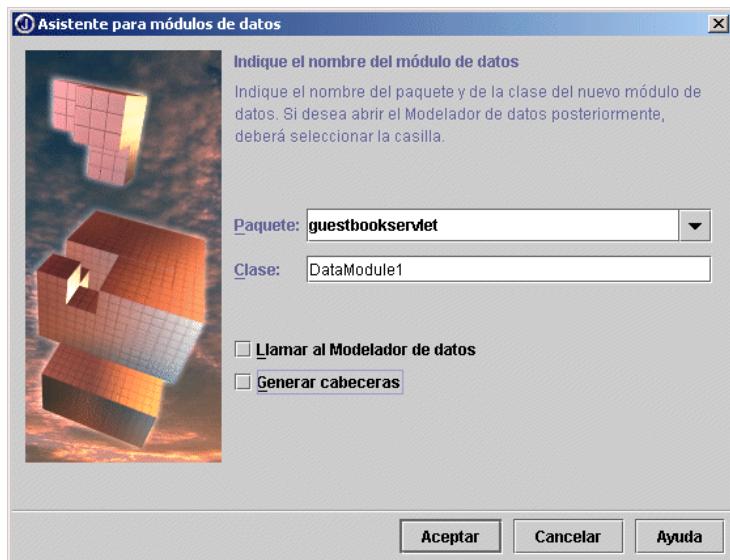


Paso 5: Creación del módulo de datos

Con el fin de crear el módulo de datos que conecte con el JDataStore del libro de visitas y realice las tareas de actualización:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo. En la ficha General, seleccione Módulo de datos y pulse Aceptar.
- 2 Se abre el Asistente para módulos de datos.
- 3 Asigne el nombre `guestbookservlet` al Paquete. Mantenga el nombre de Clase por defecto de `DataModule1`.
- 4 Desactive las opciones Llamar al Modelador de datos y Generar cabeceras.

Cuando acabe, el Asistente para módulos de datos tendrá este aspecto.



- 4 Pulse Aceptar para crear el módulo de datos.
- 5 Seleccione Archivo | Guardar todo o pulse en la barra de herramientas para guardar su trabajo.
- 6 Seleccione Proyecto | Ejecutar Make del proyecto "GuestbookServlet.jpx" con el fin de compilarlo y crear los archivos de clase.

Paso 6: Cómo añadir componentes de bases de datos al módulo de datos

En este paso, se utilizará el Diseñador de interfaces de usuario de JBuilder para añadir componentes de base de datos al módulo de datos. Si desea información sobre la forma de utilizar el diseñador de interfaces de usuario, consulte "Introducción al diseñador", en *Diseño de aplicaciones con JBuilder*. Si desea obtener más información acerca de los componentes de base de datos, consulte "Conexión con bases de datos" en la *Guía del desarrollador de aplicaciones de bases de datos*.

Para abrir el diseñador, haga doble clic en `DataModule1.java` en el panel del proyecto. A continuación, haga clic en la pestaña Diseño en la parte inferior del panel de contenido.

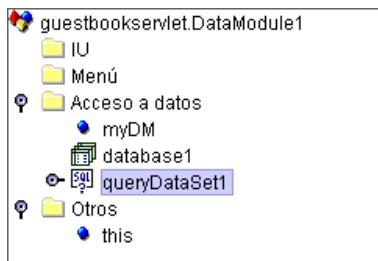
Para añadir componentes de acceso a datos al módulo de datos:

- 1 Seleccione la pestaña DataExpress de la paleta de componentes.



- 2** Pulse el ícono Base de datos. Después pulse en el árbol de componentes del panel de estructura, con el fin de añadir el componente de base de datos al módulo de datos.
- 3** Pulse en el ícono QueryDataSet. Haga clic en el árbol de componentes.

Cuando haya acabado, el árbol de componentes debe tener este aspecto.

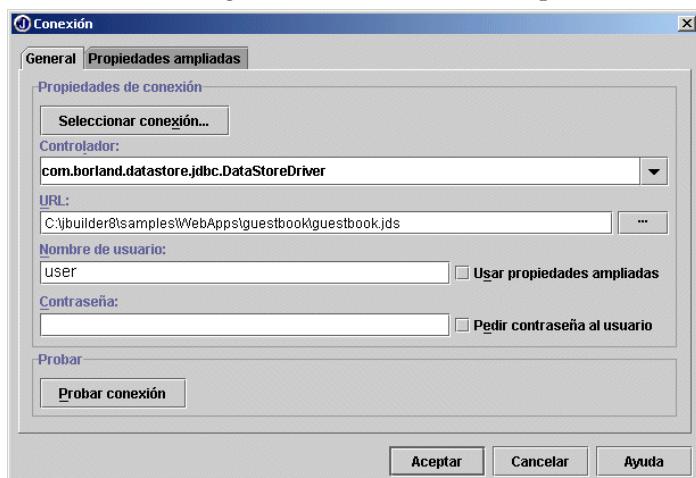


Para conectar con el JDataStore del libro de visitas:

- 1** En el árbol de componentes, seleccione `database1`.
- 2** Pulse en la propiedad `connection` en el Inspector, luego pulse el botón puntos suspensivos a la derecha del nombre de la propiedad.
Esto abre el cuadro de diálogo Conexión.
- 3** En la ficha General del cuadro de diálogo Conexión, verifique que en el Controlador se lee `com.borland.datastore.jdbc.DataStoreDriver`.
- 4** Pulse el botón puntos suspensivos a la derecha del campo URL para mostrar el cuadro de diálogo Crear URL para el DataStore. Este cuadro de diálogo se utiliza para elegir el JDataStore al que hay que conectarse.
- 5** Pulse la opción Base de datos DataStore local.
- 6** Seleccione el botón puntos suspensivos para poder buscar el JDataStore del libro de visitas (`guestbook.jds`) en el directorio `samples/WebApps/guestbook` de su instalación de JBuilder. Pulse Abrir para seleccionar el JDataStore.
- 7** Pulse Aceptar para cerrar el cuadro de diálogo Crear URL para el DataStore.

- 8 En la ficha General del cuadro de diálogo Conexión, escriba user en el campo Nombre de usuario.

El cuadro de diálogo Conexión tendrá este aspecto.



- 9 Pulse el botón Probar conexión con el fin de comprobar la conexión con el JDataStore del libro de visitas. Si se realiza la conexión, aparecerá la palabra Correcto a la derecha del botón. Si la conexión no se realiza, un mensaje de error intentará explicar por qué ha fallado.

- 10 Pulse Aceptar para cerrar el cuadro de diálogo Conexión.

JBuilder añade el método `databasel.setConnection()` al método `jbInit()` del módulo de datos.

Para acceder a los datos de Guestbook, debe definir una consulta. Para obtener más información acerca de las consultas, consulte “Consultas en bases de datos” en la *Guía del desarrollador de aplicaciones de bases de datos*. Para crear la consulta en JBuilder:

- 1 Seleccione `queryDataSet1` en el árbol de componentes.
- 2 Haga clic en la propiedad `query` en el Inspector y, a continuación, pulse el botón de puntos suspensivos.

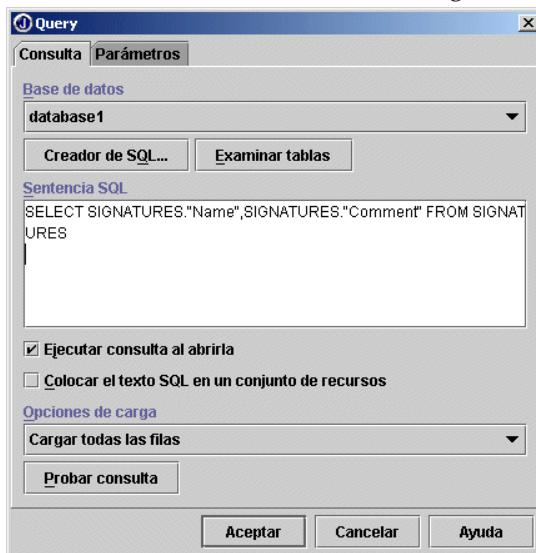
Se abre el cuadro de diálogo Query.
- 3 En la ficha Consulta, seleccione `databasel` de la lista desplegable Base de datos.
- 4 En el cuadro de texto Sentencia SQL, escriba:

```
SELECT SIGNATURES."Name",SIGNATURES."Comment" FROM SIGNATURES
```

Esta consulta carga todos los valores en los campos Nombre y Comentario de la tabla SIGNATURES del JDataStore del libro de visitas. Cuando introduzca la consulta, utilice las mayúsculas como se ha mostrado anteriormente.

- 5 Verifique que está seleccionado Ejecutar consulta al abrirla.
- 6 En la lista desplegable de Opciones de carga, compruebe que está seleccionada la opción Cargar todas las filas.

La ficha Consulta del cuadro de diálogo Query debe tener este aspecto.



- 7 Pulse el botón Probar consulta con el fin de comprobarla. Si se realiza la conexión, aparecerá la palabra **Correcto** a la derecha del botón. Si la consulta no puede ejecutarse, un mensaje de error intentará explicar por qué falló.

- 8 Pulse Aceptar para cerrar el cuadro de diálogo Query.

JBuilder añade el método `queryDataSet1.setQuery()` al método `jbInit()`.

Nota Puede ver el mensaje siguiente escrito en la pestaña Diseñador del panel de mensajes:

```
No se creó la instancia dinámica de la variable 'myDM'  
guestbookservlet.DataModule1
```

Por el momento, puede pasar por alto este mensaje. Ya lo arreglará en un paso posterior. Haga clic en la X de la pestaña Diseñador, de la parte inferior del Visualizador de aplicaciones, para cerrarla.



- 9 Pulse el icono Guardar todo de la barra de herramientas para guardar su trabajo.

Paso 7: Creación de la conexión de datos con el DBServlet

En este paso, se creará una conexión de datos con DBServlet. La conexión permite que el servlet pase datos al módulo de datos.

- 1** Haga doble clic en `DBServlet.java` en el panel del proyecto con el fin de abrirla en el editor.
- 2** Seleccione Proyecto | Ejecutar Make del proyecto “GuestbookServlet.jpx” para compilarlo.
- 3** Seleccione Asistentes | Usar módulo de datos con el fin de que se abra el Asistente para usar módulos de datos.
- 4** Mantenga el nombre por defecto para la clase DataModule. Cambie el nombre del campo por `dm`. Deje la declaración de instancias en `Share (static) instance of dataModule`.
- 5** Pulse Aceptar para cerrar el asistente. El asistente añade la siguiente línea de código a la declaración de clase `DBServlet`:

```
DataManager dm;
```

También añade el siguiente constructor:

```
public DBServlet() {
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
```

A continuación del constructor, se añade el siguiente código:

```
private void jbInit() throws Exception {
    dm = guestbookservlet.DataManager1.getDataModule();
}
```

- 
- 6** Pulse el ícono Guardar todo de la barra de herramientas para guardar su trabajo.

Ahora que el servlet está conectado al módulo de datos, es necesario que ambos hagan algo. A partir de este punto en nuestro tutorial, usted mismo introducirá el código directamente en el editor. El primer paso será crear un formulario de entrada en `FormServlet`.

Paso 8: Creación de un formulario de entrada en el FormServlet

En este paso, añadirá código al método `doGet()` de `FormServlet`. Este código crea un formulario mediante la etiqueta `<form>` de HTML. El formulario leerá dos valores, `UserName` y `UserComment`, introducidos por el usuario. Estos datos serán enviados a `DBServlet`, el servlet que se comunica con el módulo de datos.

Una etiqueta `<form>` es una etiqueta HTML estándar que crea un formulario de entrada para recoger los datos y mostrar la información a un usuario. La etiqueta contiene los atributos `action` y `method`. Estos atributos le dicen al servlet qué hacer cuando se pulsa el botón Enviar del formulario. En nuestro tutorial, el atributo `action` llama a `DBServlet`. El atributo `method` envía los parámetros `UserName` y `UserComment` a `DBServlet`.

Para añadir código a `FormServlet`:

- 1** Haga doble clic en `FormServlet` en el panel del proyecto con el fin de que se abra en el editor. (Puede estar ya abierta).
- 2** Busque el método `doGet()` en la parte superior del archivo.

Sugerencia

Puede buscarlo situando el cursor en el panel de estructura y escribiendo `doGet`.

- 3** Borre todas las líneas que empiecen por `out.println`.
- 4** Añada las siguientes líneas de código al método `doGet()`:

```
PrintWriter out = response.getWriter();

out.println("<html><body>");
out.println("<h1>Sign the guestbook</h1>");
out.println("<strong>Enter your name and comment in the input fields
below.</strong>");
out.println("<br><br>");
out.println("<form action=table method=POST>");
out.println("Name<br>");
out.println("<input type=text name=UserName value=\"\""
size=20 maxlength=150>");
out.println("<br><br>");
out.println("Comment<br>");
out.println("<input type=text name=UserComment value=\"\""
size=50 maxlength=150>");
out.println("<br><br><br><br>");
out.print("<input type=submit value=Submit>");
out.println("</form>");
out.println("</html></body>");
```

Sugerencia

Es posible copiar y pegar directamente el código en el editor, o copiarlo del ejemplo que se encuentra en el directorio `samples/WebApps/GuestbookServlet` de la instalación de JBuilder.



- 5** Pulse el icono Guardar todo de la barra de herramientas para guardar su trabajo.

Paso 9: Adición de código al método DBServlet doPost()

En el siguiente paso, añadirá el código que realiza las siguientes funciones dentro del método `doPost()` de DBServlet:

- Lee en los parámetros `UserName` y `UserComment` de FormServlet.
- Llama al método `DataModule` que actualiza el JDataStore del libro de visitas, pasando los valores de los parámetros `UserName` y `UserComment`.
- Llama al método del módulo de datos y guarda los cambios en el JDataStore.

Para ello:

1 Haga doble clic en `DBServlet` en el panel del proyecto con el fin de abrirla en el editor. (Puede estar ya abierta).

2 Busque el método `doPost()`.

3 Elimine la siguiente línea de código del método `doPost()`:

```
out.println("<p>The servlet has received a POST. This is the reply.</p>");
```

4 Inserte las siguientes líneas de código, manteniendo el cursor en el lugar del que acaba de eliminar código:

```
String userName = request.getParameter("UserName");
String userComment = request.getParameter("UserComment");
dm.insertNewRow(userName, userComment);
dm.saveNewRow();
doGet(request, response);
```

Sugerencia

Es posible copiar y pegar directamente el código en el editor, o copiarlo del ejemplo que se encuentra en el directorio `samples/WebApps/GuestbookServlet` de la instalación de JBuilder.

Las dos primeras líneas de código reciben los valores de los parámetros `UserName` y `UserComment` que se pasan desde FormServlet. Las siguientes líneas llaman a dos métodos del módulo de datos:

- `insertNewRow()`—inserta los nuevos valores de Nombre y Comentario en la última fila de la tabla.
- `saveNewRow()`—guarda los cambios en el JDataStore del libro de visitas.

La última línea llama al método `doGet()` del servlet que convierte la tabla Guestbook (libro de visitas) en HTML.

Nota

Estos métodos no han sido aún añadidos al módulo de datos, por lo que verá errores en la carpeta Errores del panel de estructura. Por el momento, puede hacer caso omiso de ellos.



5 Pulse el icono Guardar todo de la barra de herramientas para guardar su trabajo.

Paso 10: Adición de código para convertir la tabla SIGNATURES de Guestbook

En este paso, añadirá un método `doGet()` a DBServlet para convertir la tabla SIGNATURES de Guestbook en HTML. Se presentan tanto las filas existentes como la nueva.

- 1 Inserte el siguiente código después del método `doPost()` del DBServlet:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("<h2>" + dm.getQueryDataSet1().getTableName() + "</h2>");
    Column[] columns = dm.getQueryDataSet1().getColumns();
    out.println ("<table border = 1><tr>");
    for (int i=1; i < columns.length; i++) {
        out.print ("<th>" + columns[i].getCaption() + "</th>");
    }
    out.println("</tr>");
    dm.getQueryDataSet1().first();
    while (dm.getQueryDataSet1().inBounds()) {
        out.print("<tr>");
        for (int i = 1; i < columns.length; i++) {
            out.print ("<td>" + dm.getQueryDataSet1().format(i) + "</td>");
        }
        out.println("</tr>");
        dm.getQueryDataSet1().next();
    }
    out.println("</table>");
    out.println("</body>");
    out.println("</html>");
}
```

Sugerencia

Es posible copiar y pegar directamente el código en el editor, o copiarlo del ejemplo que se encuentra en el directorio `samples/WebApps/GuestbookServlet` de la instalación de JBuilder.

- 2 Añada los paquetes siguientes a la lista de sentencias `import` de la parte superior del archivo. Esto le asegura que el servlet se compilará.

```
import com.borland.dx.dataset.*;
import com.borland.dx.sql.dataset.*;
import com.borland.datastore.*;
```

Sugerencia

Puede utilizar MemberInsight (*Ctrl+H*) para que le ayude a completar las sentencias `import`.

Nota

No tenga en cuenta los errores que vea en la carpeta Errores del panel de estructura.



- 3 Pulse el icono Guardar todo de la barra de herramientas para guardar su trabajo.

Qué hace el método doGet()

El método `doGet()` que acaba de añadir convierte la tabla SIGNATURES del Libro de visitas en HTML. Se mueve cíclicamente a través de las filas de la tabla JDataStore y las muestra en el navegador web.

Las siguientes líneas de código contienen la declaración de un método de servlet estándar, configuran el tipo de contenido del servlet y definen el escritor de salida:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
```

La dos líneas siguientes de código configuran la salida como HTML e inician la página HTML.

```
out.println("<html>");
out.println("<body>");
```

La siguiente línea de código imprime el nombre de la tabla JDataStore, SIGNATURES, en la parte superior de la página HTML. El código utiliza el método `queryDataSet1.getTableName()` para obtener el nombre de la tabla.

```
out.println("<h2>" + dm.getQueryDataSet1().getTableName() + "</h2>");
```

La siguiente línea llama al método `queryDataSet1.getColumns()` para recoger el nombre de las columnas, y las devuelve como una matriz.

```
Column[] columns = dm.getQueryDataSet1().getColumns();
```

La línea siguiente crea la tabla con la etiqueta `<table>` y crea la primera fila de la tabla.

```
out.println ("<table border = 1><tr>");
```

Entonces, el código utiliza un bucle `for` para pasar a través de los nombres de las columnas de la matriz de columnas, recupera los títulos de las columnas y muestra cada uno en una fila de la tabla. En este tutorial, se presentan solamente la segunda y la tercera columna del JDataStore. No se presenta la primera columna, el número interno de fila.

```
for (int i = 1; i < columns.length; i++) {
    out.print ("<th>" + columns[i].getCaption() + "</th>");
```

La línea que sigue al bloque `for` cierra la fila de la tabla.

```
out.println("</tr>");
```

La línea siguiente sitúa el cursor en la primera fila del JDataStore.

```
dm.getQueryDataSet1().first();
```

El bucle `while` pasa a través del `JDataStore` y presenta los datos de la segunda y de la tercera columna de la tabla. La primera vez que pasa, presenta los datos de la primera fila. A continuación, el método `next()` sitúa el cursor en la siguiente fila del `JDataStore`. El bucle `while` continúa mostrando los datos mientras que el cursor esté dentro de los límites, o lo que es lo mismo, mientras el método `inBounds()` informe que la navegación está entre el primer y el último registro visible para el cursor. Cuando esta condición no se cumple, se cierra la tabla y la página HTML.

```
while (dm.getQueryDataSet1().inBounds()) {
    out.print("<tr>");
    for (int i = 1; i < columns.length; i++) {
        out.print ("<td>" + dm.getQueryDataSet1().format(i) + "</td>");
    }
    out.println("</tr>");
    dm.getQueryDataSet1().next();
}
out.println("</table>");
out.println("</body>");
out.println("</html>");
```

Paso 11: Cómo añadir la lógica empresarial al módulo de datos

El proceso está casi terminado. En este momento, todavía no hay código para escribir los datos recién añadidos al Libro de visitas `JDataStore` y guardarlos. Este código se añadirá al `Module1`. Este código abrirá el conjunto de datos, insertará la nueva fila (mediante las cadenas `userName` y `userComment` que se pasan desde `DBServlet`) y guardará la nueva fila en el `JDataStore`.

Siga estos pasos y añada lógica empresarial al módulo de datos:

- 1** Haga doble clic en `Module1.java` en el panel del proyecto con el fin de abrirlo en el editor. (Puede que ya esté abierto en el Diseñador de interfaces de usuario; si es así, pulse la pestaña Fuente de la parte inferior del panel de contenido.)
- 2** Busque el método `jbInit()` mediante el comando Buscar | Buscar. Añada el código siguiente antes de la llave de cierre del método:

```
queryDataSet1.open();
```

Este código abre el conjunto de datos. Debe estar abierto para poder insertar o guardar datos. En el módulo de datos, el conjunto de datos se abre justo después del código que conecta con la base de datos y configura la consulta.

- 3** Añada el código para el método que inserta una fila nueva. Con el fin de añadir una fila, se necesita crear un objeto `DataRow`, luego hay que pasar los datos de los parámetros `userName` y `userComment` al `DataRow`. Este método se añadirá después del método `jbInit()`. Simplemente mueva el cursor una

Línea hacia abajo, más allá de la llave de cierre del método y pulse Enter unas cuantas veces. Añada el siguiente método:

```
public void insertNewRow(String userName, String userComment) {  
    try {  
        try {  
            DataRow dataRow1 = new DataRow(queryDataSet1, new String[]  
                new String[] { "Name", "Comment" });  
            dataRow1.setString("Name", userName);  
            dataRow1.setString("Comment", userComment);  
            queryDataSet1.addRow(dataRow1);  
        }  
        catch (DataSetException ex) {  
            ex.printStackTrace();  
        }  
    }  
}
```

La primera línea del método crea un objeto `DataRow` que contiene los nuevos valores de `Name` y `Comment`. La segunda y la tercera fila pasan los valores de los parámetros `userName` y `userComment` a los campos `Nombre` y `Comentario`. La última fila añade el objeto `DataRow` al conjunto de datos.

- 4 Añada el siguiente método para guardar la nueva fila en el conjunto de datos tras el método `insertNewRow()`:

```
public void saveNewRow() {  
    try {  
        database1.saveChanges(queryDataSet1);  
    }  
    catch (DataSetException ex) {  
        ex.printStackTrace();  
    }  
}
```



- 5 Pulse el ícono Guardar todo de la barra de herramientas para guardar su trabajo.

Ahora ya se ha añadido todo el código al programa. En el paso siguiente, se compilará y se ejecutará.

Paso 12: Compilación y ejecución del proyecto

Antes de compilar y ejecutar, necesita comprobar las dependencias de la WebApp y modificar la configuración de ejecución.

Para comprobar las dependencias de la WebApp:

- 1 Haga clic con el botón derecho sobre la WebApp `guestbook` en el panel del proyecto y seleccione Propiedades.
- 2 Elija la ficha Dependencias en el cuadro de diálogo Propiedades.

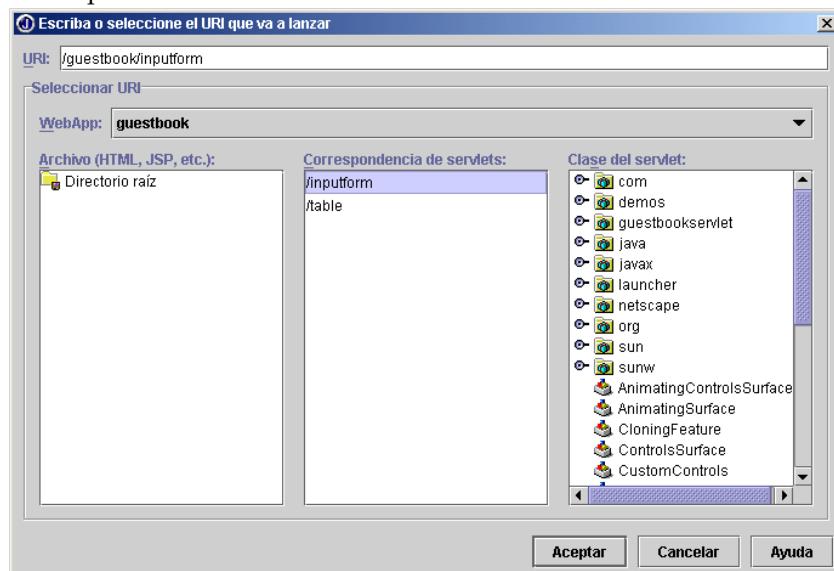
Asegúrese de que los parámetros de las bibliotecas de servidor Data Express y JDataStore Server, en el lado derecho de la ficha, están configurados como Incluir todo. Si están configurados con cualquier otro valor, seleccione la opción Incluir siempre todas las clases y recursos, en la parte inferior de la ficha, para cada biblioteca.

- 3 Pulse Aceptar para cerrar el cuadro de diálogo.

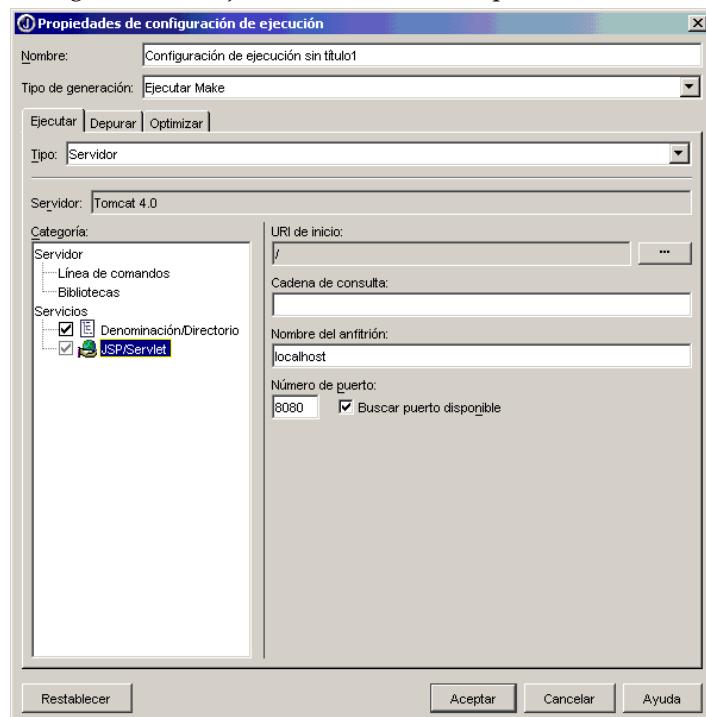
Para modificar la configuración de ejecución:

- 1 Seleccione Ejecutar | Configuraciones con el fin de mostrar la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto. Elija la configuración FormServlet y haga clic en Modificar con el fin de que aparezca el cuadro de diálogo Propiedades de configuración de ejecución.
- 2 Elija el servicio JSP/Servlet en el árbol de la izquierda de la ficha.
- 3 Pulse sobre el botón con puntos suspensivos a la derecha del campo URI de inicio con el fin de mostrar el cuadro de diálogo Escriba o seleccione el URI que se va a lanzar, donde escogerá el nombre del servlet a ejecutar.
- 4 Seleccione /inputform en el árbol de directorio Correspondencia de servlets, en el centro del cuadro de diálogo. El campo URI en la parte superior del cuadro de diálogo ahora contiene: /guestbook/inputform. Este es el nombre de la WebApp que creó en el Asistente para aplicaciones web, seguido por el nombre del servlet.

El cuadro de diálogo Escriba o seleccione el URI que va a lanzar tendrá este aspecto:



- 5 Pulse Aceptar para cerrar el cuadro de diálogo Escriba o seleccione el URI que va a lanzar. El cuadro de diálogo Propiedades de configuración de ejecución tendrá este aspecto:



- 6 Pulse Aceptar dos veces más con el fin de cerrar los cuadros de diálogo Propiedades de configuración de ejecución y Propiedades de proyecto.



- 7 Pulse el icono Guardar todo de la barra de herramientas para guardar su trabajo.

Para compilar y ejecutar el proyecto:

- 1 Seleccione Proyecto | Ejecutar Make del proyecto "GuestbookServlet.jpx."
- 2 Seleccione Ejecutar | Ejecutar proyecto.

El servidor web Tomcat se presenta en el panel de mensajes.

- 3** El formulario de entrada de FormServlet se presenta en la vista web. El URI es /guestbook/inputform/ y coincide con el seleccionado en el cuadro de diálogo URI de inicio.

Firmar el guestbook

Escriba su nombre y comentario en los campos siguientes.

Nombre:

Comentario:

http://localhost:8080/guestbook/inputform

Fuente Vista web Ver código fuente web Diseño Bean Doc Histórico

- 4** Introduzca MyName en el campo Name y MyComment en el campo Comment.
5 Pulse el botón Submit (Enviar).

La tabla SIGNATURES del Libro de visitas se convierte en HTML. MyName y MyComment aparecen en la última fila de la tabla. Observe que la URI ha cambiado a <http://localhost:8080/guestbook/table>, indicando que el programa está ejecutando DBServlet.

SIGNATURES

Name	Comment
Aries	I am
MyName	MyComment

http://localhost:8080/guestbook/table

Fuente Ver web Ver código fuente web Diseño Bean Doc Histórico

Si desea obtener más información acerca de las URL, URI y servlets, consulte “[Cómo las URL ejecutan los servlets](#)” en la página 10-11.

-  **6** Puede pulsar la flecha de retroceso a la izquierda del campo Ubicación del URL con el objeto de volver al formulario de entrada e introducir otro nombre y otro comentario.
 -  **7** Si desea detener el servidor web, pulse el botón Terminar el programa directamente encima de la pestaña del servidor web. Se debe detener el servidor web antes de compilar y ejecutar el servlet de nuevo, después de efectuar cambios.
- Nota** Puede abrir el JDataStore del Guestbook en el Explorador de JDataStore (Herramientas | Explorador de JDataStore) para verificar que los nuevos datos se han guardado en la tabla.

Ha finalizado este tutorial. Ahora ya sabe cómo crear un formulario de entrada HTML para utilizar un servlet, pasar un parámetro de un servlet a otro, conectar un servlet con un módulo de datos, pasar parámetros desde un servlet a un módulo de datos y utilizar un módulo de datos para actualizar un JDataStore.

18

Tutorial: Creación de una JSP mediante el Asistente para JSP

El desarrollo web es una función de JBuilder Enterprise

Este tutorial le acompaña a través del desarrollo de una Página JavaServer (JSP) utilizando el Asistente para JSP de JBuilder. Esta JSP toma un texto como entrada, lo muestra como salida cuando se pulsa Enviar y utiliza un JavaBean para contar el número de visitas de la página.

El Asistente para JSP constituye un buen punto de partida para la creación de páginas JSP. No genera aplicaciones completas, sino que se encarga de todos los detalles repetitivos necesarios para que la aplicación empiece a funcionar. Para abrir este asistente, seleccione Nuevo del menú Archivo, pulse la pestaña Web y seleccione entonces JavaServer Page. Para obtener una completa información sobre las opciones del Asistente para JSP, consulte el tema Asistente para JSP en la ayuda en línea.

Para realizar pruebas en este tutorial, utilizará Tomcat. Este tutorial utiliza Tomcat porque viene incluido con JBuilder y no requiere una configuración adicional. Tomcat es la implementación de referencia de las especificaciones Java Servlet y Páginas JavaServer. Esta implementación se puede utilizar en Apache Web Server y en otros servidores web y herramientas de desarrollo. Si desea más información sobre Tomcat, consulte <http://jakarta.apache.org>.

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

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

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte “[Convenciones de la documentación](#)” en la página 1-4.

Paso 1: Creación de proyectos

- 1 Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
- 2 En el campo Nombre, introduzca un nombre de proyecto, como jsptutorial
- 3 Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.

Se crea un proyecto.

Paso 2: Selección de un servidor

En este paso, se seleccionará el servidor Tomcat 4.0 como servidor de este proyecto.

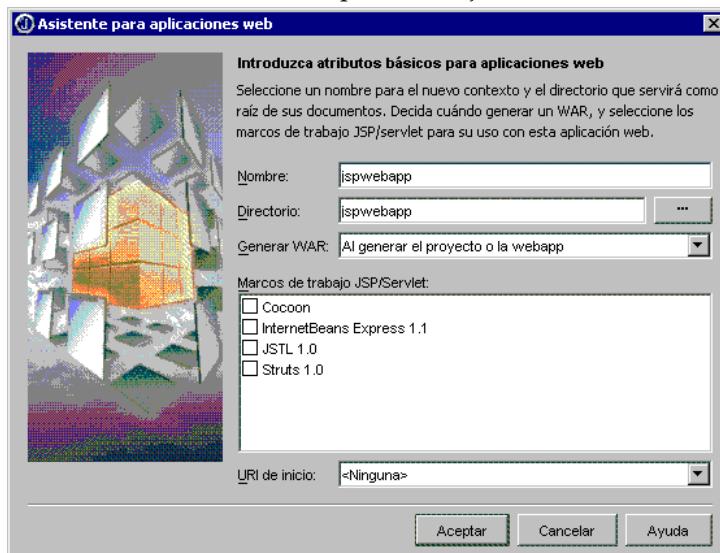
- 1 Seleccione Proyecto | Propiedades de proyecto. Aparece el cuadro de diálogo Propiedades de proyecto.
- 2 Pulse la pestaña Servidor.
- 3 Asegúrese de que el botón de radio Servidor único para todos los servicios del proyecto está seleccionado.
- 4 Asegúrese de que Tomcat 4.0 está seleccionado en la lista desplegable de servidores.
- 5 Pulse Aceptar.

Paso 3: Creación de una WebApp

Este paso es optativo pero aconsejable. Puede utilizar la WebApp por defecto, pero suele ser menos confuso crear una WebApp con un nombre personalizado. Si desea obtener más información acerca de las WebApps y los archivos WAR, consulte el [Capítulo 3, “Las WebApps y los archivos WAR”](#).

- 1 Seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web. Seleccione Aplicación web.
- 3 Pulse Aceptar. Aparece el Asistente para aplicaciones web.
- 4 Escriba un nombre para la WebApp, como `jspwebapp`. El campo Directorio se rellena automáticamente con el mismo nombre.
- 5 Deje la opción por defecto para Generar WAR, si bien no va a necesitar un archivo WAR ya que, probablemente, no deseará distribuir la aplicación de este tutorial.
- 6 No seleccione ningún marco de trabajo JSP/Servlet ni especifique una URI de inicio.

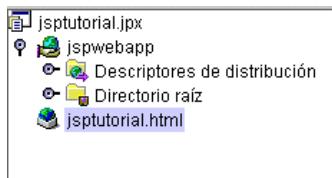
El asistente debe tener un aspecto semejante a éste:



- 7 Pulse Aceptar para cerrar el asistente.

En el panel del proyecto, se presenta un nodo de WebApp, `jspwebapp`. Expanda el nodo para ver el directorio raíz y los nodos de los descriptores de distribución.

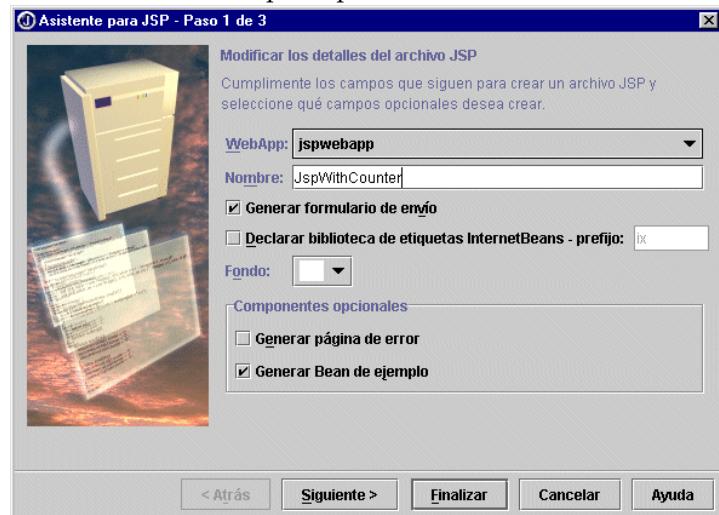
Figura 18.1 Nodo WebApp en el panel de proyecto



Paso 4: Creación de la JSP

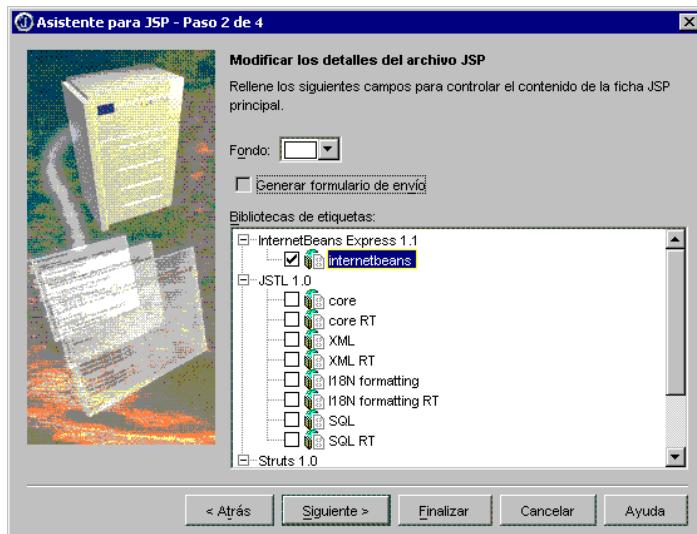
En este paso creará el esqueleto de una JSP con el Asistente para JSP.

- 1 Seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web. Seleccione Página JavaServer.
- 3 Pulse Aceptar. Se abre el Asistente para JSP.
- 4 En el campo Nombre, escriba `JSPWithCounter`. Este es el nombre de la JSP.
- 5 Active Generar Bean de ejemplo y desactive Generar página de error. El asistente tendrá un aspecto parecido a:



- 6 Pulse Siguiente.

- 7 Marque Generar formulario de envío. El asistente tendrá un aspecto parecido a:



- 8 Pulse Finalizar para aceptar todos los valores por defecto de las restantes fichas del asistente.

Se añade un archivo `JSPWithCounter.jsp` al directorio raíz de su WebApp. Expanda el nodo del Directorio raíz en el panel del proyecto con el fin de verlo. También se ha añadido a su proyecto un bean de ejemplo `JSPWithCounterBean.java`. Este bean es llamado por la JSP. El Asistente para JSP también crea automáticamente una configuración de ejecución de manera que la JSP pueda ejecutarse en el IDE. Para obtener más información sobre las configuraciones de ejecución, consulte “Definición de las configuraciones de ejecución” en *Creación de aplicaciones con JBuilder*. Si desea más información sobre la creación de una configuración de ejecución con un asistente, consulte “[Creación de una configuración de ejecución con los asistentes](#)” en la página 10-2.

Paso 5: Adición de funciones al JavaBean

En este momento se han creado una página JSP y un JavaBean que aquella puede utilizar.

El siguiente paso de este tutorial consiste en crear un método para contar el número de accesos a una página web.

- 1 Haga doble clic en `JspWithCountingBeanBean.java` en el panel del proyecto.
- 2 Modifique el código fuente como sigue, introduciendo los fragmentos en **negrita** en el código existente:

Paso 6: Modificación del código de la JSP

```
package jsptutorial;

public class JSPWithCounterBean {
    /**inicializar la variable aquí*/
    private int myCount=0;
    private String sample = "Start value";
    //Acceder a la propiedad sample
    public String getName() {
        return sample;
    }
    //Acceder a la propiedad sample
    public void setSample(String newValue) {
        if (newValue!=null) {
            sample = newValue;
        }
    }
    /**Nuevo método para el recuento de veces*/
    public int count() {
        return ++myCount;
    }
}
```

- 3 Seleccione Archivo | Guardar todo y se guardará el trabajo.

Paso 6: Modificación del código de la JSP

En este paso, modificará el código de la JSP con el fin de contar el número de visitas.

- 1 Haga doble clic en `JSPWithCounter.jsp` en el panel del proyecto con el fin de abrirlo en el editor. Recuerde que está en el nodo del Directorio raíz de la WebApp.
- 2 Modifique el archivo generado como sigue, añadiendo el código en **negrita**. Puede utilizar CodeInsight y resaltar el código fuente JSP para orientarse.

```
<html>
<head>
<title>
JspWithCounter
</title>
</head>

<jsp:useBean id="jSPWithCounterBeanId" scope="session"
    class="jsptutorial.JSPWithCounterBean" />
<jsp:setProperty name="jSPWithCounterBeanId" property="*" />
<body>
<h1>
JBuilder Generated JSP
</h1>
<form method="post">
<br>Enter new value: <input name="sample"><br>
```

```

<br><br>
<input type="submit" name="Submit" value="Submit">
<input type="reset" value="Reset">
<br>
Value of Bean property is: <jsp:getProperty name="JSPWithCounterBeanId"
    property="sample" />
<p>Esta página ha sido visitada: <%= jSPWithCounterBeanId.count() %> veces.</p>
</form>
</body>
</html>

```

3 Seleccione Archivo | Guardar todo y se guardará el trabajo.

La línea de código que acaba de añadir utiliza una etiqueta expression de JSP para llamar al método `count()` de la clase `JSPWithCounterBean` e inserta el valor devuelto en el HTML generado. Para obtener más información sobre las JSP, consulte “[Etiquetas JSP](#)” en la página 6-3.

Fíjese en las etiquetas `<jsp:useBean>`, `<jsp:setProperty>` y `<jsp:getProperty>` en el código anterior. Estas fueron añadidas por el Asistente para JSP. La etiqueta `useBean` crea una instancia de la clase `JSPWithCounterBean`. Si la instancia ya existe, se recupera. Si no existe, se crea. Las etiquetas `setProperty` y `getProperty` permiten manipular propiedades del bean.

El resto del código que generó el Asistente para JSP es HTML estándar.

Paso 7: Ejecución de la página JSP

Con el fin de poder ejecutar la JSP, las propiedades de ejecución del proyecto deben configurarse correctamente en Proyecto | Propiedades de proyecto | Ejecutar. En este tutorial, las propiedades de ejecución fueron ya fijadas por el Asistente para JSP, de modo que pueda seguir adelante y ejecutar la JSP.

- 1 Haga clic con el botón derecho en `JSPWithCounter.jsp` en el Directorio raíz del nodo WebApp. Elija Ejecutar web mediante “JSPWithCounter” en el menú.

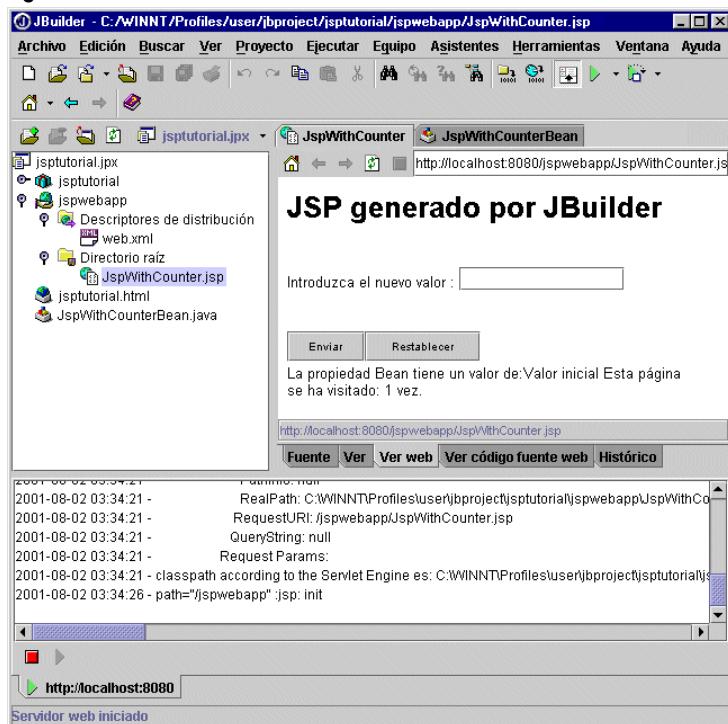
Nota

El comando Ejecutar web incluye el nombre de una configuración de ejecución, en este caso, “JSPWithCounter”. Esta configuración de ejecución fue creada automáticamente por el Asistente para JSP. Por defecto, el asistente asigna a la configuración el mismo nombre que la JSP. Para obtener más información sobre las configuraciones de ejecución, consulte “Definición de las configuraciones de ejecución” en *Creación de aplicaciones con JBuilder*.

El proyecto se compila y se ejecuta. Los errores de compilación se muestran en el panel de mensajes. Si hay errores, consulte “[Depuración web del servlet o de la JSP](#)” en la página 10-19.

Si no hay errores, se inicia el servidor web y en el panel de contenido aparecen dos nuevas pestañas, Ver web y Ver código fuente web. Tomcat, que se instala junto con JBuilder, es un motor de servlet que admite archivos servlet y JSP. La Vista web es un navegador que muestra la salida de la ejecución de la JSP. La pestaña Ver código fuente web muestra el código HTML real que la JSP ha generado dinámicamente. Si se han seguido todos los pasos correctamente, la página JSP en ejecución presentará este aspecto:

Figura 18.2 JSP en Ver Web



La pestaña Ver web del panel de contenido presenta la página JSP. Para la comprobación local, la URL señala a localhost:8080, que es donde se está ejecutando Tomcat. Para probar la JSP:

- 1 Escriba algo de texto en el campo correspondiente.
- 2 Pulse el botón Enviar. El valor introducido se muestra debajo del botón, y el contador de la página aumenta.

Los datos de salida y registro de Tomcat aparecen en una nueva pestaña, en el panel de mensajes. La salida de servlets, beans, comandos HTTP y valores de parámetros se muestra en el panel de mensajes. La configuración de ejecución de una JSP puede modificarse, o se puede crear una en la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto (Proyecto | Propiedades de proyecto). Si desea más información sobre la configuración de las

propiedades de ejecución de la JSP, consulte “[Creación de una configuración de ejecución del servidor](#)” en la página 10-5. JBuilder utiliza el puerto 8080 por defecto. Si este puerto está en uso, JBuilder buscará por defecto un puerto disponible.

Utilización de la pestaña Ver web

La ficha Ver web del panel de contenido muestra el archivo JSP después de que lo procese el motor JSP. En este caso el motor de la JSP es Tomcat. El panel Ver web no se comporta de la misma forma que el panel de visualización. En el panel Ver web puede haber un retraso entre el momento en que se modifica el archivo JSP y cuando el cambio se muestra en él. Si desea ver los cambios más recientes en un archivo JSP, seleccione el botón Actualizar de la barra de herramientas de la vista, al igual que haría en cualquier navegador web.



Si se estuviera depurando la JSP, se podría pulsar F9 para devolver la pantalla a la vista web.

Depuración de las JSP

**La depuración de JSP es
una función de JBuilder
Enterprise**

Las páginas JSP se compilan en servlets. Desde JBuilder se pueden depurar fragmentos de código Java en el archivo JSP original, en lugar de depurarlas en el correspondiente servlet de Java generado. Para obtener más información sobre la depuración de las JSP, consulte “[Depuración web del servlet o de la JSP](#)” en la página 10-19.

Distribución de la JSP

Para la distribución en un servidor web de producción, consulte en su documentación lo relativo a la forma de distribuir archivos JSP en él. Si desea obtener más información acerca de la distribución de las JSP, consulte el [Capítulo 11, “Distribución de aplicaciones web”](#).

Como se indica en la base de datos FAQ de JSP, que se encuentra en <http://java.sun.com/products/jsp/faq.html>, existen numerosas implementaciones de la tecnología JSP para distintos servidores web. Puede encontrar la información más reciente en java.sun.com.

19

Tutorial: Creación de un servlet con InternetBeans Express

El desarrollo web es una función de JBuilder Enterprise

Este tutorial le enseña cómo crear una JSP mediante InternetBeans. Cuando finalice este tutorial, tendrá un servlet que utiliza un `DataManager` para consultar una tabla en un `JDataStore`, muestra los comentarios de un libro de visitas en una `IxTable` y permite que los visitantes introduzcan sus propios comentarios y los vean aparecer en el libro de visitas. Se puede encontrar una versión acabada de la aplicación en `<JBuilder>/samples/WebApps/guestbook`.

Este tutorial presupone que está familiarizado con Java y los servlets de Java, con JBuilder IDE y con `JDataStore`. Para obtener más información sobre las JSP, consulte *Procedimientos iniciales con Java*. Para obtener más información sobre los servlets de Java, consulte el [Capítulo 4, “Los servlets”](#). Para obtener más información sobre el IDE de JBuilder, consulte “El entorno de JBuilder” en *Introducción a JBuilder*. Si desea obtener más información sobre `JDataStore`, consulte la *Guía del desarrollador de JDataStore*.

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

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte “[Convenciones de la documentación](#)” en la página 1-4.

Paso 1: Creación de proyectos

- 1 Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
- 2 En el campo Nombre, introduzca un nombre de proyecto, como guestbooktutorial
- 3 Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.

Se crea un proyecto.

Paso 2: Selección de un servidor

En este paso, se seleccionará el servidor Tomcat 4.0 como servidor de este proyecto.

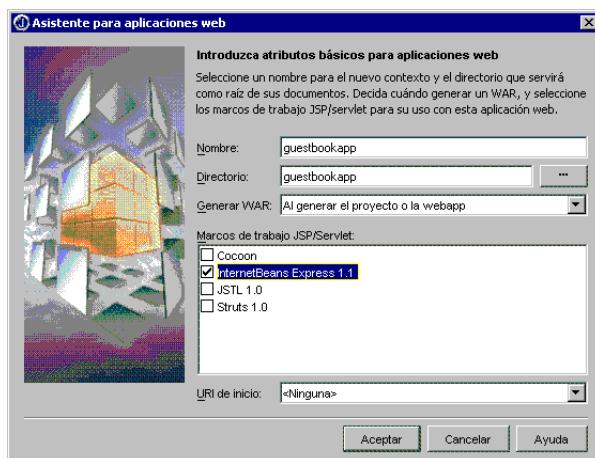
- 1 Seleccione Proyecto | Propiedades de proyecto. Aparece el cuadro de diálogo Propiedades de proyecto.
- 2 Pulse la pestaña Servidor.
- 3 Asegúrese de que el botón de radio Servidor único para todos los servicios del proyecto está seleccionado.
- 4 Asegúrese de que Tomcat 4.0 está seleccionado en la lista desplegable de servidores.
- 5 Pulse Aceptar.

Paso 3: Creación de una WebApp

Este paso es optativo pero aconsejable. Puede utilizar la WebApp por defecto, pero suele ser menos confuso crear una WebApp con un nombre personalizado. Si desea obtener más información acerca de las WebApps y los archivos WAR, consulte el [Capítulo 3, “Las WebApps y los archivos WAR”](#).

- 1 Seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web de la galería de objetos. Seleccione Aplicación web.
- 3 Pulse Aceptar. Aparece el Asistente para aplicaciones web.
- 4 Escriba un nombre para la WebApp, como guestbookapp. El campo Directorio se rellena automáticamente mientras escribe.

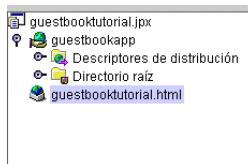
- 5 Deje la opción por defecto para Generar WAR, si bien no va a necesitar un archivo WAR ya que, probablemente, no deseará distribuir la aplicación de este tutorial.
- 6 Elija el marco de trabajo InternetBeans Express 1.1 JSP/Servlet. No especifique una URI de inicio.
- 7 El asistente debe tener un aspecto semejante a éste:



- 8 Pulse Aceptar.

En el panel del proyecto se presenta un nuevo nodo WebApp, guestbookapp. Expanda el nodo para ver el directorio raíz y los nodos de los descriptores de distribución.

Figura 19.1 Nodo WebApp en el panel del proyecto

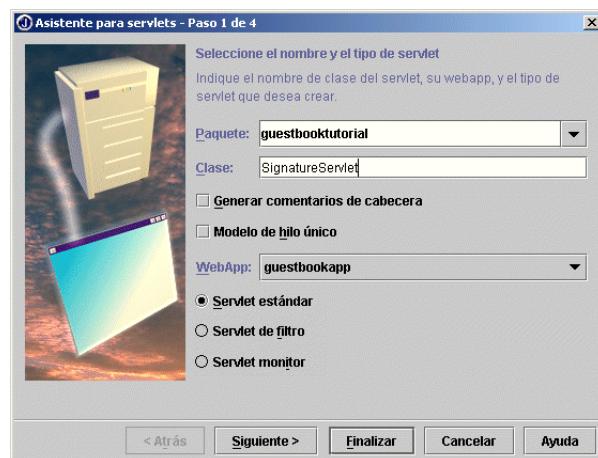


Paso 4: Creación del servlet

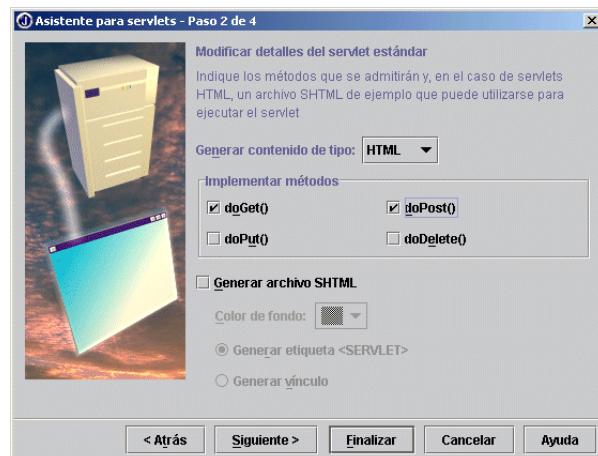
En este paso, se creará el servlet mediante el Asistente para servlets.

- 1 Seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web de la galería de objetos. Seleccione Servlet.
- 3 Pulse Aceptar. Aparece el Asistente para servlets.
- 4 Introduzca el nombre de la clase: SignatureServlet

- 5 Seleccione guestbookapp para la WebApp, si no lo está ya. El asistente debe tener un aspecto semejante a éste:



- 6 Haga clic en Siguiente para avanzar en el asistente.
- 7 Asegúrese de que en la opción Generar contenido de tipo figura HTML.
- 8 Asegúrese de que están seleccionados los métodos doGet() y doPost().
- 9 Asegúrese de que Generar archivo SHTML no está seleccionado. El asistente debe tener un aspecto semejante a éste:



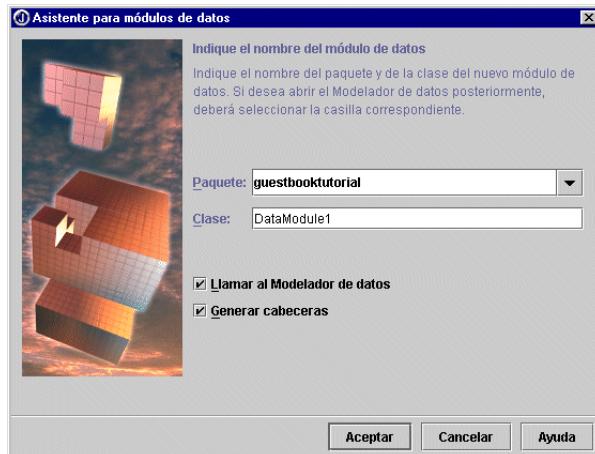
- 10 Pulse el botón Finalizar. Se añade un archivo SignatureServlet.java a su proyecto.
- 11 Pulse el botón Guardar todo de la barra de herramientas para guardar su trabajo.



Paso 5: Creación del módulo de datos

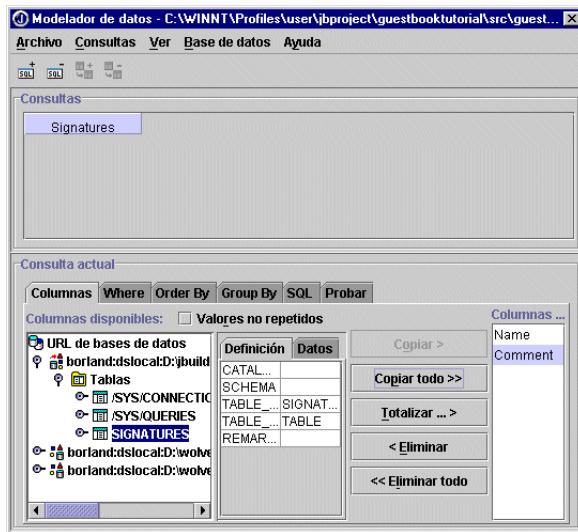
En este paso, utilizará el Asistente para módulo de datos con el fin de crear el que albergará la lógica de conexión de la base de datos.

- 1 Seleccione Archivo | Nuevo.
- 2 Seleccione Módulo de datos en la ficha General de la galería de objetos.
- 3 Pulse Aceptar. Se abre el Asistente para módulos de datos.



- 4 No modifique la configuración por defecto de los campos Paquete y Nombre de clase.
- 5 Asegúrese de que está seleccionado Llamar al Modelador de datos.
- 6 Pulse Aceptar. Se abre el Modelador de datos.
- 7 Vaya al menú Base de datos y seleccione Añadir URL de conexión.
- 8 Seleccione `com.borland.datastore.jdbc.DataStoreDriver` en la lista desplegable de controlador.
- 9 En el campo URL, busque o escriba la vía de acceso al archivo `guestbook.jds`. Se encuentra en la carpeta de `<JBuilder>/samples/WebApps/guestbook`. Pulse Aceptar para cerrar el cuadro de diálogo Crear URL para el DataStore.
- 10 Pulse Aceptar de nuevo. La nueva URL para la base de datos se añade a la lista de URL para bases de datos, en la parte inferior izquierda del Modelador de datos y se selecciona.
- 11 Haga doble clic en la URL y escriba `user` en el cuadro de diálogo de conexión. No es necesaria una contraseña. Pulse Aceptar para cerrar el cuadro de diálogo.

- 12 Abra la lista de tablas pulsando el nodo Tablas, del árbol Columnas disponibles.
- 13 Seleccione la tabla SIGNATURES pulsando sobre ella.
- 14 Pulse el botón Copiar todo. El Modelador de datos presenta un aspecto similar al siguiente:



- 15 Seleccione Guardar en el menú Archivo del Modelador de datos.
- 16 Seleccione Salir en el menú Archivo del Modelador de datos. El archivo DataModule1.java se actualiza con la información de conexión requerida.
- 17 Pulse el botón Guardar todo de la barra de herramientas de JBuilder para guardar su trabajo.



Paso 6: Diseño de la página de plantilla HTML

En este paso, creará una página HTML que puede ser utilizada por InternetBeans como plantilla para la disposición de los datos dinámicos.

- 1 Pulse el botón Añadir Archivos/Paquetes en la barra de herramientas del proyecto.
- 2 Pulse el botón Proyecto en la pestaña Explorador, del cuadro de diálogo Añadir archivos o paquetes al proyecto.
- 3 Seleccione el directorio para la WebApp (guestbookapp)
- 4 Escriba gb1.html en el campo Nombre de archivo.
- 5 Pulse Aceptar.
- 6 Pulse nuevamente Aceptar para crear el archivo.

- 7 En el panel del proyecto, haga doble clic en el archivo para abrirlo. Se abre el archivo vacío HTML.
- 8 Pulse la pestaña Fuente para abrir el código del HTML. En este momento estará vacío.
- 9 Escriba el siguiente código HTML en el archivo nuevo. También se puede copiar y pegar desde este tutorial.

```

<html>
<head>

<title>Guestbook Signatures</title>

</head>

<body>

<h1>Firmar el guestbook</h1>

<table id="guestbooktable" align="CENTER" cellspacing="0"
       border="1" cellpadding="7">
<tr>
<th>Nombre</th><th>Comentario</th>
</tr>
<tr>
<td>Leo</td><td>Todo controlado!</td>
</tr>
</table>

<form method="POST">
<p>Escriba su nombre:</p>

<input type="text" id="Name" name="Nombre" size="50">

<p>Escriba su comentario:</p>

<input type="text" id="Comment" name="Comentario" size="100">
<p>
<input type="submit" name="submit" value="Submit"></p>
</form>

</body>
</html>

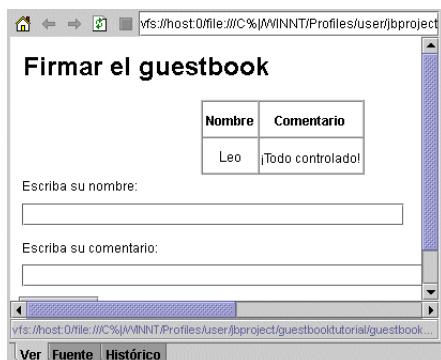
```

Observe que la etiqueta <table> contiene datos de prueba. Estos datos serán sustituidos con los datos dinámicos del JDataStore cuando se ejecute el servlet. Estos datos de prueba suministran una indicación de como se verán los datos dinámicos reales cuando se presenten. Para obtener mas información sobre como InternetBeans Express utiliza las tablas, consulte “[Generación de tablas](#)” en la página 7-6



- 10 Haga clic en el botón Guardar todo en la barra de herramientas.

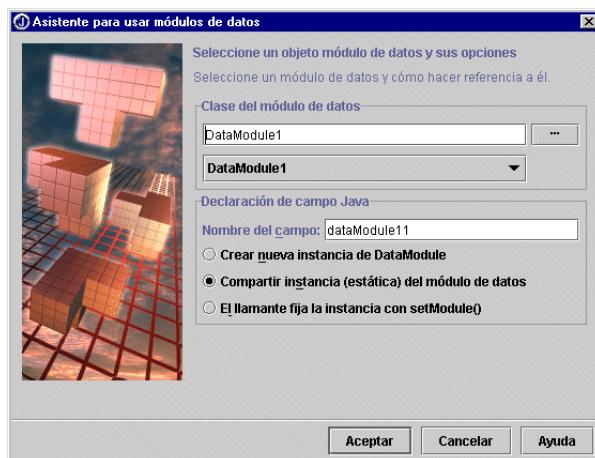
- 11** Seleccione la pestaña Ver. El HTML debe tener este aspecto:



Paso 7: Conexión del servlet al DataModule

En este paso, añadirá una línea de código al servlet que lo habilitará para utilizar el DataModule.

- 1** Seleccione Proyecto | Ejecutar Make “guestbooktutorial.jpx”. Esto genera el proyecto de manera que se crea el archivo DataModule1.class.
- 2** Abra el archivo SignatureServlet.java en el editor.
- 3** Seleccione Asistentes | Usar módulo de datos. Se abre el Asistente para usar módulos de datos. La clase DataModule1 ya está seleccionada.
- 4** Asegúrese de que está seleccionado Compartir instancia (estática) del módulo de datos. El asistente debe tener un aspecto semejante a éste:



- 5** Pulse Aceptar. Se añade una línea de código al método jbInit() para asociar el DataModule con el servlet.

Paso 8: Diseño del servlet

En este paso, se utilizará el diseñador para añadir componentes de InternetBeans al servlet. Estos componentes no serán visibles en el diseñador, ya que la GUI del servlet está realmente en el archivo HTML. No obstante, las propiedades de los componentes podrán verse en el inspector. Cuando se ejecuta el servlet, los componentes de InternetBeans que se añadan en este paso sustituirán los datos de prueba en el archivo HTML con datos del JDataStore.

- 1 Asegúrese de que el archivo `SignatureServlet.java` está abierto en el editor.
- 2 Pulse sobre la pestaña Diseño para abrir el diseñador de JBuilder.
- 3 Seleccione la pestaña InternetBeans de la paleta de componentes.
- 4 Seleccione el ícono `IxPageProducer` y suelte un `IxPageProducer` dentro del servlet pulsando en el diseñador.
- 5 Configure como sigue las propiedades del `IxPageProducer`:



Propiedad	Valor
<code>dataModule</code>	<code>DataModule11</code>
<code>htmlFile</code>	<code>gb1.html</code> — configurar esta propiedad rellena automáticamente la propiedad <code>rootPath</code>

S



- 6 Seleccione el ícono `IxControl` en la paleta. Suelte tres `IxControl` dentro del servlet pulsando en el diseñador.
- 7 Cuando quiera eliminar múltiples instancias de un control desde la paleta de componentes del diseñador, presione *Mayús* y haga clic en el ícono del control. Esto hace que el control permanezca seleccionado. Cuando haya finalizado con ese control, pulse en la herramienta de selección de la paleta de componentes con el fin de finalizar su selección.
- 7 Seleccione `ixControl1` y configure sus propiedades en el orden que se muestra a continuación:

Propiedad	Valor
<code>dataSet</code>	<code>signatures</code>
<code>columnName</code>	<code>Nombre</code>
<code>pageProducer</code>	<code>ixPageProducer1</code>
<code>controlName</code>	<code>Nombre</code>

8 Seleccione ixControl2 y configure sus propiedades como sigue:

Propiedad	Valor
dataSet	Signatures
columnName	Comentario
pageProducer	ixPageProducer1
controlName	Comentario



9 Seleccione el ícono `IxTable` de la paleta. Suelte una `IxTable` dentro del servlet pulsando en el diseñador.

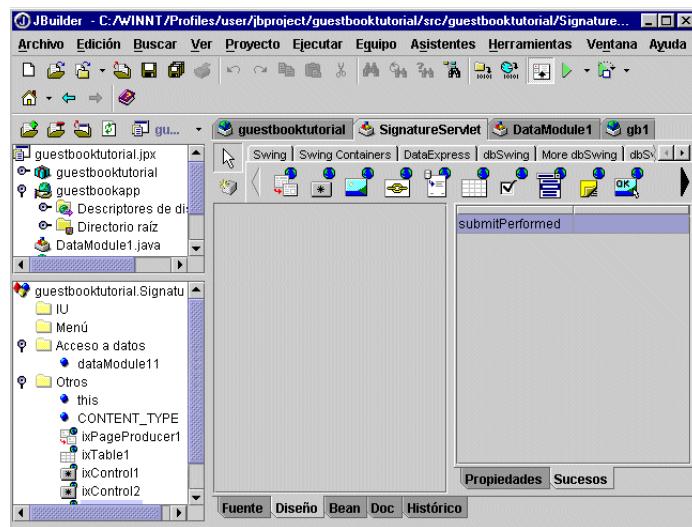
10 Configure las propiedades de `ixTable1` como sigue:

Propiedad	Valor
pageProducer	ixPageProducer1
dataSet	Signatures
elementId	guestbooktable

11 Seleccione `ixControl3` y configure sus propiedades como sigue:

Propiedad	Valor
pageProducer	ixPageProducer1
controlName	submit

12 Haga clic en la pestaña Sucesos del Inspector de propiedades. El IU presenta un aspecto similar al siguiente:



- 13** Pulse una vez en la propiedad `submitPerformed` en el inspector a fin de asignar al evento `submitPerformed()` el valor `ixControl3_submitPerformed`. Esto añade un monitor de eventos a `ixControl3`. Pulse Intro para generar el método `ixControl3_submitPerformed()`. Esto abre el editor y sitúa el cursor en el nuevo método.



- 14** Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 9: Edición del servlet

- 1** Asegúrese de que el archivo `SignatureServlet.java` está abierto en el editor.
- 2** Elimine el cuerpo del método `doGet()` del servlet generado por el asistente.
- 3** Elimine el cuerpo del método `doPost()` del servlet generado por el asistente.
- 4** Escriba la siguiente línea de código en el cuerpo del método `doGet()`:
`ixPageProducer1.servletGet(this, request, response);`
 Ahora está completo el método `doGet()`. A menudo todo lo que necesita hacer aquí es llamar al método `servletGet()` de la `IxPageProducer`.
- 5** Escriba las siguientes líneas de código dentro del cuerpo del método `doPost()`:

```
DataModule1 dm =
(DataModule1)
ixPageProducer1.getSessionDataModule(request.getSession());
dm.getSignatures().insertRow(false);
ixPageProducer1.servletPost(this, request, response);
doGet(request, response);
```

Cuando se envíe el formulario, este código obtiene una instancia por cada sesión del `DataModule`, inserta una línea vacía, llama a `IxPageProducer.servletPost()` para llenar la línea vacía con los valores escritos por el usuario y, a continuación, llama de nuevo a `doGet()` para mostrar los datos que se han enviado.

- 6** Luego necesita rellenar el cuerpo del método `ixControl3_submitPerformed()`. Este método es llamado por el método `servletPost()`. Introduzca el código siguiente en el cuerpo del método `ixControl3_submitPerformed()`:

```
DataModule1 dm =
(DataModule1) ixPageProducer1.getSessionDataModule(e.getSession());
dm.getSignatures().post();
dm.getSignatures().saveChanges();
```

Este código obtiene una instancia de `DataModule` por cada sesión y envía y guarda la entrada del usuario en el `JDataStore`. Advierta que esta

instancia por sesión es diferente de la instancia compartida almacenada en la variable `dataModule11`.



- 7 Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 10: Configuración de dependencias de la WebApp

En este paso, configurará las dependencias de la WebApp con el fin de asegurarse de que se incluyen las bibliotecas necesarias.

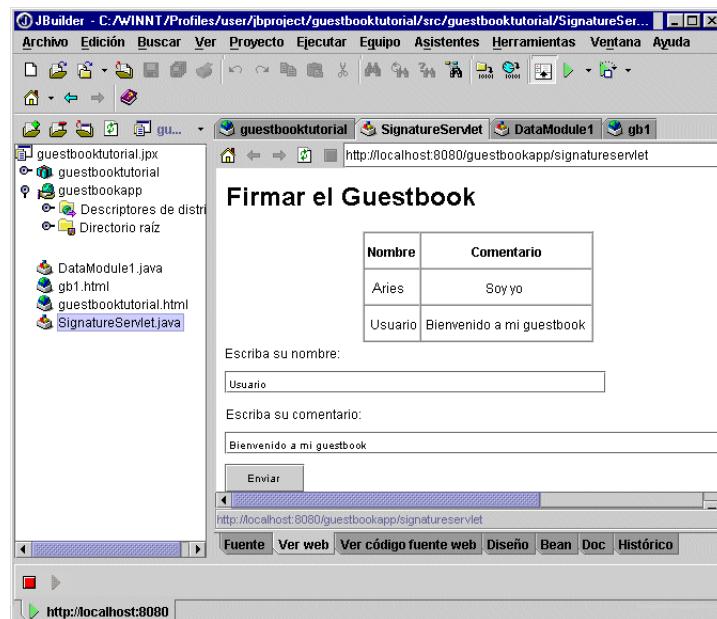
- 1 Haga clic con el botón derecho del ratón en el nodo `guestbookapp` del panel del proyecto.
- 2 Seleccione Propiedades en el menú contextual de la WebApp.
- 3 Seleccione la pestaña Dependencias del cuadro de diálogo de Propiedades.
- 4 Asegúrese de que las dependencias de las siguientes bibliotecas están configuradas como Incluir todo:
 - Data Express
 - Servidor de JDataStore
 - InternetBeans Express
 - dbSwing
- 5 Pulse Aceptar.

Paso 11: Ejecución del servlet

En este paso, ejecutará y probará el servlet.

- 1 Haga clic con el botón derecho en el archivo `SignatureServlet.java` en el panel del proyecto.
- 2 Elija Ejecutar web mediante “SignatureServlet” en el menú. El servlet se ejecuta en el IDE de JBuilder.
- 3 Pruebe el servlet borrando los valores existentes en los campos Nombre y Comentario. Introduzca su nombre y su comentario y pulse Enviar.

Su nombre y su comentario aparecen en la tabla y se guardan en el JDataStore.



- 4 Detenga el servlet pulsando el botón Terminar el programa en la pestaña Servidor web del panel de mensajes.

Distribución del servlet

Si desea obtener más información acerca de la distribución de servlets, consulte el [Capítulo 11, “Distribución de aplicaciones web”](#).

Tutorial: Creación de una página JSP con InternetBeans Express

El desarrollo web es una función de JBuilder Enterprise

Este tutorial le enseña cómo crear una JSP que contenga InternetBeans. Cuando acabe con el tutorial, tendrá una JSP que consulta una tabla de un JDataStore, muestra los comentarios del libro de visitas en una `IxTable` y permite que los visitantes introduzcan sus propios comentarios y los vean aparecer en el libro de visitas. Puede encontrar una versión acabada de la aplicación creada en `<JBuilder>/samples/WebApps/jspinternetbeans`.

Este tutorial presupone que está familiarizado con Java y las Páginas JavaServer (JSP), con el JBuilder IDE y con JDataStore. Para obtener más información sobre las JSP, consulte *Procedimientos iniciales con Java*. Si desea más información acerca de las JSP, consulte el [Capítulo 6, “Desarrollo de Páginas JavaServer”](#). Para obtener más información sobre el IDE de JBuilder, consulte “El entorno de JBuilder” en *Introducción a JBuilder*. Si desea obtener más información sobre JDataStore, consulte la *Guía del desarrollador de JDataStore*.

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

Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte “[Convenciones de la documentación](#)” en la página 1-4.

Paso 1: Creación de proyectos

- 1 Seleccione Archivo | Nuevo proyecto para abrir el Asistente para proyectos.
- 2 En el campo Nombre, introduzca un nombre de Proyecto, como jspixtutorial
- 3 Pulse Finalizar para cerrar el asistente para Proyectos y crear el proyecto. No se necesita hacer ningún cambio en los valores por defecto en los Pasos 2 y 3 del asistente.

Se crea un proyecto.

Paso 2: Selección de un servidor

En este paso, se seleccionará el servidor Tomcat 4.0 como servidor de este proyecto.

- 1 Seleccione Proyecto | Propiedades de proyecto. Aparece el cuadro de diálogo Propiedades de proyecto.
- 2 Pulse la pestaña Servidor.
- 3 Asegúrese de que el botón de radio Servidor único para todos los servicios del proyecto está seleccionado.
- 4 Asegúrese de que Tomcat 4.0 está seleccionado en la lista desplegable de servidores.
- 5 Pulse Aceptar.

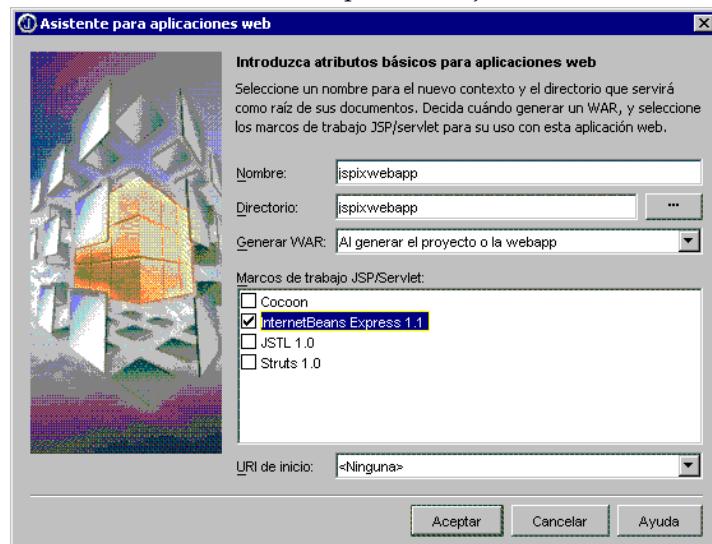
Paso 3: Creación de una WebApp

Este paso es optativo pero aconsejable. Puede utilizar la WebApp por defecto, pero suele ser menos confuso crear una WebApp con un nombre personalizado. Si desea obtener más información acerca de las WebApps y los archivos WAR, consulte el [Capítulo 3, “Las WebApps y los archivos WAR”](#).

- 1 Seleccione Nuevo del menú Archivo.
- 2 Haga clic en la pestaña Web de la galería de objetos. Seleccione Aplicación web.
- 3 Pulse Aceptar. Aparece el Asistente para aplicaciones web.
- 4 Escriba un nombre para la WebApp, como jspixwebapp. El campo Directorio se rellena automáticamente mientras escribe.

- 5 Deje la opción por defecto para Generar WAR, si bien no va a necesitar un archivo WAR ya que, probablemente, no deseará distribuir la aplicación de este tutorial.
- 6 Elija el marco de trabajo InternetBeans Express 1.1 JSP/Servlet. No especifique una URI de inicio.

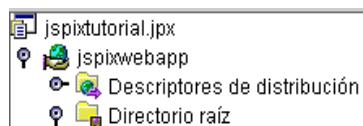
El asistente debe tener un aspecto semejante a éste:



- 7 Pulse Aceptar.

En el panel del proyecto se presenta un nuevo nodo WebApp, `jspixwebapp`. Expanda el nodo para ver el directorio raíz y los nodos de los descriptores de distribución.

Figura 20.1 Nodo WebApp en el panel del proyecto



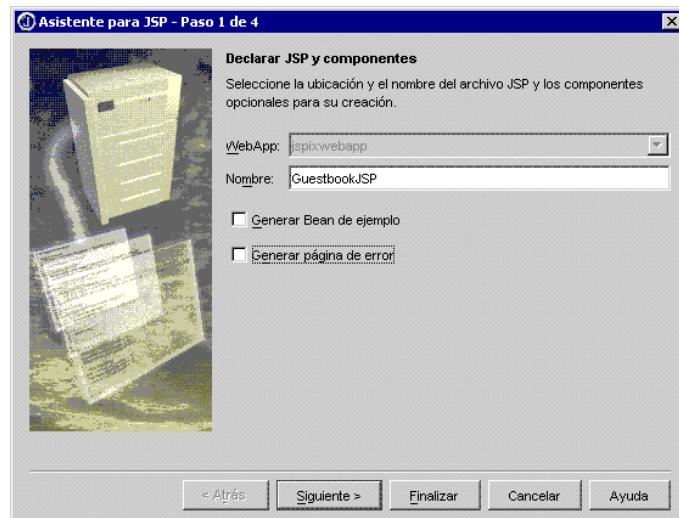
Paso 4: Utilización del Asistente para JSP

En este paso creará el esqueleto de una JSP mediante el Asistente para JSP.

- 1 Seleccione Archivo | Nuevo.
- 2 Haga clic en la pestaña Web. Seleccione Página JavaServer.
- 3 Pulse Aceptar. Se abre el Asistente para JSP.
- 4 En el campo nombre, escriba uno para la JSP: GuestbookJSP

5 Desactive Generar Bean de ejemplo.

El asistente para JSP presenta un aspecto similar al siguiente:

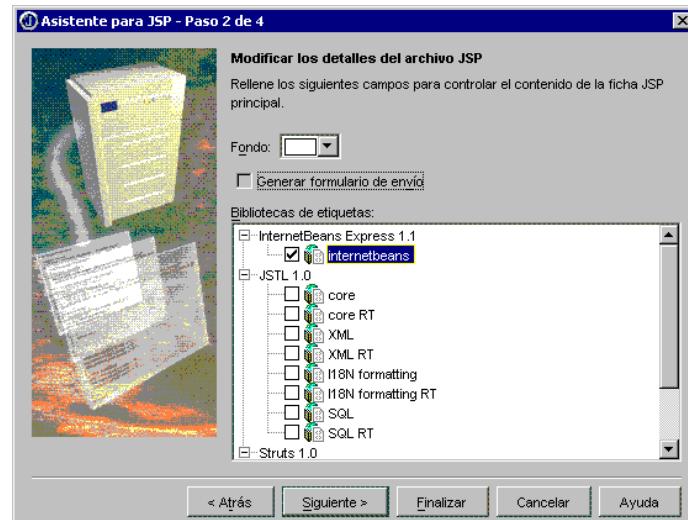


6 Pulse Siguiente.

7 Desactive Generar formulario de envío.

8 Marque internetbeans en InternetBeans Express 1.1, en el árbol Bibliotecas de etiquetas.

El asistente para JSP presenta un aspecto similar al siguiente:



9 Pulse el botón Finalizar. Se añade un archivo GuestbookJSP.jsp al nodo del Directorio raíz de su WebApp en el panel del proyecto. Expanda el nodo del Directorio raíz para ver el archivo. La JSP contiene la directiva

page y la directiva taglib, necesarias para utilizar la biblioteca de etiquetas de InternetBeans. El Asistente para JSP también se ocupa de los pasos necesarios para añadir la biblioteca InternetBeans al proyecto, según se describe en “[Utilización de InternetBeans Express con JSP](#)” en la [página 7-6](#). El Asistente para JSP también crea automáticamente una configuración de ejecución de manera que la JSP pueda ejecutarse en el IDE. Para obtener más información sobre las configuraciones de ejecución, consulte “Definición de las configuraciones de ejecución” en *Creación de aplicaciones con JBuilder*. Si desea más información sobre la creación de una configuración de ejecución con un asistente, consulte “[Creación de una configuración de ejecución con los asistentes](#)” en la [página 10-2](#).

Paso 5: Diseño de la parte HTML de la JSP

En este paso añadirá código HTML al archivo GuestbookJSP.jsp. El HTML que añada proporciona una plantilla para la forma en que los datos se muestran cuando se ejecuta la JSP.

- 1 Abra el archivo GuestbookJSP.jsp en el editor, si no lo está ya. Se encuentra en el nodo del Directorio raíz de la WebApp, en el panel del proyecto.
- 2 Cambie el contenido de la etiqueta <title> de modo que diga JSP/InternetBeans Tutorial.
- 3 Cambie el contenido de la etiqueta <h1> de modo que diga Sign the Guestbook
- 4 Escriba el siguiente código HTML en el cuerpo del archivo, debajo de la etiqueta </h1>:

```
<table id="guestbooktable" align="CENTER" cellspacing="0"
       border="1" cellpadding="7">
  <tr>
    <th>Nombre</th><th>Comentario</th>
  </tr>
  <tr>
    <td>Leo</td><td>Todo controlado!</td>
  </tr>
</table>

<form method="POST">
<p>Escriba su nombre:</p>

<input type="text" name="Nombre" size="50">

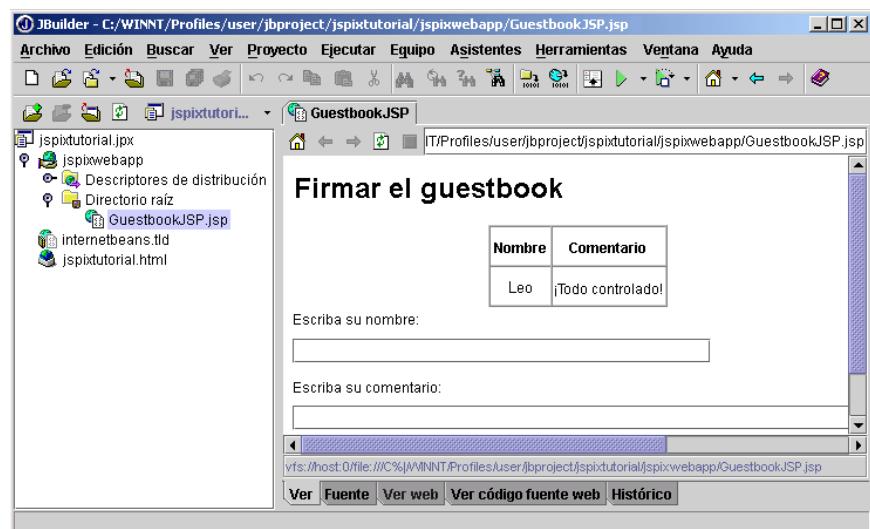
<p>Escriba su comentario:</p>

<input type="text" name="Comentario" size="100">
<p>
```

Paso 6: Adición de la etiqueta de base de datos de InternetBeans

```
<input type="submit" name="submit" value="Submit"></p>
</form>
```

Cuando haya acabado, el HTML debe tener esta apariencia en la pestaña Ver:



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 6: Adición de la etiqueta de base de datos de InternetBeans

En este paso, añadirá una etiqueta database de InternetBeans al JSP. La etiqueta database proporciona información de conexión con una fuente de datos.

- 1 Si cambió a la pestaña Ver en el paso anterior, vuelva al editor.
- 2 Añada la etiqueta de apertura database mostrada en negrita. Cambie el valor del atributo url de la etiqueta database para que apunte hacia el JDataStore guestbook.jds en <JBUILDER>\samples\WebApps\guestbook.

```
<h1>
Sign the guestbook
</h1>

<iix:database id="database1"
driver="com.borland.datastore.jdbc.DataStoreDriver"
url="jdbc:borland:dslocal:jbuilder\samples\WebApps\guestbook\
guestbook.jds"
username="user">

<table id="guestbooktable" align="CENTER" cellspacing="0"
border="1" cellpadding="7">
```

- 3 Añada la etiqueta de cierre database mostrada en negrita.**

```
</form>
</ix:database>
</body>
```



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 7: Adición de la etiqueta consulta de InternetBeans

En este paso, añadirá una etiqueta query a la JSP. La etiqueta query especifica una sentencia de consulta SQL.

- 1 Añada la etiqueta de apertura query mostrada en negrita.**

```
<ix:database id="database1"
driver="com.borland.datastore.jdbc.DataStoreDriver"
url="jdbc:borland:dslocal:jbuilder\\samples\\WebApps\\guestbook\\
guestbook.jds"
username="user">

<ix:query id="signatures" statement="select * from signatures">

<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
```

Observe que está anidando la etiqueta query dentro de la etiqueta database. Esto se hace de forma que el atributo database de la etiqueta query no necesite especificarse, dado que está implícito. También hace que el código sea más elegante.

- 2 Añada la etiqueta de cierre query mostrada en negrita.**

```
</ix:query>

</ix:database>
```



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 8: Adición de la etiqueta de tabla de InternetBeans

- 1 Añada las etiquetas de apertura y cierre table mostradas en negrita.**

```
<ix:query id="signatures" statement="select * from signatures">

<ix:table dataSet="signatures">

<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
<tr>
<th>Name</th><th>Comentario</th>
</tr>
<tr>
```

Paso 9: Adición de las etiquetas de control de InternetBeans

```
<td>Leo</td><td>¡Todo controlado!</td>
</tr>
</table>

</ix:table>

<form method="POST">
```

Observe que se está englobando la etiqueta `table` de HTML en la etiqueta `table` de InternetBeans. Esto permite que `IxTable` de InternetBeans entienda implícitamente a qué tabla está sustituyendo. La etiqueta `table` de InternetBeans está anidada dentro de la etiqueta `query` de InternetBeans. Esto no es necesario, ya que el atributo `dataSet` de la tabla establece una relación clara. El anidamiento de la tabla de InternetBeans dentro de una etiqueta `query` hace que el código sea más elegante.



- 2 Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 9: Adición de las etiquetas de control de InternetBeans

Ahora es el momento de añadir las dos etiquetas de `control` de los dos campos de introducción de texto. Las etiquetas `control` permiten suministrar datos dinámicos en lugar de los controles HTML que se utilizan como sustitutos de los datos. Añada las etiquetas mostradas en **negrita**.

```
<form method="POST">
<p>Escriba su nombre:</p>

<ix:control dataSet="signatures" columnName="Nombre">
<input type="text" name="Nombre" size="50">
</ix:control>

<p>Escriba su comentario:</p>

<ix:control dataSet="signatures" columnName="Comentario">
<input type="text" name="Comentario" size="100">
</ix:control>

<p>
```

Observe que se está englobando cada una de las etiquetas `input` de texto HTML en una etiqueta `control` de InternetBeans. Esto permite que `IxControls` de InternetBeans entiendan implícitamente a qué campos de introducción de texto están sustituyendo.



- Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 10: Adición de la etiqueta de envío de InternetBeans

Añada las etiquetas `submit` de apertura y cierre mostradas en **negrita**.

```
<input type="text" name="Comentario" size="100">
</ix:control>

<p>
<b><ix:submit methodName="submitPerformed"></b>

<input type="submit" name="submit" value="Submit"></p>

</ix:submit>

</form>
```

Observe que se está englobando la etiqueta `input` de envío HTML en una etiqueta `submit` de InternetBeans. Esto permite que `IxSubmitButton` de InternetBeans entienda implícitamente a qué botón de envío está sustituyendo.



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 11: Adición del método submitPerformed()

Todavía no hemos acabado con el botón de envío. Aún hay que añadir el método que se ejecutará cuando se presione el botón. Lo hará en este paso. Añada el código mostrado en **negrita**.

```
<ix:submit methodName="submitPerformed">

<%
    public void submitPerformed(PageContext pageContext){
        DataSet signatures = (DataSet) pageContext.findAttribute( "signatures" );
        signatures.post();
        signatures.saveChanges();
    }
%>

<input type="submit" name="submit" value="Submit"></p>

</ix:submit>
```

El método `submitPerformed()` está contenido dentro de una etiqueta de declaración de una JSP. Esta declaración del método no tiene que estar anidada dentro de la etiqueta `submit` de InternetBeans, pero es una forma elegante de escribir el código. El parámetro que se pasa a este método es `PageContext`. Este es un objeto que contiene información de la página, que existe para cada JSP.

Paso 12: Adición del código para insertar una fila

El método recupera un `DataSet` de DataExpress hallando el atributo `page` que se corresponde con el conjunto de datos “signatures”. Luego envía la entrada del usuario al conjunto de datos y guarda los cambios en el conjunto de datos.



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 12: Adición del código para insertar una fila

Aún hay un fragmento de código que es necesario añadir antes de que la JSP funcione correctamente. Cuando se envía el formulario, se necesita añadir una fila vacía al conjunto de datos para que contenga la entrada del usuario. Añada el código mostrado en **negrita**.

```
</ix:table>

<%
    signatures.insertRow(false);
%>

<form method="POST">
```

Este último fragmento de código Java puede parecer un poco confuso, ya que parece que no está encerrado en una declaración de método. Realmente lo está. Cuando se compila la JSP esto formará parte del método `service()` en el servlet generado (que no se puede ver, pero que sigue estando ahí). Cualquier línea de código dentro de una etiqueta de un scriplet de una JSP pasará a formar parte del método `service()`.

Este fragmento de código inserta una línea en el conjunto de datos justo antes de que se muestre el formulario. El formulario presenta valores vacíos. Entonces, cuando se envíe el formulario se escribirán los datos en la fila vacía antes de llamar al método `submitPerformed()`.



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 13: Adición de la biblioteca JDataStore Server al proyecto

Este proyecto necesita la biblioteca JDataStore Server. Para añadir esta biblioteca a las propiedades del proyecto:

- 1 Seleccione Propiedades de proyecto del menú Proyecto.
- 2 Pulse sobre la pestaña Vías de acceso.
- 3 Haga clic en la pestaña Bibliotecas necesarias.
- 4 Pulse Añadir.
- 5 Seleccione JDataStore Server en la carpeta JBuilder. Pulse Aceptar.

- 6** Pulse Aceptar para cerrar el cuadro de diálogo Propiedades de proyecto. En este punto, debería asegurarse de que las dependencias de las bibliotecas del proyecto están correctamente configuradas. Para ello:
- 7** Pulse con el botón derecho del ratón el nodo `jspxwebapp` del panel del proyecto y seleccione Propiedades en el menú contextual.
- 8** Seleccione la pestaña Dependencias del cuadro de diálogo de Propiedades.
- 9** Asegúrese de que las dependencias de las siguientes bibliotecas están configuradas como Incluir todo:
 - InternetBeans Express
 - dbSwing
 - Data Express
 - Servidor de JDataStore
- 10** Si alguna de estas bibliotecas no está configurada como Incluir todo, selecciónela y marque el botón de radio Incluir siempre todas las clases y recursos. El texto junto al nombre de la biblioteca cambia a Incluir todo. Una vez que todas las dependencias de bibliotecas estén correctamente configuradas, pulse Aceptar.



Haga clic en el botón Guardar todo en la barra de herramientas.

Paso 14: Ejecución de la página JSP

Ahora es el momento de ejecutar y probar la JSP.

- 1** Asegúrese de que el nodo del Directorio raíz de la WebApp está ampliado en el panel del proyecto.
- 2** Haga clic con el botón derecho en `GuestbookJSP.jsp` en el panel del proyecto.
- 3** Elija Ejecutar web mediante “GuestbookJSP” en el menú.

Se inicia Tomcat y la JSP se ejecuta dentro del IDE de JBuilder.

Nota

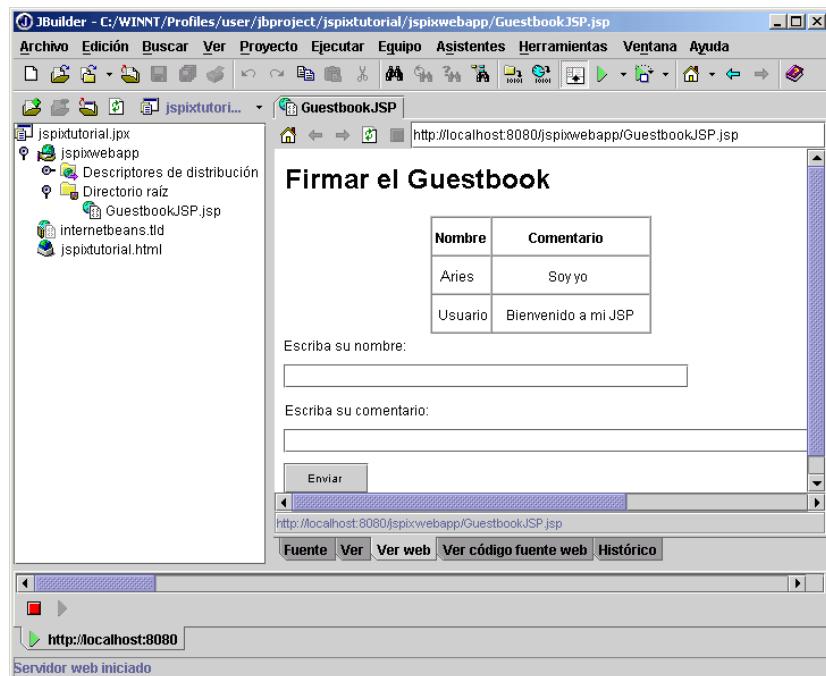
El comando Ejecutar web incluye el nombre de una configuración de ejecución, en este caso, “GuestbookJSP”. Esta configuración de ejecución fue creada automáticamente por el Asistente para JSP. Por defecto, el asistente asigna a la configuración el mismo nombre que la JSP. Para obtener más información sobre las configuraciones de ejecución, consulte “Definición de las configuraciones de ejecución” en *Creación de aplicaciones con JBuilder*.

- 4** Seleccione la URL que aparece en la parte superior de la vista web y cópiela. Péguela en el campo URL en el navegador con todas las prestaciones que prefiera y vaya a la URL.

Nota La vista web de JBuilder ofrece algunas funciones de navegador, pero no es un navegador con todas las prestaciones. Para obtener un mejor resultado, copie la URL de la aplicación web en ejecución en un navegador externo y ejecútela desde allí.

- 5 Escriba su nombre y sus comentarios en la JSP en ejecución en el navegador externo.
- 6 Pulse el botón Enviar del navegador externo. Su nombre y su comentario se añaden a la tabla (y se almacenan en el JDataStore).

Figura 20.2 Ejecución de la JSP en la vista web



Distribución de la JSP

Las JSP se distribuyen más fácilmente que los servlets. Esto se debe a que un servidor web las encuentra de la misma forma que procede con los archivos HTML. No se tiene que hacer una instalación especial, ya que el servidor Web es competente para saber lo que tiene que hacer con la JSP. Para obtener más información sobre la distribución de las JSP, consulte el Capítulo 11, “Distribución de aplicaciones web”.

21

Tutorial: Ejecución de la aplicación de ejemplo **CheckBoxControl con Java Web Start**

Este tutorial le acompañará a través de los pasos necesarios para iniciar una aplicación de ejemplo basada en Swing con WebStart. El ejemplo, CheckBoxControl, está situado en el directorio samples/Swing de su instalación de JBuilder. Este tutorial da por supuesto que en su ordenador está instalado Java WebStart. Si desea obtener las instrucciones de instalación, consulte “[Instalación de Java Web Start](#)” en la página 15-3.

Si desea obtener más información acerca de Java WebStart y JBuilder, consulte el [Capítulo 15, “Ejecución de aplicaciones web con Java Web Start”](#).

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

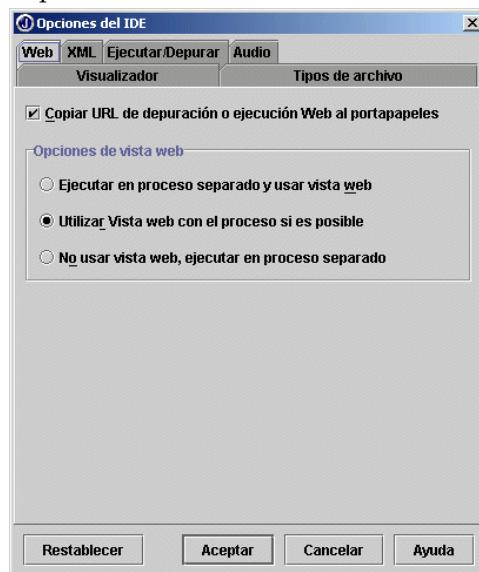
Para obtener información sobre las convenciones utilizadas en este tutorial y en otra documentación de JBuilder, consulte “[Convenciones de la documentación](#)” en la página 1-4.

Paso 1: Apertura y configuración del proyecto

Primero, abra el proyecto en JBuilder y fije las opciones del IDE de la vista web. Para ello:

- 1 Seleccione Archivo | Abrir proyecto con el fin de presentar el cuadro de diálogo homónimo.
- 2 En el cuadro de diálogo Abrir proyecto, pulse el botón Ejemplos. Busque Swing/CheckBoxControl/. Haga clic en CheckBoxControl.jpx y pulse Aceptar para abrir el proyecto.
- 3 Seleccione Herramientas | Opciones del IDE con el fin de abrir el cuadro de diálogo homónimo. En la ficha Web, compruebe que está seleccionada la opción Copiar URL de depuración o ejecución web al portapapeles. Esta opción copia la URL generada por una ejecución web en el portapapeles, de manera que pueda pegarse directamente en un visualizador externo.

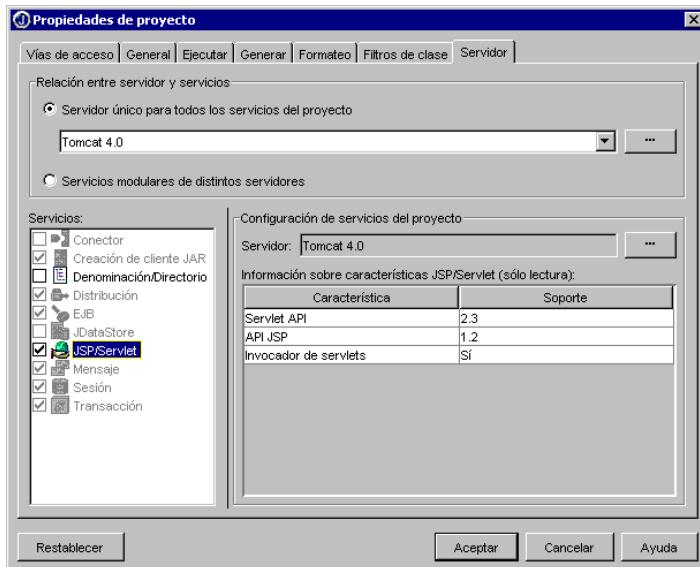
La ficha Web del cuadro de diálogo Opciones del IDE tiene este aspecto:



- 4 Pulse Aceptar para cerrar el cuadro de diálogo.
- 5 Elija Proyecto | Propiedades de proyecto y se abrirá el cuadro de diálogo homónimo. Abra la pestaña Servidores.
- 6 Elija Tomcat 4.0 en la lista desplegable Servidor único para todos los servicios del proyecto.

- 7** Compruebe que sólo está seleccionado el servicio JSP/Servlet en la lista Servicios, a la izquierda del cuadro de diálogo.

La ficha Servidor, con el servicio JSP/Servlet seleccionado, tiene la siguiente apariencia:



- 8** Pulse Aceptar para cerrar el cuadro de diálogo.

Si desea lo que hace esta aplicación, puede ejecutarla eligiendo Proyecto | Ejecutar el proyecto. La aplicación demuestra cómo utilizar el control CheckBox de Swing. (No está utilizando WebStart en este momento).

Tenga en cuenta que esta aplicación no contiene operaciones de manipulación de ficheros, recomendadas para su ejecución con Java WebStart. Una aplicación WebStart, al no distribuirse igual que un applet, no puede realizar operaciones de gestión de archivos a menos que, digitalmente, firme el archivo JAR o utilice las clases JNLP. Para obtener más información acerca de Java Web Start y de las cuestiones de seguridad, consulte ["Consideraciones sobre las aplicaciones Java Web Start"](#) en la página 15-2.

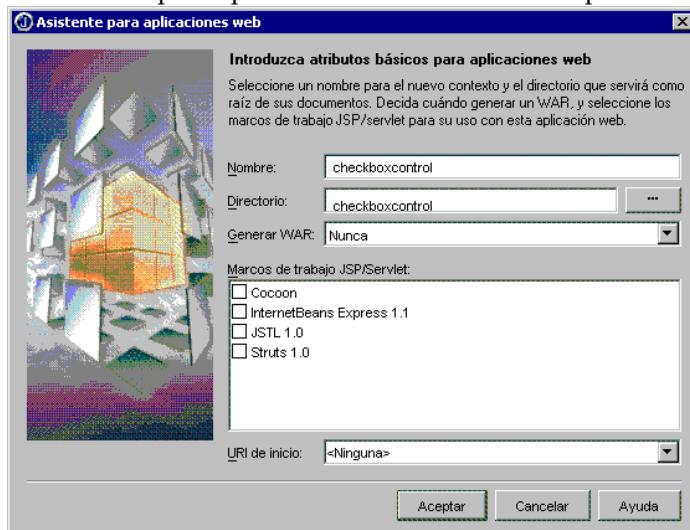
Paso 2: Creación de la WebApp de la aplicación

Para crear una WebApp, utilice el Asistente para aplicaciones web. Si desea obtener más información sobre las clases, consulte el [Capítulo 3, "Las WebApps y los archivos WAR"](#).

Para crear la WebApp de la aplicación:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo. En la ficha Web, seleccione Aplicación web y pulse Aceptar.
- Se abre el Asistente para aplicaciones web.
- 2 Introduzca `checkboxcontrol` en el campo Nombre. El campo Directorio se rellena mientras escribe.
- 3 Asigne el valor Nunca a la opción Generar WAR. No seleccione ningún marco de trabajo JSP/Servlet.

El Asistente para aplicaciones web tendrá este aspecto:



- 4 Pulse Aceptar para cerrar el asistente.

La WebApp `checkboxcontrol` se presenta en el panel de proyecto como un nodo. Despliegue el nodo con el fin de ver el Descriptor de distribución y los nodos del directorio raíz.

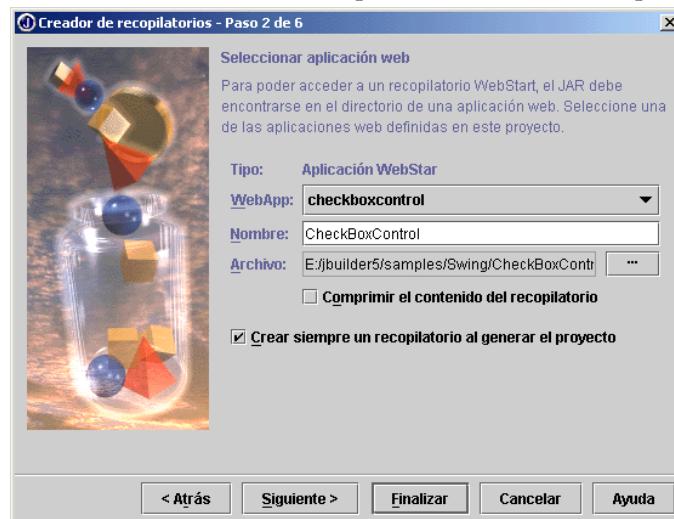
Paso 3: Creación del archivo JAR de la aplicación

Para iniciar una aplicación como aplicación Web Start, debe crear una archivo JAR. Para crear archivos JAR se utiliza el Creador de recopilatorios de JBuilder:

- 1 Compile el proyecto. Seleccione Proyecto | Ejecutar Make del proyecto "CheckBoxControl.jpr".
- 2 Seleccione Asistentes | Creador de recopilatorios.
- 3 Cambie el Tipo de recopilatorio a Aplicación Web Start en la ficha Seleccione un tipo de recopilatorio, del Creador de recopilatorios. Pulse Siguiente.
- 4 En el paso 2, compruebe que está seleccionado checkboxcontrol en la lista desplegable de la ficha Seleccionar aplicación web.
- 5 Cambie Nombre a CheckBoxControl.

El campo Archivo se ha llenado automáticamente. El nombre del archivo JAR se basa en el nombre del proyecto. El archivo JAR se coloca en la carpeta samples/Swing/CheckBoxControl/webapp de su proyecto.

El Paso 2 del Creador de recopilatorios tendrá este aspecto:



- 6 Pulse Finalizar para crear el archivo y cerrar el asistente. No tiene que cambiar ninguna opción en los pasos restantes del asistente.
- 7 Elija Archivo | Guardar todo.
- 8 Seleccione Proyecto | Ejecutar Make del proyecto "CheckBoxControl.jpx" para crear el archivo JAR.

El asistente crea un nodo de recopilatorio y lo presenta en el panel de proyecto. El recopilatorio será generado cada vez que se genere el proyecto.

Paso 4: Creación de la página de inicio y del archivo JNLP de la aplicación

En este paso, utilizará el Asistente para el inicio de Web Start con el fin de crear el archivo JNLP y la página de inicio de la aplicación. La página de inicio es un archivo HTML que se abre en su visualizador externo. Contiene un enlace a su aplicación. Cuando se pulsa el enlace, el archivo JNLP le dice a Java Web Start cómo iniciar su aplicación.

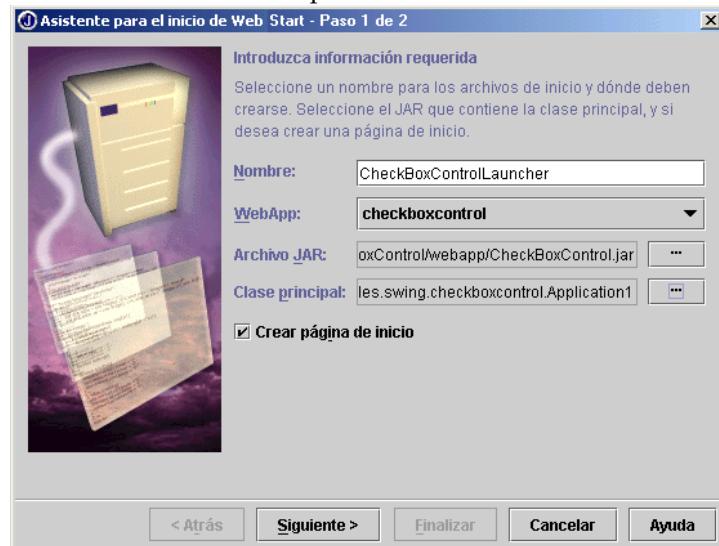
Para crear estos archivos:

- 1 Para mostrar la galería de objetos, seleccione Archivo | Nuevo.
- 2 Seleccione Inicio de Web Start y pulse Aceptar en la ficha Web.
Se presenta el Asistente para el inicio de Web Start.
- 3 Introduzca `CheckBoxControlLauncher` en el campo Nombre.
Esta opción da nombre al archivo HTML y al archivo JNLP.
- 4 Compruebe que `checkboxcontrol` está seleccionado en la lista desplegable de WebApp.
- 5 Pulse el botón de puntos suspensivos, situado a la derecha del campo Archivo JAR. Esto abre el cuadro de diálogo Seleccionar JAR para Web Start en donde se escoge el nombre del archivo JAR generado con el Creador de recopilatorios. Este es `CheckBoxControl.jar`. Está en el directorio `CheckBoxControl/webapp`. Seleccione el archivo JAR en el cuadro de diálogo Seleccionar JAR para Web Start y pulse Aceptar para cerrarlo.
- 6 Haga clic sobre el botón puntos suspensivos, situado a la derecha del campo Clase principal, si éste no está ya relleno. Aparece el cuadro de diálogo Seleccionar Clase principal. Abra la pestaña Examinar. Despliegue la carpeta `com` en la parte superior del cuadro de diálogo para seleccionar `com.borland.samples.swing.checkboxcontrol.Application1`. Pulse Aceptar para cerrar el cuadro de diálogo.
- 7 Asegúrese de que la opción Crear página de inicio está seleccionada en el Asistente para el inicio de Web Start. Esta opción crea el archivo HTML que inicia la aplicación.

Advertencia

Si el nombre introducido en el campo Nombre coincide con el nombre de un archivo JNLP o HTML ya existente en su proyecto, se le pregunta si quiere sobreescribirlo.

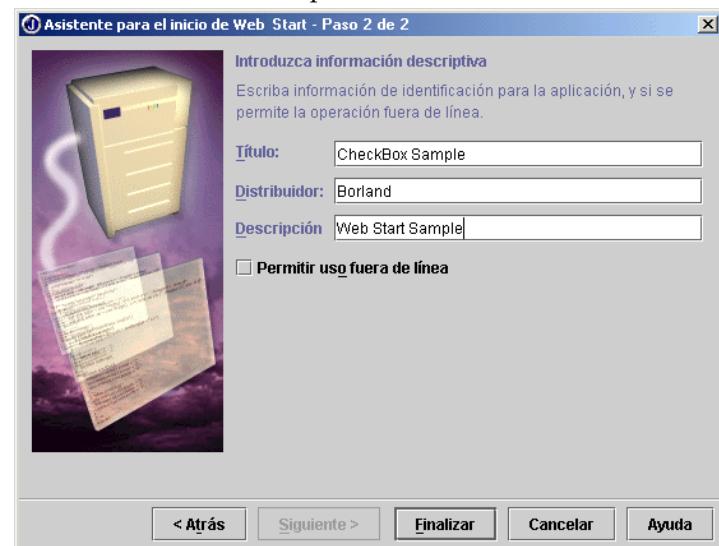
La ficha Introduzca información requerida del Asistente para el inicio de Web Start tiene este aspecto:



8 Pulse Siguiente.

9 Introduzca CheckBox Sample en el campo Título. Introduzca Borland en el campo Distribuidor y Web Start Sample en el campo Descripción. Asegúrese de que no está seleccionada la opción Permitir uso fuera de línea.

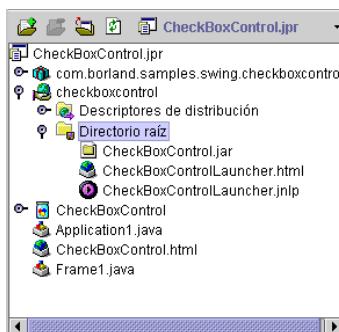
La ficha Introduzca información descriptiva del Asistente para el inicio de Web Start tiene este aspecto:



10 Pulse el botón Finalizar.

El asistente crea un archivo HTML y un archivo JNLP de nombre CheckBoxControlLauncher y los coloca en el directorio raíz de su aplicación web. Si desea ver estos archivos en el panel de proyecto, despliegue el nodo Directorio raíz del nodo de la WebApp checkboxcontrol.

El panel de proyecto debe ser similar a:



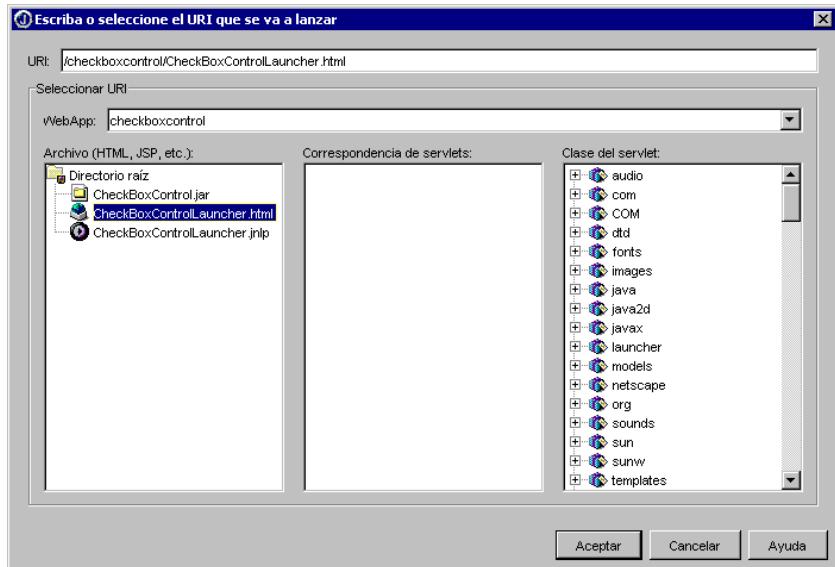
Nota Estos archivos pueden abrirse en el editor; no obstante, no los modifique.

Paso 5: Creación de una configuración de ejecución del servidor

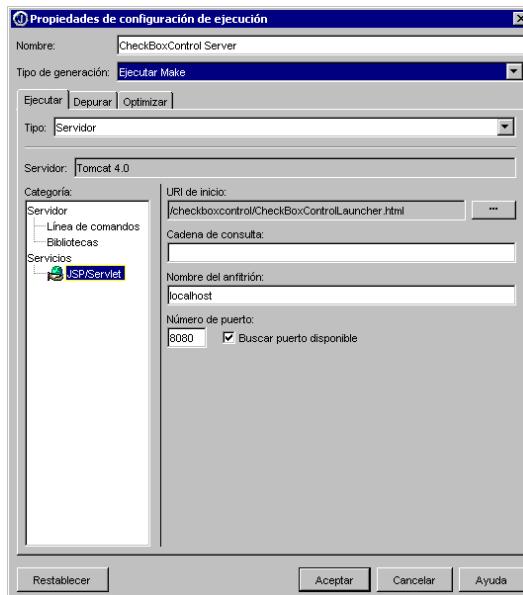
En este paso se crea una configuración de ejecución del servidor. Esto permite ejecutar el ejemplo utilizando un servidor.

- 1** Elija Ejecutar | Configuraciones. Se abre la ficha Ejecutar del cuadro de diálogo Propiedades de proyecto.
- 2** Pulse el botón Nuevo con el fin de abrir el cuadro de diálogo Propiedades de configuración de ejecución.
- 3** Introduzca CheckBoxControl Server en el campo Nombre.
- 4** Seleccione Servidor en la lista desplegable Tipo.
- 5** Pulse JSP/Servlet en la lista Categorías de la parte izquierda del cuadro de diálogo.
- 6** Pulse el botón de puntos suspensivos junto al cuadro de diálogo URI de inicio con el objeto de abrir el cuadro Escriba o seleccione el URI que se va a lanzar.

- 7 En la lista Archivo de la parte izquierda del cuadro de diálogo, elija CheckBoxControlLauncher.html. Observe que este archivo se encuentra en el directorio raíz de la aplicación web. El cuadro de diálogo Escriba o seleccione el URI que se va a lanzar tendrá este aspecto:



- 8 Pulse Aceptar para cerrar el cuadro de diálogo.
- 9 El cuadro de diálogo Propiedades de configuración de ejecución tendrá un aspecto parecido a:



- 10** Pulse Aceptar dos veces para cerrar los cuadros de diálogo Propiedades de configuración de ejecución y Propiedades de proyecto.

Para obtener más información, consulte “[Creación de una configuración de ejecución del servidor](#)” en la página 10-5.

Paso 6: Inicio de la aplicación

Este paso indica cómo iniciar su aplicación con Web Start. Para ello:

- 1** Haga clic con el botón derecho en CheckBoxControlLauncher.html en el panel del proyecto y seleccione Ejecutar web mediante “CheckBoxControl Server”. (Se encuentra en la carpeta Directorio raíz del nodo WebApp del panel del proyecto).

JBuilder compila los archivos e inicia el servidor web Tomcat. A causa de que el archivo JNLP especifica que esta aplicación debe ejecutarse con Web Start, JBuilder muestra un mensaje de advertencia en la vista web. La vista web tiene este aspecto:

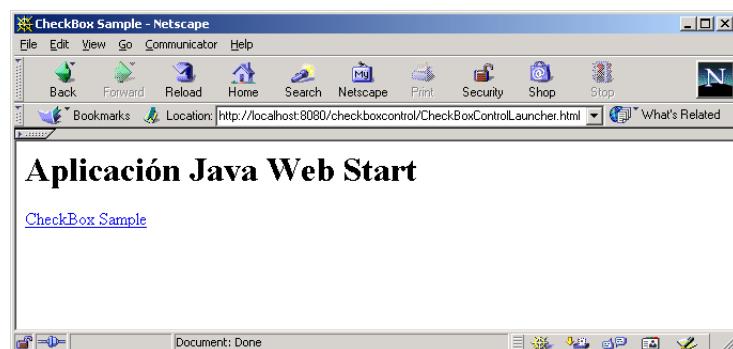


- 2** Sitúe el cursor en el campo Ubicación del visualizador externo y presione *Ctrl+V*. Esta acción copia la URL de Ejecutar web desde el portapapeles. JBuilder copió esta URL en el portapapeles basándose en su selección en la ficha Web del cuadro de diálogo Opciones del IDE. (Esta opción se seleccionó en un paso anterior de este tutorial). Pulse Intro para ir a la URL.

Importante

Si su navegador externo es Netscape 6, es necesario que lleve a cabo una configuración adicional. Si desea obtener más información, diríjase a “Using Java Web Start Technology with NetscapeTM 6 Web Browsers” de *Java Web Start Installation Guide* en <http://java.sun.com/products/javawebstart/docs/installguide.html>.

Su visualizador web muestra la página de inicio de la aplicación, CheckBoxControlLauncher.html. La página web contiene un enlace a la aplicación.



- 3 Haga clic sobre el enlace en la página web. Java Web Start carga e inicia la aplicación. La aplicación se está ejecutando ahora desde un enlace en un visualizador web externo. Advierta que la pantalla de inicio muestra la información que se introdujo en el Asistente para el inicio de Web Start.
- 4 Elija Archivo | Salir y se cerrará la aplicación. Si desea detener el servidord web, pulse el botón Terminar el programa en la pestaña Servidor web.

En este tutorial, ha aprendido a configurar un proyecto para una aplicación Web Start, utilizar el Asistente para el inicio de Web Start para crear la página de inicio y el archivo JNLP de la aplicación y, además, iniciar la aplicación con Web Start.

Índice

A

API de servlet 4-3
API JSP
 directivas
 page 7-7
 taglib 7-7
aplicaciones distribuidas
 y aplicaciones web 2-12
aplicaciones web 1-1, 2-12, 3-1, 10-1
applets 14-1
 bibliotecas de terceros 14-14
 comprobar 14-15
 depurar 14-23
 depurar en un navegador 14-25
 descripción general 2-7, 14-2
 ejecutar 14-21
 ejecutar applets JDK 1.1.x en JBuilder 14-23
 ejecutar JDK 1.2 14-23
 implementación de Java en navegadores 14-6
 incluir en un archivo WAR 3-20
 Java Web Start 14-8
 Máquina virtual de Java 14-2
 navegadores 14-5, 14-8
 Plug-in de Java 14-8
 recopilar 14-10
 sugerencias 14-9
 utilizar paquetes 14-10
AppletTestbed 14-22
 depurar applets 14-23
appletviewer 14-22
 depurar applets 14-23
archivo web.xml 3-1
archivos internetbeans.tld 7-7
 formato 7-10
 tabla de etiquetas JSP 7-9
archivos JAR
 atributo archive de applets 14-3
 autenticar 14-12
 incluir en archivo WAR 3-20
archivos recopilatorios
 distribuir a un servidor web 11-1
archivos SHTML 5-7
archivos struts-config.xml
 editar 13-1
archivos WAR (recopilatorio web)
 añadir applets 3-20
 añadir archivos JAR 3-20
 compresión 3-10
 crear 3-10

definición de 3-18
directorios a incluir 3-11
distribuir 11-1
generar 3-3
herramientas 3-2
nombre de configuración de 3-10
propiedades 3-18
relación con la WebApp 3-18
tipos de archivo incluidos 3-18
ubicación de la configuración de 3-10
ver contenido de 3-18
archivos web.xml 11-4
 añadir restricciones de seguridad 12-2
 añadir un filtro 12-2
 añadir un servlet 12-2
 añadir una colección de recursos web 12-2
 asignar página de error 12-14
 asignar Tipo de MIME 12-13
 bibliotecas de etiquetas 12-11
 creación de 3-8
 editar 3-2, 12-1
 entradas de entorno 12-14
 etiquetas de asignación de filtro 12-5
 etiquetas de monitor 12-7
 etiquetas de servlet 12-8
 etiquetas taglib 12-11
 más información 11-6
 método de autentificación 12-17
 referencias de entorno de recurso 12-16
 Referencias de la factoría de conexión del gestor de recursos 12-16
 referencias EJB 12-15
 referencias EJB locales 12-16
 Restricciones de seguridad 12-18
 y Struts 8-17
 y struts-config.xml 8-17
archivos ZIP
 atributo archive de applets 14-3
asistentes
 para Action 8-10
 para ActionForm 8-7
 para aplicaciones web 3-3
 compatibilidad con Struts 8-6
 para applets 14-18
 para conversión de Struts 8-14
 para JSP 6-11, 7-7
 compatibilidad con Struts 8-7
 e InternetBeans Express 7-7
 para JSP a partir de ActionForm 8-12
 para servlets

opciones 5-1
atributos
archive, etiqueta de applets 14-3
code, etiqueta de applets 14-3
codebase, etiqueta de applets 14-3
height, etiqueta de applets 14-3
hspace, etiqueta de applets 14-3
name
 archivo internetbeans.tld 7-10
name, etiqueta de applets 14-3
required, archivo internetbeans.tld 7-10
rtexprvalue, archivo internetbeans.tld 7-10
tagclass, archivo internetbeans.tld 7-10
vspace, etiqueta de applets 14-3
width, etiqueta de applets 14-3
autenticar applets 14-12
autentificación
 para una WebApp 12-17

B

bibliotecas de etiquetas
 configurar 6-7
 importar en JSP 8-7
 Struts 8-1, 8-4
 utilizar en webapp 8-6
 y JBuilder 8-4
bibliotecas de etiquetas InternetBeans Express 7-7
 etiquetas
 control 7-9
 database 7-9
 image 7-9
 query 7-9
 submit 7-9
 table 7-9
bibliotecas de etiquetas JSP 6-4
 configurar 6-7
 InternetBeans Express 7-7
bibliotecas de terceros
 applets 14-14
Borland
 asistencia
 a desarrolladores 1-6
 técnica 1-6
 contacto 1-6
 correo electrónico 1-8
 grupos de noticias 1-8
 informar sobre errores 1-8
 recursos en línea 1-7
 World Wide Web 1-7

C

CGI (Interfaz de enlace común)
 en comparación con los servlets 2-2
clases
 Action 8-1, 8-10
 y struts-config.xml 8-10
 ActionForm 8-1, 8-7, 8-12
 y struts-config.xml 8-7
 AtionServlet 8-1
database
 utilizar en un servlet 7-3
 utilizar en una JSP 7-9
IxCheckBox 7-2
IxComboBox 7-2
IxControl 7-2
 utilizar en un servlet 7-3
IxHidden 7-2
IxImage 7-2
 utilizar en una JSP 7-9
IxImageButton 7-2
IxLink 7-2
IxListBox 7-2
IxPageProducer 7-2
 método servletGet() 7-3
 método servletPost() 7-5
 utilizar en un servlet 7-3
IxPassword 7-2
IxPushButton 7-2
IxRadioButton 7-2
IxSpan 7-2
IxSubmitButton 7-2
 utilizar en un servlet 7-5
 utilizar en una JSP 7-9
IxTable 7-2
 generar tablas 7-7
 utilizar en una JSP 7-9
IxTextArea 7-2
IxTextField 7-2
QueryDataSet
 utilizar en un servlet 7-3
 utilizar en una JSP 7-9
colección de recursos web
 añadir a archivo web.xml 12-2
comandos
 Depurar web 10-1
 JSP 6-14
 Ejecutar web 10-1, 10-16, 10-17, 10-18
 JSP 6-14
comandos web
 configuración del IDE para 9-7
compatibilidad con Java 14-5
compilar
 applets 14-21

JSP 6-13, 10-14
servlets 10-14
comprobar
 applets 14-15
configuración de bibliotecas 6-7
configuración del servidor web 9-1
configuraciones de ejecución
 JSP 10-2
configuraciones de ejecución de applets 10-2
 crear con asistentes 10-2
 crear mediante Ejecutar | Configuraciones 10-3
configuraciones de ejecución de JSP 10-2
configurar archivos ejecutables 10-5
 crear con asistentes 10-2
 crear mediante Ejecutar | Configuraciones 10-5
configuraciones de ejecución de servlets 5-11, 10-2
configurar archivos ejecutables 10-5
 crear con asistentes 10-2
 crear mediante Ejecutar | Configuraciones 10-5
convenciones de la documentación 1-4, 1-5
cuadros de diálogo
 Configurar bibliotecas 6-7

D

DataExpress
 utilizar en un servlet 7-1, 7-3
 utilizar en una JSP 7-1, 7-7
denominación de servlets 5-7, 10-10
depuración del código fuente de las JSP 10-19
depuración web
 JSP 10-19
 servlets 10-19
depurar
 applets 14-23
 applets en un navegador 14-25
desarrollo web
 proceso básico 2-11
descriptores de distribución 3-1
 aplicación web 11-4
 archivo web.xml 3-8, 12-1
 editar 3-2
Editor DD de WebApp 12-1
Editor de configuración de Struts 13-1
más información 11-6
nodo de WebApp 3-8
para una WebApp 3-6
propios del fabricante para una WebApp 11-5
 struts-config.xml 13-1
directorio raíz
 de una WebApp 3-6
directorio WEB-INF 3-6
directorios
 de una WebApp 3-11

distinción entre mayúsculas y minúsculas
en applets y etiquetas APPLET 14-10
distribuir
 aplicaciones 15-1
 applets 14-9, 14-14
 en recopilatorios 14-10
 archivo recopilatorio 11-1
 archivo WAR 11-1
 archivos recopilatorios de applets 14-10
 JSP 11-3
 por tipo de archivo 3-18
 servlets 4-9, 11-2
 WebApp 11-1
documentación
 convenciones 1-4
 de plataformas 1-5

E

Editor DD de WebApp 12-1
añadir restricciones de seguridad 12-2
añadir un filtro 12-2
añadir un servlet 12-2
añadir una colección de recursos web 12-2
fichas
 Bibliotecas de etiquetas 12-11
 Conexión 12-17
 Descriptor de distribución para la
 WebApp 12-3
 Entradas de entorno 12-14
 Filtros 12-5
 Monitores 12-7
 Páginas de error 12-14
 Referencias de la factoría de conexión del
 gestor de recursos 12-16
 Referencias del entorno de recursos 12-16
 Referencias EJB 12-15
 Referencias EJB locales 12-16
 Seguridad 12-18
 Servlets 12-8
 Tipo de MIME 12-13
menú contextual 12-2
Editor de configuración de Struts 8-16
fichas
 Correspondencias Action 13-14
 Form Beans 13-6
 Fuentes de datos 13-3
 Reenvíos globales 13-10
 menú contextual 13-3
ejecutar applets 14-21
ejecutar web
 JSP 10-15
 servlets 10-15
Ejecutar web (comando) 10-16, 10-18

elemento action-mappings
struts-config.xml 13-14

elemento data-source
struts-config.xml 13-3

elemento form-beans
struts-config.xml 13-6

elemento global-forwards
struts-config.xml 13-10

enlazados a datos
JSP 7-1
servlets 5-15, 7-1

espacio de contención
seguridad de las aplicaciones Web Start 15-2
seguridad de las applets 14-11

etiquetas
control, InternetBeans Express 7-9
database, InternetBeans Express 7-9
de applets 14-2
atributos 14-3
errores 14-4
image, InternetBeans Express 7-9
IxControl
utilizar en una JSP 7-9
param, applets 14-3
query, InternetBeans Express 7-9
submit, InternetBeans Express 7-9
table, InternetBeans Express 7-9

etiquetas JSP 6-3
comment 6-3
declaration 6-3
directivas
page 6-3
taglib 6-3
expression 6-3
getProperty 6-3
scriptlet 6-3
setProperty 6-3
useBean 6-3

F

fichas
Bibliotecas de etiquetas
en Editor DD de WebApp 12-11

Clases, de Propiedades de WebApp 3-13

Conexión
en Editor DD de WebApp 12-17

Correspondencias Action
Editor de configuración de Struts 13-14

Dependencias, de Propiedades de WebApp 3-16

Descriptor de distribución para la WebApp 12-3

Descriptor, de Propiedades de WebApp 3-18

Entradas de entorno
en Editor DD de WebApp 12-14

Filtros
en Editor DD de WebApp 12-5

Form Beans
Editor de configuración de Struts 13-6

Fuentes de datos
Editor de configuración de Struts 13-3

Monitores
en Editor DD de WebApp 12-7

Páginas de error
en Editor DD de WebApp 12-14

Reenvíos globales
Editor de configuración de Struts 13-10

Referencias de la factoría de conexión del gestor de recursos
en Editor DD de WebApp 12-16

Referencias del entorno de recursos
en Editor DD de WebApp 12-16

Referencias EJB
en Editor DD de WebApp 12-15

Referencias EJB locales
en Editor DD de WebApp 12-16

Seguridad
en Editor DD de WebApp 12-18

Servlets
en Editor DD de WebApp 12-8

Tipo de MIME
en Editor DD de WebApp 12-13

filtro
añadir a archivo web.xml 12-2

fuentes
convenciones empleadas en la documentación de JBuilder 1-4

G

generar tablas
con InternetBeans Express 7-7

grupos de noticias
Borland 1-8
public 1-8

H

hilos de servlet
de hilo simple 5-1
multihilo 4-7

I

importar
aplicación web 3-3
archivos para una WebApp 3-7

iniciar

JSP 10-5
 servlets 10-5
 URI 10-2, 10-5
instalar Web Start 15-3, 15-5
Interfaz de enlace común (CGI)
 en comparación con los servlets 2-2
InternetBeans
 tutorial 19-1, 20-1
InternetBeans Express 7-1
 analizar HTML 7-5
 biblioteca de etiquetas 7-7
 descripción general 2-6
 envío de los datos del servlet 7-5
 formato de archivo de biblioteca de
 etiquetas 7-10
 generar tablas 7-7
 tabla de clases 7-2
 tabla de etiquetas JSP 7-9
 utilizar con JSP 6-4
visualización de datos de servlet 7-3
y JSP 7-7
y servlets 7-3

J

Java Network Launching Protocol 15-1
 Consulte también JNLP
Java Web Start 15-1
 Consulte también Web Start
JBuilder
 utilización de WebApps 10-1
JNLP 15-1
 archivo JNLP 15-8
JSP (Páginas JavaServer)
 asignar bibliotecas de etiquetas 12-11
 bibliotecas de etiquetas 6-4
 características de JBuilder 6-6
 compilar 6-13, 10-14
 convertir en Struts 8-14
 crear a partir de ActionForm 8-12
 crear con el asistente 6-11
 depuración del código fuente 10-19
 depuración web 6-14, 10-19
 desarrollo 6-10
 descripción general 2-4
 distribuir 11-3
 e InternetBeans Express 7-7
 ejecutar web 6-14, 10-15
 enlazados a datos 7-1
 incluir en archivo WAR 3-18
 iniciar 10-5
 InternetBeans Express 6-4
 JSTL 6-4
 marcos de trabajo 6-4, 6-6

propiedades de ejecución 10-13
sintaxis 6-3
Struts 6-4
tutorial 18-1, 20-1
vínculos 6-14
y servlets 4-2
JSTL (Biblioteca de etiquetas estándar para
 Páginas JavaServer) 6-4
 descripción general 2-7

L

llamar a los servlets 5-13

M

mapeo de servlets 5-7, 10-2, 10-10
marcos de trabajo 6-6
 Cocoon 8-4
 configurar 6-7
 cuadro de diálogo Configurar bibliotecas 8-4
 definidos por el usuario 6-7
 InternetBeans 8-4
 JSTL 8-4
 Struts 8-4
 utilizar con JSP 6-4
métodos
 de servlets
 redefinir estándar 5-5
 servletGet() 7-3
 servletPost() 7-5

N

navegadores 14-5
 diferencias en la implementación de Java 14-6
 ejecutar applets 14-5
 llamada a servlets desde 5-13
 Plug-in de Java 14-8
 sobre la compatibilidad con JDK 14-5
 soluciones para ejecutar applets 14-8
nodo WebApp 3-6

P

página de inicio de la aplicación
 Web Start 15-8
página HTML
 convertir en Struts 8-14
 llamada a servlets desde 5-14
paquetes
 en applets 14-10
 servlet HTTP 4-4
parámetros
 de servlets 5-10

etiqueta de applets 14-3
patrón de URL 5-7, 10-10
plataformas
 convenciones 1-5
Plug-in de Java 14-8
propiedades
 de archivo WAR 3-18
propiedades de ejecución
 JSP 10-13
 servlets 10-13
propiedades de WebApp 3-9
 compresión del archivo WAR 3-10
 directorio 3-10
 directorios 3-11
 ficha WebApp 3-10
 fichas
 Clases 3-13
 Dependencias 3-16
 Descriptor 3-18
 generación del archivo WAR 3-10
 nombre del archivo WAR 3-10
 tipos de ficheros incluidos 3-10
 ubicación del archivo WAR 3-10
proyecto
 configurar Web Start 15-5
 seleccionar servidor web 9-4

R

recopilatorio Web 3-2
restricción de seguridad
 añadir a archivo web.xml 12-2

S

seguridad
 administrador de seguridad 14-11
 autenticar applets 14-12
 en las aplicaciones Web Start 15-2
 espacio de contención 14-11
 para una WebApp 12-18
 restrictiones de un applet 14-12
seguridad de las applets
 administrador de seguridad 14-2, 14-11
 autenticar 14-12
 espacio de contención 14-11
 restrictiones 14-12
 soluciones 14-12
servidores
 iPlanet de Sun
 configurar 9-3
 servidores web
 configurar 9-1
 finalizar 10-19
 iniciar 10-16

iPlanet de Sun 9-3
salida con formato 10-17
salida en bruto 10-18
seleccionar para proyecto 9-4
Tomcat 9-1
WebLogic 9-3
 configurar 9-3
WebSphere 9-3
 configurar 9-3
ServletContext 3-6
servlets 4-1, 5-1
 añadir a archivo web.xml 12-2
 archivo SHTML 5-7
 ciclo de vida 4-6
 compilar 10-14
 crear con asistentes 5-1
 de filtro 5-1, 12-5
 de hilo simple 5-1
 denominación 5-7, 10-10
 depuración web 10-19
 descripción general 2-2
 destruir 4-7
 distribuir 4-9, 11-2
 e InternetBeans Express 5-15, 7-3
 ejecutar web 10-15
 ejemplos de uso 4-9
 en comparación con la CGI (Interfaz de enlace común) 2-2, 4-1
 enlace a HTML 4-7
 enlazados a datos 5-15, 7-1
 específicos para HTTP 4-8
 estándar 5-1
 archivo SHTML 5-7
 HTML 4-7, 5-4
 HTTP 4-8
 incluir en archivo WAR 3-18
 inicializar 4-6
 iniciar 10-5
 internacionalización 5-14
 llamada 5-13, 5-14
 mapeo 5-7, 10-2, 10-10
 monitor 5-1, 12-7
 interfaces 5-10
 multihilo 4-7
 paso de las respuestas 4-6
 patrón de URL 5-7, 10-10
 propiedades de ejecución 10-13
 ServletContext 3-6
 solicitudes del cliente 4-6
 tipo de contenido 5-4
 tutorial 16-1, 17-1, 19-1
 WML 5-4
 XHTML 5-4
 XML 5-4

y JSP 4-2
y servidores web 4-1, 4-3
y URI 10-10
y URL 5-7, 10-10
y WebApp 5-1, 5-7, 10-10
servlets monitor 12-7
solicitudes del cliente al servlet 4-6
Struts 8-1
 Asistente para Action 8-10
 Asistente para ActionForm 8-7
 Asistente para aplicaciones web 8-6
 Asistente para conversión de Struts 8-14
 Asistente para JSP a partir de ActionForm 8-12
bibliotecas de etiquetas 8-1
bibliotecas de etiquetas y archivo web.xml 8-17
clase Action 8-1, 8-10
clase ActionForm 8-1, 8-7, 8-12
clase ActionServlet 8-1
compatibilidad del marco de trabajo 8-4
compatibilidad en JBuilder 8-3
controlador 8-1
descripción general 2-6
editar struts-config.xml 8-16
modelo 8-1
pasos para crear webapp preparada para
 Struts 8-19
URI de inicio 8-6
utilizar con JBuilder 8-19
utilizar con JSP 6-4
versión 1.0 8-3
versión beta 1.1 8-3
vista 8-1
y archivo web.xml 8-17
y asistente para JSP 8-7
y struts-config.xml 8-17
struts-config.xml 13-1
 elementos
 action-mappings 13-14
 data-source 13-3
 form-beans 13-6
 global-forwards 13-10
menú contextual de la ficha Correspondencias
 Action 13-18
menú contextual de la ficha Form Beans 13-10
menú contextual de la ficha Fuentes de
 datos 13-6
menú contextual de la ficha Reenvíos
 globales 13-14
y Struts 8-17
y web.xml 8-17

T

tablas
 generar con InternetBeans Express 7-7
tipos de archivo
 incluir en archivo WAR 3-18
tipos de servlet
 estándar 5-1
 filtro 5-1, 12-5
 monitor 5-1, 5-10
Tomcat
 configurar 9-1
tutoriales
 JSP 18-1
 JSP e InternetBeans 20-1
 servlet 16-1, 17-1
 servlet e InternetBeans 19-1
 Web Start 21-1

U

ubicación de archivos
 en una WebApp 3-6
URI y servlets 10-10
URL generado 10-17
URL y servlets 5-7, 10-10
Usenet, grupos de noticias 1-8

V

vista de código fuente web 10-18
vista web 10-17

W

Web Start 15-1
 applets 14-8
 archivo JAR 15-8
 archivo JNLP 15-8
 asistente 15-8
 configurar proyecto para 15-5
 instalar 15-3, 15-5
 modificar definición de bibliotecas 15-5
 página de inicio de la aplicación 15-8
 seguridad de las aplicaciones 15-2
 tutorial 21-1
 y JBuilder 15-5
WebApp
 asistente 3-3
 comprobar 11-3
 descriptores de distribución 3-6, 11-4
 descriptores de distribución, específicos del
 fabricante 11-5
 directorio raíz 3-6
 directorio WEB-INF 3-6

distribuir 11-1
editor del descriptor de distribución 12-1, 13-1
estructura 3-1
herramientas 3-2
importación a JBuilder 3-3
importar archivos 3-7
propiedades 3-9
ubicación de archivos 3-6

webapp preparada para struts
 crear en JBuilder 8-19
webapps
 marco de trabajo Struts 8-1
WebLogic
 archivo weblogic.xml 11-4
 descriptor de distribución 11-4