Java™

Interfaces gráficas y aplicaciones para Internet

Microsoft Windows, Linux y otros

4.ª EDICIÓN

JDK 8
•
EDI NetBeans
•
Interfaces gráficas
•
Menús y barras de herramientas
•
Controles y cajas de diálogo
•
Tablas y árboles
•
Enlace de datos
•
Acceso a BBDD (JDBC)
•
Persistencia

Java EE
•
Clientes web (HTML, applets)
•
Servlets y JDBC
•
Diseño de páginas web con JSP, JSTL y JDBC
•
Servicios web XML y RESFUL
•
JavaServer Faces
•
AJAX
•
Spring, Hibernate y JSF

Fco. Javier Ceballos

WWW

CD-ROM descargable (ISO) con ejemplos, apéndices y URL de descarga del software necesario para el estudio del libro disponible en **www.ra-ma.es**.

Ra-Ma®

Java™

Interfaces gráficas y aplicaciones para Internet

4.ª edición

Fco. Javier Ceballos Sierra
Profesor titular de la
Escuela Politécnica Superior
Universidad de Alcalá

http://www.fjceballos.es/

Ra-Ma®

Java: Interfaces gráficas y aplicaciones para Internet, 4.ª edición

Editado por:
RA-MA Editorial
C/ Jarama, 3A, Polígono Industrial Igarsa
28860 PARACUELLOS DEL JARAMA, Madrid
Teléfono: 91 658 42 80
Telefax: 91 662 81 39
Correo electrónico: *editorial@ra-ma.com*
Internet: *www.ra-ma.es* y *www.ra-ma.com*
ISBN: 978-84-9964-522-3
Depósito Legal: M-32202-2014
Autoedición: Fco. Javier Ceballos
Filmación e impresión: Copias Centro, S.L.
Impreso en España
Primera impresión: marzo 2015

Ayer es historia, mañana un misterio y
hoy es un regalo, por eso se llama presente.

Dedico esta obra
a María del Carmen, mi esposa,
y a mis hijos Francisco y Javier.

CONTENIDO

PRÓLOGO ..... **XXI**

CAPÍTULO 1. MI PRIMERA APLICACIÓN **1**

FORMULARIOS 3
BIBLIOTECA JFC 5
ESTRUCTURA DE UNA APLICACIÓN 6
Compilar y ejecutar la aplicación 10
DISEÑO DE LA INTERFAZ GRÁFICA 12
Crear un componente Swing 12
Componentes Swing más comunes 12
Contenedores 13
Administradores de diseño 14
Añadir los componentes al contenedor 15
Asignar un administrador de diseño 16
Añadir una etiqueta y editar sus propiedades 16
Añadir un botón de pulsación y editar sus propiedades 17
MANEJO DE EVENTOS 19
Asignar manejadores de eventos a un objeto 20
Adaptadores 22
Responder a los eventos 23
ESTABLECER LA APARIENCIA DE LAS VENTANAS 24
RESUMEN 25
EJERCICIOS PROPUESTOS 25

CAPÍTULO 2. JFC (SWING) 27

COMPONENTES SWING 28
ARQUITECTURA MODELO-VISTA-CONTROLADOR 31
MANEJADORES DE EVENTOS DE SWING 33
JERARQUÍA DE COMPONENTES DE UNA APLICACIÓN 36
CAJAS DE TEXTO, ETIQUETAS Y BOTONES 37
Desarrollo de la aplicación 37
Objetos 38
Eventos 38
Pasos a seguir durante el desarrollo 38
El formulario, los componentes y sus propiedades 39
Tecla de acceso 43
Botón por omisión 43
Responder a los eventos 43
Enfocar un objeto 46
Seleccionar el texto de una caja de texto 47
INTERCEPTAR LA TECLA PULSADA 48
VALIDAR UN CAMPO DE TEXTO 50
Expresiones regulares 53
Ejemplos de expresiones regulares 53
El motor de expresiones regulares 56
CREAR UN CAMPO DE TEXTO VALIDADO 57
RESUMEN 59
EJERCICIOS RESUELTOS 59
Diseño de una calculadora 60
Objetos 60
Eventos 60
Pasos a seguir durante el desarrollo 61
Diseño de la ventana y de los controles 61
Establecer una fuente 65
Color 65
Escribir el código 66
EJERCICIOS PROPUESTOS 75

CAPÍTULO 3. MENÚS Y BARRAS DE HERRAMIENTAS 77

MENÚS 77
DISEÑO DE UNA BARRA DE MENÚS 78
Manejadores de eventos 82
Aceleradores y nemónicos 84
BARRA DE HERRAMIENTAS 85

Utilizar imágenes en botones 85
Diseño de una barra de herramientas 86
BARRA DE ESTADO 89
Diseño de una barra de estado 89
DESARROLLO DE UN EDITOR DE TEXTOS 92
Caja de texto multilínea 93
Panel de desplazamiento 93
Diseño del editor 94
El portapapeles 97
Trabajar con texto seleccionado 99
Diseño de la barra de menús 99
Diseño de la barra de herramientas 102
Asociar un método con un elemento de un menú 103
Archivo - Salir 103
Edición - Cortar 104
Edición - Copiar 105
Edición - Pegar 106
Opciones - Fuente 107
Opciones - Tamaño 109
Habilitar o inhabilitar los elementos de un menú 110
Marcar el menú seleccionado 111
Grupo de botones 112
Deshacer y rehacer 113
Recordar las ediciones reversibles 113
Añadir a la interfaz las órdenes Deshacer y Rehacer 114
MENÚS EMERGENTES 116
ASOCIAR UN ICONO A LA APLICACIÓN 118
REDIMENSIONAR UN COMPONENTE 118
RESUMEN 120
EJERCICIOS PROPUESTOS 121

CAPÍTULO 4. CONTROLES Y CAJAS DE DIÁLOGO 125

CAJAS DE DIÁLOGO MODALES Y NO MODALES 126
CAJAS DE DIÁLOGO PREDEFINIDAS 126
Visualizar datos con showMessageDialog 126
Confirmar datos con showConfirmDialog 127
Requerir datos con showInputDialog 128
Diálogo modal personalizado 130
Utilización de diálogos predefinidos 131
CAJAS DE DIÁLOGO PERSONALIZADAS 133
CASILLAS DE VERIFICACIÓN 137

BOTONES DE OPCIÓN 140
LISTAS SIMPLES 146
Diseñar la lista 147
Iniciar la lista 149
Acceder a los elementos seleccionados 150
Modelos de una lista simple 151
LISTAS DESPLEGABLES 153
Diseñar la lista 154
Iniciar la lista 156
Acceder al elemento seleccionado 156
Modelo de una lista desplegable 156
COMPONENTES DE RANGO DEFINIDO 158
JScrollBar 159
JSlider 162
JProgressBar 164
CAJAS DE DIÁLOGO ESTÁNDAR 167
Cajas de diálogo Abrir y Guardar 168
Propiedades 170
Filtros 170
Caja de diálogo Color 172
TEMPORIZADORES 173
RESUMEN 177
EJERCICIOS RESUELTOS 177
EJERCICIOS PROPUESTOS 191

CAPÍTULO 5. TABLAS Y ÁRBOLES 193

TABLAS 193
Construir una tabla 195
Iniciar la tabla 197
Modelos de una tabla 197
Crear un nuevo modelo para la tabla 198
Tamaño de las celdas 199
Acceder al valor de la celda seleccionada 200
ÁRBOLES 202
Construir un árbol 202
Iniciar el árbol 205
Modelos de un árbol 205
Acceder al nodo seleccionado 207
Añadir y borrar nodos 210
Añadir nodo 212
Borrar nodo 213

Borrar todos los nodos 214
Personalizar el aspecto de un árbol 214
EJERCICIOS RESUELTOS 215
Iniciar la tabla 222
Iniciar la ventana de la aplicación 226
Manejo de la aplicación 228
EJERCICIOS PROPUESTOS 236

CAPÍTULO 6. ENLACE DE DATOS 237

ASPECTOS BÁSICOS 237
Enlace de datos manual 237
Notificar cuándo cambia una propiedad 242
Enlace de datos con la biblioteca Beans Binding 246
La clase Binding 247
Crear un enlace 250
Enlaces con otros controles 253
Aplicar conversiones 253
Aplicar validaciones 256
Controlar eventos 258
ENLACES COMPLEJOS 259
Enlace a colecciones de objetos 261
JList 261
JComboBox 269
JTable 270
RESUMEN 274

CAPÍTULO 7. ACCESO A UNA BASE DE DATOS 275

SQL 276
Crear una base de datos 276
Crear una tabla 276
Escribir datos en la tabla 278
Modificar datos de una tabla 278
Borrar registros de una tabla 279
Seleccionar datos de una tabla 279
Un ejemplo con una sola tabla 281
Descripción del escenario 281
Creación de la base de datos y de sus tablas 281
Utilizando MySQL 282
Insertar datos en la base de datos 283
Modificar datos en la base de datos 284

Borrar registros de una tabla 284
Obtener datos de la base de datos 284
Un ejemplo con varias tablas 285
ACCESO A UNA BASE DE DATOS CON JDBC 289
Controladores 291
Descripción del escenario 291
Creación de la base de datos 291
Creación de las tablas 292
Conectando con la base de datos 294
Cargar el controlador 294
Conectar con la fuente de datos 295
Recuperar datos de la base de datos 297
Metadatos 298
Obtener datos de un conjunto de resultados 298
Insertar, actualizar y borrar datos en la base de datos 300
Navegar por la base de datos 301
Integridad referencial 303
EJEMPLO DE ACCESO A DATOS 305
UTILIZANDO UNA INTERFAZ GRÁFICA 311
GESTIÓN DE EXCEPCIONES Y DE FICHEROS LOG 318
EJERCICIOS RESUELTOS 324
EJERCICIOS PROPUESTOS 337

CAPÍTULO 8. PERSISTENCIA 339

API DE PERSISTENCIA DE JAVA 340
MAPEO OBJETO-RELACIONAL CON JPA 340
ENTIDADES 342
APLICANDO JPA 344
Unidad de persistencia 346
Definir el modelo de entidades 348
Entidad Alumno 348
Entidad Asignatura 349
Entidad AlumnoAsignatura 350
Clase insertable AlumnoAsignaturaPK 352
Asociaciones entre entidades 352
Definir el administrador de entidades 355
Operaciones con las entidades 357
Lectura 357
Actualización 357
Persistencia 358
Eliminación 360

Operaciones en cascada 361
Ciclo de vida de una entidad 362
Eventos durante el ciclo de vida de una entidad 363
Lenguaje de Consulta de Persistencia en Java 367
Clases de entidad a partir de una BBDD existente 370
Cadena de conexión 370
Modelo de entidades 371
Unidad de persistencia 374
Acceder a los datos 375
Clases controladoras JPA de clases de entidad 385
Utilizando enlaces de datos 388

CAPÍTULO 9. Java EE 401

INTRODUCCIÓN 401
¿QUÉ ES Java EE? 403
ARQUITECTURA Java EE MULTICAPA 404
Componentes Java EE 406
Contenedores Java EE 407
Tipos 408
Otros servicios Java EE 409
PRÁCTICAS EN EL DESARROLLO 410
La capa cliente 410
La capa web 411
La capa EJB 412
CREAR UNA APLICACIÓN JEE CON NETBEANS 413
Base de datos 416
Clases de entidad y unidad de persistencia 417
Crear los Enterprise Java Beans 420
Añadir un servlet 421
Crear la página que mostrará la interfaz gráfica 424
Crear una etiqueta personalizada 426
Refrescar la vista 430
RESUMEN 432

CAPÍTULO 10. CLIENTES 433

¿QUÉ ES INTERNET? 433
Intranet 434
Terminología Internet 434
SERVICIOS EN INTERNET 437
PÁGINAS WEB 440

Qué es HTML 441
Etiquetas básicas HTML 441
Etiquetas de formato de texto 442
URL 444
Enlaces entre páginas 444
Gráficos 446
Marcos 447
Formularios 448
Entrada básica de datos 449
Caja de texto 449
Caja de clave de acceso 449
Casilla de verificación 450
Botón de opción 450
Parámetros ocultos 450
Enviar datos 451
Reiniciar los datos de un formulario 451
Imágenes 451
Orden de tabulación 451
Caja de texto multilínea 452
Listas desplegables 452
Tablas 454
HOJAS DE ESTILO 455
Clases 457
Etiquetas y 459
XML 460
XHTML 461
PÁGINAS WEB DINÁMICAS 461
APPLETS 464
Crear un applet 464
Restricciones de seguridad con los applets 466
Ejecutar un applet 467
La clase JApplet 468
public void init() 468
public void start() 468
public void paint(Graphics g) 468
public void stop() 469
public void destroy() 469
Un ejemplo simple 469
Ciclo de vida de un applet 472
Pasar parámetros a un applet 473
Mostrar una imagen 474
Reproducir un fichero de sonido 477
Mostrar información en la barra de estado 479

Crear una animación 479
Desplegar un applet en Apache Tomcat 484
Desplegar un applet en GlassFish 485
APLICACIÓN CLIENTE DE UNA APLICACIÓN JEE 488
Interfaz remota 489
Aplicación Java EE 489
Aplicación cliente 492
Java Web Start 495
EJERCICIOS RESUELTOS 495
EJERCICIOS PROPUESTOS 499

CAPÍTULO 11. SERVLETS 501

¿QUÉ ES UN SERVLET? 501
Características de un servlet 502
ESTRUCTURA DE UN SERVLET 502
Ciclo de vida de un servlet 504
Un servlet sencillo 506
Software necesario para ejecutar un servlet 507
EJECUTAR UN SERVLET EN EL SERVIDOR 508
INCLUIR PROCESOS ESCRITOS EN JAVA 510
INVOCAR AL SERVLET DESDE UNA PÁGINA HTML 514
PROCESAR FORMULARIOS 515
Tipos de peticiones 517
Petición HTTP GET 518
Petición HTTP POST 518
LEER LOS DATOS ENVIADOS POR EL CLIENTE 519
DESCRIPTOR DE DESPLIEGUE 522
Anotación WebServlet 524
INICIACIÓN DE UN SERVLET 524
SEGUIMIENTO DE UNA SESIÓN 525
Cookies 526
Identificar al cliente 529
Reescritura del URL 532
Parámetros ocultos en los formularios 532
Interfaz HttpSession 533
Obtener una sesión 533
Datos asociados con una sesión 533
Cancelar una sesión 536
SERVLETS Y JDBC 538
Creación de la base de datos 538
Creación de las tablas 538

Creación de la aplicación web ... 540
CONJUNTO DE CONEXIONES ... 544
EMPAQUETAR UNA APLICACIÓN WEB ... 552
INSTALAR UNA APLICACIÓN WEB EN EL SERVIDOR ... 553
APLICACIÓN JEE ... 556
EJERCICIOS RESUELTOS ... 568
EJERCICIOS PROPUESTOS ... 573

CAPÍTULO 12. JSP ... 575

¿CÓMO TRABAJA UNA PÁGINA JSP? ... 575
Ciclo de vida de una página JSP ... 578
Objetos implícitos ... 580
Ámbito de los atributos ... 581
Ámbito de aplicación ... 582
Ámbito de sesión ... 582
Ámbito de petición ... 582
Ámbito de página ... 582
Fijar un atributo con un ámbito específico ... 582
¿Cuándo utilizar uno u otro ámbito? ... 583
Ejemplo ... 583
LENGUAJE DE EXPRESIÓN EN JSP ... 584
Objetos implícitos ... 585
Comentarios ... 586
Directrices ... 587
Directriz page ... 587
Directriz include ... 588
Directriz taglib ... 588
Elementos de programación ... 589
Declaraciones ... 589
Expresiones ... 590
Fragmentos de código Java ... 590
Ejemplo ... 590
Activar el lenguaje de expresión (LE) ... 590
Ejemplo ... 591
Variables y expresiones ... 592
Operadores ... 594
Prioridad y orden de evaluación ... 595
Ejemplo ... 595
Palabras reservadas ... 596
Funciones ... 596
COMPONENTES SOFTWARE: JavaBeans ... 599

Normas de diseño ... 599
Crear y utilizar un componente JavaBean ... 601
Establecer y obtener el valor de las propiedades ... 602
Instalación en el servidor ... 603
BIBLIOTECA ESTÁNDAR DE ETIQUETAS ... 604
Operaciones con etiquetas básicas ... 604
Operaciones con etiquetas SQL ... 607
Conectar con la base de datos ... 607
Realizar una consulta a la base de datos ... 608
Realizar una modificación sobre la base de datos ... 608
Ejemplo ... 609
API de Java ... 610
ETIQUETAS PERSONALIZADAS ... 611
Tipos de etiquetas ... 611
Etiqueta definida mediante una clase ... 612
Etiqueta con atributos ... 614
Etiqueta definida mediante un fichero ... 618
Ejemplo ... 618
Directrices ... 619
Atributos de la directriz attribute ... 619
Atributos de la directriz variable ... 621
Fragmentos ... 623
ETIQUETAS PERSONALIZADAS VS. JavaBeans ... 624
MANIPULACIÓN DE EXCEPCIONES ... 625
TRANSFERIR EL CONTROL A OTRO COMPONENTE WEB ... 626
FORMULARIOS ... 627
Parámetros de las casillas de verificación ... 628
Solicitar datos mediante listas (menús) ... 629
APLICACIONES WEB UTILIZANDO JSP ... 632
Modelo 1 ... 632
Modelo 2 ... 633
JSP Y JDBC ... 634
EJERCICIOS RESUELTOS ... 634
EJERCICIOS PROPUESTOS ... 645

CAPÍTULO 13. SERVICIOS WEB ... 653

SERVICIOS WEB XML ... 654
Crear un nuevo servicio web XML ... 654
Cómo se construye un servicio web ... 656
Escribir la interfaz del servicio web ... 657
Crear un cliente del servicio web ... 661

Aplicación Java como cliente de un servicio web ... 662
Descubrimiento de servicios web XML ... 664
Obtener acceso al servicio web XML ... 668
Aplicación web como cliente de un servicio web ... 670
Invocar al servicio web desde una página JSP ... 670
Invocar al servicio web desde un servlet ... 673
SERVICIOS WEB XML SIN HERRAMIENTAS RAD ... 676
Crear un servicio web XML ... 677
Compilar el servicio web ... 679
Generar los artefactos del servicio web en el lado del servidor ... 679
Empaquetar el servicio web ... 680
Desplegar el servicio web en el servidor GlassFish ... 680
Probar el servicio ... 680
Generar los artefactos del servicio web en el lado del cliente ... 680
Escribir la clase correspondiente al cliente ... 681
Compilar el cliente del servicio web ... 682
Ejecutar el cliente ... 683
SERVICIOS WEB RESTFUL ... 683
Crear un nuevo servicio web RESTful ... 684
Crear el servicio web RESTful y configurar REST ... 688
Probar el recurso web ... 690
Petición con parámetros ... 691
Cliente Java del servicio web RESTful ... 693
EJERCICIOS RESUELTOS ... 699
EJERCICIOS PROPUESTOS ... 716

CAPÍTULO 14. JSF ... 717

DESARROLLO DE UNA APLICACIÓN JSF ... 718
FacesServlet ... 720
Crear las páginas JSF ... 720
Añadir un componente ... 722
Añadir un fichero de propiedades ... 723
Añadir una nueva página ... 725
Definir la navegación entre páginas ... 726
Desarrollar los beans de apoyo ... 729
Convertidores ... 732
Configuración para los beans de apoyo ... 733
Desplegar la aplicación ... 734
Validación de los datos ... 734
Caja de texto vacía ... 734
Visualizando mensajes ... 735

Valor fuera de rango ... 736
Mensaje personalizado ... 736
Iniciación de las propiedades de un bean ... 737
Facelets ... 738
Crear el proyecto ... 739
Desarrollar los beans de apoyo ... 740
Añadir un fichero de propiedades ... 742
Crear las páginas JSF ... 742
Usar una plantilla Facelets ... 745
Escuchadores de eventos ... 750
CICLO DE VIDA DE UNA PÁGINA JSF ... 752
CONECTANDO CON BASES DE DATOS ... 754
Clases de entidad y unidad de persistencia ... 756
Crear los Enterprise Java Beans ... 757
Crear los beans de apoyo ... 760
Crear la interfaz gráfica ... 764
EJERCICIOS RESUELTOS ... 770
EJERCICIOS PROPUESTOS ... 784

CAPÍTULO 15. AJAX ... 785

FUNDAMENTOS DE AJAX ... 787
XMLHttpRequest ... 788
GENERACIÓN DE CÓDIGO JAVASCRIPT ... 796
Fichero JavaScript ... 797
AÑADIR AJAX A UNA APLICACIÓN WEB ... 797
JAVASERVER FACES MÁS AJAX ... 799
Validación usando AJAX ... 803
Proyecto JSF con tecnología AJAX ... 807
Crear la base de datos ... 807
Construir el proyecto ... 807
Crear la página JSF con AJAX ... 811
ICEFACES ... 814
Actualizaciones síncronas y asíncronas ... 816
Procesamiento de formularios ... 818
Componentes ... 819
Utilizando ICEfaces ... 820
Construir el proyecto ... 820
Crear la página web ... 821
EJERCICIOS PROPUESTOS ... 824

CAPÍTULO 16. SPRING 827

ACOPLAMIENTO ENTRE COMPONENTES 830
ELIMINAR EL ACOPLAMIENTO ENTRE COMPONENTES 834
INTRODUCCIÓN A SPRING 836
El contenedor de IoC de Spring 840
Metadatos de configuración 841
Crear el contenedor de IoC de Spring 843
Anotaciones 845
SPRING, ACCESO A DATOS Y JSF 850
Crear el proyecto 852
Capa de acceso a datos 854
Objetos de negocio 860
Capa de lógica de negocio 861
Capa de presentación 862
Integrar Spring con JSF 870
EJERCICIOS PROPUESTOS 873

APÉNDICES 875

ÍNDICE 957

PRÓLOGO

Java es un lenguaje de programación introducido por Sun Microsystems cuyas características lo sitúan, junto con Microsoft C# (C Sharp), en uno de los productos ideales para desarrollar programas para la Web.

Cuando Java se introdujo de forma importante, allá por 1995, fue cuando su uso en el diseño de páginas web revolucionó la naturaleza de estas. ¿Recuerda? Todo el mundo hablaba de *applets*, esos pequeños programas que se ejecutan en el contexto de una página web en cualquier ordenador, introduciendo animación y efectos especiales. Y quizás, esta idea esté enmascarando que Java no solo es eso. Java está también disponible para desarrollar aplicaciones de uso general; esto es, Java le permite crear programas para su uso personal, para su grupo de trabajo, para una empresa, aplicaciones distribuidas a través de Internet, aplicaciones de bases de datos, aplicaciones para móviles y otras muchas que usted pueda imaginar.

Actualmente, en el mercado, hay multitud de herramientas de programación Java, siendo una de las más conocidas *NetBeans*, actualmente soportada por Oracle. No obstante, la mejor forma de ver el alcance de Java es desarrollando directamente a través del kit de desarrollo de Java (JDK). Se trata de un paquete que se puede obtener de la Red basado en un conjunto de herramientas de órdenes en línea para editar, compilar, ejecutar y depurar programas Java.

Este libro, en su última versión, fue actualizado con *JDK 8/Java EE 7* y *NetBeans 8* (véanse los apéndices A y B). Está dedicado al diseño de interfaces gráficas, al desarrollo de aplicaciones con acceso a bases de datos, al diseño de páginas web y al desarrollo de aplicaciones para Internet con JSF, persistencia de los datos, servicios web, AJAX y otras técnicas.

Todos los capítulos expuestos se han documentado con abundantes problemas resueltos, de forma que cuando complete su estudio sabrá cómo escribir aplicaciones que presentan una interfaz gráfica, así como aplicaciones para Internet.

Considero importante que, antes de continuar, eche una ojeada a los apéndices. En ellos se expone cómo utilizar el entorno de desarrollo (EDI) *NetBeans*. Instalando este EDI tendrá todo lo necesario para desarrollar las aplicaciones mencionadas, incluso para trabajar con bases de datos Java DB.

Para quién es este libro

Este libro está pensado para aquellas personas que quieran aprender a desarrollar aplicaciones que muestren una interfaz gráfica al usuario, aplicaciones para acceso a bases de datos y para Internet (páginas web). Para ello, ¿qué debe hacer el lector? Pues simplemente leer ordenadamente los capítulos del libro, resolviendo cada uno de los ejemplos que en ellos se detallan.

Evidentemente, no vamos a enseñar a programar aquí, por eso es necesario tener algún tipo de experiencia con un lenguaje de programación orientado a objetos (Java, C#, Visual Basic, etc., son lenguajes orientados a objetos). Haber programado en Java sería lo ideal, así como tener conocimientos de HTML y XML. Estos requisitos son materia de mis otros libros *Java: Lenguaje y aplicaciones* y *Java: Curso de programación*, ambos editados también por las editoriales RA-MA y Alfaomega Grupo Editor.

Java: Lenguaje y aplicaciones se centra en la programación básica: tipos, sentencias, matrices, métodos, ficheros, etc., y hace una introducción a las interfaces gráficas, a las bases de datos y a las aplicaciones para Internet, y *Java: Curso de programación* cubre la programación básica (expuesta en menor medida en el libro anterior) y la programación orientada a objetos (POO) en detalle: clases, clases derivadas, interfaces, paquetes, excepciones, etc.; después, utilizando la POO, añade otros temas como estructuras dinámicas de datos, algoritmos de uso común, hilos (programación concurrente), etc. Este sí que es un libro de programación con Java en toda su extensión. Puede ver más detalles de cada uno de ellos en mi web: *www.fjceballos.es*.

Cómo está organizado el libro

El libro se ha estructurado en 16 capítulos más algunos apéndices que a continuación se relacionan. El capítulo 1 nos introduce en el desarrollo de aplicaciones de escritorio. Los capítulos 2 al 5 nos enseñan a desarrollar aplicaciones de escritorio que muestran una interfaz de ventanas al usuario. Los capítulos 6 al 8 cubren el enlace a datos, el acceso a bases de datos (JDBC) y la persistencia de los datos. Y

los capítulos 9 al 16 nos enseñan cómo desarrollar aplicaciones para Internet a base de formularios web, *servlets*, servicios web, JSP, JSF y AJAX.

CAPÍTULO 1. MI PRIMERA APLICACIÓN
CAPÍTULO 2. JFC (SWING)
CAPÍTULO 3. MENÚS Y BARRAS DE HERRAMIENTAS
CAPÍTULO 4. CONTROLES Y CAJAS DE DIÁLOGO
CAPÍTULO 5. TABLAS Y ÁRBOLES
CAPÍTULO 6. ENLACE DE DATOS
CAPÍTULO 7. ACCESO A UNA BASE DE DATOS
CAPÍTULO 8. PERSISTENCIA
CAPÍTULO 9. Java EE
CAPÍTULO 10. CLIENTES
CAPÍTULO 11. SERVLETS
CAPÍTULO 12. JSP
CAPÍTULO 13. SERVICIOS WEB
CAPÍTULO 14. JSF
CAPÍTULO 15. AJAX
CAPÍTULO 16. SPRING
APÉNDICES

Qué se necesita para utilizar este libro

Este libro ha sido escrito utilizando el paquete *JDK 8/Java EE 7* y *NetBeans 8*, que incluye todo lo necesario para escribir, construir, verificar y ejecutar aplicaciones Java. Por lo tanto, basta con que instale en su máquina ese software para poder reproducir todos los ejemplos durante el estudio.

Sobre los ejemplos del libro

La imagen del CD de este libro, con las aplicaciones desarrolladas y el software para reproducirlas, puede descargarla desde *http://www.fjceballos.es* (sección *Mis publicaciones > Java > CD*) o desde *http://www.ra-ma.com* (en la página correspondiente al libro). La descarga consiste en un fichero ZIP con una contraseña que encontrará en el libro.

Agradecimientos

He recibido ayuda de algunas personas durante la preparación de este libro y, por ello, estoy francamente agradecido; pero en especial quiero expresar mi agradecimiento a mi colega **Óscar García Población** y a mi amigo **Roberto Canales Mora**, empresario dedicado al desarrollo de soluciones informáticas (puede saber más de él en *http://www.autentia.com*), por sus buenas recomendaciones y aportaciones en la corrección que hicieron de la segunda edición de este libro. También

deseo expresar mi agradecimiento a Oracle por poner a mi disposición en particular y de todos los lectores, en general, los productos que la creación y el estudio de esta obra requieren.

Francisco Javier Ceballos Sierra
http://www.fjceballos.es/

APÉNDICE A

ENTORNO DE DESARROLLO INTEGRADO PARA JAVA

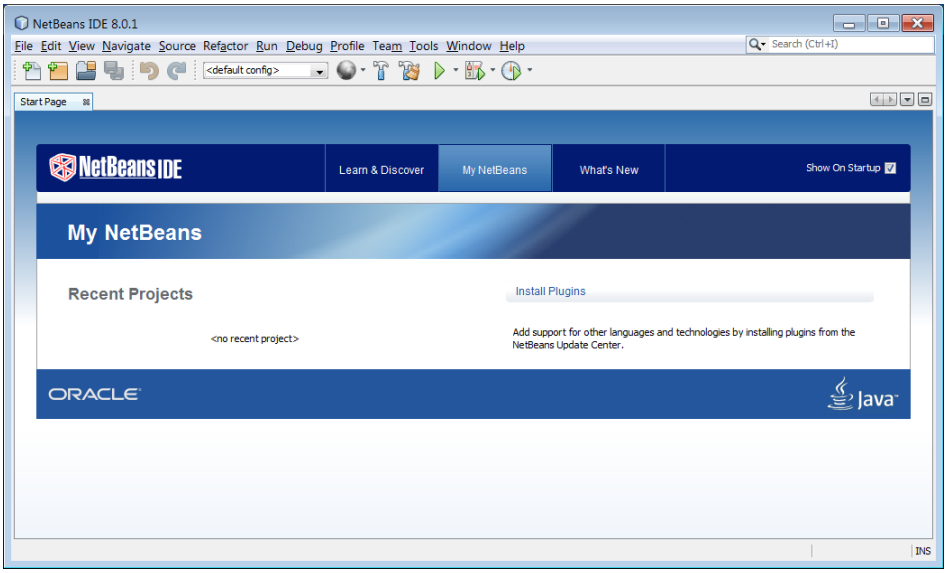
Evidentemente, para poder escribir programas se necesita un entorno de desarrollo Java. Oracle, propietario de Java, proporciona uno de forma gratuita, ***Java SE Development Kit 8*** (*JDK 8*) para Microsoft Windows, en todas sus versiones, y para Linux. En el apéndice B se explica cómo obtenerlo e instalarlo.

Opcionalmente, puede instalar un entorno de desarrollo integrado (EDI) que le facilite las tareas de creación de la interfaz gráfica de usuario, edición del código, compilación, ejecución y depuración, como, por ejemplo, *NetBeans* de Oracle. Para instalarlo véase el apéndice B.

Asegúrese de que las variables de entorno *PATH* y *CLASSPATH* están perfectamente establecidas (en el caso de instalar *NetBeans*, esta operación se realizará automáticamente).

DISEÑO DE UNA APLICACIÓN DE CONSOLA

Cuando se utiliza un entorno de desarrollo integrado (EDI), lo primero que hay que hacer una vez instalado es asegurarse de que las rutas donde se localizan las herramientas, las bibliotecas, la documentación y los ficheros fuente hayan sido establecidas; algunos EDI solo requieren la ruta donde se instaló el compilador. Este proceso normalmente se ejecuta automáticamente durante el proceso de instalación de dicho entorno. Si no es así, el entorno proporcionará algún menú con las órdenes apropiadas para realizar dicho proceso. Por ejemplo, en el entorno de desarrollo integrado *NetBeans* que se presenta a continuación, esas rutas a las que nos referimos quedan establecidas durante la instalación del mismo.



Para personalizar el EDI, ejecute la orden *Options* del menú *Tools*.

Para editar y ejecutar la aplicación realizada en el capítulo 1, "¡¡¡Hola, mundo!!!", utilizando este EDI, los pasos a seguir se indican a continuación:

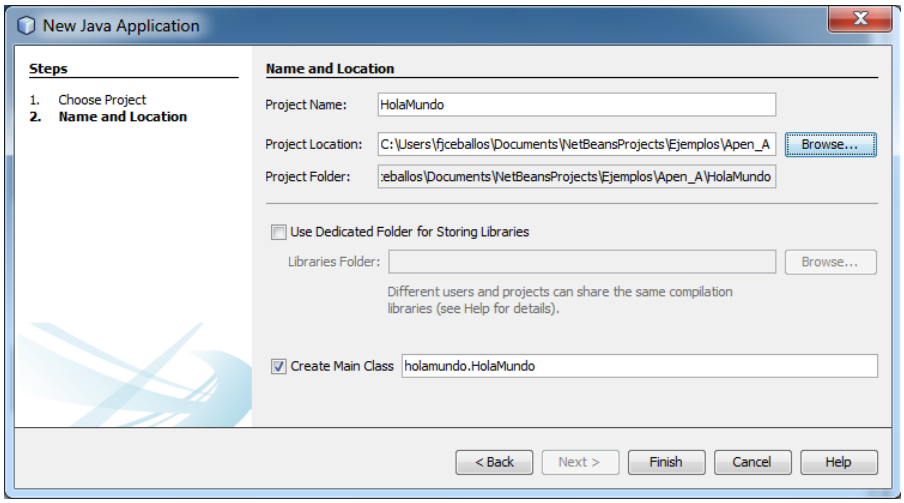
1. Suponiendo que ya se está visualizando el entorno de desarrollo, ejecute la orden *File* > *New Project* (Archivo > Nuevo Proyecto). Se muestra la ventana *New Project*.

2. Seleccione *Java* en la lista *Categories* (Categorías), y en la lista *Projects* (Proyectos) seleccione *Java Application* (Aplicación Java). Después haga clic en el botón *Next* (siguiente). Se muestra la ventana *New Java Application*.

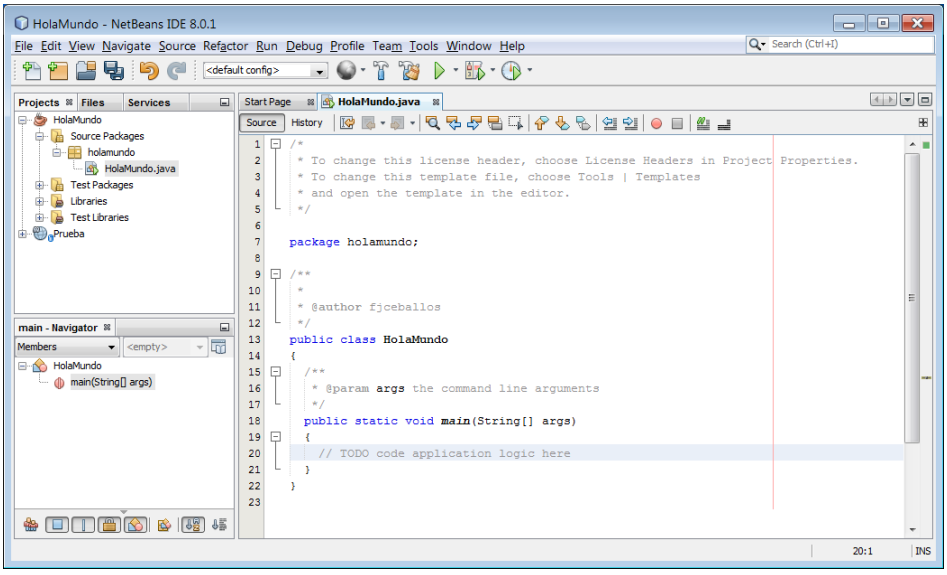
3. Escriba el nombre del proyecto (*Project Name*); en nuestro caso será *HolaMundo* y, a continuación, seleccione la carpeta donde quiere guardarlo.

4. Asegúrese de que la casilla *Create Main Class* (crear clase principal) está marcada.

5. Observe la caja de texto correspondiente al nombre de la clase principal; muestra *holamundo.HolaMundo*. Esto significa que la clase principal se llama *HolaMundo* y que pertenece al paquete *holamundo*. Asumiremos el paquete por omisión.



6. Para finalizar haga clic en el botón *Finish*. El resultado será el siguiente:



El EDI crea la carpeta *Ejemplos\Apen_A\HolaMundo* en la que guardará el proyecto compuesto en este caso por un solo fichero, *HolaMundo.java*, que almacena el código correspondiente a la clase *HolaMundo*.

En la ventana mostrada en la figura anterior distinguimos otras tres ventanas, algunas de ellas, con varios paneles. La que está en la parte superior derecha está mostrando el panel de edición para el código fuente de nuestra aplicación y tiene oculto el panel de inicio. La que está en la parte superior izquierda muestra el pa-

nel de proyectos; este lista el nombre del proyecto y el nombre de los ficheros que componen el proyecto. Observe el fichero *HolaMundo.java*; contiene el código de las acciones que tiene que llevar a cabo nuestra aplicación. También distinguimos un elemento *Libraries* que hace referencia a las bibliotecas que pueden ser necesarias para compilar la aplicación. Finalmente, la ventana que hay debajo de la de proyectos permite navegar por el código del proyecto. Puede visualizar otras ventanas desde el menú *Window*; por ejemplo, la ventana *Output*, que será utilizada para mostrar los resultados de la compilación y de la ejecución.

Una vez creado el esqueleto de la aplicación, editamos el código de la misma. En nuestro caso, simplemente hay que completar el método *main* como se indica a continuación:

```
public static void main(String[] args)
{
  System.out.println("Hola mundo!!!");
}
```

El paso siguiente es construir el fichero ejecutable (fichero *HolaMundo.class*). Para ello ejecute la orden *Run > Build Project*, o bien pulse la tecla *F11*. Si la compilación es correcta, puede pasar a ejecutar la aplicación ejecutando la orden *Run > Run Project*, o bien pulsando la tecla *F6*; observe el resultado en la ventana *Output*.

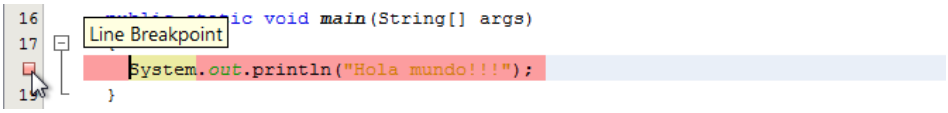
Cuando la aplicación necesite más de un fichero, el proceso es igual de sencillo. Añadir otro fichero a una aplicación, por ejemplo un nuevo fichero que almacene una nueva clase, supone hacer clic con el botón derecho del ratón sobre el nombre del proyecto, elegir la orden *New* y seleccionar del menú contextual que se visualiza el tipo de elemento que se desea añadir.

DEPURAR UNA APLICACIÓN CON NETBEANS

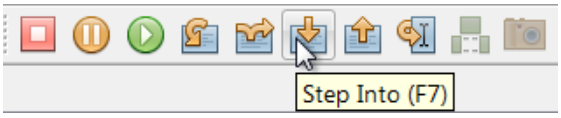
¿Por qué se depura una aplicación? Porque los resultados que estamos obteniendo con la misma no son correctos y no sabemos por qué. El proceso de depuración consiste en ejecutar la aplicación paso a paso, indistintamente por sentencias o por métodos, con el fin de observar el flujo seguido durante su ejecución, así como los resultados intermedios que se van sucediendo, con la finalidad de detectar las anomalías que producen un resultado final erróneo.

Por ejemplo, para depurar una aplicación utilizando el depurador del entorno de desarrollo *NetBeans*, debe establecer un punto de parada inicial. Para ello haga clic con el botón derecho del ratón sobre la sentencia a partir de la cual quiere ejecutar el código de su aplicación paso a paso y ejecute la orden *Toggle Line*

Breakpoint (poner un punto de parada) del menú contextual que se visualiza, o haga clic en la zona sombreada a su izquierda:



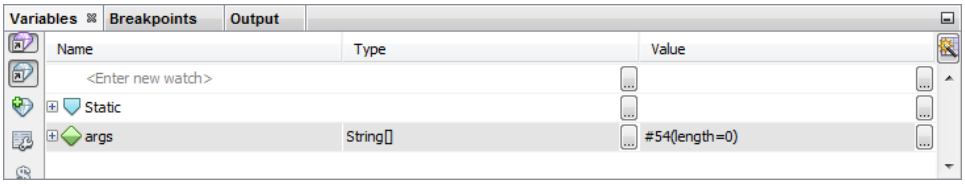
Después ejecute la orden *Debug* > *Debug Project*, o bien pulse la tecla *Ctrl+F5* para iniciar la depuración. Continúe la ejecución paso a paso utilizando las órdenes del menú *Debug* o los botones correspondientes de la barra de herramientas *Debug* (para saber el significado de cada botón, ponga el puntero del ratón sobre cada uno de ellos).



De forma resumida, las órdenes disponibles para depurar una aplicación son las siguientes:

- *Debug Project* o *Ctrl+F5*. Inicia la ejecución de la aplicación en modo depuración hasta encontrar un punto de parada o hasta el final si no hay puntos de parada.
- *Toggle Line Breakpoint* o *Ctrl+F8*. Pone o quita un punto de parada en la línea sobre la que está el punto de inserción.
- *Finish Debugger Session* o *Mayús+F5*. Detiene el proceso de depuración.
- *Step Into* o *F7*. Ejecuta la aplicación paso a paso. Si la línea a ejecutar coincide con una llamada a un método definido por el usuario, dicho método también se ejecuta paso a paso.
- *Step Over* o *F8*. Ejecuta la aplicación paso a paso. Si la línea a ejecutar coincide con una llamada a un método definido por el usuario, dicho método no se ejecuta paso a paso, sino de una sola vez.
- *Step Out* o *Ctrl+F7*. Cuando un método definido por el usuario ha sido invocado para ejecutarse paso a paso, utilizando esta orden se puede finalizar su ejecución en un solo paso.
- *Run to Cursor* o *F4*. Ejecuta el código que hay entre la última línea ejecutada y la línea donde se encuentra el punto de inserción.

Para ver los valores intermedios que van tomando las variables ponga el cursor sobre ellas, o bien utilice las ventanas *Variable*, etc., del fondo del EDI. Para añadir o quitar ventanas, ejecute la orden *Window* > *Debuggin*.

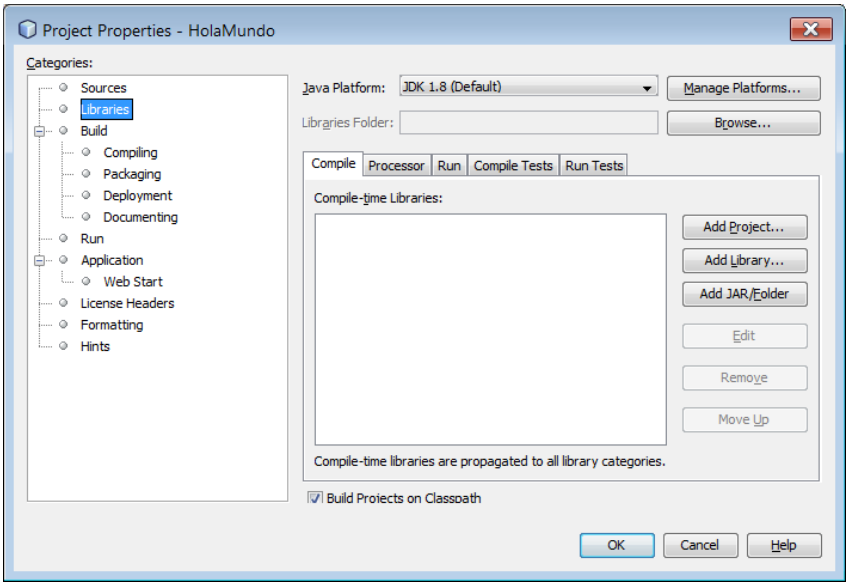


VARIABLE CLASSPATH

La opción **–classpath** del compilador debe incluir las rutas de todas las carpetas donde se deben buscar las clases necesarias para compilar una aplicación. Algunas de estas clases podrían, incluso, encontrarse empaquetadas en un fichero *.jar*.

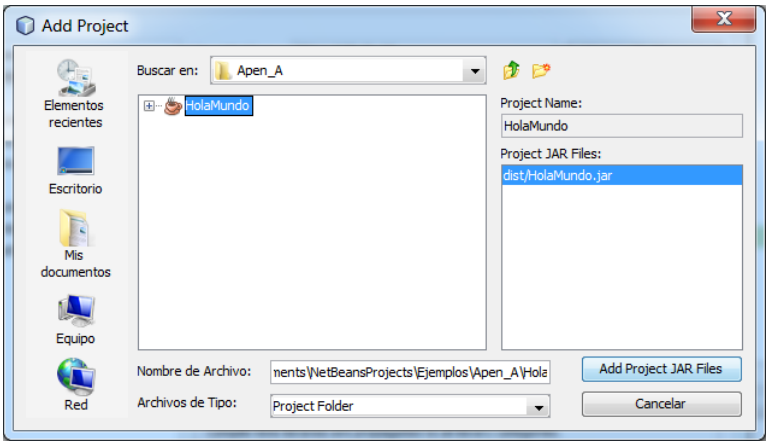
Cuando necesite especificar estas rutas:

1. Diríjase al panel del proyecto, haga clic con el botón derecho del ratón sobre el nombre del mismo y seleccione la orden *Properties* del menú contextual que se visualiza. Se muestra el diálogo siguiente:



2. Seleccione el nodo *Libraries*. Haga clic en la pestaña *Compile* y después en el botón *Add Project*.

3. Seleccione la carpeta correspondiente al proyecto y, en el diálogo que se visualiza, observe la lista *Project JAR Files* (ficheros JAR del proyecto); muestra los ficheros JAR que pueden ser añadidos al proyecto. Observe que se muestra también el fichero JAR correspondiente a nuestra aplicación. Este fichero se crea una vez que hayamos compilado y ejecutado el proyecto.



4. Una vez seleccionado el fichero que desea añadir, haga clic en el botón *Add Project JAR Files* (añadir ficheros JAR al proyecto) y, finalmente, cierre el diálogo. El fichero JAR seleccionado será añadido.

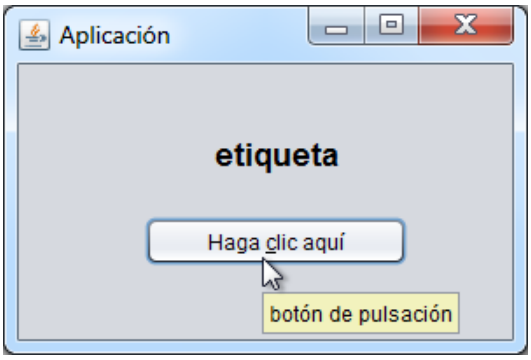
Desde la línea de órdenes, esta variable se especificaría análogamente a como indica el ejemplo siguiente:

```
set classpath=%classpath%;.;c:\java\ejemplos;c:\lib\milib.jar
```

DISEÑO DE UNA APLICACIÓN CON INTERFAZ GRÁFICA

Para implementar y ejecutar una aplicación que muestre una interfaz gráfica como la de la figura mostrada a continuación, utilizando el entorno de desarrollo integrado *NetBeans*, los pasos a seguir son los siguientes:

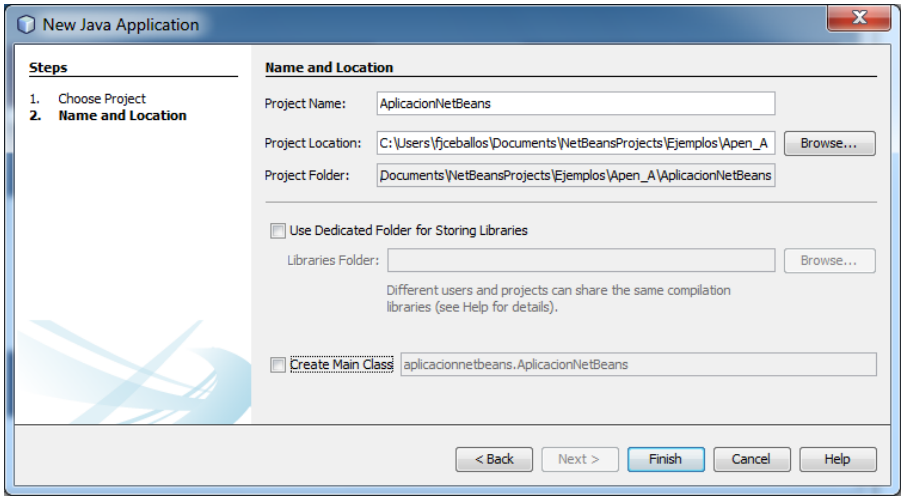
1. Suponiendo que ya está visualizado el entorno de desarrollo, ejecute la orden *File* > *New Project* (Archivo > Nuevo Proyecto). Se muestra la ventana *New Project*.



2. Seleccione *Java* en la lista *Categories*, y en la lista *Projects* seleccione *Java Application* (Aplicación Java). Después haga clic en el botón *Next* (siguiente). Se muestra la ventana *New Java Application*.

3. Escriba el nombre del proyecto (*Project Name*); en nuestro caso será *AplicacionNetBeans*; y, a continuación, seleccione la carpeta donde quiere guardarlo.

4. No marque la opción *Create Main Class* (crear clase principal). De esta forma se creará un proyecto vacío. Si marca esta opción, el nombre de la clase será el último especificado, y los anteriores, separados por puntos, darán lugar al nombre del paquete al que pertenecerá la clase.

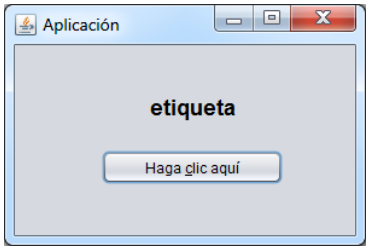


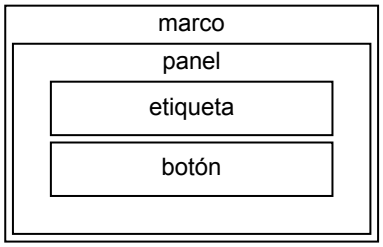
5. Para finalizar haga clic en el botón *Finish*. El EDI crea la carpeta *Ejemplos\Apen_A\AplicacionNetBeans* en la que guardará el proyecto.

Una vez creado un proyecto vacío, el paso siguiente consistirá en añadir una nueva clase derivada de **JFrame** que nos sirva como contenedor de la interfaz gráfica.

Contenedores

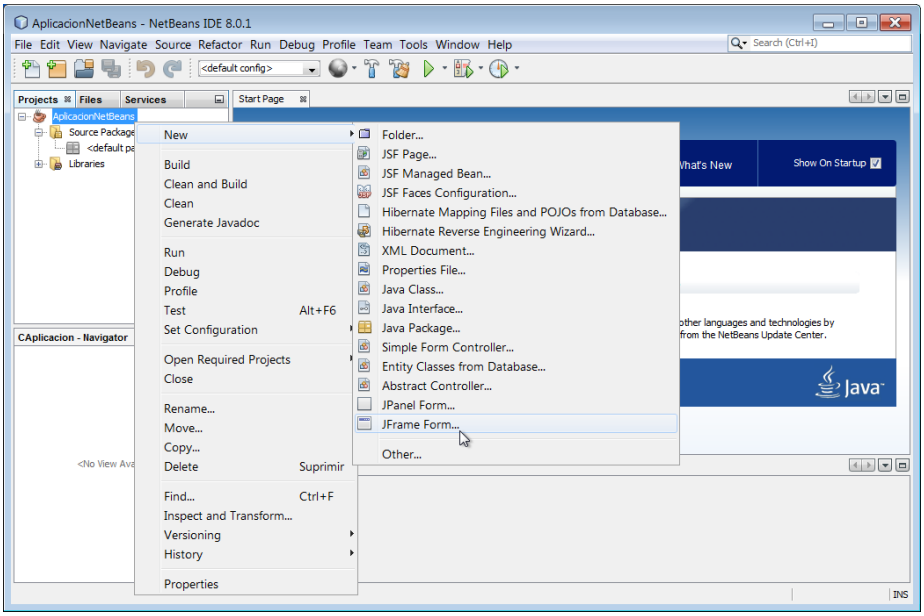
Los contenedores son componentes *Swing* utilizados para ubicar otros componentes. Por ejemplo, la figura siguiente, correspondiente a la interfaz gráfica de la aplicación que queremos desarrollar, explica cómo se acomodan los componentes en uno de estos contenedores.



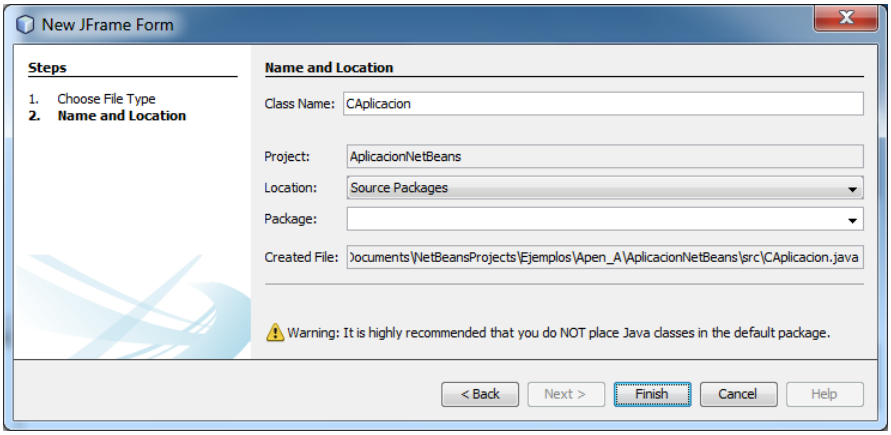


Para agregar componentes a una ventana es aconsejable utilizar un *contenedor* intermedio; esto facilitará la realización de otras operaciones posteriores, como, por ejemplo, añadir un borde alrededor de los componentes. Siempre que sea necesario, un contenedor puede contener a otros contenedores, lo que dará lugar a una jerarquía de contenedores que facilitará la distribución de los componentes. La raíz de esa jerarquía es el *contenedor del nivel superior* definido por la ventana. Las clases **JFrame**, **JDialog** y **JApplet** son contenedores del nivel superior.

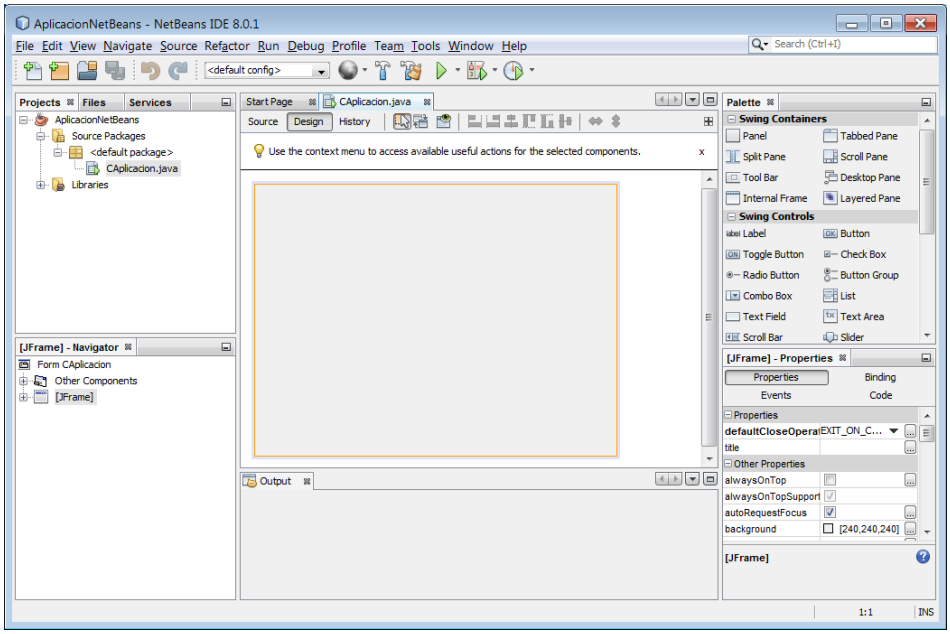
Para añadir una ventana marco (objeto **JFrame**) al proyecto, haga clic sobre el nombre del mismo, utilizando el botón derecho del ratón, y seleccione del menú contextual que se visualiza la orden *New > JFrame Form...*.



Se visualizará una ventana en la que puede elegir el nombre para la nueva clase de objetos que vamos a añadir. En nuestro caso, elegiremos como nombre, por ejemplo, *CAplicacion*.



Para continuar, haga clic en el botón *Finish*. Se visualizará la ventana mostrada a continuación, en la que observamos que la clase *CAplicacion* almacenada en el fichero *CAplicacion.java* es la que da lugar a la ventana marco, también llamada formulario, que se visualiza en el centro del EDI (objeto **JFrame**).



Observe también que encima del formulario hay una barra de herramientas con una serie de botones. Los dos primeros (*Source* y *Design*) le permitirán alternar entre el panel de edición (el que muestra el código fuente de la clase *CAplicacion*) y el panel de diseño (el que se está mostrando).

También, a la derecha del formulario, se observan otras dos ventanas: una muestra varias paletas de herramientas (la que se está mostrando es la paleta de componentes *Swing*; si no se muestra ejecute *Window* > *IDE Tools* > *Palette*) y la otra está mostrando el panel de propiedades con las propiedades del formulario. La primera le muestra los controles o componentes que puede seleccionar y colocar sobre el formulario; observe que, además de la paleta de componentes *Swing*, hay otras como *AWT* (kit de herramientas de ventanas abstractas) y *Beans* (componentes reutilizables); y la segunda le permite mostrar y editar las propiedades del componente seleccionado (tamaño, color, fuente...), los manejadores de eventos (botón *Events*), etc.

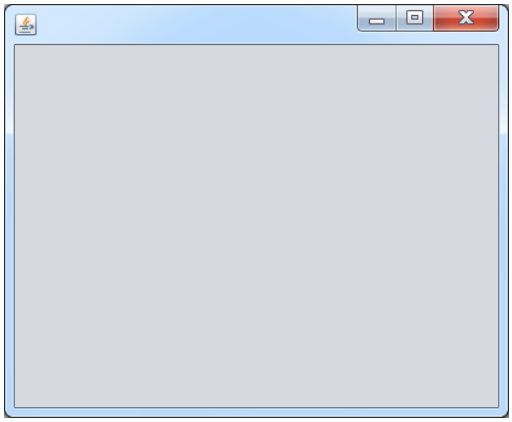
A la izquierda del formulario, debajo del panel del proyecto, hay otra ventana con el navegador (*Navigator*). El navegador permite navegar por los componentes software de la aplicación (controles, clases, métodos, etc.).

En el panel de edición de código (clic en el botón *Source*) se puede observar que se ha generado una clase *CAplicacion*, con un constructor público y una serie de métodos. En los pasos siguientes añadiremos los componentes al formulario,

utilizando el panel de diseño, y el código necesario para que la aplicación realice lo deseado.

Ejecutar la aplicación

Si ahora compilamos y ejecutamos esta aplicación, para lo cual tendremos que elegir la orden *Run Project* (*F6*) del menú *Run*, o bien hacer clic en el botón correspondiente de la barra de herramientas del EDI, aparecerá sobre la pantalla la ventana de la figura mostrada a continuación y podremos actuar sobre cualquiera de sus controles (minimizar, maximizar, mover, ajustar el tamaño, etc.).



Esta es la parte que *NetBeans* realiza por nosotros y para nosotros; pruébelo. Para finalizar, haga clic en el botón  para cerrar la ventana.

Observe en el panel de propiedades que el valor de la propiedad **defaultCloseOperation** es *EXIT_ON_CLOSE*. Esto indica que al hacer clic en el botón  la ventana se cerrará y la aplicación finalizará invocando al método **exit**.

Editar el código fuente

Supongamos ahora que deseamos añadir un título a la ventana y fijar su tamaño inicial. Una forma de hacer esto es proporcionar tanto el título como el tamaño en el momento de crear el objeto **JFrame**. Esto quiere decir que los métodos del objeto que permitan manipular estas propiedades deben ser invocados desde el constructor *CAplicacion*. El método que permite establecer el título es **setTitle** y el que permite establecer el tamaño es **setSize**.

Una forma rápida de situarse en el editor sobre un determinado método para modificarlo es localizarlo en el *navegador* y hacer doble clic sobre él. Por lo tan-

to, muestre el panel de edición (clic en el botón *Source*), diríjase al *navegador* y haga doble clic sobre *CAplicacion*; esta acción colocará el punto de inserción sobre el constructor *CAplicacion*. Modifique este constructor según se muestra a continuación:

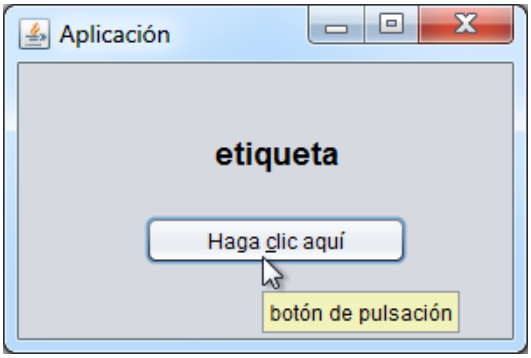
```
public CAplicacion()
{
  initComponents();
  setTitle("Aplicación"); // establecer el título de la ventana
  setSize(300, 200); // fijar el ancho y el alto de la ventana
}
```

También podría establecer el título de la ventana a través del panel de propiedades del formulario; esto es, estando en el panel de diseño, seleccionamos el formulario, nos dirigimos al panel de propiedades y asignamos a la propiedad **title** el valor "Aplicación".

Observe que el constructor invoca a *initComponents* para ejecutar las operaciones de iniciación requeridas para los componentes de la aplicación. Después hemos añadido dos líneas: una para establecer el título de la ventana y otra para fijar su tamaño. Observe también que *initComponents* había ejecutado el método **pack** para ajustar el tamaño de la ventana al tamaño preferido o al mínimo que permita visualizar todos sus componentes. Esta característica queda ahora anulada por **setSize** (la llamada al método **pack** se puede quitar desde el panel *Code* en la ventana de propiedades). Otra alternativa para establecer el tamaño del formulario es fijarlo sobre la plantilla de diseño haciendo doble clic sobre el borde cuando el cursor del ratón toma la forma que permite redimensionar la misma.

Añadir los componentes al contenedor

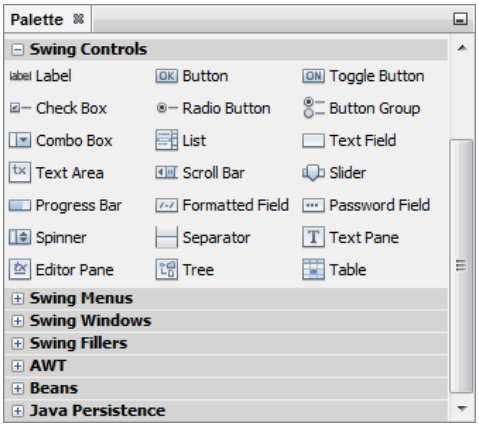
Nuestro objetivo es diseñar una aplicación que muestre una ventana principal con un botón y una etiqueta:



Cuando el usuario haga clic en el botón, la etiqueta mostrará el mensaje "¡¡¡Hola mundo!!!". La figura anterior muestra el aspecto de esta ventana. En ella se puede observar que el botón puede activarse, además de con un clic del ratón, utilizando las teclas *Alt+c*, y que tiene asociada una breve descripción.

Los componentes, tales como cajas de texto, botones, etiquetas, marcos, listas o temporizadores, son objetos gráficos que permiten introducir o extraer datos. El contenedor más los componentes dan lugar al formulario que hace de interfaz o medio de comunicación con el usuario.

Para añadir un componente a un contenedor, primero visualizaremos el panel de diseño. Para ello haga clic en el botón *Design*. Después nos dirigiremos a la paleta de componentes para seleccionar el deseado. La figura siguiente muestra la paleta de componentes *Swing* proporcionada por *NetBeans*:



Cada elemento de la paleta crea un único componente. Para saber de qué componente se trata, mueva el ratón encima del componente y espere a que se muestre la descripción asociada. Como podrá observar, estos componentes están clasificados en grupos; por ejemplo:

- *Swing*. Grupo de componentes de interfaz gráfica de usuario independientes de la plataforma por estar escritos en Java. Estos componentes ofrecen más y mejores funciones que los *AWT*.

- *AWT*. Grupo de componentes de interfaz gráfica de usuario (IGU) que se han implementado utilizando versiones dependientes de la plataforma, sustituido en gran medida por el grupo de componentes *Swing*.

- *Beans*. Componentes software reutilizables en cualquier programa. Mientras que los componentes anteriores son intrínsecos a Java, estos otros existen co-

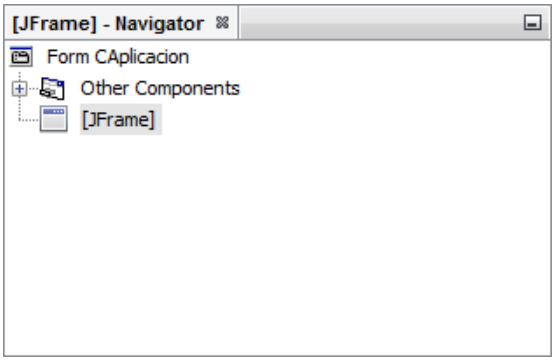
mo ficheros independientes con extensión *.jar*. Se trata de programas Java que nosotros mismos podemos crear con una funcionalidad específica, con la intención de insertarlos posteriormente en otros programas.

- *Layouts*. Administradores de diseño para organizar los componentes que se añaden a un contenedor. Para visualizarlos, haga clic con el botón derecho del ratón sobre el formulario y seleccione *Set Layout* en el menú contextual.

Dibujar los componentes

Añadir uno o más de estos componentes a un contenedor implica dos operaciones: seleccionar un administrador de diseño para dicho contenedor y dibujar sobre él los componentes requeridos.

Por ejemplo, volviendo a la aplicación que estamos desarrollando, diríjase al navegador de componentes (*Navigator*) y observe que el contenedor del formulario (*JFrame*) no tiene asociado un administrador de diseño.



Por omisión, los nuevos formularios creados utilizan un esquema de diseño libre (*Free Design*) que permite distribuir los componentes libremente usando unas líneas guía que automáticamente sugieren la alineación y el espaciado óptimo para los mismos, todo esto sin requerir un administrador de diseño de los definidos en Java. Esto es así porque *Free Design* utiliza el administrador de diseño **GroupLayout** especialmente construido para ser utilizado con el IDE *NetBeans* de forma totalmente transparente para facilitar el diseño de formularios.

Asignar un administrador de diseño

Un administrador de diseño determina el tamaño y la posición de los componentes dentro de un contenedor. *Swing* proporciona varios administradores de diseño: **FlowLayout**, **GridBagLayout**, **BorderLayout**, **CardLayout**, **GridLayout**, **BoxLayout**, etc.

Para asignar a un contenedor un administrador de diseño específico basta con que:

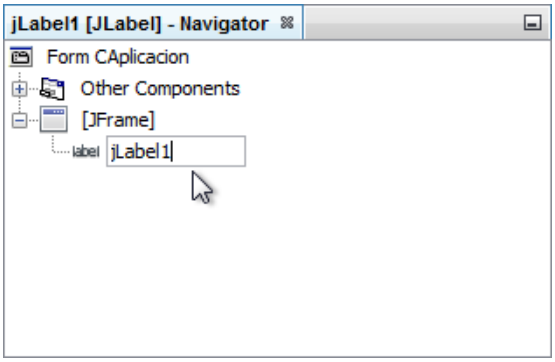
1. Haga clic con el botón derecho del ratón sobre el contenedor y seleccione *Set Layout* en el menú contextual.
2. Y, a continuación, haga clic en el administrador de diseño deseado.

Nosotros seguiremos utilizando *Free Design* por la automatización que aporta a los formularios para colocar los componentes. Otra opción sería utilizar el componente *Null Layout*, pero sabiendo que no proporciona el posicionamiento automático de componentes.

Añadir una etiqueta y editar sus propiedades

Para añadir una etiqueta en el contenedor **JFrame**, siga estos pasos:

1. Seleccione la paleta de componentes *Swing Controls*.
2. Haga clic en el componente *Label* y después diríjase al formulario y haga clic en cualquier parte del contenedor. Ajuste su tamaño y su posición.
3. Cambie su nombre para que sea *jEtSaludo*. ¿Dónde podríamos hacerlo? En el navegador de componentes (*Navigator*):



4. Haga que el texto de la etiqueta aparezca centrado, en negrita y de tamaño 18. Para ello diríjase al formulario y haga clic sobre la etiqueta, o bien diríjase al navegador de componentes y seleccione el componente *jEtSaludo* para visualizar sus propiedades en la ventana de propiedades.
5. Cambie el valor de la propiedad *horizontalAlignment* a *CENTER*. Observará en el formulario que ahora la etiqueta muestra el texto centrado.
6. Cambie el valor de la propiedad *font*. Para ello, una vez seleccionada la propiedad, haga clic en el botón que hay a la derecha del valor que tiene asignado

actualmente para visualizar el diálogo que le permitirá elegir el tipo de fuente, así como su estilo y tamaño.

7. Cambie el valor de la propiedad *text* a "etiqueta".

El trabajo realizado ha generado en la clase *CAplicacion* el siguiente código, lo que puede comprobar a través del editor.

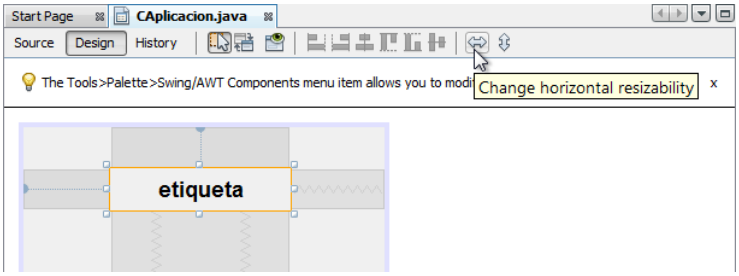
```
private javax.swing.JLabel jEtSaludo;
// ...
jEtSaludo = new javax.swing.JLabel();
// ...
jEtSaludo.setFont(new java.awt.Font("Dialog", 1, 18));
jEtSaludo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jEtSaludo.setText("etiqueta");
// ...
```

Lo que esta etiqueta muestre durante la ejecución habrá que asignárselo a su propiedad **text** directamente por medio de la ventana de propiedades, o indirectamente a través de su método **setText**:

```
jEtSaludo.setText("¡¡¡Hola mundo!!!");
```

Redimensionamiento automático

Un control puede ser anclado arriba y a la izquierda, abajo y a la derecha, etc., seleccionándolo, haciendo clic sobre él con el botón derecho del ratón y eligiendo *Anchor* > *Left* (o *Anchor* > *Bottom*, etc.) del menú contextual. Así mismo, puede ser redimensionado horizontalmente y/o verticalmente cuando se redimensione el formulario, haciendo clic sobre él con el botón derecho del ratón y eligiendo *Auto Resizing* > *Horizontal* (o *Auto Resizing* > *Vertical*) del menú contextual; también puede utilizar los botones equivalentes de la barra de herramientas.



Añadir un botón y editar sus propiedades

Para añadir un botón, puede repetir los pasos descritos en el apartado anterior, o bien realizar los siguientes:

1. Seleccione la paleta de componentes *Swing Controls*.
2. Haga clic en el componente *Button* y después diríjase al formulario y haga clic en cualquier parte del panel. Ajuste su tamaño y su posición.
3. Cambie su nombre para que sea *jBtSaludo*.
4. Modifique su propiedad **text** para que muestre el título "Haga clic aquí".
5. Modifique su propiedad **toolTipText** para que muestre el mensaje "botón de pulsación".
6. Modifique su propiedad **mnemonic** para asociarle la tecla de acceso *c*.

El trabajo realizado ha generado en la clase *CAplicacion* el siguiente código, lo que puede comprobar a través del editor.

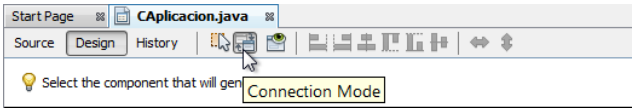
```
private javax.swing.JButton jBtSaludo;
// ...
jBtSaludo = new javax.swing.JButton();
// ...
jBtSaludo.setMnemonic('c');
jBtSaludo.setText("Haga clic aquí");
jBtSaludo.setToolTipText("botón del pulsación");
// ...
```

Asignar manejadores de eventos a un objeto

Considere el botón *jBtSaludo* que anteriormente añadimos a la interfaz gráfica de *CAplicacion*. Cada vez que el usuario haga clic sobre este botón, se generará un evento de acción que podrá ser interceptado por un manejador de este tipo de eventos, si es que tiene uno asociado. La respuesta será mostrar en la etiqueta *jEtSaludo* el mensaje "¡¡¡Hola mundo!!!". Para ello tiene que existir una conexión entre el botón y la etiqueta, lo que se traducirá en la ejecución de un método que asigne esa cadena de caracteres a la etiqueta como respuesta al evento clic.

Para facilitar la realización del tipo de conexiones al que nos hemos referido, *NetBeans* proporciona un asistente para conexiones, el cual permite añadir el código que un componente tiene que ejecutar para responder al mensaje que otro componente le ha enviado, y todo ello con muy poca intervención del desarrollador. Para utilizar este asistente, debe realizar los pasos indicados a continuación:

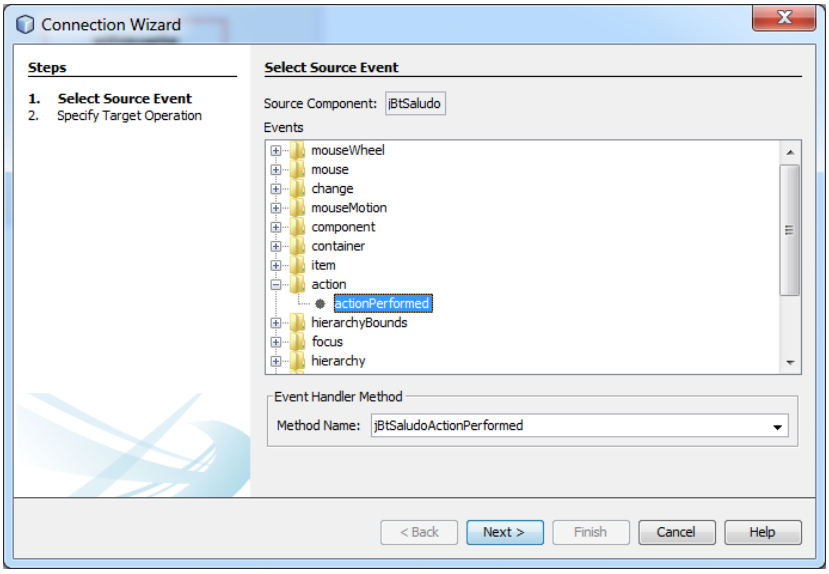
1. Diríjase a la ventana de diseño. Observe los botones que hay en su parte superior.
2. Cambie al modo de conexión haciendo clic en el botón *Connection Mode*:



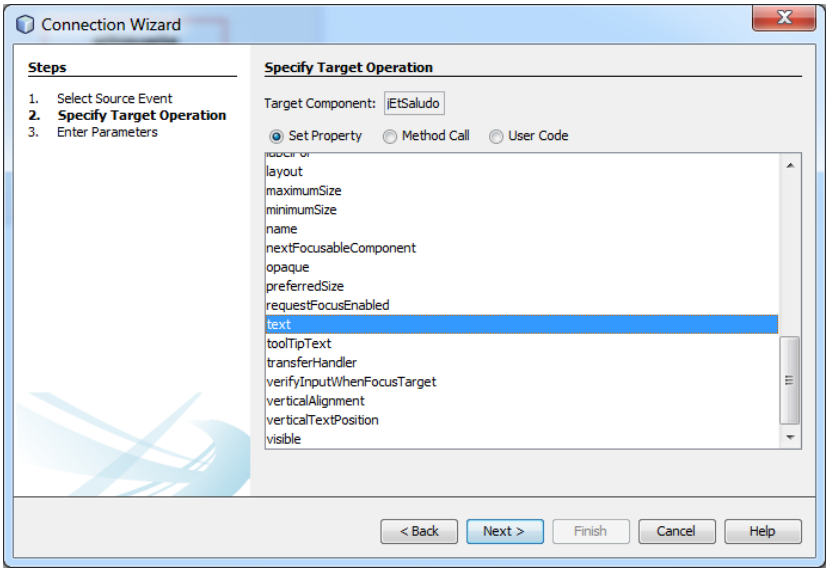
3. A continuación haga clic primero en el componente sobre el que ocurrirá el evento (en el ejemplo, sobre el botón) y después en el componente sobre el que se realizará la operación (en el ejemplo, sobre la etiqueta). Puede hacer las operaciones indicadas directamente sobre los componentes del formulario, o bien sobre el navegador de componentes.

   Cuando haya ejecutado este punto, *NetBeans* abre el asistente que le permitirá realizar la conexión en dos o tres pasos.

4. Seleccione el evento del componente origen (*Select Source Event*). En este primer paso, el diálogo que se muestra permitirá seleccionar el evento que se desea manejar del componente seleccionado y el nombre del método que responderá a ese evento.



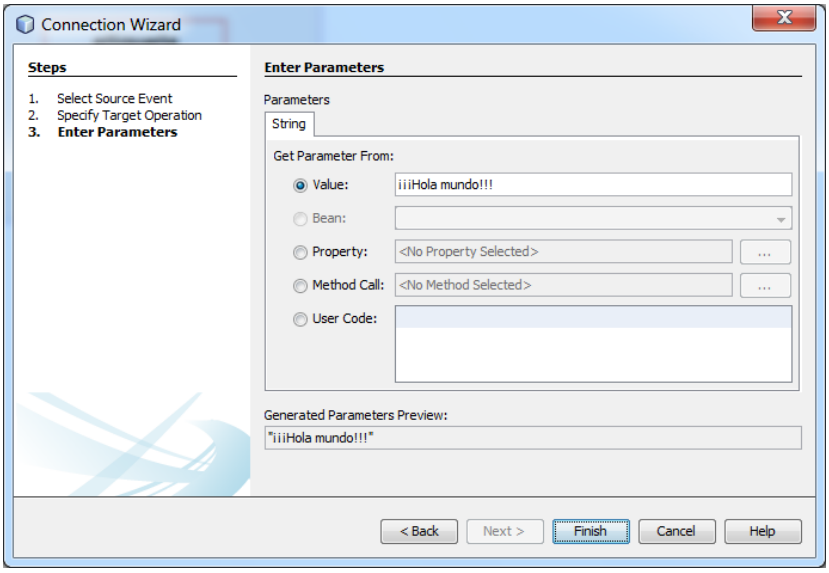
5. Haga clic en el botón *Next* para especificar la operación en el destino (*Specify Target Operation*). En este segundo paso, el diálogo que se muestra permite especificar la operación que tiene que realizarse en el componente seleccionado como destino de la misma (*Set Property > text*):



6. Para especificar esta operación, disponemos de tres opciones: establecer el valor de una propiedad, llamar a un método o escribir código.

   - Si elige establecer el valor de una propiedad, le serán mostradas todas las propiedades del componente destino de la conexión. Seleccione una y en el siguiente paso establezca su valor.
   - Si elige llamar a un método, le serán mostrados los posibles métodos a los que puede invocar. Seleccione uno y especifique los parámetros en el siguiente paso.
   - Si elige escribir código, se añadirá al programa el método que debe responder al evento seleccionado, pero sin código (en este caso no hay un tercer paso).

   En nuestro ejemplo elegiremos la primera opción, aunque también podría valer la tercera.



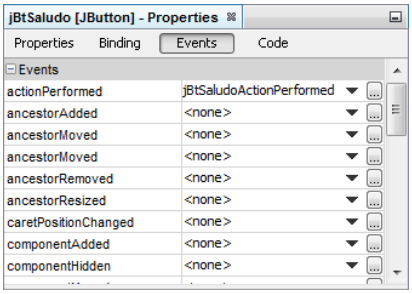
El resultado será que a *CAplicacion* se añade el código siguiente:

```
jBtSaludo.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(java.awt.event.ActionEvent evt)
  {
    jBtSaludoActionPerformed(evt);
  }
});

// ...

private void jBtSaludoActionPerformed(java.awt.event.ActionEvent evt)
{
  jEtSaludo.setText("¡¡¡Hola mundo!!!");
}
```

Este manejador también podemos añadirlo seleccionando el botón *jBtSaludo*, el panel *Events* (eventos) en la ventana de propiedades, el evento *actionPerformed* en este panel al que, finalmente, asignaremos el nombre del método que debe responder a tal evento:



Para este evento, también podemos añadir su manejador haciendo doble clic sobre el botón *jBtSaludo*.

Eliminar un método añadido por el asistente

Si quisiéramos eliminar el método *actionPerformed* añadido anteriormente, seleccionaríamos el botón *JBtSaludo*, el panel *Events* (eventos) en la ventana de propiedades, el nombre del método asignado al evento *actionPerformed*, pulsaríamos la tecla *Del*, para borrar el valor actual, y finalizaríamos esta operación pulsando la tecla *Entrar*.

Añadir otro código

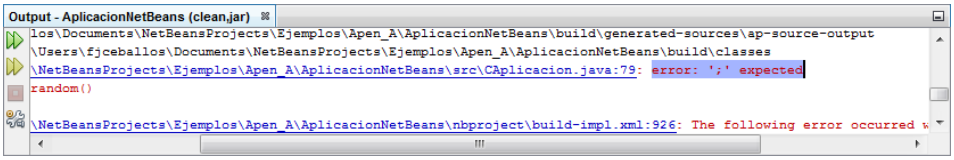
Para finalizar la aplicación que estamos escribiendo, diríjase a la ventana de edición y complete el método como se indica a continuación con el fin de cambiar de forma aleatoria el color del texto:

```
private void jBtSaludoActionPerformed(java.awt.event.ActionEvent evt)
{
  float rojo = (float)Math.random();
  float verde = (float)Math.random();
  float azul = (float)Math.random();
  jEtSaludo.setForeground(new java.awt.Color(rojo, verde, azul));
  jEtSaludo.setText("¡¡¡Hola mundo!!!");
}
```

Compilar la aplicación

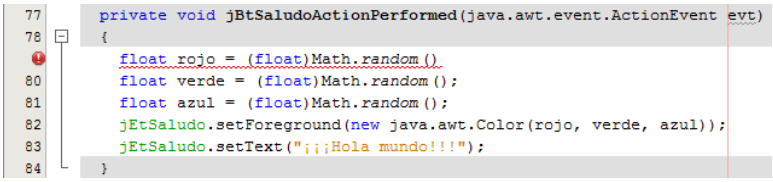
Una vez finalizada la aplicación, puede compilarla y ejecutarla. Para compilar la aplicación, ejecute la orden *Build Project* (*F11*) del menú *Run*. En realidad, esta operación se hace automáticamente cuando se guarda un fichero. Si quiere recompilar la aplicación, ejecute *Run* > *Clean and Build Project*.

Si la construcción del fichero *.class* resulta satisfactoria, verá en la ventana de salida del compilador el mensaje "*BUILD SUCCESSFUL*". Si hay problemas con la construcción, verá los mensajes de error correspondientes en la misma ventana. Por ejemplo, suponga que en la primera línea de código del método *jBtSaludoActionPerformed* olvidó el punto y coma final. Cuando ejecute la orden *Build*, le será mostrada una ventana como la siguiente:



La ventana de la figura anterior indica que el compilador ha detectado que falta un punto y coma. Para ir a la línea de código donde el compilador ha detectado el error y corregirlo, puede hacer clic sobre el mensaje de error. Una vez que obtenga una compilación sin errores, puede ejecutar la aplicación.

No obstante, este error también se hace visible antes de la compilación en la ventana de edición. Observe la línea subrayada en la figura siguiente; es la que contiene el error. Si pone el puntero del ratón sobre ella, le será mostrado un mensaje corto indicando el error cometido:

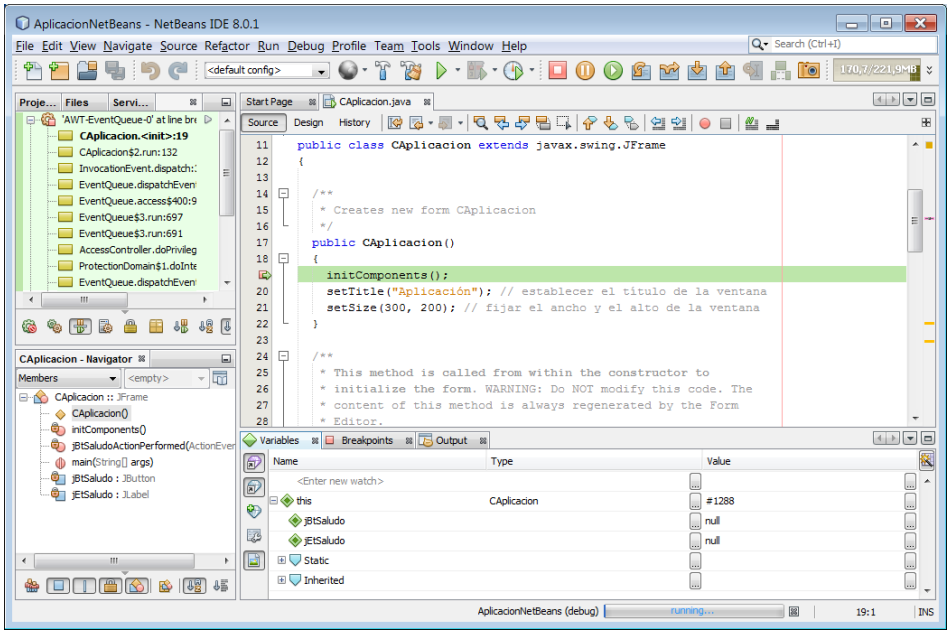


Para ejecutar la aplicación, seleccione la orden *Run Project* del menú *Run*. Si no hay errores de ejecución, *NetBeans* mostrará los resultados.

No obstante, cuando la solución obtenida no sea satisfactoria y no seamos capaces de localizar dónde se está produciendo el error (imaginemos una expresión *a/(2*b)* en la que olvidamos poner los paréntesis), podemos utilizar el depurador para ejecutar el programa paso a paso y poder investigar los resultados parciales que se van produciendo a medida que avanza la ejecución.

Depurar la aplicación

Apoyándonos en la aplicación que acabamos de diseñar, enumeramos una serie de pasos que le enseñarán cómo trabajar con el depurador:



1. Diríjase a la ventana de edición y haga clic con el botón izquierdo del ratón a la izquierda de la llamada al método *initComponents* que hay en el constructor *CAplicacion*, sobre la columna que muestra el número de línea. Ha establecido un punto de parada; esto es, cuando se ejecute la aplicación según indica el punto siguiente, la ejecución será detenida en ese punto. Para quitar el punto de parada, proceda de igual forma.

   También puede establecer un punto de parada haciendo clic con el botón derecho del ratón sobre la línea de código y ejecutando la orden *Toggle Line Breakpoint* (*Ctrl+F8*) del menú contextual que se visualiza.

2. Ejecute la orden *Debug Project* (*Ctrl+F5*) del menú *Debug*. La ejecución del programa se inicia y se detiene en el punto de parada anteriormente añadido, instante en el que se muestra el espacio de trabajo de depuración compuesto por los paneles *Variables*, *Breakpoints*, etc. que puede observar en la figura anterior.

   La depuración puede también iniciarse ejecutando la orden *Step Into* (*F7*) del menú *Debug*. Esta forma de proceder no requiere poner un punto de parada inicial.

   En cualquier instante, puede detener la depuración ejecutando la orden *Finish Debugger Session* (*Mayúsculas+F5*) del menú *Debug*.

3. Puede continuar la ejecución paso a paso por sentencias pulsando la tecla *F7* (*Step Into*), paso a paso por métodos pulsando la tecla *F8* (*Step Over*), hasta la posición del cursor pulsando la tecla *F4* (*Run To Cursor*), hasta que el método actual finalice y el control pase al método que lo invocó pulsando las teclas *Ctrl+F7* (*Step Out*), o continuar la ejecución hasta el siguiente punto de parada o hasta el final de la aplicación pulsando la tecla *F5*.

   Paso a paso por sentencias significa ejecutar cada método del programa paso a paso. Si no quiere que los métodos invocados por el método actualmente en ejecución se ejecuten sentencia a sentencia, sino de una sola vez, utilice la tecla *F8* en vez de *F7*.

4. Cuando la ejecución está detenida, puede inspeccionar los valores que van tomando las variables del programa. El panel *Variables* del espacio de trabajo de depuración será el encargado de presentarnos los valores deseados. Para añadir una variable que aparece en el código fuente a esta lista, añádala en el panel *Variables* (*<Enter new watch>*). Para eliminar una variable del panel de inspección, selecciónela y pulse la tecla *Supr* o haga clic sobre ella utilizando el botón derecho del ratón y ejecute la orden *Delete* del menú contextual que se visualiza.

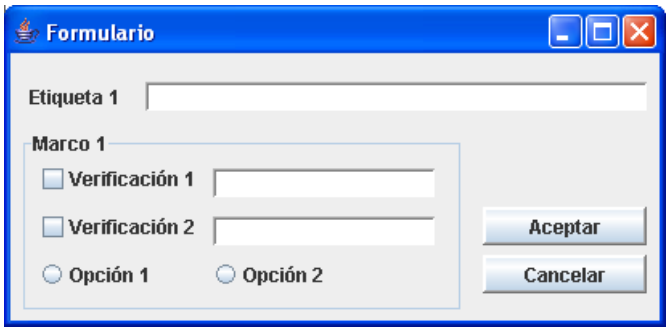
   Así mismo, en el panel *Call Stack* puede ver la pila de llamadas, en el panel *Breakpoints* los puntos de parada, etc. Observe el menú mostrado cuando ejecuta *Window > Debugging*, muestra los distintos paneles que puede utilizar.

   También, colocando el punto de inserción sobre una variable en una línea de código ya ejecutada, le será mostrada una descripción con el valor de dicha variable.

Como alternativa a las órdenes mencionadas del menú *Debug* en los puntos anteriores, puede utilizar los botones correspondientes de la barra de herramientas o las teclas de acceso directo.

Administradores de diseño nulo y absoluto

El administrador de diseño nulo, denominado *Null Layout*, no es estándar y permite colocar los componentes exactamente donde se quiera, pero sin utilizar objetos para definir el tamaño y las coordenadas del componente colocado. A diferencia de este, el administrador de diseño absoluto, denominado *AbsoluteLayout*, no estándar, hace lo mismo y, además, utiliza objetos para definir el tamaño y las coordenadas del componente colocado. De cualquiera de las dos formas resulta muy sencillo diseñar interfaces como la siguiente:



Además, para colocar cada uno de los componentes, nos podremos ayudar de una rejilla invisible sobre el editor de formularios, que se puede configurar desde el diálogo *Options* (menú *Tools*, orden *Options*, botón *Java*, panel *GUI Builder*, valor *Grid Size*).

Es importante saber que el empleo de la plantilla *AbsoluteLayout* (o no utilizar plantilla) no es recomendado en general, ya que cuando el entorno cambia, las posiciones y el tamaño de los componentes permanecen inalterables. Además, pueden aparecer distorsiones cuando tal aplicación corra en una plataforma diferente o cuando decida mostrar los componentes con una apariencia diferente. No obstante, ambas formas de diseño se pueden utilizar por la facilidad que ofrecen y, posteriormente, antes de distribuir la aplicación, cambiar al administrador de diseño *GridBagLayout*. Además, este cambio es facilitado por *NetBeans* al proporcionar un asistente que simplifica la creación de formularios que utilizan el administrador de diseño *GridBagLayout*. Esto es, una vez que haya cambiado a este administrador, para utilizar el asistente al que nos hemos referido, diríjase al navegador de componentes (*Navigator*) y utilizando el botón derecho del ratón haga clic sobre el nodo *GridBagLayout* y ejecute la orden *Customize* del menú contextual que se visualiza.

AÑADIR OTROS FORMULARIOS A LA APLICACIÓN

Normalmente, una aplicación que muestra una interfaz gráfica al usuario despliega una ventana principal y a partir de ella pueden mostrarse otras ventanas de diálogo de alguno de los grupos siguientes:

- *Ventanas de diálogo predefinidas*. Son ventanas de diálogo creadas por medio de los métodos proporcionados por la clase **JOptionPane** de la biblioteca JFC (*Java Foundation Classes* – clases base de Java); por ejemplo, **showMessageDialog**.
- *Ventanas de diálogo personalizadas*. Son ventanas de diálogo hechas a medida, para lo cual la biblioteca JFC proporciona la clase **JDialog**.

- *Ventanas de diálogo estándar*. Son ventanas muy comunes; por ejemplo, las ventanas de diálogo *Abrir* o *Guardar* proporcionadas por la clase **JFileChooser**, o el diálogo *Color* proporcionado por la clase **JColorChooser**.

Para añadir uno de estos diálogos, por ejemplo un **JDialog**, puede optar por alguna de las siguientes opciones:

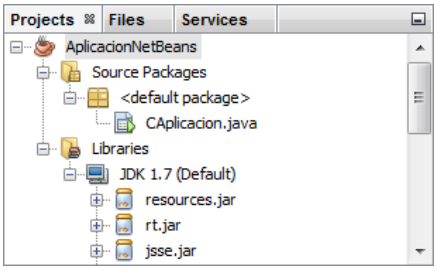
1. Añada un objeto **JDialog** desde la paleta de componentes *Swing*. Una vez añadido, diríjase al navegador de componentes, haga doble clic sobre el nuevo diálogo añadido y complételo de acuerdo con sus necesidades.

2. Añada al proyecto una nueva clase derivada de la clase *JPanel Form* (para ello haga clic con el botón derecho del ratón sobre el nombre del proyecto) y modifique la definición de la misma para que se derive de **JDialog** en lugar de derivarse de **JPanel**. Después, desde cualquier otra parte de su aplicación, podrá crear objetos de la nueva clase añadida.

Si opta por la opción 1, el objeto **JDialog** quedará integrado en su aplicación como cualquier otro control que haya añadido a su ventana principal. En cambio, si opta por la opción 2, se añadirá una nueva clase derivada de **JDialog**.

PROYECTOS

Un proyecto permite agrupar los ficheros requeridos para producir una aplicación o un *applet*. Esto presenta ventajas como poder compilar todo el proyecto sin tener que especificar los ficheros que incluye, especificar la clase principal del proyecto, ver bajo el panel *Projects* del explorador todos los ficheros que componen el proyecto, configurar el entorno de desarrollo integrado (EDI) para cada proyecto, etc. De esto se deduce que para un determinado proyecto podemos configurar un escenario particular que será guardado cuando se finalice la sesión, lo que permitirá recuperarlo automáticamente la próxima vez que se cargue ese proyecto.



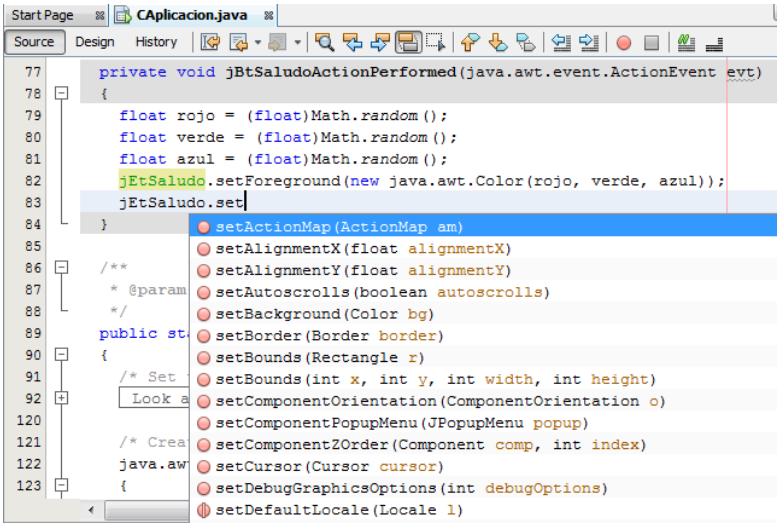
Puede tener cargados varios proyectos simultáneamente y activar en cada instante aquel sobre el que desea trabajar. Para ello tiene que hacer clic sobre el

nombre del proyecto que desea sea el actual, o bien establecerlo como tal ejecutando la orden *Set Main Project* del menú *Run*.

De forma predeterminada la codificación de los proyectos *NetBeans* es UTF-8. Si necesita cambiarla, haga clic con el botón secundario del ratón sobre el nombre del proyecto y especifique este código en *Propiedades* > *Sources* > *Encoding*.

COMPLETAR EL CÓDIGO MIENTRAS SE ESCRIBE

NetBeans proporciona la característica de completar el código mientras lo escribe:



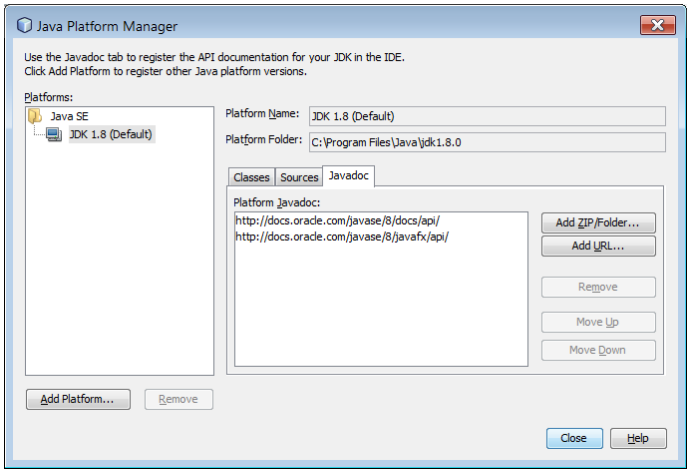
Por ejemplo, según puede verse en la figura anterior, cuando haya escrito *jEtSaludo*, aparecerá una lista de los posibles métodos que pueden ser invocados (si no aparece dicha lista, pulse las teclas *Ctrl+espacio*); seleccione el adecuado y pulse la tecla *Entrar*.

Este tipo de ayuda solo estará disponible si está activada la opción de completar código. Para verlo, ejecute la orden *Options* del menú *Tools* y haga clic en el botón *Editor* > *Code Completion*.

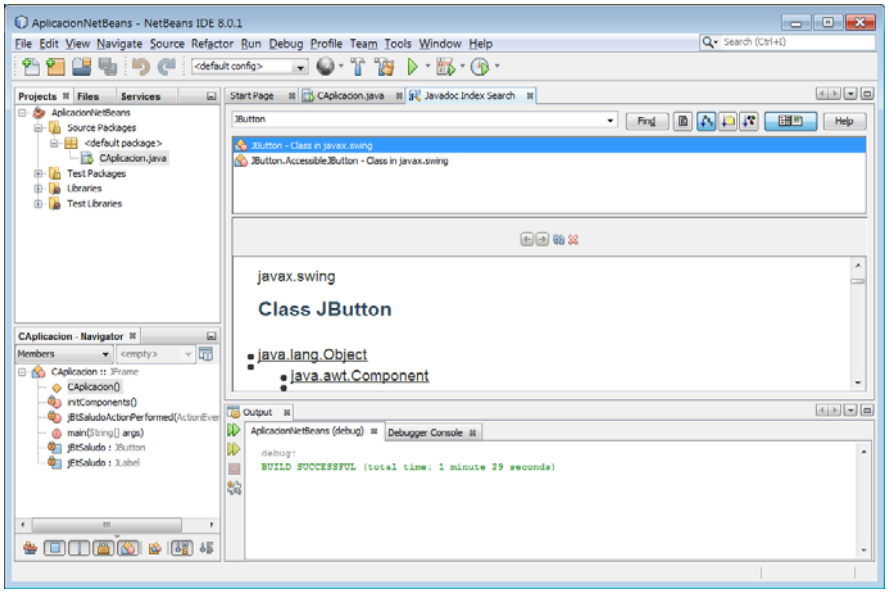
OBTENER AYUDA

La orden *Javadoc Index Search* del menú *Help* (*Mayúsculas+F1*) permite obtener ayuda acerca de múltiples temas. Pero si quiere obtener ayuda acerca de la biblioteca de clases de Java (suponiendo que la ha instalado), es preciso que dicho entorno tenga conocimiento de la ruta de acceso a la misma. Para ello ejecute la

orden *Java Platform* del menú *Tools* y asegúrese de que en la lista de rutas mostrada en el panel *Javadoc* hay una que hace referencia a la carpeta donde se encuentra la ayuda mencionada; si no es así, añádala.



Después del proceso anterior, cuando pulse las teclas *Mayúsculas+F1*, se visualizará el panel que se muestra en la figura siguiente, donde podrá solicitar ayuda acerca de la clase que quiera. También puede dirigirse al editor de código, colocar el cursor sobre el nombre de la clase, método, etc. de la cual quiere ayuda y pulsar las teclas *Mayúsculas+F1*.



Así mismo, puede obtener ayuda acerca del desarrollo de aplicaciones y del EDI pulsando la tecla *F1*, o bien ejecutando la orden *Help Contents* del menú *Help*.

CREAR UN APPLET

Para ejecutar un *applet* en una aplicación web, hay que crearlo separadamente como un proyecto de tipo *Class Library* (biblioteca de clases) y empaquetar después el fichero JAR del *applet* en la aplicación web.

Para crear un *applet* desde el entorno de desarrollo *NetBeans* siga los pasos indicados a continuación:

1. Seleccione *File* > *New Project* > *Java* > *Class Library*.

2. Escriba el nombre del proyecto y seleccione la carpeta donde desea almacenarlo.

3. Una vez creado el proyecto, haga clic sobre el nombre del mismo utilizando el botón secundario del ratón y seleccione *New* > *Other* > *Java* > *JApplet*.

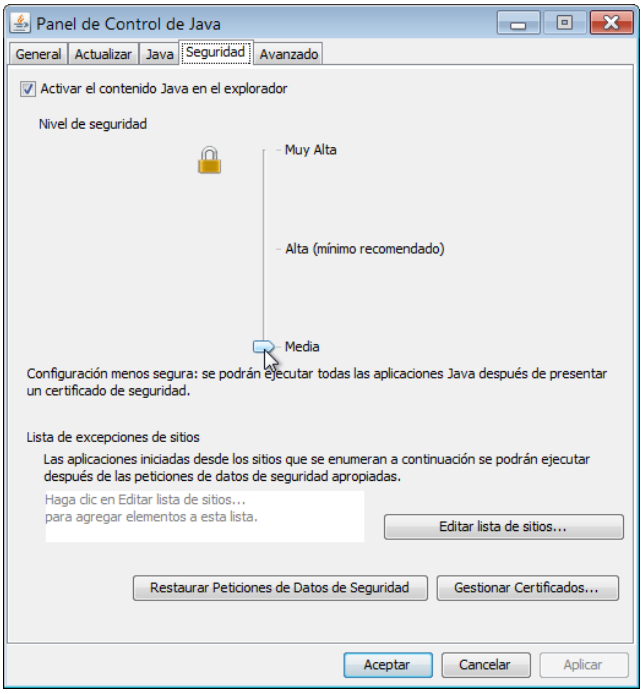
4. Escriba el nombre del *applet* y seleccione la carpeta donde desea almacenarlo.

5. Si es necesario, añada una página *html* para desplegarlo. Para ello seleccione *New* > *Other* > *Other* > *HTML File*.

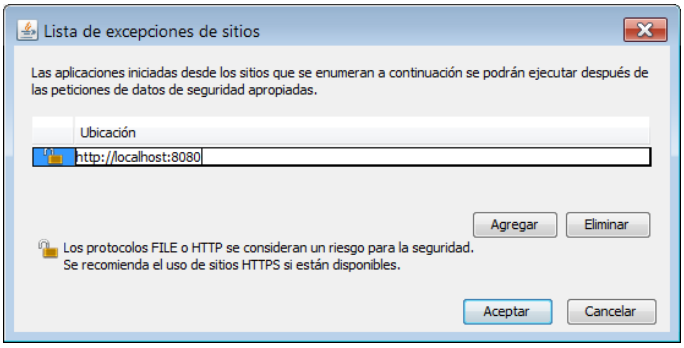
Ejecutar un applet

Supongamos que hemos construido un *applet* formado por los ficheros *MiApplet.java* y *MiApplet.html* (véase el capítulo *Clientes*). Para ejecutarlo, basta con hacer doble clic sobre la página *MiApplet.html*; esta acción abrirá el explorador que tenga instalado y mostrará la página web con la ejecución del *applet*. Ahora bien, sepa que desde la versión 1.7 U51 de Java, los *applets* con acceso al sistema local autofirmados (puede echar una ojeada a las utilidades *keytool* y *jarsigner* de Java) ya no pueden ejecutarse. Si lo intenta le será mostrado un mensaje de advertencia grave. Para poder ejecutarlos, hay que obtener un certificado y firmarlos.

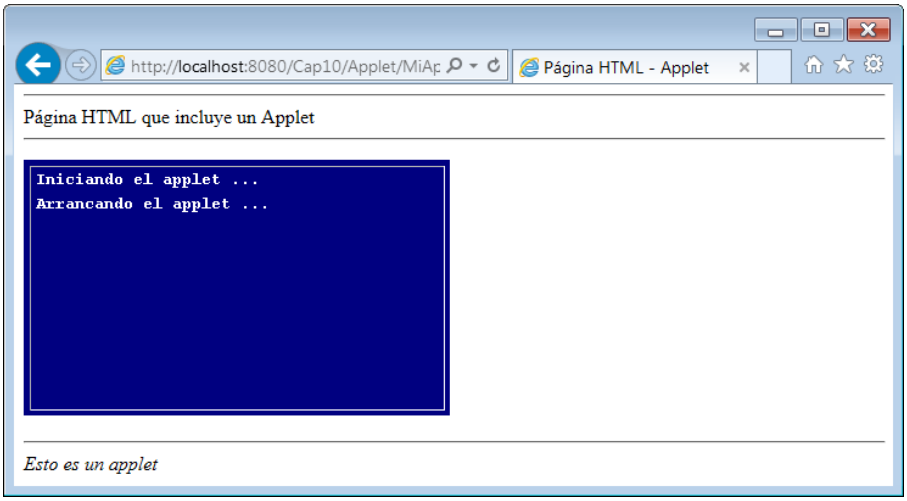
Entonces, ¿cómo podemos probarlos en nuestro sistema local sin tener que realizar el proceso de firma? En una primera aproximación, en Windows, mostramos el panel de control de Java: *Inicio* > *Todos los programas* > *Java* > *Configurar Java* > *Seguridad* y bajamos el nivel de seguridad al mínimo posible.



Esta acción será suficiente para ejecutar el *applet* si este no tiene que acceder a los recursos de nuestra máquina. Como alternativa, puede probar esto otro: añada a la lista de excepciones de sitios administrada por el panel de control de Java el URL del sitio del *applet* que deseamos ejecutar. En nuestro caso, editamos esta lista haciendo clic en el botón *Editar lista de sitios* del panel (véase la figura anterior) y agregamos el URL del sitio de nuestro *applet* (*http://localhost:8080*):



Esta acción también será suficiente para ejecutar el *applet* si este no tiene que acceder a los recursos de nuestra máquina; por ejemplo, para leer un fichero que contiene una imagen.



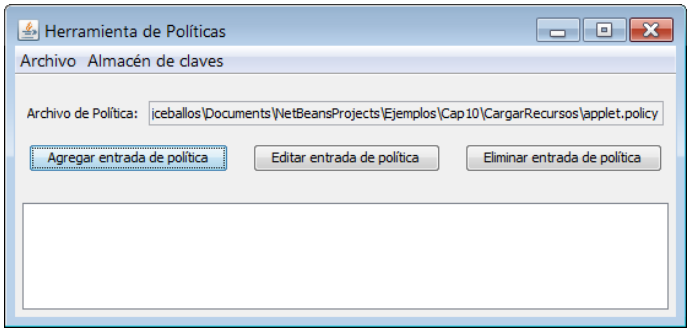
En otro caso, por seguridad, ningún *applet* tiene permitido el acceso a los recursos de nuestra máquina, a menos que explícitamente se lo concedamos incluyendo el permiso correspondiente en la política de seguridad. Para ello tendremos que editar un fichero de política que incluya los permisos adecuados y cargar dicho fichero desde el fichero *java.security* de Java, según exponemos a continuación.

Editar la política de seguridad

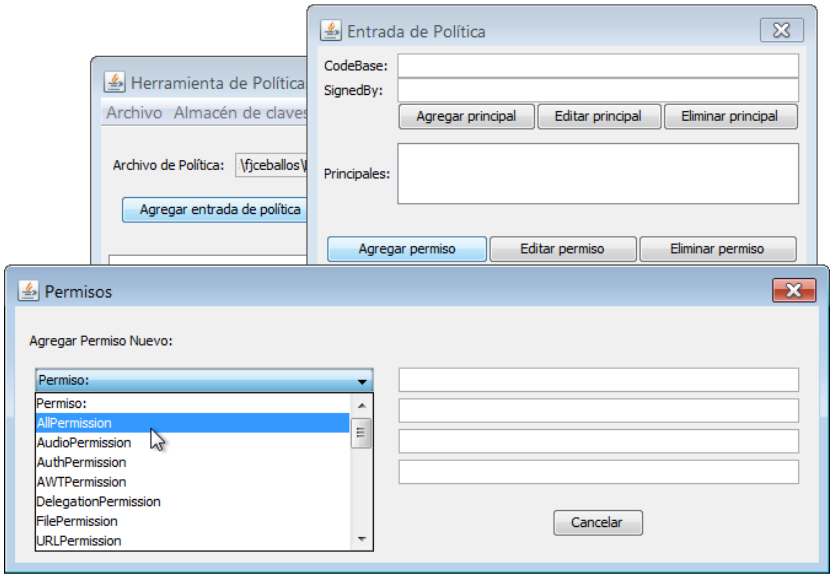
Actualmente, en las plataformas Java los ficheros relacionados con la seguridad residen en la carpeta *java.home\lib\security* (*java.home* es la carpeta en la que se instaló el JRE). En esta carpeta se encuentran, entre otros, el fichero de propiedades de seguridad *java.security* y el fichero de política del sistema *java.policy*.

En el fichero *java.security* se configuran propiedades de seguridad que son utilizadas con las clases del paquete **java.security**; por ejemplo, el URL para el fichero de política que deba ser utilizado con una aplicación concreta.

Pensando en nuestro *applet*, vamos a añadir un fichero de política, por ejemplo *applet.policy*, que añada un permiso para que el dicho *applet* pueda acceder a los recursos de nuestra máquina. Para ello ejecutamos la herramienta *java.home\bin\policytool*. A continuación creamos el fichero *applet.policy*, o lo editamos si ya existe:



En el siguiente paso ejecutamos *Agregar entrada de política* > *Agregar permiso* > *Permiso* > *AllPermission* > *Aceptar* > *Listo* > *Archivo* > *Guardar*:



A continuación editamos el fichero *java.security* y añadimos la línea que especifica la ruta del fichero de política que hay que cargar:

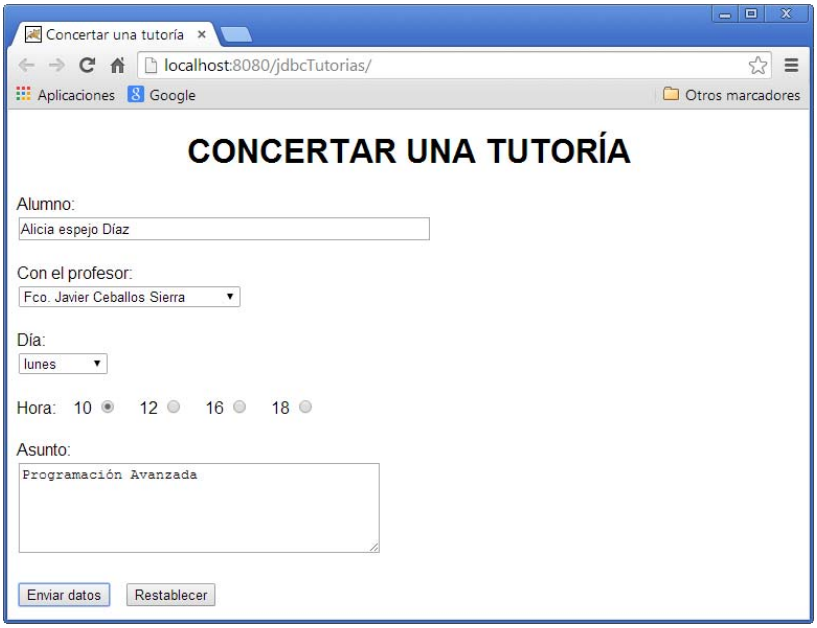
```
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
policy.url.3=file:/C:/.../CargarRecursos/applet.policy
```

Ahora, cuando ejecutemos el *applet*, previamente se establecerán los permisos establecidos en los ficheros especificados por las líneas anteriores.

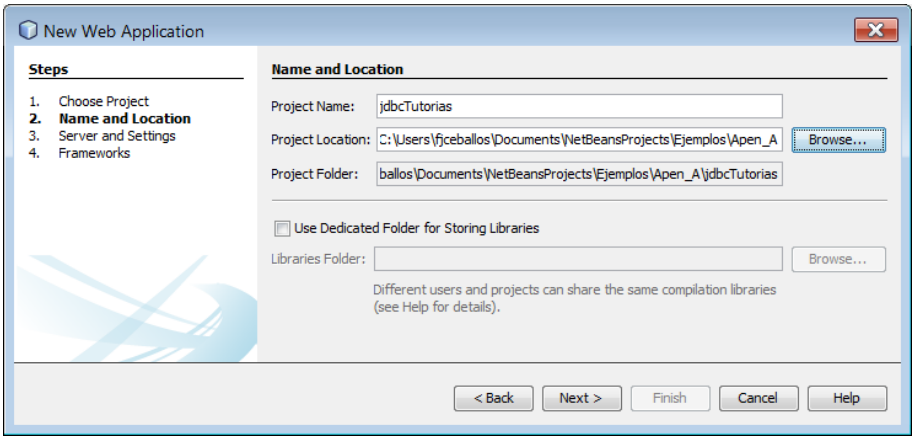
APLICACIÓN JSP-SERVLET

En los capítulos 11 y 12 explicamos cómo desarrollar aplicaciones para Internet utilizando *servlets* y *JSP*. *NetBeans* también permite este tipo de desarrollos.

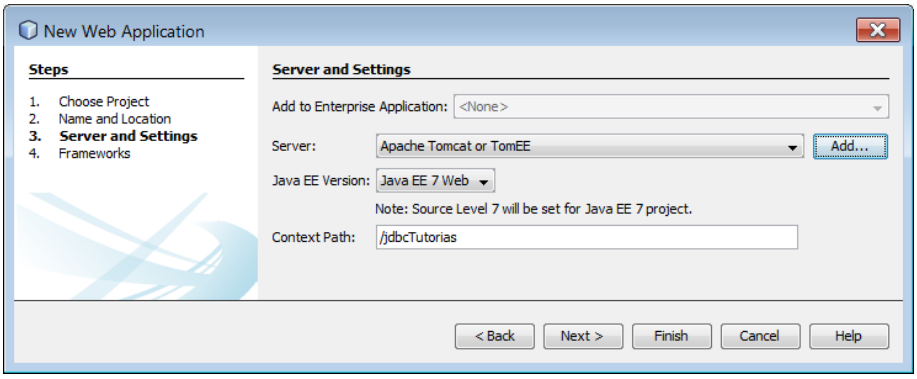
Como ejemplo de desarrollo de una aplicación para Internet con *NetBeans*, vamos a realizar otra versión de la aplicación que desarrollamos en el capítulo 11 (*cap11\jdbcTutorias1*) para que permita a un alumno concertar una tutoría, según muestra la figura siguiente. Recuerde que esta aplicación estaba formada por el *servlet SvTutorias*, la clase *BDTutorias*, el fichero *index.html*, y que accedía a la base de datos *bd_tutorias*.



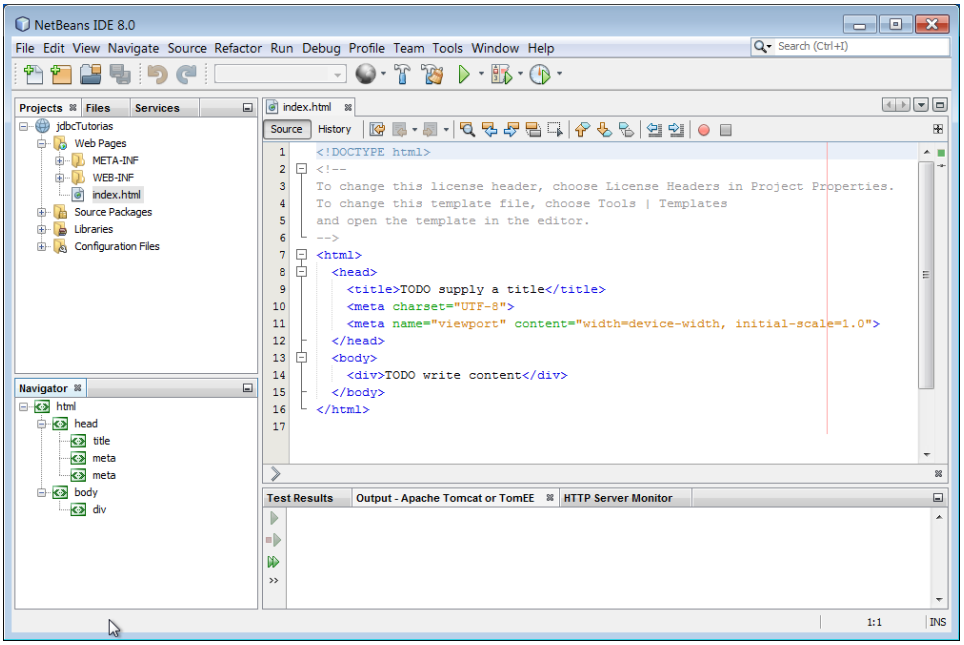
Inicie una nueva aplicación: *File* > *New Project* > *Java Web* > *Web Application*.



Pulse el botón *Next*. Se visualizará la ventana siguiente:



Pulse el botón *Finish*. Se visualizará la ventana siguiente:



Ya tiene creado el módulo web. Los pasos siguientes tienen como fin añadir los ficheros correspondientes al *servlet SvTutorias*, a la clase *BDTutorias* y al fichero *tutorias.html* que creamos en el capítulo 11 para que un alumno pudiera solicitar una tutoría.

Escriba entre las etiquetas *<html>* y *</html>* del fichero *index.html* el contenido del fichero *tutorias.html*. Así, este fichero pasa a llamarse ahora *index.html*. Queda como ejercicio para el lector crear la lista de profesores de esta página web dinámicamente, obteniendo los datos de la tabla *profesores* de la base de datos; esto puede hacerlo, por ejemplo, creando una etiqueta personalizada según se explicó en el apartado *Crear una aplicación JEE con NetBeans* del capítulo *Java EE*. Puede también, si lo desea, cambiar su nombre y editar el fichero *web.xml* de la carpeta *WEB-INF* para hacer referencia al mismo.

Cuando edite el fichero *index.html*, asegúrese de que el URL especificado por el atributo **action** de **form** es:

```
<form action="http://localhost:8080/jdbcTutorias/SvTutorias"
```

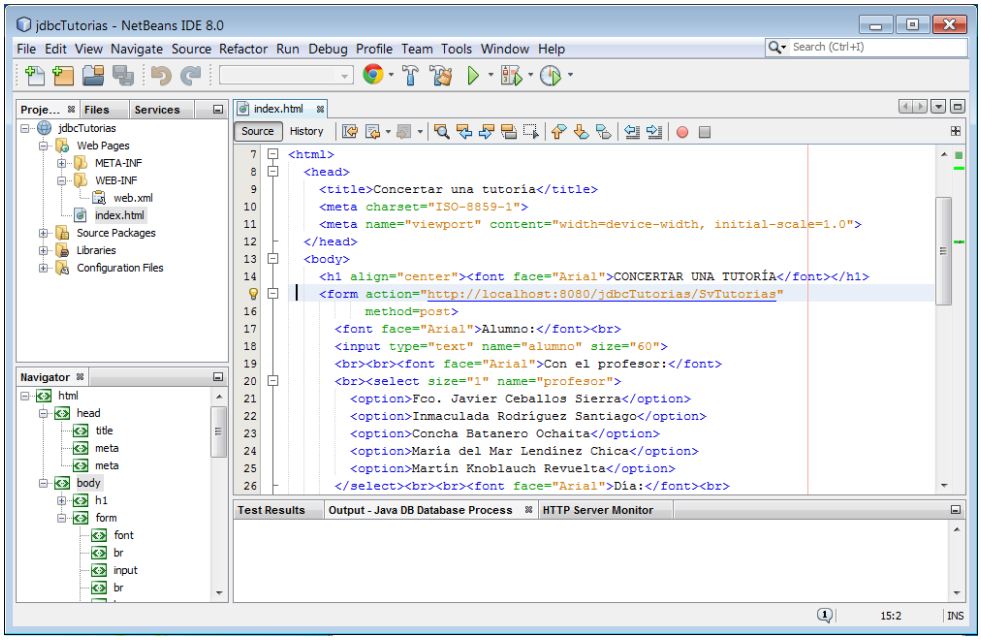
Si utiliza variables o constantes que incluyen vocales acentuadas y otras letras especiales del idioma español, quizás tenga que especificar que se utilice el juego de caracteres adecuado. En una página HTML puede hacerlo así:

```
<html>
  <head>
```

```
    <title>TODO supply a title</title>
    <meta charset="ISO-8859-1">
    <meta name="viewport" content="width=device-width,
                initial-scale=1.0">
  </head>
  <body>
    ...
  </body>
</html>
```

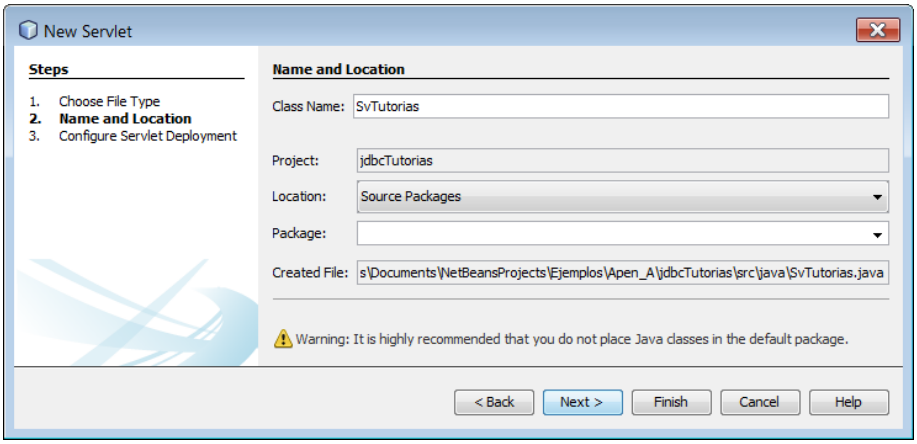
En una página JSP puede hacerlo así:

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>JSP Page</title>
  </head>
  <body>
    ...
  </body>
```

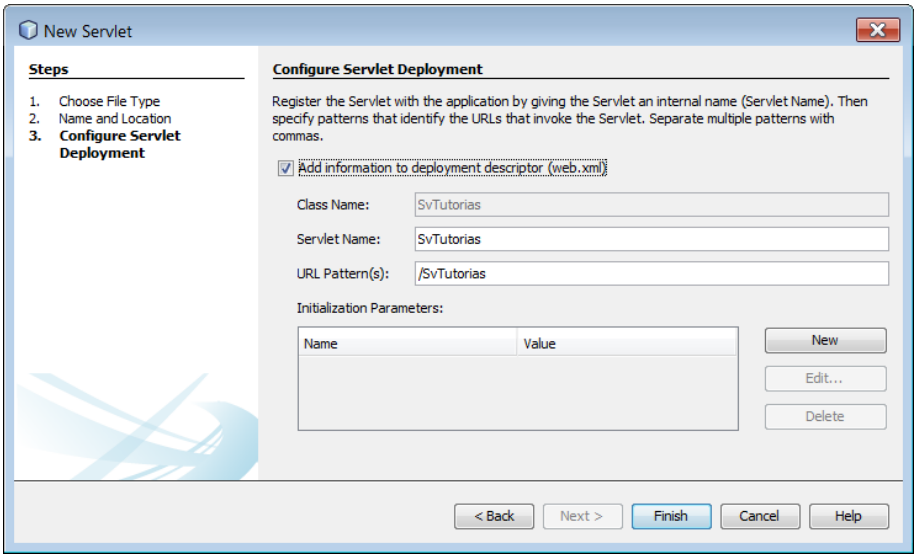


Para añadir a la aplicación un *servlet* o cualquier otro elemento, los pasos siempre son los mismos: clic con el botón derecho del ratón sobre la carpeta donde se va a guardar el elemento, y seleccionar *New* > *elemento a añadir*.

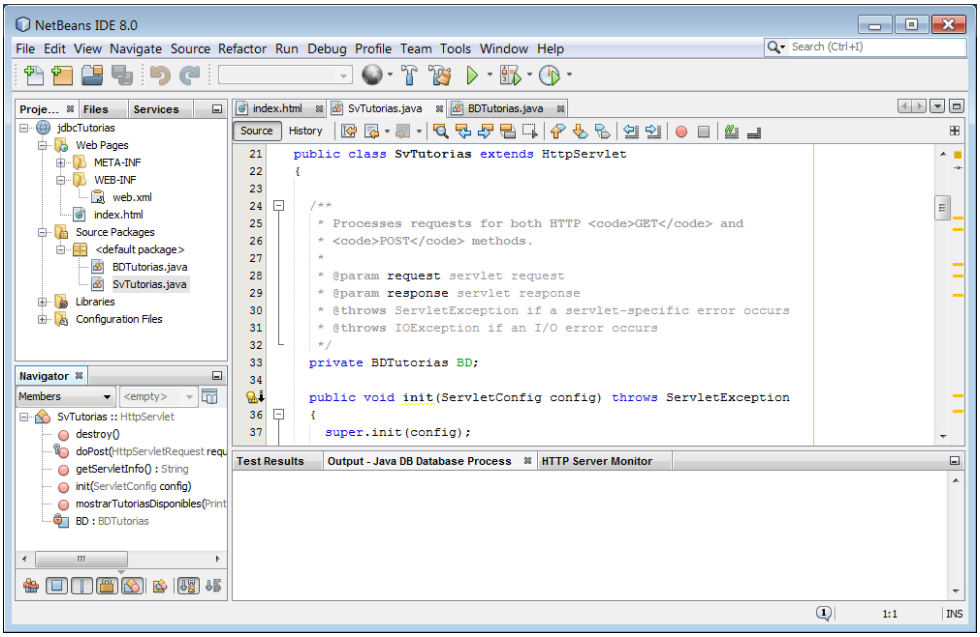
Así, para añadir el *servlet SvTutorias*, haga clic con el botón derecho del ratón sobre el nombre *jdbcTutorias* del proyecto, y seleccione *New* > *Servlet*. En la ventana que se visualiza, introduzca el nombre del fichero *SvTutorias*.



Pulse el botón *Next*.



Pulse el botón *Finish* y escriba el código del fichero.



Para añadir la clase *BDTutorias*, proceda de forma análoga. Esto es, haga clic con el botón derecho del ratón sobre el nombre *jdbcTutorias* del proyecto, y seleccione *New* > *Java Class*. En la ventana que se visualiza, introduzca el nombre del fichero *BDTutorias* y pulse el botón *Finish*. Después escriba el código del fichero.

Finalmente, edite el fichero *web.xml* y asegúrese de que la etiqueta <*url-pattern*> especifica el patrón */servlet/SvTutorias* y de que la etiqueta <*welcome-file*> especifica el fichero *index.html*.

Asegúrese también de que *NetBeans* conoce la ruta donde se localiza el conector *mysql-connector-java-bin.jar* (véase en el apéndice B el apartado *Utilizar el controlador MySQL con NetBeans*).

Después asegúrese de que ha iniciado el servidor *MySQL* y de que el usuario y la contraseña que especificó en *BDTutorias* para realizar la conexión con la base de datos son correctos.

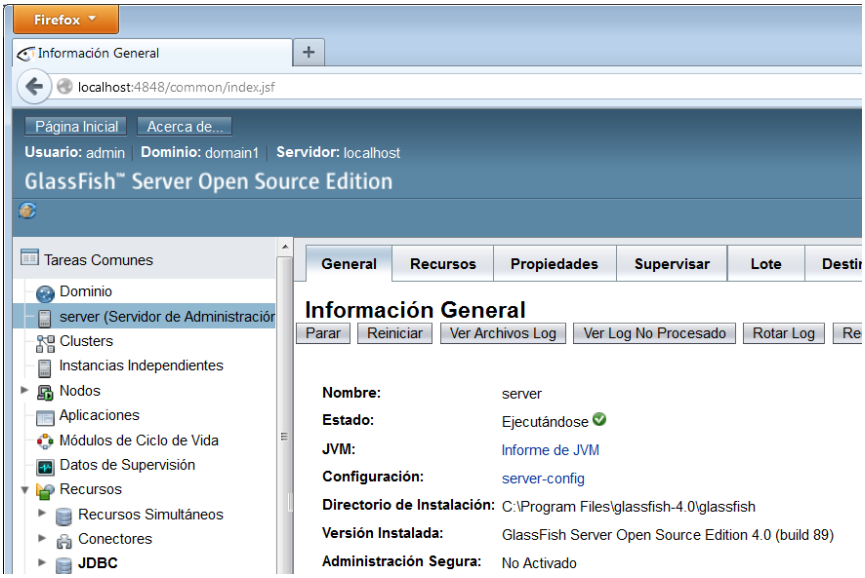
Compile la aplicación y ejecútela.

CONFIGURACIÓN DE GLASSFISH

Cuando quiera ver el archivo *log* del servidor de aplicaciones *GlassFish* o quiera configurar este servidor, por ejemplo, porque tiene que crear un *pool* de conexiones o un recurso de JDBC, inicie el servidor (puede hacerlo desde el panel *Services* de *NetBeans*) y abra el explorador web para acceder al administrador de *GlassFish* por medio del URL:

```
localhost:4848
```

La siguiente figura muestra el aspecto del administrador de configuración de *GlassFish*:



Cuando realizamos una aplicación JEE, como la expuesta en el capítulo *Java EE*, se crea un fichero de configuración, *persistence.xml*, y un fichero de recursos, *glassfish-resources.xml*, que informa al servidor de los datos necesarios para acceder a la base de datos. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish ..."
"http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <jdbc-connection-pool ...
     name="mysql_bd_telefonos_userPool"
     res-type="javax.sql.DataSource"
   ...>
   <property name="serverName" value="localhost"/>
   <property name="portNumber" value="3306"/>
   <property name="databaseName" value="bd_telefonos"/>
```

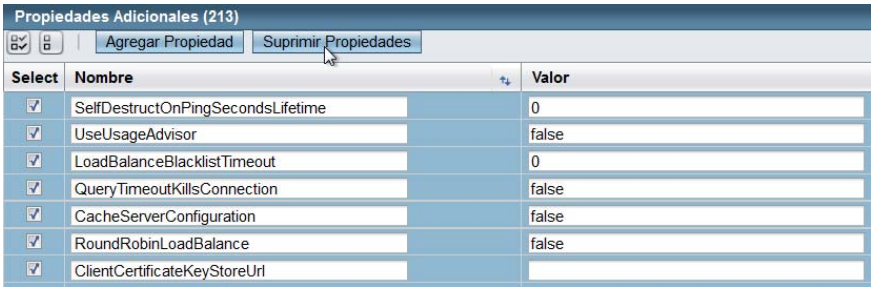
```
    <property name="User" value="user"/>
    <property name="Password" value="password"/>
    <property name="URL" value="jdbc:mysql://localhost:3306/bd_telefonos"/>
    <property name="driverClass" value="com.mysql.jdbc.Driver"/>
  </jdbc-connection-pool>
  <jdbc-resource enabled="true"
     jndi-name="jdbc/bd_telefonos" object-type="user"
     pool-name="mysql_bd_telefonos_userPool"/>
</resources>
```

Este fichero fue creado por *NetBeans* cuando obtuvimos las entidades correspondientes al modelo relacional de la base de datos *bd_telefonos*, pero *GlassFish* no fue configurado de forma automática con esos datos. Piense que la aplicación, cuando necesite acceder a la base de datos lo va a hacer solicitando al servidor *GlassFish* el recurso *jdbc/bd_telefonos* que está vinculado con la conexión *mysql_bd_telefonos_userPool* del *pool*. Por lo tanto, lo que tenemos que hacer es dar de alta el recurso JDBC y la conexión en el *pool* de conexiones. La figura siguiente muestra los enlaces a los que tiene que acceder:

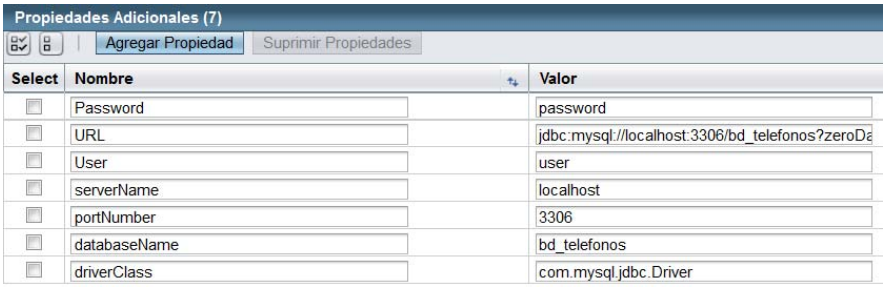
Para crear un *pool* de conexiones para que nuestra aplicación pueda obtener una conexión para acceder a la base de datos, haga clic en *Pools de conexiones JDBC* > *Nuevo*:

Haga clic en *Siguiente*.

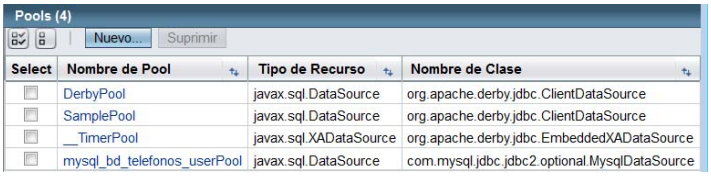
Continúe en esta página estableciendo los valores de las propiedades que aparecían en *glassfish-resources.xml*. Quizás lo más sencillo sea borrar todas las propiedades que aparecen:



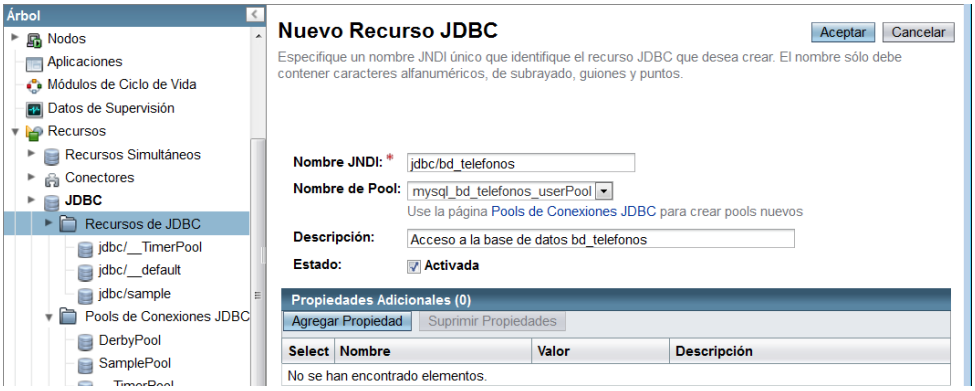
Y a continuación agregar las propiedades que especifica *glassfish-resources.xml*, según muestra la figura siguiente:



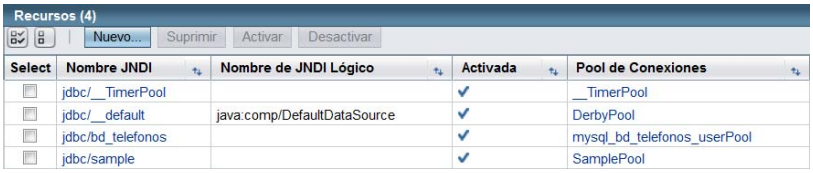
Haga clic en *Finalizar*. Observará el nombre del *pool* de conexiones añadido. Si hace clic en él, puede hacer un *ping* (clic en el botón correspondiente) y probar que tiene acceso a la base de datos.



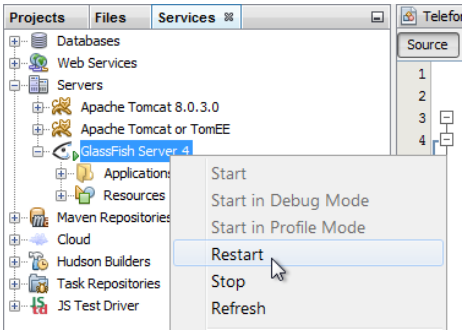
Para dar de alta el recurso de JDBC, haga clic en *Recursos de JDBC* > *Nuevo*. Después escriba el nombre JNDI y seleccione la conexión que acaba de configurar:



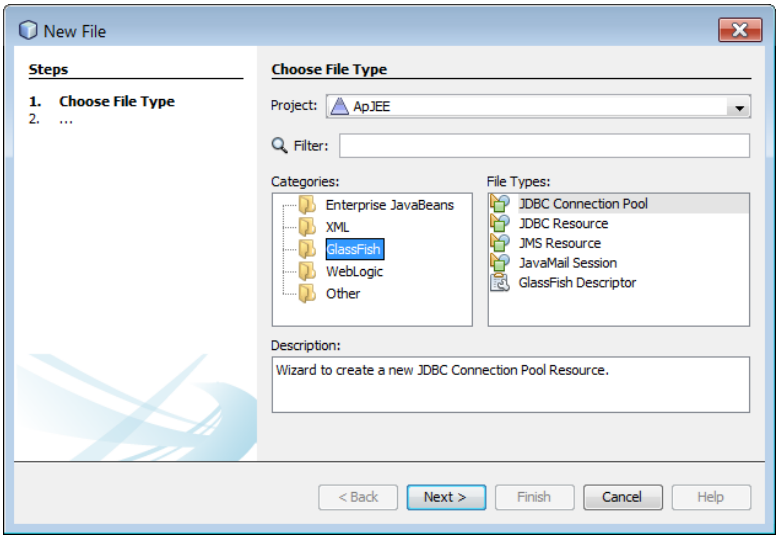
Para finalizar, haga clic en *Aceptar*. Podrá observar que el recurso ha sido añadido.



Puede ahora reiniciar el servidor, bien desde el administrador, o bien desde *NetBeans*:



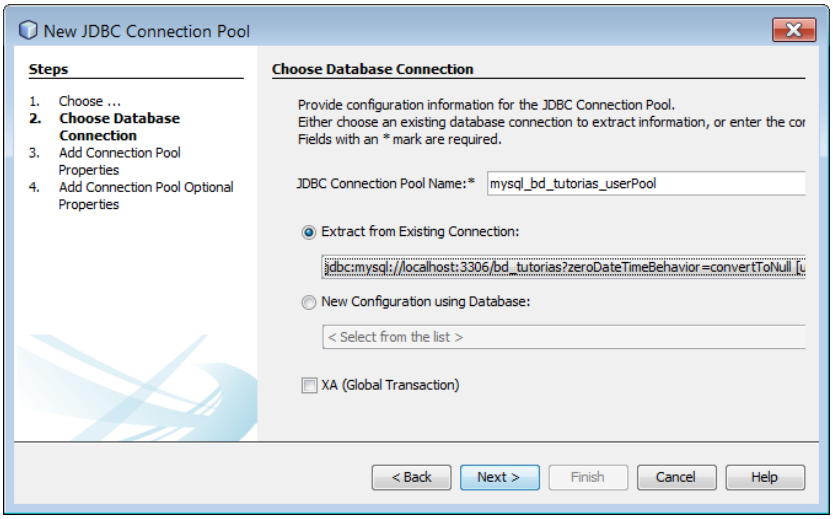
El proceso que acabamos de describir podría también realizarse desde *NetBeans*. Observe en la figura siguiente la categoría *GlassFish* y los tipos de fichero *JDBC Connection Pool* y *JDBC Resource*.



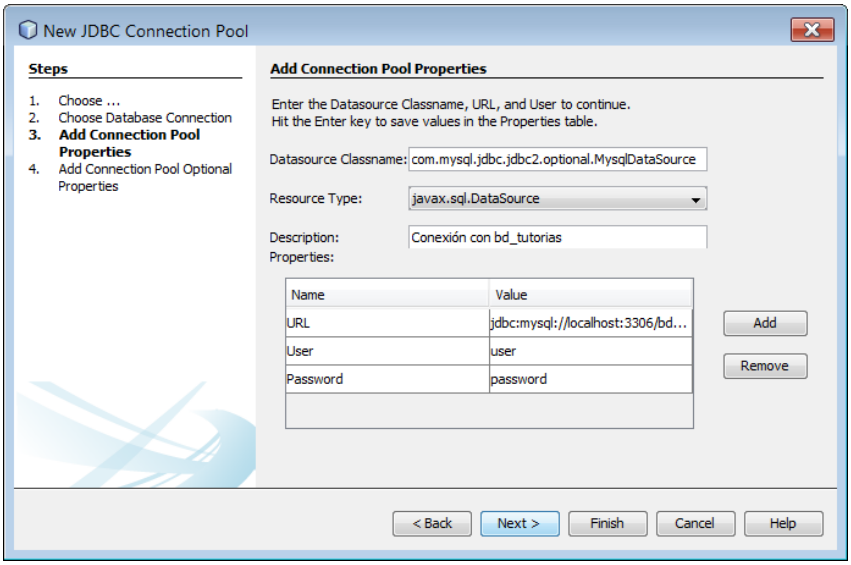
Observaremos que se crea un fichero *sun-resources.xml* en nuestro proyecto. El *JDBC Connection Pool* y el *JDBC Resource* creados desde *NetBeans* se aplicarán al servidor *GlassFish* únicamente cuando se despliegue el proyecto. Ahora bien, cuando la configuración se hace desde *NetBeans*, es solo con fines de desa-

rrollo. Lógicamente, cuando despleguemos la aplicación en un servidor real, tendremos que ir a ese servidor y hacer la configuración nosotros mismos según hemos explicado anteriormente.

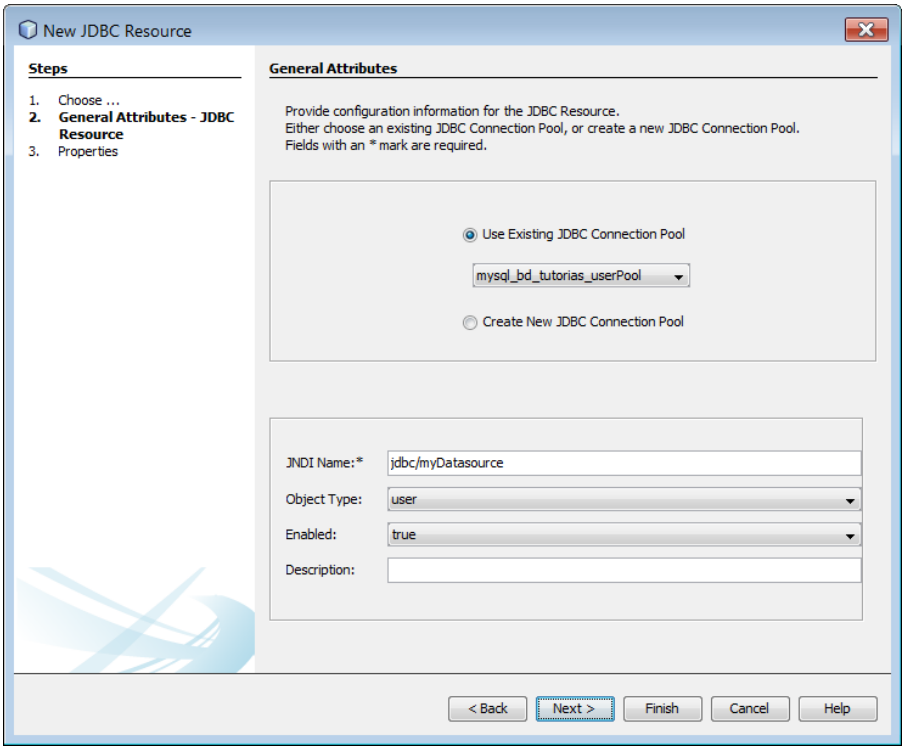
Entonces, para crear un *pool* de conexiones para que una aplicación pueda obtener una conexión para acceder a una base de datos, desde *NetBeans*, haga clic con el botón secundario del ratón sobre el nombre del proyecto > *New* > *Other* > *GlassFish* > *JDBC Connection Pool*:



Haga clic en *Next* para establecer el usuario y la contraseña de acceso a la base de datos. El resto de los datos, generalmente, son los que aparecen.



A continuación establecemos el recurso de JDBC; esto es, el nombre JNDI que apunta al *pool* de conexiones: clic con el botón secundario del ratón sobre el nombre del proyecto > *New* > *Other* > *GlassFish* > *JDBC Resource*:

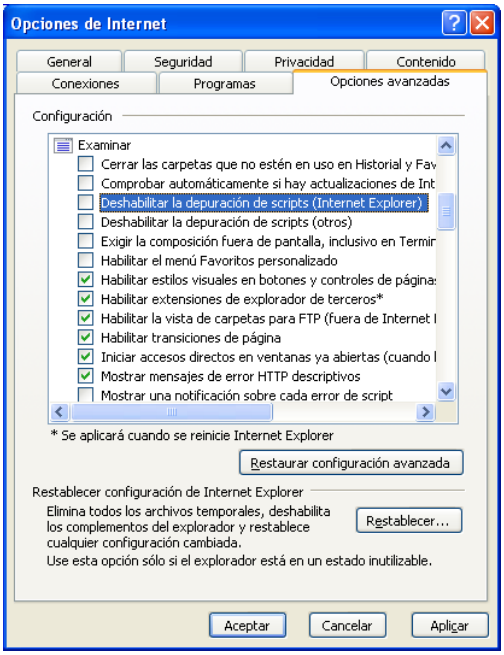


Compile y despliegue la aplicación. Si ahora abre el administrador de *GlassFish* observará en el nodo JDBC el *pool* de conexiones y el recurso añadidos.

DEPURAR CÓDIGO JAVASCRIPT

En una aplicación web Java que incluya código JavaScript (véase el capítulo 13) es posible utilizar el depurador de Microsoft Visual Studio 2005, o algún otro que lo soporte, para depurar ese código.

Para poder depurar código JavaScript, el primer paso es habilitar esta opción en el navegador. En el caso de Internet Explorer, seleccione *Herramientas* > *Opciones de Internet* > *Opciones avanzadas* y asegúrese de que no esté seleccionada la opción *Deshabilitar la depuración de scripts*:

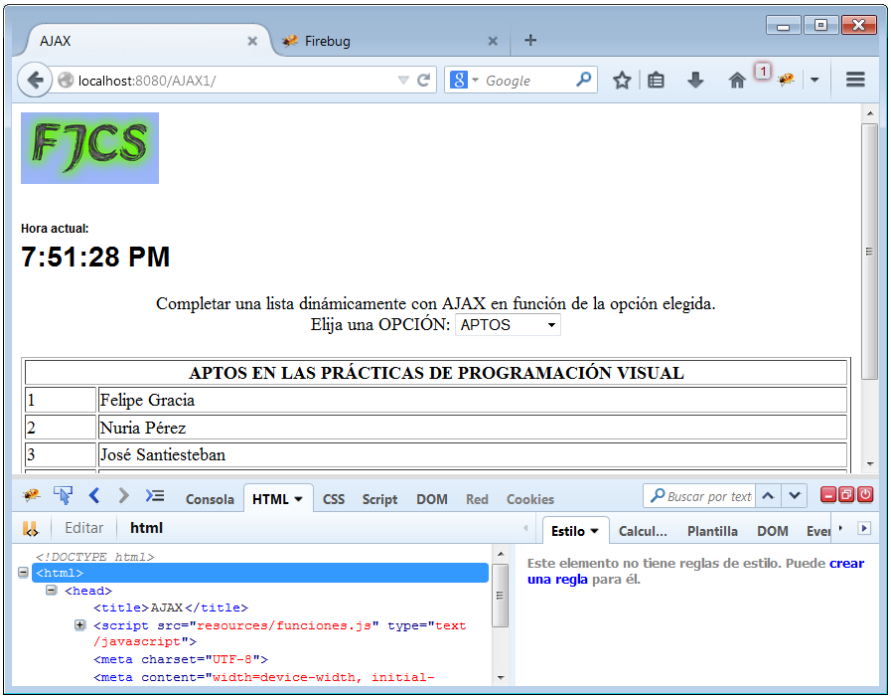


Cumplido el requisito anterior, solo queda poner en el código JavaScript a depurar la sentencia **debugger**. De esta forma, cuando inicie la depuración de la aplicación y el flujo de ejecución pase por esta sentencia, la ejecución se detendrá y podrá continuarla paso a paso.

```
function CargarTabla(resultado, contexto)
{
  debugger;
  var elementos =
     LoadXmlFromString(resultado).getElementsByTagName("string");
```

```
TablaResultados = document.getElementById("TablaDeResultados");
if (TablaResultados != null)
{
  // ...
```

Puede también echar una ojeada a este complemento del navegador *Firefox*: *Firebug* (*http://getfirebug.com/*).



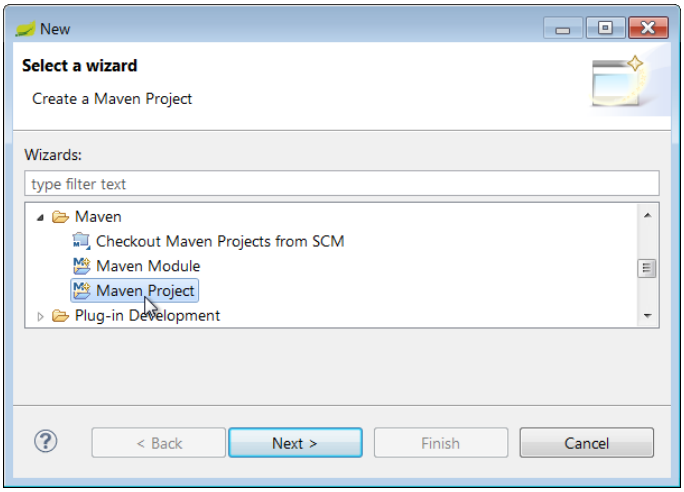
SPRING TOOL SUITE

En este apartado vamos a exponer los pasos básicos para realizar un proyecto utilizando el EDI *Spring Tool Suite* y *Maven* para la creación del proyecto. *Maven* es una herramienta para la gestión y construcción de proyectos Java.

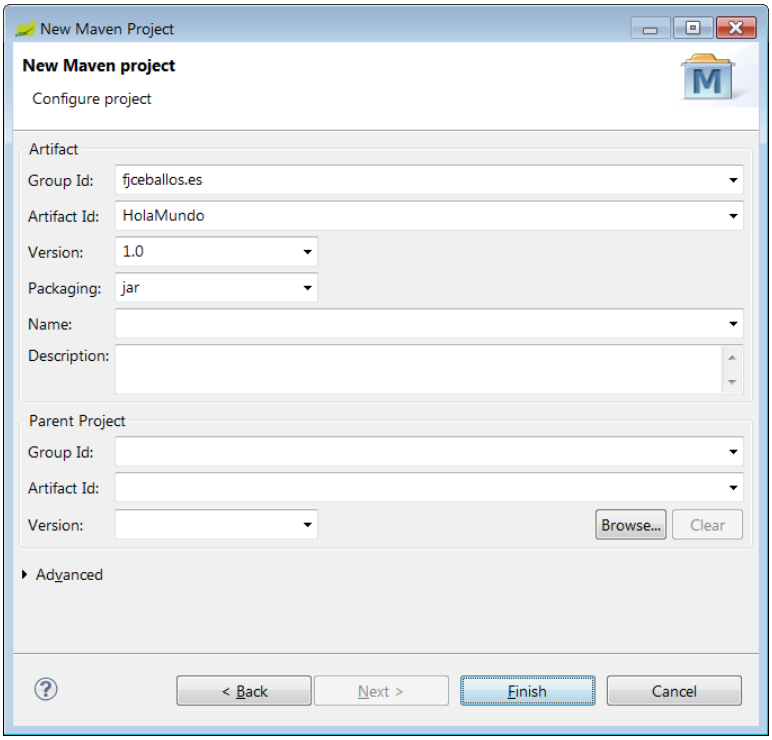
Para editar y ejecutar la aplicación realizada en el capítulo 1, "¡¡¡Hola, mundo!!!", utilizando este EDI, los pasos a seguir se indican a continuación:

1. Suponiendo que ya se está visualizando el entorno de desarrollo, ejecute la orden *File* > *New* > *Other*…

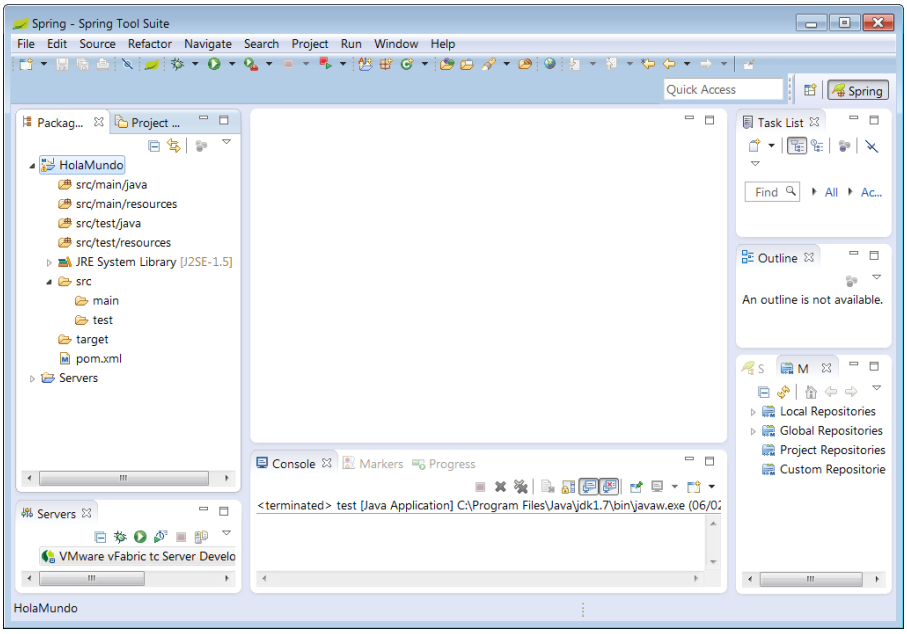
2. Se muestra la ventana *New*. Seleccione el asistente *Maven Project* y haga clic en el botón *Next*.



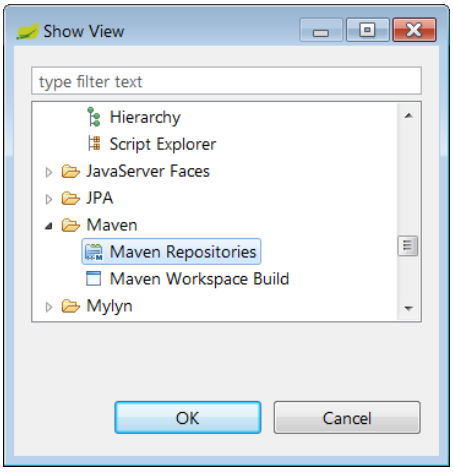
3. Se muestra la ventana *New Maven Project*. Marque la opción *Create a Simple Project* (crear un proyecto simple) y haga clic en el botón *Next*. Complete la ventana *New Maven Project* de forma análoga a como se muestra a continuación. Observe que se ha especificado el identificador de grupo, el nombre del proyecto, la versión y el tipo de empaquetado.



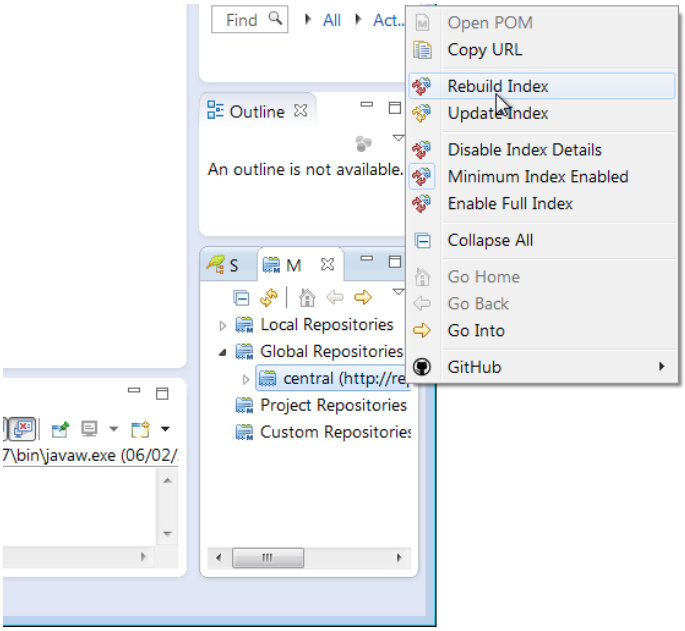
4. Para finalizar haga clic en el botón *Finish*. El resultado será el siguiente:



Es recomendable que tenga actualizado el repositorio *Maven*: menú *Window* > *Show View* > *Others*:

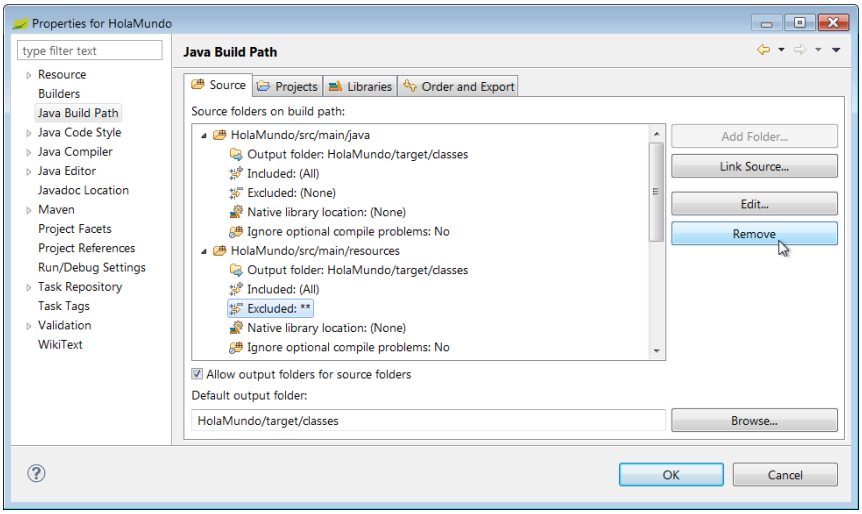


En la ventana *Show View*, seleccione *Maven Repositories* y, a continuación, proceda según muestra la figura siguiente (clic con el botón secundario del ratón sobre *Global Repositories* > *central…* > *Rebuild Index*):



Una vez terminada la actualización, tendremos actualizadas las bibliotecas de Spring listas para usar.

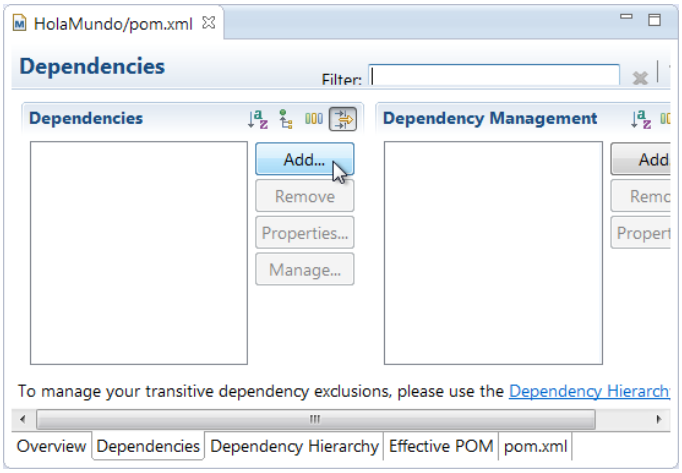
El siguiente paso será configurar el proyecto. Para ello haga clic con el botón secundario del ratón sobre el nombre del proyecto y seleccione la opción *Properties* del menú contextual que se visualiza. Se muestra la ventana de propiedades del proyecto:



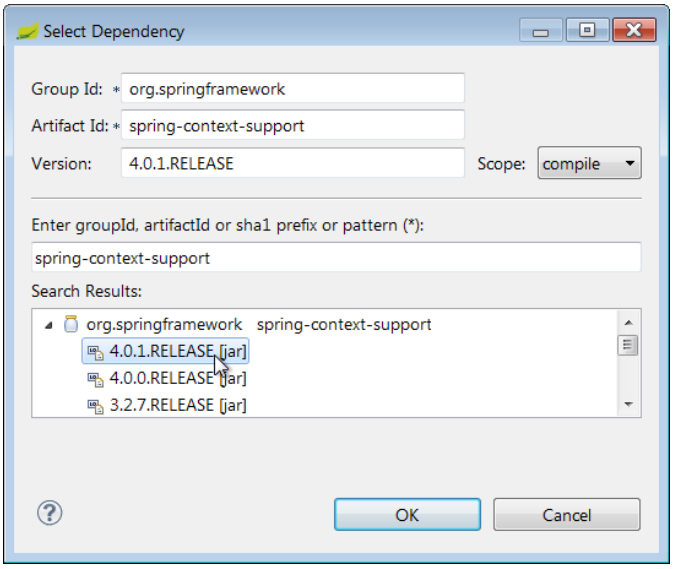
Seleccione el panel *Source* y no excluya ningún archivo en las carpetas *resources* del proyecto (proceda según muestra la figura anterior).

A continuación seleccionamos el panel *Libraries* para verificar que la biblioteca JRE de Java es la que queremos utilizar. Si quiere elegir otra versión, haga clic en el botón *Edit* y selecciónela.

El siguiente paso será agregar al proyecto las dependencias con respecto a los módulos de Spring que sean necesarios. Por ejemplo: *spring-core*, *spring-context-support*, *spring-test*, *junit*, etc. Para ello haga doble clic sobre el fichero *pom.xml* del proyecto, seleccione el panel *Dependencies* y haga clic en el botón *Add…*

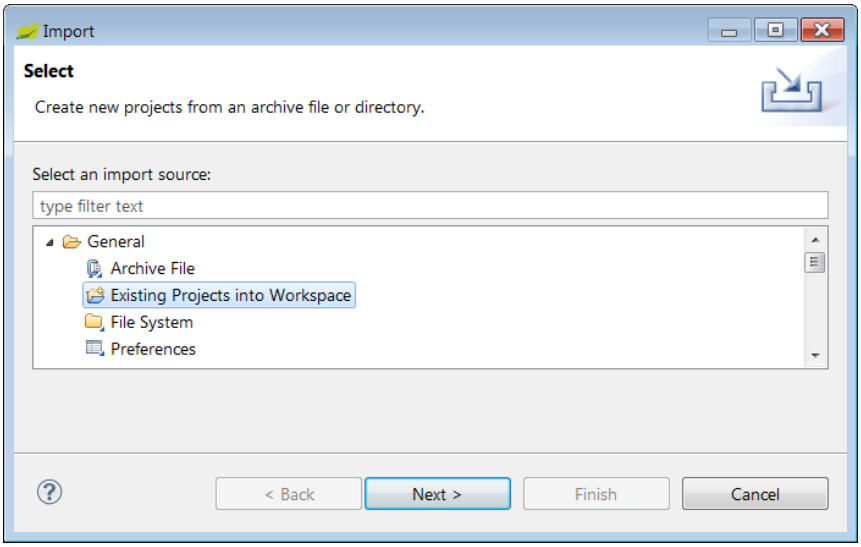


Después, en la ventana que se muestra, en la caja de texto *Enter groupId, artifactId or sha1 prefix or pattern (*):*, escriba, por ejemplo, el patrón relativo al módulo que desea buscar (en la figura, el módulo *spring-context-support*, el único necesario para este proyecto) y selecciónelo:



Repita los mismos pasos para añadir el resto de las dependencias que necesite en un determinado proyecto. Por ejemplo, para *junit*:

Guarde los cambios: *File* > *Save* o *Save All*. Observe el explorador de proyectos o de paquetes (*Project* o *Package Explorer*); ya tenemos un proyecto vacío configurado para crear una aplicación utilizando Spring. Puede cerrar y abrir de nuevo el proyecto desde el menú *Project*. Puede también cerrar el proyecto, eliminarlo del explorador (no del sistema de ficheros) y cambiarlo a otra carpeta, simplemente arrastrando la carpeta del proyecto a la nueva ubicación. En este caso, para recuperarlo, cambie al nuevo espacio de trabajo (*File* > *Switch Workspace* > *Other…*) y ejecute la orden *File* > *Import*. Se muestra la ventana siguiente:



Seleccione como origen *Existing Projects into Workspace* y después la carpeta raíz del proyecto.

Una vez creado el proyecto, editamos el código correspondiente a la aplicación. En nuestro caso vamos a añadir la interfaz *ServicioDeMensajes* y la clase *ImpresoraDeMensajes* en el paquete *beans*, y la clase *Aplicacion* en el paquete *holamundo*.

Spring incluye una serie de módulos entre los cuales se encuentra el que vamos a utilizar aquí: *spring-context*. Entonces, si el proyecto ya tiene establecida la dependencia con este módulo, podemos pasar a escribir el siguiente código.

beans/ServicioDeMensajes.java

```
package beans;

public interface ServicioDeMensajes
{
  String obtenerMensage();
}
```

beans/ImpresoraDeMensajes.java

```
package beans;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component  // Spring registra la clase como bean de la aplicación
public class ImpresoraDeMensajes
{
```

```
  @Autowired // decimos a Spring que inyecte una dependencia
  private ServicioDeMensajes servicio;

  public void escribirMensaje()
  {
    System.out.println(this.servicio.obtenerMensage());
  }
}
```

holamundo/Aplicacion.java

```
package holamundo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.*;

import beans.ImpresoraDeMensajes;
import beans.ServicioDeMensajes;

@Configuration // la clase anotada es de configuración de Spring
@ComponentScan("beans") // buscar componentes en paquetes
public class Aplicacion
{
  @Bean // indica que un método produce una definición de bean
  ServicioDeMensajes mockServicioDeMensajes()
  {
    return new ServicioDeMensajes()
    {
      public String obtenerMensage()
      {
        return "Hola Mundo!!!";
      }
    };
  }

  public static void main(String[] args)
  {
    ApplicationContext contexto =
      new AnnotationConfigApplicationContext(Aplicacion.class);
    ImpresoraDeMensajes impresora =
      contexto.getBean(ImpresoraDeMensajes.class);
    impresora.escribirMensaje();
  }
}
```

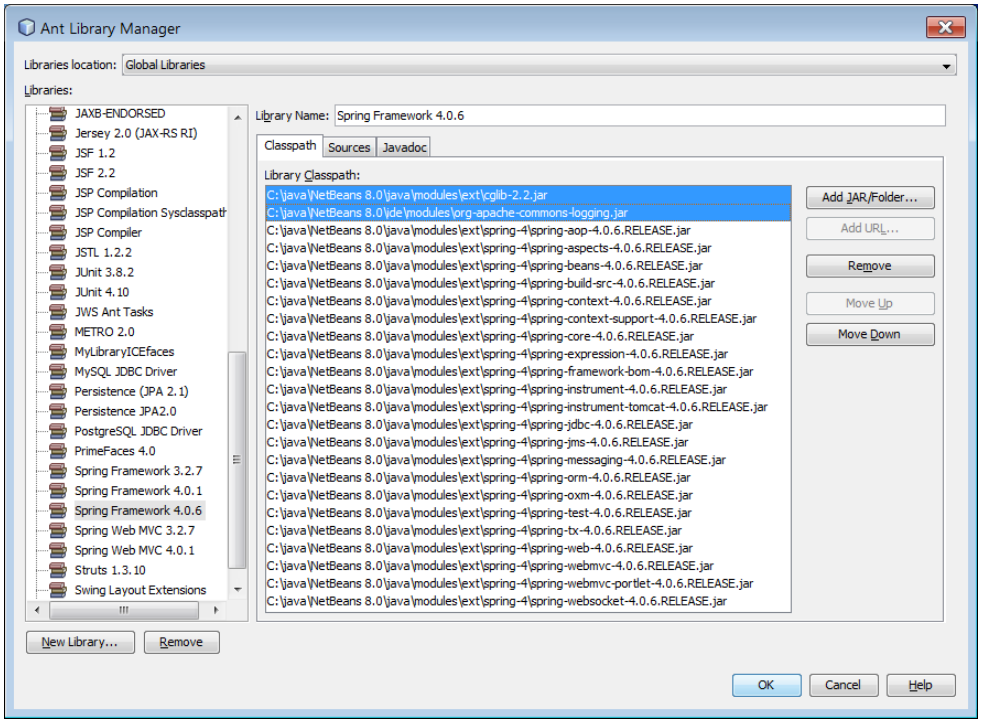
El ejemplo anterior muestra el concepto básico de la inyección de dependencia. Observamos que la clase *ImpresoraDeMensajes* está desacoplada de la implementación *ServicioDeMensajes*, para lo cual, utilizando las anotaciones de Spring Framework, se inyecta una dependencia.

Como ejercicio, puede realizar el mismo proyecto, pero ahora utilizando *NetBeans*. Cree un nuevo proyecto *Java Application*, añada al proyecto la biblioteca Spring y después añada la interfaz y las clases escritas anteriormente.

Si dispone de una biblioteca Spring más reciente que la proporcionada por *NetBeans* y quiere utilizarla, tendrá que incorporarla al repositorio de *NetBeans*. Para ello descargue la biblioteca desde esta dirección:

```
http://repo.spring.io/release/org/springframework/spring/
```

y ejecute *Tools* > *Libraries* > *New Library…* (por ejemplo, *spring-framework-4.0.1.RELEASE*). Después añada los ficheros *.jar* según muestra la figura siguiente. Observe que no hemos añadido ni los *sources* ni los *javadoc*.



Si observa las otras bibliotecas de Spring que ya tiene añadidas *NetBeans*, verá que tiene que añadir dos *.jar* más: los dos primeros del listado.

APÉNDICE B

INSTALACIÓN DEL SOFTWARE

En los capítulos de este libro, vamos a necesitar utilizar distintos paquetes de software para poder implementar y probar las aplicaciones que en ellos se explican; por ejemplo, el gestor de bases de datos (*PostgreSQL* o *MySQL*), el entorno de desarrollo (*NetBeans* o *Eclipse*) o el servidor de aplicaciones (*Apache Tomcat* o *GlassFish Server*). Este apéndice le indicará de forma breve cómo instalar estos paquetes en una máquina Windows y cómo ponerlos en marcha.

JDK 8

Java Platform Standard Edition Development Kit 8 (abreviadamente, ***J**ava SE **D**evelopment **K**it 8* o *JDK 8*) proporciona la base para desarrollar y distribuir aplicaciones que se podrán ejecutar en un servidor o en un ordenador personal con distintos sistemas operativos. Actualiza las versiones anteriores e incluye nuevas características (desarrollo más fácil, más rápido y a menor coste, y ofrece mayor funcionalidad para servicios web, soporte de lenguajes dinámicos, diagnósticos, aplicaciones de escritorio, bases de datos, etc.), pero conservando la compatibilidad y la estabilidad.

Instalación

Para instalar el paquete de desarrollo JDK 8, siga los pasos indicados a continuación:

1. Descargue de Internet el paquete *jdk-8uxx-windows-i586.exe* y ejecútelo.

2. Siga los pasos indicados durante la instalación.

3. Si a continuación desea instalar la documentación, descargue de Internet y descomprima el fichero *jdk-8uxx-apidocs.zip*. Dicha documentación puede instalarla en *jdk1.8.0_xx\docs*.

NetBeans 8.x

NetBeans 8.x es un entorno de desarrollo integrado para desarrollar y distribuir aplicaciones multicapa distribuidas. Incluye un entorno gráfico de desarrollo de Java, una utilidad para desarrollar aplicaciones web, otra para desarrollar aplicaciones para dispositivos móviles y un servidor de aplicaciones (*GlassFish Server* y *Apache Tomcat*), así como una serie de herramientas que facilitan el desarrollo y la distribución de las aplicaciones.

Instalación

Para instalar el entorno de desarrollo *NetBeans*, siga los pasos indicados a continuación:

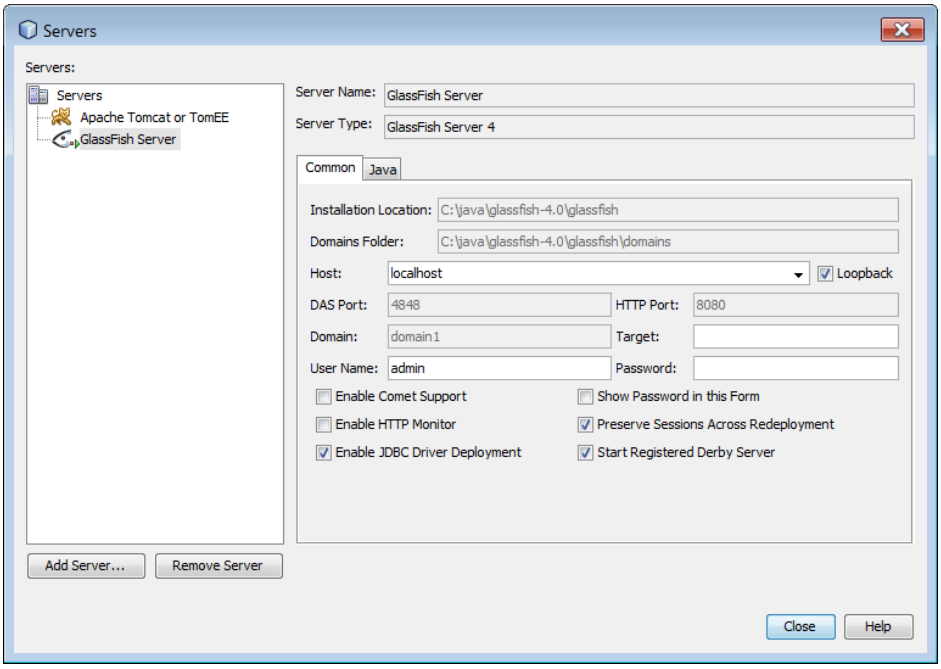
1. Descargue de *http://www.NetBeans.org* el fichero *netbeans-8.xxx-javaxx-windows.exe* y ejecútelo.

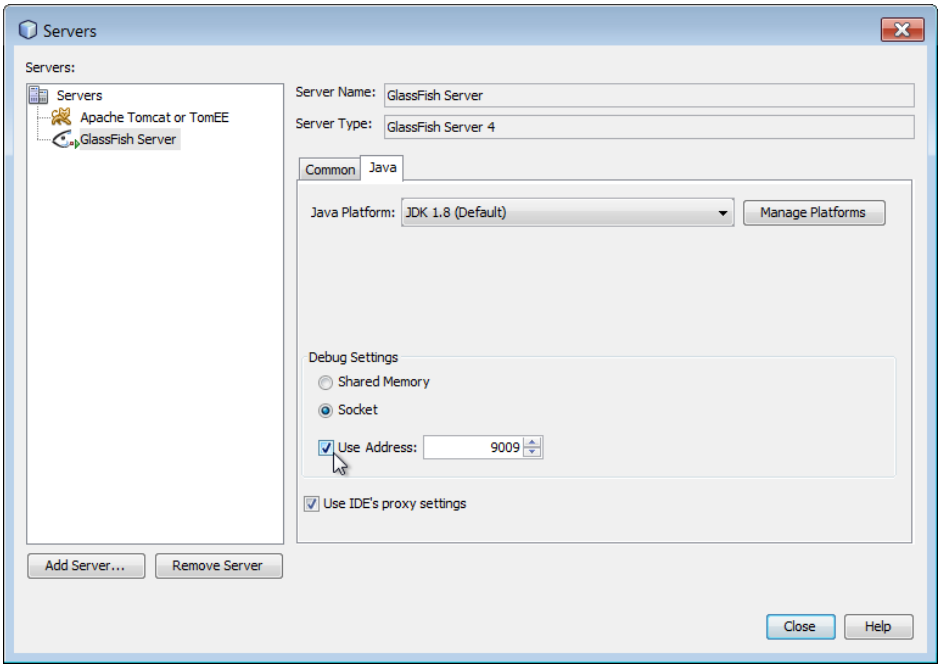
2. Puede instalar uno de los dos o los dos servidores: *GlassFish Server* y *Apache Tomcat*. El servidor de aplicaciones *GlassFish Server* es más completo que *Apache Tomcat*.

3. Para obtener ayuda acerca de la biblioteca de Java SE 8, solo si instaló dicha ayuda, es preciso que dicho entorno tenga conocimiento de la ruta de acceso a la misma. Para ello ejecute la orden *Java Platform* del menú *Tools* y asegúrese de que en la lista de rutas mostrada en *Javadoc* hay una que hace referencia a la carpeta donde se encuentra la ayuda mencionada; si no es así, haga clic en el botón *Add Folder* para añadirla.

Arrancar GlassFish en modo debug

Para arrancar *GlassFish* en modo debug, observe que la configuración es la que muestran las figuras siguientes; en especial, en la opción *Use address*: actívela y especifique el puerto 9009.


Servers
Servers:
Servers
Apache Tomcat or TomEE
GlassFish Server
Server Name: GlassFish Server
Server Type: GlassFish Server 4
Common
Java
Installation Location: C:\java\glassfish-4.0\glassfish
Domains Folder: C:\java\glassfish-4.0\glassfish\domains
Host: localhost
Loopback
DAS Port: 4848
HTTP Port: 8080
Domain: domain1
Target:
User Name: admin
Password:
Enable Comet Support
Show Password in this Form
Enable HTTP Monitor
Preserve Sessions Across Redeployment
Enable JDBC Driver Deployment
Start Registered Derby Server
Add Server...
Remove Server
Close
Help


Servers
Servers:
Servers
Apache Tomcat or TomEE
GlassFish Server
Server Name: GlassFish Server
Server Type: GlassFish Server 4
Common
Java
Java Platform: JDK 1.8 (Default)
Manage Platforms
Debug Settings
Shared Memory
Socket
Use Address: 9009
Use IDE's proxy settings
Add Server...
Remove Server
Close
Help

GESTOR DE BASES DE DATOS MySQL

MySQL es un gestor de bases de datos relacionales con licencia pública GNU; esto es, se trata de un software de libre distribución del cual puede obtener una versión actualizada de la dirección de Internet *http://dev.mysql.com/*.

Instalación

Para instalar el gestor de bases de datos *MySQL*, descárguelo del sitio web *http://dev.mysql.com/* (fichero *mysql-installer-web-community-5.6.xx.x.msi*):

1. Ejecute el fichero *.msi*.

2. Siga los pasos indicados durante la instalación. Las opciones propuestas son las adecuadas en la mayoría de los casos. Cuando finalice la instalación se ejecutará el asistente de configuración. Éste, entre otras cosas, le permitirá dar de alta un usuario anónimo y establecer la contraseña para el usuario *root*.

3. Diríjase a la carpeta donde ha instalado *MySQL*, edite el fichero ***my.ini*** y verifique que en la sección [mysqld] se han establecido las líneas indicadas a continuación. Como se puede observar, especifican las rutas de la carpeta donde ha instalado el paquete y de la carpeta donde se van a almacenar los datos (los separadores tienen que ser / o \\):

```
[mysqld]
basedir="C:/Archivos de programa/MySQL/MySQL Server 5.x/"
datadir="C:/Archivos de programa/MySQL/MySQL Server 5.x/Data/"
```

Una vez finalizada la instalación, es necesario instalar también el controlador para poder conectarse a *MySQL* vía *JDBC*, solo si no fue instalado durante la instalación de *MySQL*. Para ello:

1. Descargue el fichero *mysql-connector-java-5.x.xx.zip* y descomprímalo. Se obtendrá el fichero *mysql-connector-java-5.x.xx-bin.jar* (este fichero, seguramente, lo podrá obtener en la instalación de *MySQL* que acaba de realizar).

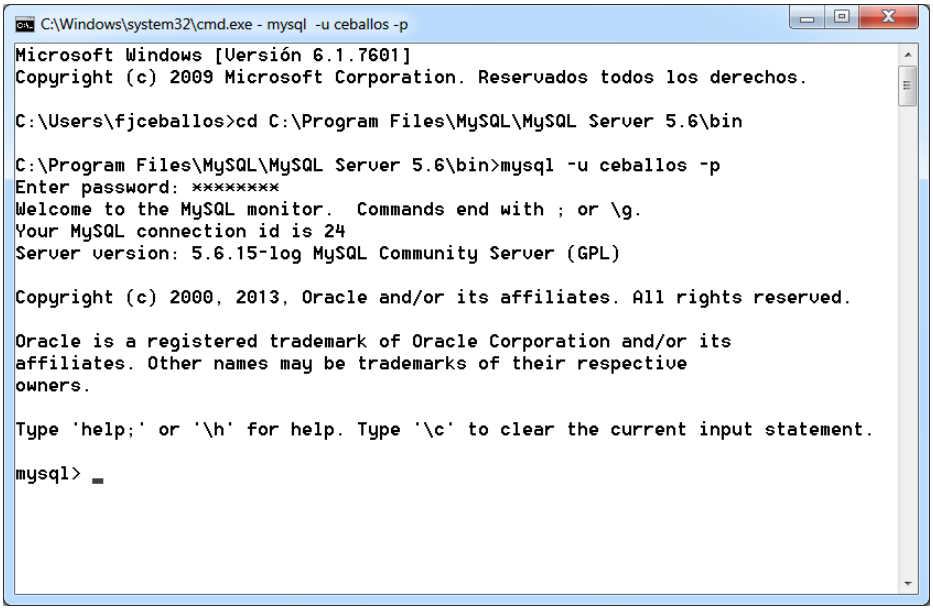
2. Después copie el fichero *mysql-connector-java-5.x.xx-bin.jar* en la carpeta *…/jre/lib/ext* (vea el ejemplo a continuación) de la instalación de Java y, cuando lo necesite, establezca la variable *classpath* para que incluya esa ruta. Por ejemplo:

```
set classpath=.;C:\Java\jdk1.8.x_xx\jre\lib\ext\mysql-connector-java-5.1.x-bin.jar
```

Poner en marcha MySQL en Windows

Normalmente, *MySQL* en Windows se instala como un servicio. Esta operación fue realizada, por omisión, por el asistente para la configuración de *MySQL* que se ejecutó automáticamente al finalizar la instalación. Puede comprobar la existencia de este servicio ejecutando *Panel de control* > *Sistema y seguridad* > *Herramientas administrativas* > *Servicios*. Si no está iniciado, puede iniciarlo desde esta ventana.

Una vez iniciado el servicio, puede arrancar el *monitor MySQL* ejecutando la orden *mysql* desde una consola del sistema (los usuarios de Windows pueden abrir una consola ejecutando *cmd* o *command* desde la ventana *Inicio – Ejecutar*).



Esta orden inicia el *monitor de MySQL* con el que podrá realizar cualquier operación permitida sobre una base de datos, ejecutando la sentencia SQL apropiada. El usuario, por omisión, es anónimo y es el que menos privilegios tiene.

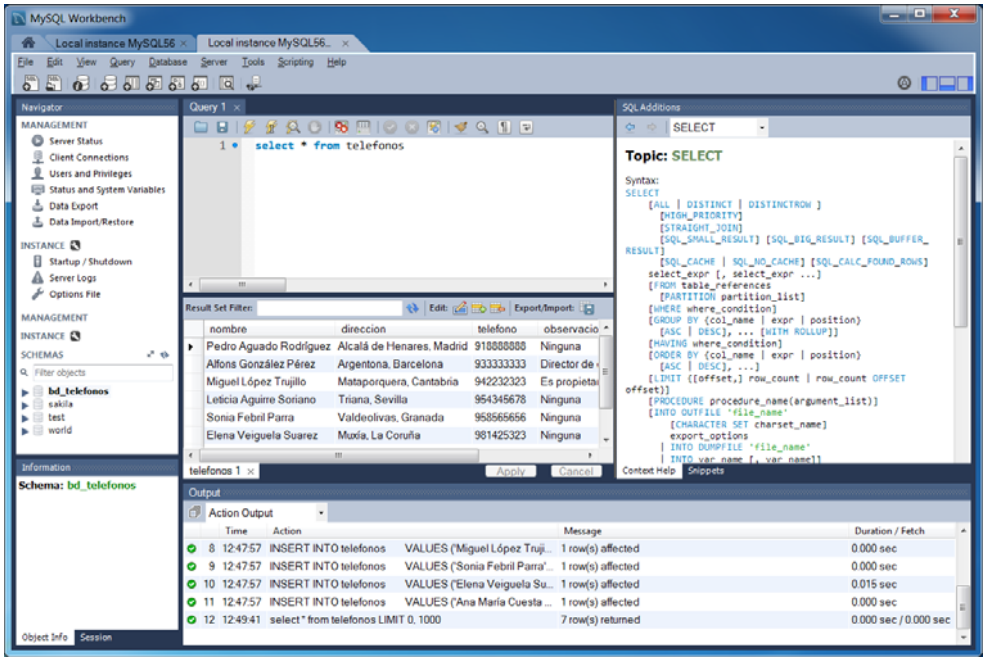
Otro usuario es *root*; este tiene todos los privilegios. Para arrancar *MySQL* bajo este usuario, ejecute la orden siguiente: *mysql -u root -p*. La opción *-p* hace que le sea solicitada la contraseña.

Para finalizar el monitor, ejecute la orden *quit*.

UTILIDADES DE MySQL

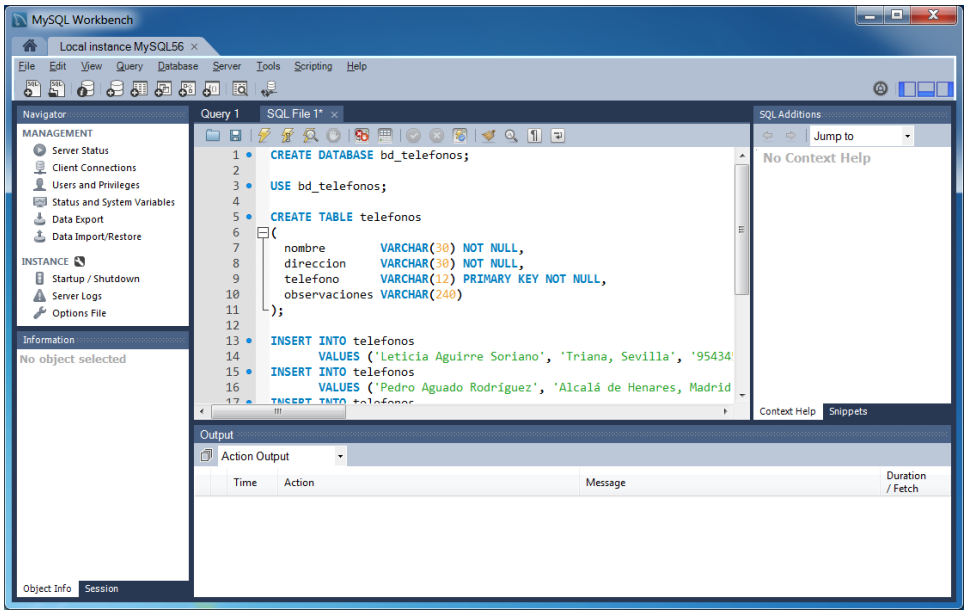
Para administrar tanto *MySQL* como las bases de datos, puede utilizar la herramienta con interfaz gráfica *MySQL Workbench*. Se trata de un software de libre distribución (se instala cuando instala *MySQL*) para ejecutar operaciones de administración, tales como configuración del servidor *MySQL*, supervisión de su estado y rendimiento, arranque y parada del mismo, gestión de usuarios y conexiones, realización de copias de seguridad y un buen número de otras tareas administrativas.

Incluye también un analizador de consultas diseñado para permitir la ejecución de *scripts SQL*, así como para facilitar las consultas y el análisis de los datos almacenados en una base *MySQL*.



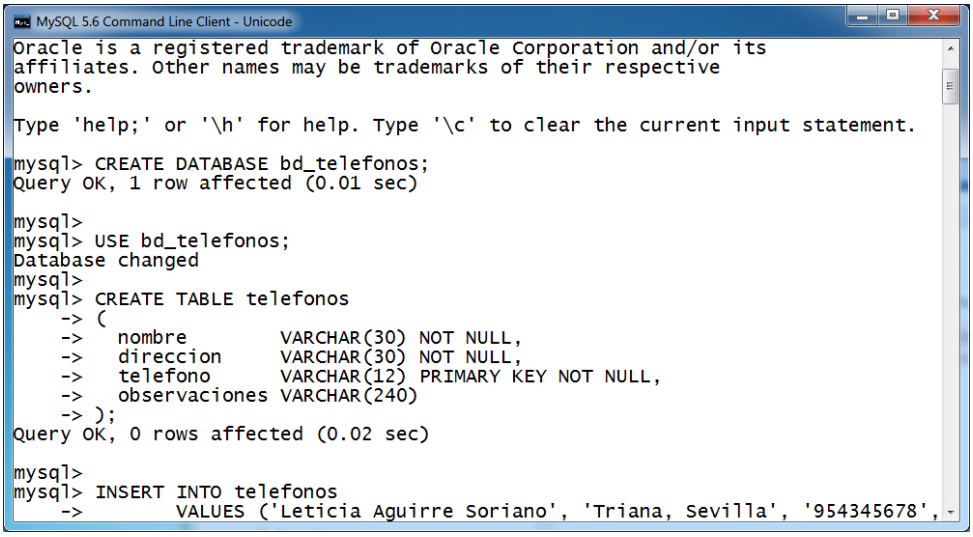
CREAR UNA BASE DE DATOS

Para crear una base de datos, podemos utilizar la herramienta *MySQL Workbench*, el *monitor MySQL* desde una consola del sistema, o bien el EDI *NetBeans*. En todos los casos podemos escribir un guión (*script*) que genere la base de datos y ejecutarlo.

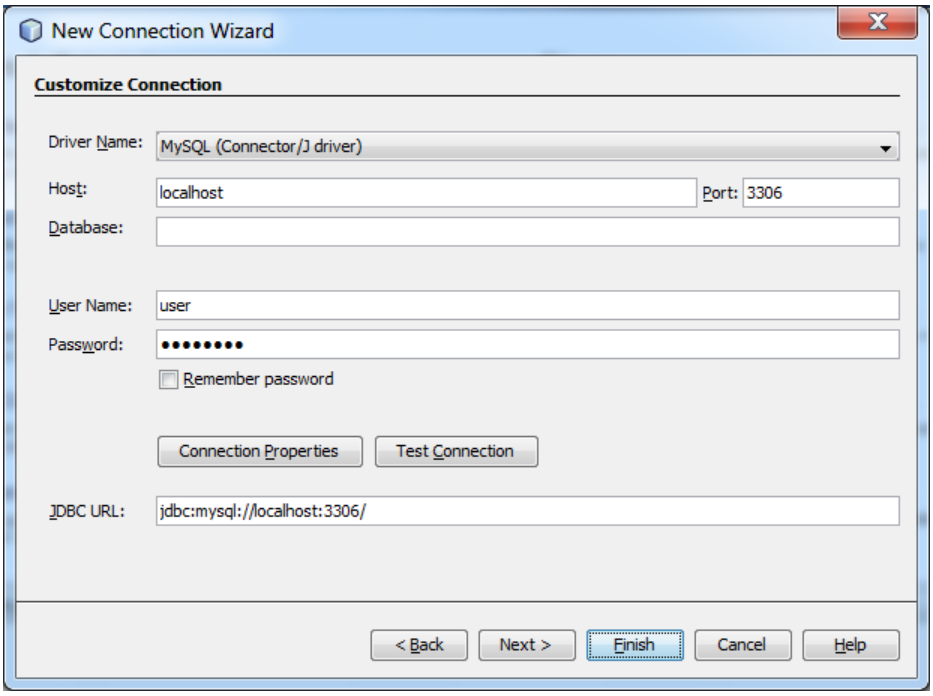


El guión que muestra la figura anterior crea la base de datos *bd_telefonos*. Dicho guión puede obtenerlo de la carpeta *Apen_B* del CD que acompaña al libro.

Si en lugar de emplear la utilidad *MySQL Workbench* utilizamos el *monitor MySQL* desde una consola del sistema, bastaría con copiar el guión en el portapapeles de nuestro sistema y pegarlo en la consola a continuación del símbolo “>”.

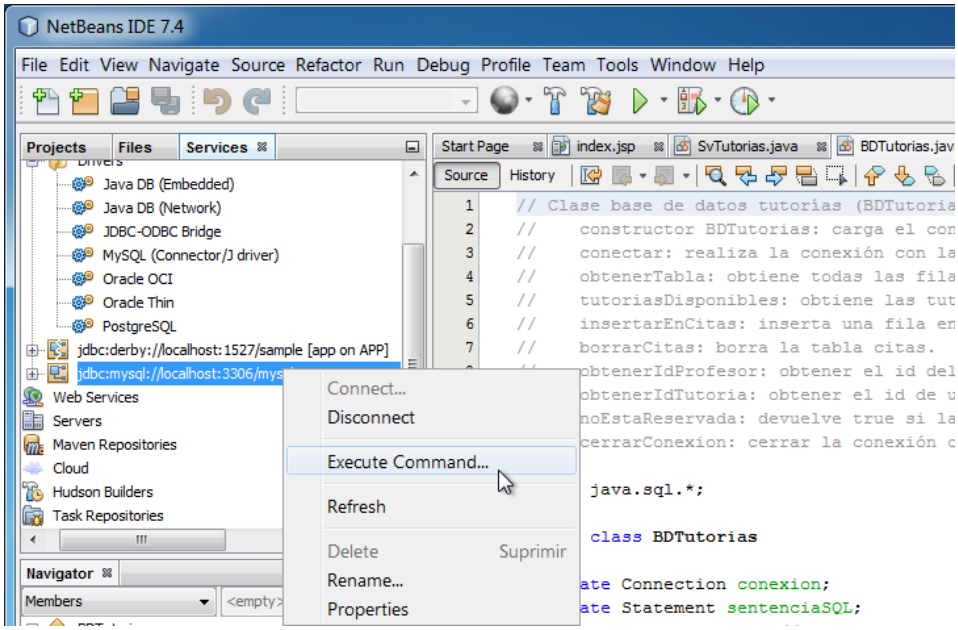


Si utilizamos el EDI *NetBeans 8.x* procederíamos según se expone a continuación. Mostramos el panel *Services* y expandimos el nodo *Databases* > *Drivers*, hacemos clic con el botón secundario del ratón en *MySQL (Connector/J driver)* y seleccionamos *Connect Using*. Se muestra el diálogo *New Connection Wizard*:

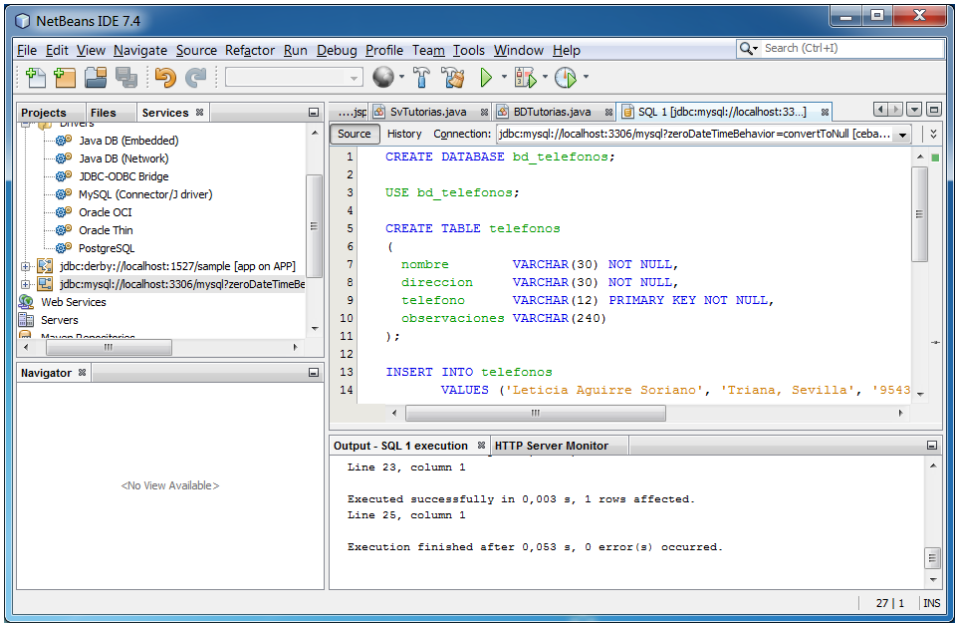


A continuación haga clic en el botón *Finish*. Esto generará una nueva conexión.

Haga clic con el botón secundario del ratón en la conexión *jdbc:mysql:…* y seleccione *Execute Command*:

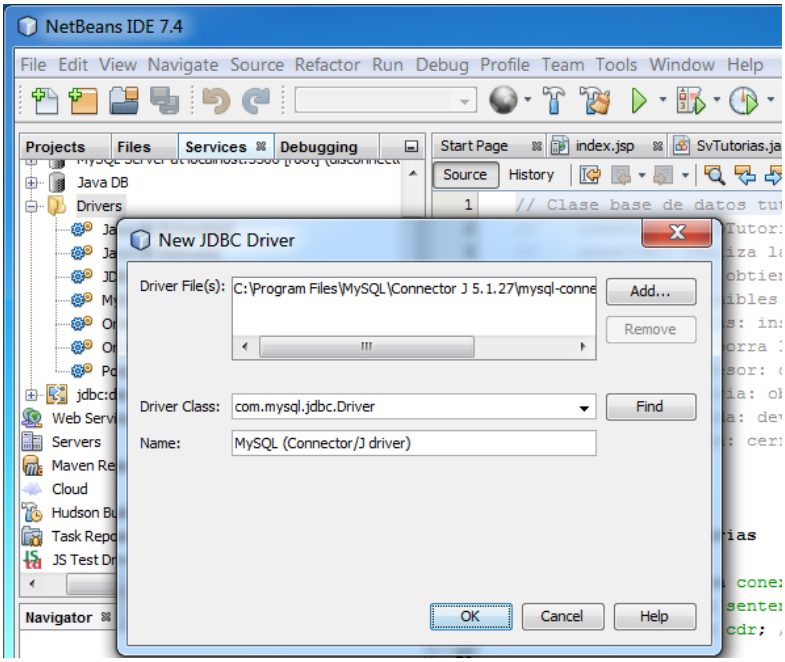


Se muestra el panel *SQL Command* donde podremos escribir sentencias SQL como *CREATE DATABASE nombre_bd* o cualquier guión (*script*) que genere la base de datos. Cuando abra de nuevo la conexión para la base de datos creada, podrá observar todos los datos (nombre de la base de datos, tablas, etc.).



UTILIZAR EL CONTROLADOR MySQL CON NetBeans

Para poder utilizar el explorador de bases de datos del EDI *NetBeans* con una base de datos *MySQL*, hay que poner a disposición del EDI el controlador *MySQL*, si aún no lo está. Para ello haga clic con el botón derecho del ratón en el nodo *Drivers* del panel *Services* (véase la figura siguiente) y ejecute la orden *New Driver* del menú contextual que se visualiza. Se muestra el diálogo *New JDBC Driver*. Haga clic en el botón *Add* y seleccione el controlador (fichero JAR o ZIP) de la carpeta en la que fue almacenado.



SQL SERVER EXPRESS

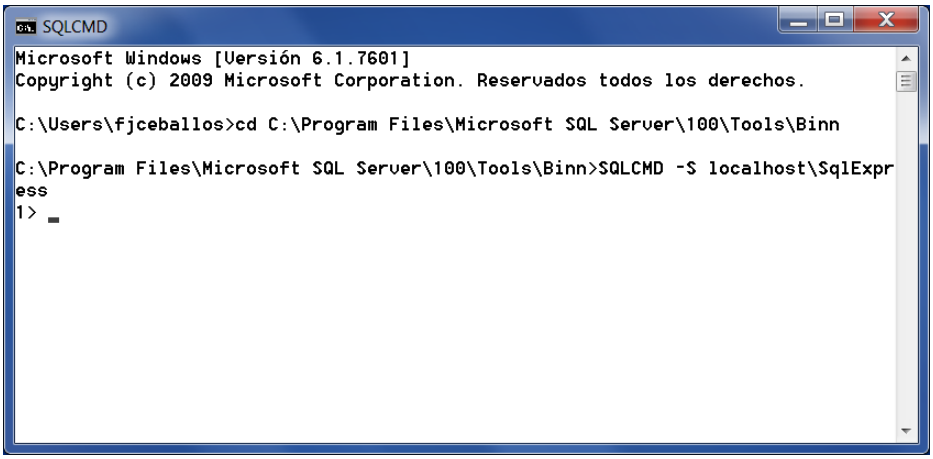
SQL Server Express es el motor de base de datos gratuito, potente, pero sencillo, que se integra perfectamente con el resto de productos Express. Se trata de una versión aligerada de la nueva generación de SQL Server.

Este producto tiene el mismo motor de base de datos que toda la familia SQL Server y utiliza el mismo lenguaje SQL.

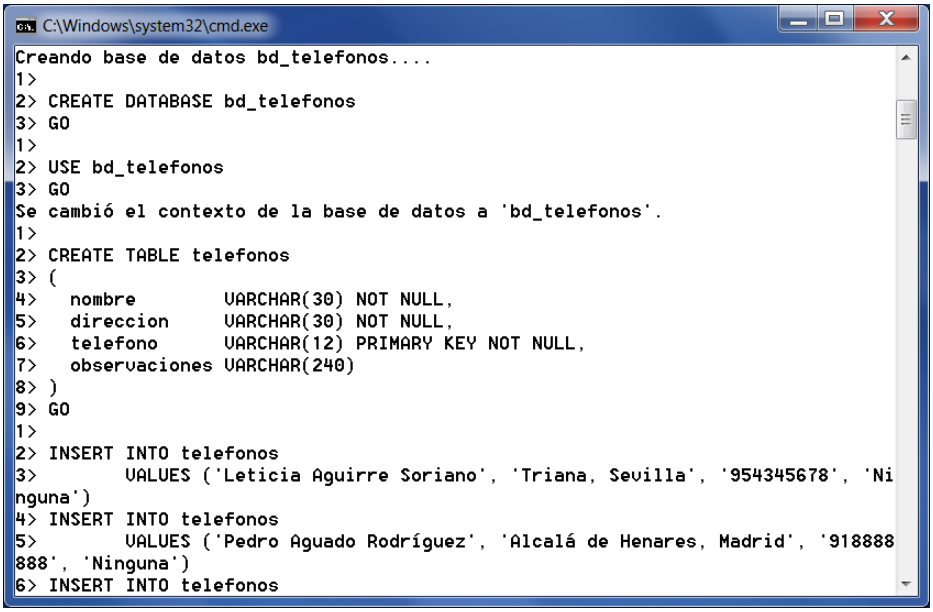
Para crear una base de datos utilizando SQL Server Express, tiene que hacerlo desde la línea de órdenes. Para iniciar la consola que le permita trabajar contra el motor de base de datos SQL Server, localice en su instalación (*C:\Archivos de*

programa\Microsoft SQL Server\110\Tools\Binn) el fichero SQLCMD.EXE, cambie a ese directorio y ejecute la orden:

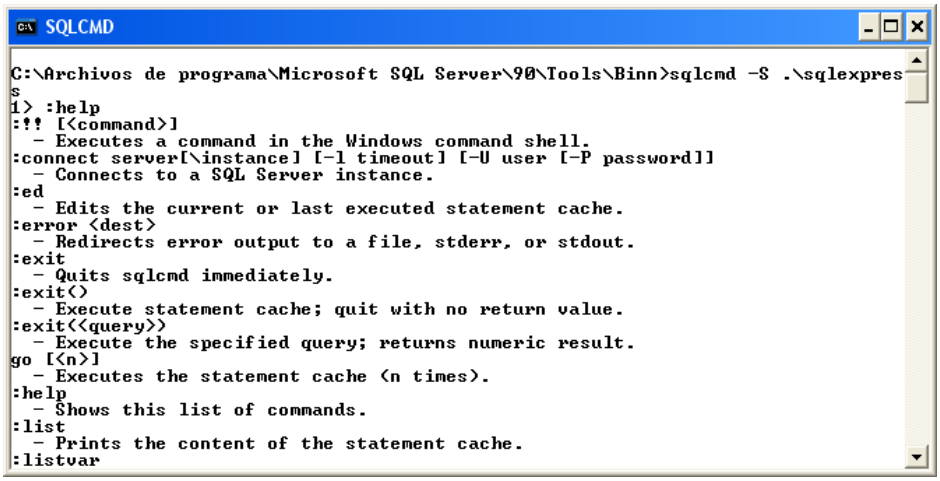
```
SQLCMD -S nombre-del-ordenador\SqlExpress
```



Una vez iniciada la consola, puede escribir órdenes SQL a continuación del símbolo ">". Para ejecutar un bloque de sentencias, escriba GO. Para salir, escriba QUIT. Por ejemplo, el guión que muestra la figura siguiente crea la base de datos *tfnos* con una tabla *telefonos*, añade tres filas a la tabla y, finalmente, selecciona todas las filas de la tabla con todas sus columnas:



Para ver la relación de órdenes que puede utilizar a través de la aplicación *SQLCMD*, ejecute la orden **help** como se muestra en la figura siguiente. Obsérvese que cada orden va precedida por dos puntos (:).



SQL SERVER MANAGEMENT STUDIO EXPRESS

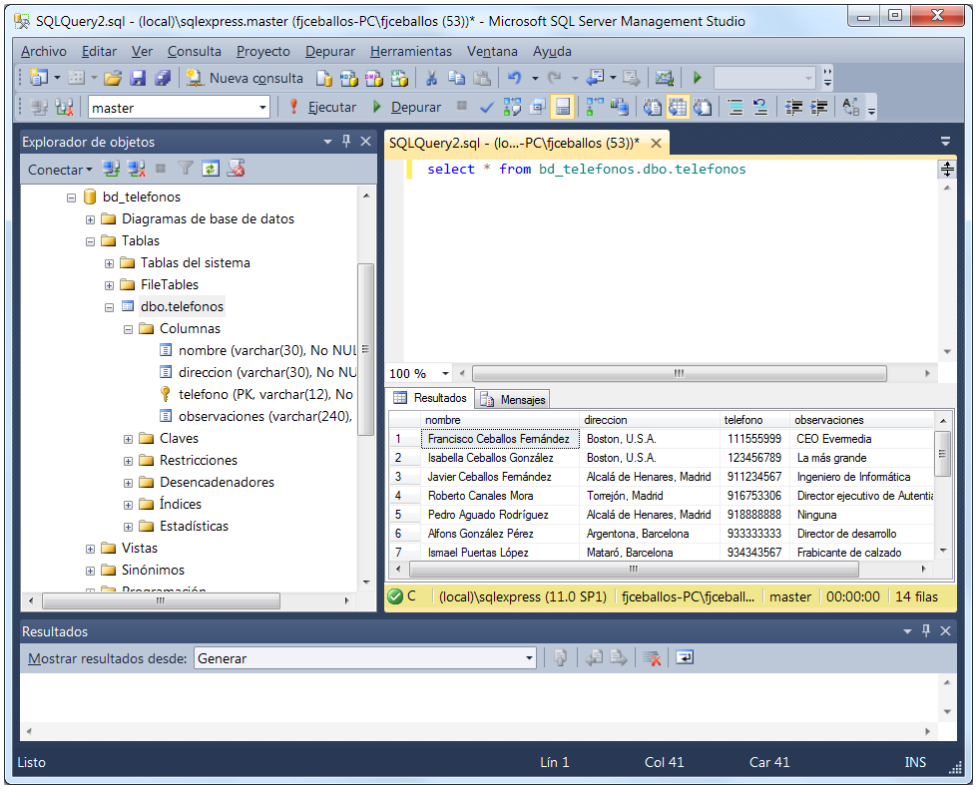
Si instaló SQL Server, habrá comprobado que no dispone de una herramienta de administración de bases de datos. Por tal motivo, Microsoft también ha desarrollado una nueva aplicación para gestionar bases de datos que puede obtener de forma gratuita de Internet en la dirección especificada a continuación:

http://www.microsoft.com/es-es/downloads

Esta aplicación presenta una interfaz gráfica, muy sencilla de utilizar, para realizar tareas típicas como crear bases de datos, gestionar las tablas de la base y los procedimientos almacenados, crear usuarios, etc.

Cuando inicie *SQL Server Management Studio Express*, le serán solicitados el nombre del servidor de bases de datos, el tipo de autenticación, y el usuario y la contraseña solo si eligió autenticación SQL Server:

Una vez realizada la conexión con el gestor de bases de datos, le será mostrada la ventana de la figura siguiente. Seleccione en la lista del panel de la izquierda la base de datos con la que desea trabajar, haga clic en el botón *Nueva consulta* de la barra de herramientas y, después, escriba en el mismo las sentencias SQL que desee ejecutar. Para ejecutar una sentencia SQL, haga clic en el botón *Ejecutar* de la barra de herramientas.



CONTENEDOR DE SERVLET/JSP TOMCAT 8

Tomcat 8.0 es un servidor de aplicaciones que implementa las tecnologías *Java Servlet 2.5* y *JavaServer Pages 2.1*. Puede obtenerlo de la dirección de Internet *http://jakarta.apache.org/tomcat*. Un servidor de aplicaciones, a diferencia de un servidor web, como es *Apache*, incluye un contenedor web que permite servir páginas dinámicas (un servidor web solo sirve páginas HTML estáticas, recursos CGI, páginas PHP y accesos SSL). Evidentemente, si se trata de servir páginas estáticas, es más eficiente un servidor web, pero también podemos utilizar para este cometido un servidor de aplicaciones –es menos eficiente, pero es más seguro– o bien podríamos utilizar ambos conectados entre sí.

Exactamente, *Tomcat* es un contenedor de *servlets*. Entonces, ¿por qué *Tomcat* puede trabajar también como un servidor web? Porque incluye un software (una serie de *servlets*) que le permite realizar este cometido. Eche una ojeada al fichero *<TOMCAT_HOME>\conf\web.xml* y verá cómo se describe un *servlet* por omisión para permitir a cualquier aplicación servir recursos estáticos. También hay otro, denominado *invoker*, que permite ejecutar *servlets* anónimos; esto es, *servlets* que no han sido descritos en el fichero *web.xml* de su aplicación. Tradicionalmente, este *servlet* se identifica con el patrón "/servlet/*". Así mismo, en este fichero podemos ver que un compilador se encarga de compilar las páginas JSP y convertirlas en *servlets*. Resumiendo, y según lo expuesto, podemos decir que *Tomcat* es un servidor web con soporte para *servlets* y páginas JSP.

El fichero *web.xml* de *Tomcat* define los valores por defecto para todas las aplicaciones web cargadas por este servidor de aplicaciones. Cada vez que se carga una aplicación se procesa este fichero, seguido del fichero *web.xml* propio de la aplicación (*/WEB-INF/web.xml*).

Otro fichero de interés es *<TOMCAT_HOME>\conf\server.xml*, donde se define la configuración del servidor.

Instalación

Suponiendo que ya tiene instalado JSE 8.x, la instalación sobre Windows de *Tomcat 8.x* es muy sencilla:

1. Ejecute el fichero *apache-tomcat-8.x.xx.exe* que haya descargado de Internet. El instalador utilizará la variable de entorno *JAVA_HOME* registrada en el registro de Windows para determinar la ruta de la máquina virtual de Java en el JDK o en el JRE, o bien puede usted localizarla manualmente. También creará accesos directos que permitirán arrancar y configurar el servidor.

2. Siga los pasos indicados durante la instalación. En uno de ellos se le pedirá el puerto HTTP; asegúrese de que el especificado (por defecto el 8080) no está siendo utilizado por otra aplicación.

Opcionalmente puede activar la ejecución *servlets* anónimos; esto es, *servlets* que no han sido declarados en un fichero *web.xml*. Para ello edite el fichero *<TOMCAT_HOME>\conf\web.xml* y permita que se ejecute el código siguiente:

```
<servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>
      org.apache.catalina.servlets.InvokerServlet
    </servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>

...

<servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Este código, por seguridad, viene comentado. Si durante la fase de desarrollo permite que se ejecute, podrá invocar la ejecución de *servlets* anónimos de la forma:

```
http://servidor[:puerto]/localización/servlet/nombre_servlet
```

Es importante notar que el administrador de aplicaciones de *Tomcat* solo podrá ser utilizado cuando el servidor esté arrancado.

APÉNDICE C

FICHEROS JAR

Los ficheros Java (JAR – *Archive Java*) son ficheros comprimidos con el formato ZIP. Típicamente, un fichero JAR contendrá los ficheros de clases y de recursos de *applets* o de aplicaciones. La utilización de este tipo de ficheros proporciona, entre otras, las siguientes ventajas:

- *Seguridad*. Ya que se pueden firmar digitalmente para certificar que su contenido no es malicioso.
- *Disminución del tiempo de descarga*. Si un *applet* está empaquetado en un fichero JAR, los ficheros de clases y recursos asociados pueden ser descargados por el navegador en una sola transacción HTTP sin necesidad de abrir una nueva conexión para cada fichero.
- *Ahorro de espacio*.
- *Portabilidad*. Ya que el mecanismo para manejar los ficheros JAR es un estándar en Java.

HERRAMIENTA JAR DE JAVA

Para trabajar con ficheros JAR, Java proporciona la herramienta *jar*. La sintaxis para invocarla y las opciones que puede utilizar para realizar distintas operaciones (crear, extraer, mostrar, etc.) se pueden ver ejecutando la orden:

```
jar
```

Las operaciones básicas que se pueden realizar se resumen en la tabla siguiente:

Operación	*Sintaxis*
Crear un fichero JAR	`jar cf f.jar fe1 [fe2]...`
Ver el contenido de un fichero JAR	`jar tf f.jar`
Extraer el contenido de un fichero JAR	`jar xf f.jar`
Extraer los ficheros especificados	`jar xf f.jar fx1 [fx2]...`

Ejecutar un *applet* empaquetado en un fichero JAR	`<applet`
	`  code=ClaseApplet.class`
	`  archive="ap.jar"`
	`  width=ancho height=alto>`
	`</applet>`

f Representa un fichero JAR.
fe Representa un fichero a empaquetar en un fichero JAR (nombre completo).
fx Representa un fichero empaquetado (nombre completo).
ap Nombre del fichero JAR que empaqueta la aplicación.

c Esta opción indica que se quiere crear un fichero JAR.
f Esta opción indica que a continuación se especifica el nombre del fichero JAR. Si se omite, se asume la salida estándar cuando se trate de almacenar datos en un fichero de salida, o la entrada estándar cuando se trate de tomar datos de un fichero de entrada.
t Esta opción indica que se desea ver la tabla de contenido del fichero JAR.
x Esta opción especifica que se trata de una operación de extracción.
v Mostrar mensajes en la salida estándar acerca de las operaciones realizadas.

EJECUTAR UNA APLICACIÓN EMPAQUETADA EN UN JAR

Según lo expuesto en el apartado anterior, utilizando la herramienta *jar* podemos empaquetar todos los ficheros *.class* que componen una aplicación en un fichero JAR y después ejecutarla. Por ejemplo:

```
jar cf MiAplicacion.jar *.class
```

Cuando se crea un fichero JAR, también se crea un manifiesto por omisión (para observarlo desempaque el fichero):

```
META-INF\MANIFEST.MF
```

Para que el fichero JAR pueda ser ejecutado, el manifiesto debe contener una línea que especifique el nombre de la clase principal:

```
Main-Class: nombre-clase-principal
```

Por lo tanto, tendrá que editar este fichero y añadir esta línea.

Finalmente, para ejecutar la aplicación empaquetada en el fichero JAR, ejecute la orden siguiente:

```
java -jar MiAplicacion.jar
```

Como ejemplo, volvamos a la clase *CAplicacion* que escribimos en el capítulo 1. Observe la primera línea de código; especifica el paquete al que pertenece dicha clase:

```
package Cap01.EstructuraAp;
```

Según esto, *CAplicacion* es una clase ubicada en *...\Cap01\EstructuraAp*, donde los tres puntos (...) sustituyen al sistema de ficheros al que pertenece *Cap01*; por ejemplo, *c:\java\ejemplos*.

Según lo expuesto, para empaquetar una aplicación ya compilada y ejecutarla desde la línea de órdenes, tendremos que realizar los pasos siguientes:

1. Especificar la ruta donde se localiza el compilador *java*. Por ejemplo:

   ```
   set path=%path%;c:\Java\jdk1.7\bin
   ```

2. Empaquetar la aplicación; por ejemplo en un fichero *EstructuraAp.jar*. Para ello escribiríamos:

   ```
   cd c:\java\ejemplos
   c:\java\ejemplos>jar -cvf Cap01\EstructuraAp\EstructuraAp.jar
   Cap01\EstructuraAp\*.class
   ```

3. Extraer del fichero *EstructuraAp.jar* el manifiesto *MANIFEST.MF*, añadirle la siguiente línea para especificar cuál es la clase principal, se trata de un fichero de texto, y volverlo a empaquetar:

   ```
   Main-Class: Cap01.EstructuraAp.CAplicacion
   ```

4. Para ejecutar *EstructuraAp.jar* hay que cambiar a la carpeta donde está y ejecutar la orden indicada a continuación:

   ```
   cd c:\java\ejemplos\Cap01\EstructuraAp
   c:\java\ejemplos\Cap01\EstructuraAp>java -jar EstructuraAp.jar
   ```

El entorno de desarrollo *NetBeans* crea automáticamente en la carpeta *dist* el fichero JAR que empaqueta la aplicación actualmente en desarrollo.

APÉNDICE D

CÓDIGOS DE CARACTERES

UTILIZACIÓN DE CARACTERES ANSI CON WINDOWS

Una tabla de códigos es un juego de caracteres donde cada uno tiene asignado un número utilizado para su representación interna. Visual C# utiliza Unicode para almacenar y manipular cadenas, pero también puede manipular caracteres en otros códigos como ANSI o ASCII.

ANSI (American National Standards Institute) es el juego de caracteres estándar más utilizado por los equipos personales. Como el estándar ANSI solo utiliza un byte para representar un carácter, está limitado a un máximo de 256 caracteres. Aunque es adecuado para el inglés, no acepta totalmente otros idiomas. Para escribir un carácter ANSI que no esté en el teclado:

1. Localice en la tabla que se muestra en la página siguiente el carácter ANSI que necesite y observe su código numérico.
2. Pulse la tecla *Bloq Núm* (Num Lock) para activar el teclado numérico.
3. Mantenga pulsada la tecla *Alt* y utilice el teclado numérico para pulsar el 0 y a continuación las teclas correspondientes al código del carácter.

Por ejemplo, para escribir el carácter ± en el entorno Windows, mantenga pulsada la tecla *Alt* mientras escribe 0177 en el teclado numérico. Pruebe en la consola del sistema (línea de órdenes).

Los 128 primeros caracteres (códigos 0 a 127) son los mismos en las tablas de códigos ANSI, ASCII y Unicode.

JUEGO DE CARACTERES ANSI

DEC	CAR
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X

DEC	CAR
89	Y
90	Z
91	[
92	\
93	]
94	^
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	▒
128	▒
129	▒
130	▒
131	▒
132	▒
133	▒
134	▒
135	▒
136	▒
137	▒
138	▒
139	▒
140	▒
141	▒
142	▒
143	▒
144	▒
145	‘

DEC	CAR
146	'
147	“
148	"
149	o
150	–
151	—
152	▒
153	▒
154	▒
155	▒
156	▒
157	▒
157	▒
159	▒
160	
161	¡
162	¢
163	£
164	¤
165	¥
166	¦
167	§
168	¨
169	©
170	ª
171	«
172	¬
173	-
174	®
175	¯
176	°
177	±
178	²
179	³
180	´
181	µ
182	¶
183	·
184	¸
185	¹
186	º
187	»
188	¼
189	½
190	¾
191	¿
192	À
193	Á
194	Â
195	Ã
196	Ä
197	Å
198	Æ
199	Ç
200	È
201	É

DEC	CAR
202	Ê
203	Ë
204	Ì
205	Í
206	Î
207	Ï
208	Ð
209	Ñ
210	Ò
211	Ó
212	Ô
213	Õ
214	Ö
215	×
216	Ø
217	Ù
218	Ú
219	Û
220	Ü
221	Ý
222	Þ
223	ß
224	à
225	á
226	â
227	ã
228	ä
229	å
230	æ
231	ç
232	è
233	é
234	ê
235	ë
236	ì
237	í
238	î
239	ï
240	ð
241	ñ
242	ò
243	ó
244	ô
245	õ
246	ö
247	÷
248	ø
249	ù
250	ú
251	û
252	ü
253	ý
254	þ
255	ÿ

UTILIZACIÓN DE CARACTERES ASCII

En MS-DOS y fuera del entorno Windows se utiliza el juego de caracteres ASCII. Para escribir un carácter ASCII que no esté en el teclado:

1. Busque el carácter en la tabla de códigos que coincida con la tabla activa. Utilice la orden **chcp** para saber qué tabla de códigos está activa.
2. Mantenga pulsada la tecla *Alt* y utilice el teclado numérico para pulsar las teclas correspondientes al número del carácter que desee.

Por ejemplo, si está utilizando la tabla de códigos 850, para escribir el carácter π mantenga pulsada la tecla *Alt* mientras escribe 227 en el teclado numérico.

JUEGO DE CARACTERES ASCII

VALOR DECIMAL	VALOR HEXA-DECIMAL	CONTROL CARACT	CARACT.	VALOR DECIMAL	VALOR HEXA-DECIMAL	CARACT.	VALOR DECIMAL	VALOR HEXA-DECIMAL	CARACT	VALOR DECIMAL	VALOR HEXA-DECIMAL	CARACT.	VALOR DECIMAL	VALOR HEXA-DECIMAL	CARACT	VALOR DECIMAL	VALOR HEXA-DECIMAL	CARACT.
000	00	NUL		043	2B	+	086	56	V	129	81	ü	172	AC	¼	215	D7	╫
001	01	SOH	☺	044	2C	,	087	57	W	130	82	é	173	AD	¡	216	D8	╪
002	02	STX	☻	045	2D	-	088	58	X	131	83	â	174	AE	«	217	D9	┘
003	03	ETX	♥	046	2E	.	089	59	Y	132	84	ä	175	AF	»	218	DA	┌
004	04	EOT	♦	047	2F	/	090	5A	Z	133	85	à	176	B0	▒	219	DB	■
005	05	ENQ	♣	048	30	0	091	5B	[	134	86	å	177	B1	▒	220	DC	▄
006	06	ACK	♠	049	31	1	092	5C	\	135	87	ç	178	B2	▓	221	DD	▌
007	07	BEL	•	050	32	2	093	5D	]	136	88	ê	179	B3	│	222	DE	▐
008	08	BS	◘	051	33	3	094	5E	^	137	89	ë	180	B4	┤	223	DF	▀
009	09	HT	○	052	34	4	095	5F	_	138	8A	è	181	B5	╡	224	E0	α
010	0A	LF	◙	053	35	5	096	60	`	139	8B	ï	182	B6	╢	225	E1	ß
011	0B	VT	♂	054	36	6	097	61	a	140	8C	î	183	B7	╖	226	E2	Γ
012	0C	FF	♀	055	37	7	098	62	b	141	8D	ì	184	B8	╕	227	E3	π
013	0D	CR	♪	056	38	8	099	63	c	142	8E	Ä	185	B9	╣	228	E4	Σ
014	0E	SO	♫	057	39	9	100	64	d	143	8F	Å	186	BA	║	229	E5	σ
015	0F	SI	☼	058	3A	:	101	65	e	144	90	É	187	BB	╗	230	E6	µ
016	10	DLE	►	059	3B	;	102	66	f	145	91	æ	188	BC	╝	231	E7	τ
017	11	DC1	◄	060	3C	<	103	67	g	146	92	Æ	189	BD	╜	232	E8	Φ
018	12	DC2	↕	061	3D	=	104	68	h	147	93	ô	190	BE	╛	233	E9	Θ
019	13	DC3	‼	062	3E	>	105	69	i	148	94	ö	191	BF	┐	234	EA	Ω
020	14	DC4	¶	063	3F	?	106	6A	j	149	95	ò	192	C0	└	235	EB	δ
021	15	NAK	§	064	40	@	107	6B	k	150	96	û	193	C1	┴	236	EC	∞
022	16	SYN	▬	065	41	A	108	6C	l	151	97	ù	194	C2	┬	237	ED	∅
023	17	ETB	↨	066	42	B	109	6D	m	152	98	ÿ	195	C3	├	238	EE	∈
024	18	CAN	↑	067	43	C	110	6E	n	153	99	Ö	196	C4	─	239	EF	∩
025	19	EM	↓	068	44	D	111	6F	o	154	9A	Ü	197	C5	┼	240	F0	≡
026	1A	SUB	→	069	45	E	112	70	p	155	9B	¢	198	C6	╞	241	F1	±
027	1B	ESC	←	070	46	F	113	71	q	156	9C	£	199	C7	╟	242	F2	≥
028	1C	FS	∟	071	47	G	114	72	r	157	9D	¥	200	C8	╚	243	F3	≤
029	1D	GS	↔	072	48	H	115	73	s	158	9E	₧	201	C9	╔	244	F4	⌠
030	1E	RS	▲	073	49	I	116	74	t	159	9F	ƒ	202	CA	╩	245	F5	⌡
031	1F	US	▼	074	4A	J	117	75	u	160	A0	á	203	CB	╦	246	F6	÷
032	20	SP	Space	075	4B	K	118	76	v	161	A1	í	204	CC	╠	247	F7	≈
033	21		!	076	4C	L	119	77	w	162	A2	ó	205	CD	═	248	F8	°
034	22		"	077	4D	M	120	78	x	163	A3	ú	206	CE	╬	249	F9	∙
035	23		#	078	4E	N	121	79	y	164	A4	ñ	207	CF	╧	250	FA	·
036	24		$	079	4F	O	122	7A	z	165	A5	Ñ	208	D0	╨	251	FB	√
037	25		%	080	50	P	123	7B	{	166	A6	ª	209	D1	╤	252	FC	n
038	26		&	081	51	Q	124	7C	¦	167	A7	º	210	D2	╥	253	FD	²
039	27		'	082	52	R	125	7D	}	168	A8	¿	211	D3	╙	254	FE	■
040	28		(	083	53	S	126	7E	~	169	A9	⌐	212	D4	╘	255	FF	
041	29		)	084	54	T	127	7F	⌂	170	AA	¬	213	D5	╒			
042	2A		*	085	55	U	128	80	Ç	171	AB	½	214	D6	╓			

JUEGO DE CARACTERES UNICODE

UNICODE es un juego de caracteres en el que se emplean 2 bytes (16 bits) para representar cada carácter. Esto permite la representación de cualquier carácter en cualquier lenguaje escrito en el mundo, incluyendo los símbolos del chino, japonés o coreano.

Códigos Unicode de los dígitos utilizados en español:

`\u0030-\u0039`	0-9 ISO-LATIN-1

Códigos Unicode de las letras y otros caracteres utilizados en español:

`\u0024`	$ signo dólar
`\u0041-\u005a`	A-Z
`\u005f`	_
`\u0061-\u007a`	a-z
`\u00c0-\u00d6`	À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö
`\u00d8-\u00f6`	Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö
`\u00f8-\u00ff`	ø ù ú û ü ý þ ÿ

Dos caracteres son idénticos solo si tienen el mismo código Unicode.

ÍNDICE

@

@Basic, 343
@Column, 343
@EJB, 489
@Embeddable, 351
@EmbeddedId, 350
@Entity, 342
@GeneratedValue, 343
@Id, 342
@IdClass, 350
@Remote, 491
@Table, 343
@Transient, 344
@WebMethod, 658
@WebService, 656
@WebServiceRef, 706
@WebServlet, 509, 524

A

a, etiqueta HTML, 445
abort, 788
absolute, 299
AbstractButton, 29
AbstractTableModel, 198, 330
Accesibilidad, 6, 27
accesos directos, 445
ace:ajax, 824
ace:column, 824
ace:dataTable, 824
ace:simpleSelectOneMenu, 823
aceleradores, 84, 100
acentos, 910
acoplamiento, 830
action, 448, 451, 516
ActionEvent, 34
ActionListener, 34
actionPerformed, 21
actualizaciones síncronas y asíncronas, ICEfaces, 816
adaptadores, 22
add, 14, 17, 152, 184
addAdjustmentListener, 34
addChoosableFileFilter, 171
addColumn, 226
addContainerListener, 35
addCookie, 527
addElement, 152
addUndoableEditListener, 114
addWindowListener, 9, 22
AdjustmentEvent, 34, 161
AdjustmentListener, 34
administrador de diseño, 142
  absoluto, 899
  null, 16, 899
administrador de entidades, 355

administrador de interfaz de usuario, 24
administradores de diseño, 889
afterLast, 299
AJAX, 785
  envío de datos utilizando el método POST, 795
  llamadas fuera de dominio, 794
  llamadas que producen errores o que no retornan, 795
AJAX+JSF, 797
align, 446
alineación del texto, 16
alineación horizontal, 64
alt, 446
ámbito de aplicación, 582
ámbito de página, 582
ámbito de petición, 582
ámbito de sesión, 582
ámbito de un atributo, 581
ámbito, variables definidas, 606
AncestorListener, 35
Anchor, 891
animación, 479
AnnotationConfigApplicationContext, 844
anotación WebMethod, 658
anotación WebService, 657
anotaciones, 341
anotaciones en Spring, 845
ANSI, 951
añadir menú, 82
añadir un botón a la barra de h., 87
añadir y borrar un nodo, 210
API, 289
API de java, JSTL, 611
aplicación
  cliente, 406, 433
  desarrollo, 37
  empresarial, 404
  Java como cliente, 662, 681
  jerarquía, 36
  mínima, 6
aplicación web, 463
  como cliente, 670
aplicaciones distribuidas, 401
Applet, 468
applet
  crear, 464, 904
  desplegar en Apache Tomcat, 484
  desplegar en GlassFish, 485
  etiqueta, 465
  parámetros, 473
applets, 403, 463, 464
application, 580
Application Scoped Entity Managers, 390
ApplicationContext, 829, 840, 844
applicationScope, 586
árbol, 202
  árbol, estilo de las líneas, 215
  iniciar, 205
  modelos, 205
  modificar aspecto, 214
área de texto, 452
área de trabajo, 3
arquitecturas JSP, 632
  modelo 1, 632
  modelo 2, 633
arrastrar y colocar, 6, 27
ArrayList, 260
ArrayList<T>, 261
ASCII, 951, 953
asistente para conexiones, 892
asociación, 352
ASP, 462, 575
atributo, 581
AudioClip, 478
Auto Resizing, 891
AWT, 5, 27, 471, 888
ayuda, 902

B

barra de desplazamiento, 2, 159
  eventos, 161
barra de direcciones, 440
barra de estado, 89
  mensajes, 479
barra de herramientas, 85
barra de menús, 1
  crear, 79
barra de progreso, 164
  eventos, 164
barra de título, 2
barras de desplazamiento, propiedades, 158
base de datos, 275
  crear, 940
bean de entidad, 407
bean de sesión, 407
bean orientado a mensajes, 407
bean, iniciar propiedades, 737
BeanFactory, 828, 840
BeanProperty, 249
beans, 601
Beans, 888
Beans Binding, 237

beans de apoyo, 729
  configuración, 733
beans manejados JSF, 730, 760
beep, 52
before, 537
beforeFirst, 299
biblioteca de componentes JSF, 721
biblioteca estándar de etiquetas de JSP, 604
binding, 731
Binding, 247
Bindings, factoría, 249
body, 441
border, 454
BorderLayout, 14, 195
borrar registros en una tabla, 279
borrar tabla de la base de datos, 279
botón borrar, 451
botón de opción, 140, 450
botón de pulsación, 37
botón enviar, 451
botón por omisión, 38, 43, 187
botón pulsado del ratón, 117
BoxLayout, 15
br, 442
BULK INSERT, 324
ButtonGroup, 112, 140

C

c, core, 604
c:catch, 606
c:choose, 605
c:forEach, 606
c:if, 605
c:out, 606
c:set, 605
caja de clave de acceso, 449
caja de diálogo
  Abrir, 168
  Color, 172
  de entrada, 128
  Guardar como, 169
  modal o no modal, 126
  modal personalizada, 130
  para confirmación, 127
  para mostrar un mensaje, 126
  personalizada, 133
caja de texto, 449
  multilínea, 93
  seleccionar el contenido, 47
cajas de diálogo común, 167
cajas de texto, 37
CallableStatement, 290
callback methods, 364
cambios en los datos, notificar, 259
campos, 275
canRedo, 115
canUndo, 115
capa, 401
  cliente, 404, 410
  de lógica de negocio, 405, 861
  de recursos de información, 405
  web, 405, 411
capas de servicios de la red, 435
cardinalidad, 353
CardLayout, 14
CaretEvent, 34
CaretListener, 34
cargar los datos desde un fichero, 284
carrito de la compra, 646
cascade, 361
casilla de verificación, 137, 450, 628
catch, 606
CDialog, 167
celda es editable, 198
center, 446
centrar ventana, 226
cerrar, 2
  conexión, 297
CGI, 462
ChangeEvent, 35
ChangeListener, 35
chat, 568
choose, 605
ciclo de vida de un applet, 472
ciclo de vida de una entidad, 362
ciclo de vida de una página JSF, 752
ciclo de vida de una página JSP, 578
clase
  anónima, 199
  clase Applet, 468
  Binding, 247
  ButtonGroup, 112, 140
  Class, 85, 118
  Clipboard, 97
  DefaultBoundedRangeModel, 158
  DefaultComboBoxModel, 156
  DefaultListModel, 152
  DefaultMutableTreeNode, 202
  DefaultTreeCellRenderer, 214
  DefaultTreeModel, 206
  File, 169
  FileFilter, 170
  Graphics, 469

clase
  Hashtable, 163
  HttpJspBase, 579
  HttpServlet, 503
  HttpServletRequest, 505
  HttpServletResponse, 505, 580
  Image, 118
  ImageIcon, 85, 476
  Integer, 145
  ItemEvent, 137
  JApplet, 468
  JButton, 12
  JCheckBox, 12, 137
  JCheckBoxMenuItem, 112
  JComboBox, 153
  JComponent, 28
  JDialog, 125, 133
  JFileChooser, 168
  JFrame, 7
  JLabel, 12
  JList, 12, 146
  JMenu, 79
  JMenuBar, 79
  JMenuItem, 80
  JOptionPane, 13, 125
  JPopupMenu, 116
  JProgressBar, 164
  JRadioButton, 12, 140
  JRadioButtonMenuItem, 113
  JScrollBar, 13, 159
  JScrollPane, 93, 95, 149
  JSeparator, 81
  JSlider, 162
  JTable, 194
  JTextArea, 12, 93, 95
  JTextField, 12
  JTree, 202
  KeyAdapter, 144
  KeyStroke, 85
  ListSelectionEvent, 201
  Pattern, 56
  SimpleTagSupport, 612
  TableColumn, 199
  Thread, 480
  Timer, 174
  Toolkit, 98, 118
  Transferable, 98
  TreePath, 210
  TreeSelectionEvent, 209
  TreeSelectionListener, 209
  UndoManager, 99, 113, 114
  URL, 85, 474
clases de entidad, definir, 348, 714
Class, 85, 118, 199, 295
classes, 510
classpath, 880, 934
ClassPathXmlApplicationContext, 843
clave foránea, 303
clave primaria, 303
clearTimeout, 795
clic, doClick, 73
cliente de servicios web XML, 661
cliente de una aplicación JEE, 488
cliente Java EE, 492
cliente Java REST, 694
cliente servidor, 401
cliente web, 406, 433
Clipboard, 97
close, 297
code, 443, 465
codebase, 465
codificación, 902
Color, 24, 65
color del primer plano, 24
color RGB, 24
color, establecer, 65
columnas, 275
comentarios, 586
Comparable, 638
compareTo, 638
compilación y ejecución, 136
compilar, 896
compilar y ejecutar la aplicación, 11
componente
  editable, 65
  eliminar, 896
  JavaBean, 599
  JSF, 722
  que tiene el foco, 47
  Swing, 12
  transferir el control, 431
componentes, 3
  de acceso a datos, 851
  de rango definido, 158
  dibujar, 889
  paleta, 888
  Swing, 28
  web, 407
ComponentEvent, 34
ComponentListener, 34, 119
componentResized, 90, 119
conectar con la fuente de datos, 295
conexión Java a una base de datos, 294
config, 580

configurar Java, 904
conjunto de conexiones, 545
conjunto de resultados desplazable, 301
Connection, 290
ConsoleHandler, 323
Container, 17
ContainerEvent, 35
ContainerListener, 35
contains, 363
contenedor de IoC, 840
contenedor de nivel intermedio, 13
contenedor de texto, 57
contenedores, 13, 883
content, 571
Context, 547
contexto de persistencia, 356
controlador, 32
  cargar, 294
  puente JDBC-ODBC, 291
  URL, 295
controladores, 291
  JDBC, 291
controles, 3, 449
conversiones, 593
  en enlaces, 253
conversor, 253
conversores intrínsecos, 254
Converter, 253
convertForward, 253
convertidores, 732
convertir una cadena de caracteres a un int, 145
convertReverse, 253
cookies, 526, 586
coordenadas de un componente, 17
copy, 99
CORBA, 409
correo electrónico, 438
crear un servicio web, 657
crear una aplicación, 6
crear una base de datos, 276, 936
crear una tabla, 276
CREATE DATABASE, 276
CREATE TABLE, 276
createAutoBinding, 249
createDefaultModel, 57
createSQLQuery, 857, 859
createStatement, 296, 301
CRUD, 357
  operaciones, 683
CSS, 455, 766, 775, 779, 785, 812, 822, 868
  atributos, 456
  class, 457
CSS
  div, 459
  id, 458
  link, 457
  pseudoclases, 458
  span, 459
  style, 456
cursor, 298
cursor, tipo, 301
cut, 99

D

Data Transfer Object, 852
DatabaseMetaData, 290, 298
DataSource, 545, 548
debugger, 921
declaraciones, 589
DefaultBoundedRangeModel, 158
defaultButton, 187
defaultCloseOperation, 886
DefaultComboBoxModel, 156
DefaultFocusManager, 47
DefaultListModel, 152
DefaultMutableTreeNode, 202
DefaultTableModel, 198, 330
DefaultTreeCellRenderer, 214
DefaultTreeModel, 206
delegado, 32
delete, 857
DELETE, 279
deslizador, eventos, 164
dependencia, 830
depurador, 897
depurar, 878
DESCRIBE, 283
descriptor de biblioteca de etiquetas, 597
descriptor de despliegue, 522
descriptor de despliegue web.xml, 576
descubrimiento de servicios web XML, 664
deshacer, 99
deshacer y rehacer, 113
deslizador, 162
  etiquetas, 163
  propiedades, 162
desplazamiento, panel, 149
despliegue de una aplicación JSF, 734
destroy, 469, 505
detach, 363
DHTML, 786
diálogo Abrir/Guardar, propiedades, 170
diálogos, añadir, 900

Dictionary, 163
Dimension, 91
dirección de Internet, 436
dirección IP, 436
directrices, 587
diseño libre, 889
display-name, 522
dispose, 135, 189
distribuir una aplicación web, 552
DNS, 435
doClick, 73
Document, 57
DocumentEvent, 35, 242
DocumentListener, 35, 59, 241
documento XML, 793
doGet, 505, 518
DOM, 786
dominio, 436
doPost, 505, 518
doTag, 612
doubleBuffered, 499
drawImage, 475
Driver, 290
DriverManager, 290
DriverPropertyInfo, 290
drivers, 291
DROP TABLE, 279

E

EDI, 901
editor de textos, 92
EJB de entidad, 412
EJB de sesión, 413
EJB dirigidos por mensajes, 413
EJB remoto, 492
ejecutar un servlet, 543
ejecutar una aplicación en un JAR, 948
elemento de un menú, señalar, 112
elementos de programación, 589
ELProperty, 249
e-mail, 438
empaquetar una aplicación web, 552
encode, 529
enfocar un componente, 46
enlace a colecciones, 261
enlace con otros controles, 253
enlace de datos, 237
enlace de datos con Beans Binding, 246
enlace de datos manual, 237
enlace, controladores de eventos, 258
enlace, crear, 250
enlaces, 445
enterprise JavaBeans (EJB), 407
entidad, 342
entidades huérfanas, 360
EntityManager, 346, 355
EntityManagerFactory, 355
entorno de desarrollo integrado, 875, 901
entrada de datos, 449
enum, 68
Enumeration, 519
errorData, 626
errores en la compilación, 897
errorPage, 625
escalabilidad, 633
escribir datos en una tabla, 278
escuchadores de eventos, 19, 33
estructura de la tabla, 283
estructura de una aplicación gráfica, 7
etiqueta html applet, 466
etiqueta html object, 466
etiqueta personalizada con atributos, 614
etiquetas, 37, 604
etiquetas HTML, 441
etiquetas personalizadas, 611
  .tag, 618
etiquetas SQL, 607
evento, 4
  blur, 806
  componentResized, 90
  de tipo OnBlur, 819
  keyTyped, 44
  stateChanged, 92
  textChanged, 241
eventos, 4, 19
  de menús, 110
  durante el ciclo de vida de una entidad, 363
  JSF, 750
  manejar, 892
  responder, 23, 43
excepción, atrapar, 606
excepciones, 318
  manipular, 625
exception, 581
executeQuery, 297
executeUpdate, 300
exit, 10
expresión LE, 594
expresiones, 590, 593
  regulares, 53

F

f:selectItem, 768
f:selectItems, 768
f:actionListener, 752
f:ajax, 800, 814
f:convertDateTime, 732
f:converter, 799
f:validator, 799
f:view, 721
Facelets, 738
Facelets Template, 746
faces-config.xml, 724, 742, 766, 775, 805, 822, 868
FacesContext, 753, 782
FacesContext.renderResponse, 753
FacesContext.responseComplete, 754
FacesMessage, 782
FacesServlet, 720
fecha y hora, 511
fichero .war, 552
fichero de propiedades, 723, 742
fichero de texto, 520
filas, 275
File, 169
FileFilter, 170
FileHandler, 323
filtros, 170
  accept y getDescription, 170
finalizar la ejecución, 886
find, 357
findAttribute, 583
Firefox
  Firebug, 922
firePropertyChange, 244
first, 299
FlowLayout, 15, 90
flush, 359
fmt, 604, 622
fmt:parseDate, 631
fn, funciones, 604
foco, 46, 96, 182
  perder en JSF, 806
FocusAdapter, 182
FocusEvent, 34
focusGained, 182
FocusListener, 34
font, 441, 890
Font, 65
forEach, 606
form, 448, 516
formatNumber, 622
formato, 232
formulario, 3
formularios, 448, 515, 627
  añadir, 900
forName, 295
fragmento de código, 590, 616, 623
frame, 569
FreeDesign, 14, 889
ftp, 437, 438
fuente, establecer, 65
funciones, 596

G

generar código JavaScript, 796
gestión de datos, 402
get, 152, 184, 516, 860
GET, 518
getActionCommand, 21
getAttribute, 534, 581, 582
getAttributeNames, 534
getAttributeNamesInScope, 583
getAttributesScope, 583
getBean, 844
getButton, 117
getChoosableFileFilters, 172
getClass, 118
getCodeBase, 475
getColumnClass, 198
getColumnCount, 201
getConnection, 296, 545
getContentPane, 13
getCookies, 527
getCreationTime, 536
getCrossPlatformLookAndFeelClassName, 24
getCurrentSession, 856
getDocument, 114
getDocumentBase, 475
getElementsByTagName, 794
getException, 581
getFocusOwner, 47
getHeight, 120
getImage, 118, 475
getInt, 299
getItem, 137
getItemSelectable, 137
getJspBody, 616
getJspContext, 613, 616
getKeyChar, 51
getKeyCode, 51
getKeyStroke, 85
getLastAccessedTime, 536

getLastPathComponent, 210
getLastSelectedPathComponent, 209
getMetaData, 298
getModel, 32, 152
getName, 169, 527
getOut, 580
getPage, 581
getPageContext, 580
getParameter, 473, 519
getParameterNames, 519
getParameterValues, 519
getResource, 85, 118
getRowCount, 201
getSelectedColumn, 200
getSelectedFile, 168
getSelectedIndex, 153, 158
getSelectedItem, 156
getSelectedRow, 200
getSelectedText, 48, 99
getSelectedValue, 150
getSelectedValuesList, 150
getSelectionEnd, 48, 99
getSelectionPath, 210
getSelectionStart, 48, 99
getServletConfig, 580
getServletContext, 580
getSession, 533, 580
getSize, 91
getSource, 21, 51, 137
getStateChange, 137
getString, 299
getSystemClipboard, 98
getSystemLookAndFeelClassName, 24
getText, 45
getUI, 32
getUserObject, 209
getValue, 527
getValueAt, 200
getWidth, 120
getWriter, 507
GlassFish, pool de conexiones, 548
Gopher, 437
gráficos en un documento HTML, 446
Graphics, 469
GridBagLayout, 14
GridLayout, 14, 142
GroupLayout, 14
grupo de botones, 112
grupos de noticias, 439
guardia de recursos, 412

H

h:column, 768
h:commandButton, 768
h:commandLink, 776
h:dataTable, 768
h:facet, 768
h:inputText, 768, 778
h:inputTextarea, 768
h:outputLabel, 778
h:outputStylesheet, 779
h:outputText, 768
h:panelGrid, 768
h:selectOneMenu, 768
h:graphicImage, 813
h:panelGroup, 806
h:selectOneMenu, 814
habilitar o inhabilitar los elementos de un menú, 110
HashSet, 374
Hashtable, 163
hasMoreElements, 520
head, 441
header, 585
headerValues, 586
height, 447
Hibernate, 850
hilo, 480
hipertexto, 444
hn, 442
hojas de estilo, 455
horizontalAlignment, 890
hr, 442
href, 445
html, 441
HTML, 441
HTTP, 437, 525
tipo de petición, 517
http-equiv, 571
HttpJspBase, 579
HttpServlet, 503
HttpServletRequest, 505
HttpServletResponse, 505, 580
HttpSession, 533, 581

I

ICEfaces, 814, 820
Icon, 476
icono, añadir a la aplicación, 118
IDE, 901
identificador de sesión único, 529

if, 605
iframe, 447, 569
Image, 118
ImageIcon, 85, 476
imagen, 451
  en un applet, 474
  en un botón, 85
imágenes en un documento HTML, 446
IMAP, 437
img, 446
import, 10
include, 588
iniciar una lista desplegable, 222
init, 468, 504, 524
init-param, 523
initParam, 586
input, 449
INSERT, 278
insertar datos en una base, 300
insertNodeInto, 206, 212
insertString, 58
instalación de J2SE 6.0 SDK, 931
instalación de Tomcat, 944
instalar una aplicación web, 553
instanceof, 21
Integer, 145
integridad referencial, 304
interceptar la tecla pulsada, 48
interfaces gráficas, 5, 881
interfaz
  AudioClip, 478
  Comparable, 638
  remota, 489
  Runnable, 480
  SingleThreadModel, 505
Internet, 433
  Explorer, 439
intranet, 434
invalidate, 536
inversión de control, 834
invocar a un servlet desde una página HTML, 514
invoke, 623
inyección de dependencias, 835
IoC, 840
IRC, 568
ISAPI, 462
isCellEditable, 198
isELIgnored, 590
isNew, 536
ISO-8859-1, 910
isRunning, 175
isSelected, 138, 141
isThreadSafe, 588
ItemEvent, 35, 137
itemLabel, 814
ItemListener, 35, 137
itemStateChanged, 137
itemValue, 814

J

JApplet, 13, 28, 468, 883
java
  comp/env, 547
Java 2D, 6, 27
Java EE, 401, 404, 556, 827
  componentes, 406
  contenedores, 407
  crear proyecto, 413, 771
  JSF y JPA, 755
Java SE, 401
Java Web Start, 495
java.sql, 290
JavaBean, 599
javac, 137
JavaMail, 409
JavaScript, 785
  depurar código, 921
  fichero .js, 797
  generar código, 796
JavaServer Faces, 717
javax.jws, 657
javax.servlet, 502
javax.sql, 290
JAXB, 677
JAX-RS, 654
JAX-WS, 653, 676
JButton, 12, 29, 37
JCheckBox, 12, 29, 137
JCheckBoxMenuItem, 29, 111
JColorChooser, 29
JComboBox, 29, 153
JComboBox, enlace a datos, 269
JComponent, 28
JCP, 403
JDBC, 289, 410
  aplicación ejemplo, 293
  y servlets, 538
JDesktopPane, 30
JDialog, 13, 28, 125, 133, 883, 900
JDK, 403, 875, 931
JEditorPane, 31
jerarquía de componentes de una aplicación, 36

Jersey, 688
JFC, 5, 27
JFileChooser, 29, 168
JFrame, 7, 28, 883
JInternalFrame, 30, 47
JLabel, 12, 30, 37
JLayeredPane, 30
JList, 12, 30, 146, 187
  actualizar la vista, 190
  enlace a datos, 261
JMenu, 29, 79
JMenuBar, 30, 79
JMenuItem, 80
JMS, 409
JNDI, 409, 547, 549
JOptionPane, 13, 30, 125, 900
JPA, 340
  ingeniería inversa, 370
JPanel, 30, 90
JPasswordField, 31, 133
JPopupMenu, 30, 116
JProgressBar, 30, 164
JRadioButton, 12, 29, 140
JRadioButtonMenuItem, 29, 111, 113
JRootPane, 30
JScrollBar, 13, 30, 159
JScrollPane, 30, 95, 149
JSeparator, 30, 81
JSF, 717
  aplicación, 718
  view, 721
JSF+AJAX, 797
JSlider, 30, 162
JSON, 792
JSP, 464, 575
  objetos implícitos, 580
jsp:attribute, 603
jsp:doBody, 620
jsp:forward, 626
jsp:getProperty, 603
jsp:invoke, 623
jsp:setProperty, 602
jsp:useBean, 601
JspFragment, 623
JSplitPane, 30
JSTL, 604
  API de Java, 611
JTA, 409
JTabbedPane, 30, 328
JTable, 31, 194
  enlace a datos, 270
  y base de datos, 330
JTableHeader, 31
JTextArea, 12, 31, 37, 95
JTextComponent, 31
JTextField, 12, 31, 37
  el texto cambia, 241
JTextPane, 31
JToggleButton, 29
JToolBar, 31, 86
JToolTip, 31
JTree, 31, 202
JTree.lineStyle, 215
juego de caracteres, 910
JUnit, 836
JViewport, 31, 226
JWindow, 28

K

KeyAdapter, 144
KeyEvent, 34, 51
KeyListener, 34, 50
keyPressed, 50
keyReleased, 50, 144
KeyStroke, 85
keyTyped, 50

L

label, 453
last, 299
Layouts, 889
leer los datos enviados por el cliente, 519
lenguaje de expresión, 585
  activar, 590
li, 444
LIKE, 288
List, 260
lista, 146
  acceder a sus elementos, 150
  añadir elemento, 152
  borrar elemento, 153, 157
  columnas, 147
  con barras de desplazamiento, 149
  desplegable JSF, 814
  desplegable, 153, 452
  diseño, 147
  editable, 153
  iniciar, 149
  selección múltiple, 150
listas, 629
  HTML, 444
ListModel, 151

ListSelectionEvent, 35, 201
ListSelectionListener, 35, 151, 201
ListSelectionModel, 396
load, 860
LOAD DATA LOCAL INFILE, 284
localhost, 518
localidad, 631
Logger, 321
logging, 321
lógica de negocio, 402
logs, 321
loop, 478

M

maestro-detalle, 390
mail, 438
main, 8
manejadores de eventos, 8, 19, 33, 892
  asignar, 20
manejadores swing, 33
mapeo objeto-relacional, 339, 341
marco, 2
  con título, 222
  de trabajo JavaServer Faces, 719, 739
marcos, 569
  en páginas HTML, 447
márgenes, 64
matches, 56
Maven, 922
maximizar, 2
mensaje breve, 18, 892
mensaje personalizado, 736
mensajes, 4
  de error, 897
  en la barra de estado, 479
  en una página JSF, gestión, 735
menú, 77
  añadir nuevo, 79
  contextual, 116
  de control, 2
  elementos, 80
  emergente, 116
  emergente, visualizar, 117
  manejador de eventos, 82, 103
  señalar elemento, 111
  separador, 81
MenuEvent, 110
MenuListener, 110
menús, 77
  diseño, 78
menuSelected, 110
merge, 363, 780
meta, 570
meta http-equiv, 570
metadatos, 298
  de configuración, 841
method, 448, 516
método conducido por un evento, 43
métodos de devolución de llamada, 364
Microsoft SQL Server, 324
middleware, 404
minimizar, 2
mnemonic, 18, 892
mock, 836
modal o no modal, 126
modelo, 32, 152
modelo – vista – controlador, 57, 634
modelo de datos de un JTable, implementar, 330
Model-View-Controller, 31
módem, 437
modificar datos en una tabla, 278
módulos de Spring, 828
mostrar las bases de datos existentes, 282
mostrar las tablas, 283
MouseAdapter, 201
mouseClicked, 201
mouseEntered, 91
MouseEvent, 34
mouseExited, 91
MouseInputAdapter, 34
MouseListener, 34, 201
MouseMotionListener, 34
multiple, 452
múltiples peticiones, 588
  despachar, 588
MVC, 634
MySQL, 282, 538, 568, 934
  conector JDBC, 934
  instalación de, 934
  puesta en marcha, 935
  utilidades, 936

N

name, 445, 449
navegación entre páginas, 726
navegar por la base de datos, 301
nemónicos, 84
NetBeans, conector JDBC, 940
newAudioClip, 478
newInstance, 295
news, 439

next, 299
nextElement, 520
n.º de filas y columnas de una tabla, 201
nodo de un árbol, acceso, 207
  añadir, 212
  borrar, 213
nodos de un árbol, 202
  borrar todos, 214
número de columnas del conjunto de resultados, 298

O

objeto, 4
objetos de negocio, 860
ObservableCollections, 260
ObservableList, 260
ol, 444
OnBlur, 819
onchange, evento, 792
onreadystatechange, 789
open, 788
openSession, 856
operaciones con las entidades, 357
operaciones en cascada, 361
operadores, 594
optgroup, 453
option, 452
orden de tabulación, 451
órdenes de acción, 21
orígenes de datos de Windows, 325
ORM, 340
out, 580, 606

P

p, 442
pack, 887
package, 136
page, 581, 587
pageContext, 580, 585, 626
PageContext, clase, 581
pageScope, 586
página HTML invoca a servlet, 570
página JSF, crear, 720
página JSP, 575
página web dinámica, 462
páginas ASP, 462
páginas web, 441
paint, 468, 498
PaintEvent, 35
palabras reservadas, 596
paleta de componentes, 888
panel de contenido, 13
panel de desplazamiento, 149
panel raíz, 13
paneles con pestañas, 328
paquete awt, 471
paquetes, 10, 949
param, 585, 608
parámetro oculto, 450, 532
parámetros de un applet, 473
paramValues, 585, 628
parseDouble, 45
parseInt, 145
password, 133
paste, 99
path, 508
Pattern, 56
persist, 358
persistence.xml, 346
persistencia, 339
  unidad, 346, 375
petición HTTP GET, 518
petición HTTP POST, 518
PHP, 575
pintar con Swing, 498
PlainDocument, 57
play, 478
POJO, 340
pool de conexiones, 545
PoolConnection, 545
POP 2 y 3, 437
portapapeles, 97
posición de un componente, modificar, 118
post, 516, 518
postbacks, 753
PreparedStatement, 290
presentación, 402
previous, 299
PRIMARY KEY, 277
println, 507
PrintWriter, 507
prioridad de los operadores, 595
proceso, 480
  escrito en Java, 510
  ligero, 480
programación orientada a objetos, 5
PropertyChangeEvent, 244
PropertyChangeListener, 35
PropertyChangeSupport, 242
propiedad cambió, notificar, 242
propiedad defaultButton, 187
propiedades de navegación, 352

protocolo, 435
protocolo de transferencia de ficheros, 438
proxy, 661
proyecto, 901
PruebaConexion, 305
punto de inserción, 46

Q

query, 608

R

random, 24
readyState, 789
recuperar datos de una base, 297
recursos, 475
redimensionamiento automático, 891
redo, 99, 115
redondear a dos cifras decimales, 232
reescribir un URL, 532
referencia al nodo seleccionado, 210
referencia web, 667
refresh, 359, 570
Refresh, página HTML, 565
registerDriver, 295
registros, 275
rehacer, 99
relative, 299
reload, 214
reloj, 173
reloj despertador, 121, 191
remove, 58, 153, 184, 360
removeAllChildren, 214
removeElementAt, 158
repaint, 469, 481
replaceSelection, 99
request, 580
RequestDispatcher, 431
requestFocus, 46, 96, 146
requestScope, 586
required, 734
reset, 451
response, 580
responseText, 789
responseXML, 789
REST, 683
RESTful y JPA, 707
ResultSet, 290, 298, 331
ResultSet, operaciones de insertar, borrar..., 331
ResultSetMetaData, 290, 298, 331
RGB, 65
rowCount, 611
rows, 611
Runnable, 480
ruta, 210

S

save, 857
Scanner, 318
scriptlet, 576, 584
scriptlets, 590
scripts, 462
scrollPathToVisible, 212
secuencia de escape, 593
seguimiento de una sesión, 525, 533
seguridad en los applets, 466
selección cambió, 396, 399
seleccionar datos de una tabla, 279
seleccionar el texto de una caja, 47, 182
seleccionar un nodo de un árbol, 209
seleccionar una fila en un JTable, 201
select, 48, 99, 452
SELECT, 279
selectAll, 48, 99
selected, 452
SelectItem, 823
send, 788
señalar un elemento de un menú, 111
Separator, 81
Serializable, 349
server.xml, 547, 549
service, 505, 517
servicio web, 676
  crear, 657
servicios de Internet, 437
servicios web RESTful, 653, 683
servicios web XML, 653
  acceso, 668
  probar, 660
servidor de aplicaciones, 944
  GlassFish, 655, 688
servidor de nombres, 436
servidor web, 944
servlet, 501, 522
  ciclo de vida, 504
  ejecutar, 508
  estructura, 504
  iniciación, 524
ServletContext, 581
servlet-name, 522
servlets, 463, 908
  características, 502

sesión, finalizar, 536
sesiones, 525
  seguimiento, 525
session, 580
Session, 857
SessionFactory, 856
sessionScope, 586
set, 605
setAccelerator, 85
setActionCommand, 21
setAttribute, 534, 581, 582
setAutoResizeMode, 200
setBackground, 65
setBounds, 17, 91
setCellRenderer, 214
setClosedIcon, 214
setContentType, 507
setDataSource, 607
setDefaultButton, 38, 43
setDelay, 175
setEditable, 65, 153
setEnabled, 111, 138
setFileFilter, 172
setFocusPainted, 88
setFont, 65
setForeground, 24, 65
setHorizontalAlignment, 64
setHorizontalScrollBarPolicy, 95
setIcon, 85, 476
setIconImage, 118
setJspBody, 616
setJspContext, 613, 616
setLayout, 15, 90
setLayoutOrientation, 147
setLeafIcon, 214
setLineWrap, 96
setListData, 150, 188
setLocation, 118
setLocationRelativeTo, 226
setLookAndFeel, 24
setMargin, 64
setMaximumRowCount, 154
setModel, 32, 152, 197
setOpenIcon, 214
setPreferredWidth, 200
setRowHeight, 200
setSelected, 138, 141
setSelectionEnd, 48, 99
setSelectionModel, 150
setSelectionStart, 48, 99
setShowsRootHandles, 214
setSize, 118, 886
setState, 112
setText, 45
setTimeout, 795
setTitle, 886
setUI, 32
setValueAt, 200
setVerticalAlignment, 64
setVerticalScrollBarPolicy, 95
setViewportView, 95
setVisible, 135, 189
setWrapStyleWord, 96
show, 117
SHOW DATABASES, 282
SHOW TABLES, 283
showConfirmDialog, 127
showInputDialog, 128
showMessageDialog, 126, 240
showOpenDialog, 168
showSaveDialog, 169
showStatus, 479
Simple Tag Extension, 611
SimpleTagSupport, 612
sincronización, 573
SingleThreadModel, 505
sistema de nombres de dominio, 435
size, 441
SMTP, 437
SOAP, 653, 676
SocketHandler, 323
sonido en un applet, 477
sonido en una aplicación, 478
Spring, 827, 836
Spring y JSF, integrar, 870
Spring, ficheros de configuración, 870
SpringLayout, 14
SQL, 276
SQL Server Express, 940
SQL Server Management Studio Express, 942
SQL, etiquetas, 607
sql:param, 608
sql:query, 608
sql:setDataSource, 607
sql:transaction, 609
sql:update, 608
SQLCMD, 941
SQLException, 295
src, 446, 447
start, 174, 468
stateChanged, 92
Statement, 290
status, 789
statusText, 789

stop, 174, 469, 478
stub, 836
subdominio, 436
submenús, 80
submit, 451
subproceso, 480
Swing, 5, 27, 28, 888
SwingBindings, factoría, 260
synchronized, 573

T

tabindex, 451
tabla, 193, 275
  construir, 195
  iniciar, 197
  modelos de las columnas, 226
  modelos, 197
  tabla con cabecera de filas, 222
tablas, 454
  insertar datos desde ficheros de texto, 324
table, 454
  dinámico, 790
TableColumn, 199
TableColumnModel, 197, 223
TableModel, 197, 223
taglib, 588
tamaño celdas, 199
tamaño de un componente, modificar, 118
tamaño de una ventana, 886
TCP/IP, 434
td, 454
tecla de acceso, 43
tecla pulsada, interceptar, 48
tecla, interceptar, 50
telnet, 437, 438
temporizador, 173, 795
text, 18, 892
textarea, 452
textChanged, 241
texto seleccionado, 110
th, 454
Thread, 480
Timer, 174
tipo de petición HTTP, 517
tipo de una columna, 198
tipo del documento, 507
tipo enumerado, 68
tipos de enlace, 259
tipos SQL, 277
title, 441
título de una ventana, 886
tld, 597
Tomcat, 508, 944
Toolkit, 52, 98, 118
toolTipText, 18, 892
toString, 183, 209, 270
tr, 454
transacción, 347
transaction, 609
Transferable, 98
transferir el control a otro componente, 431
  web, 626
trazas, 321
TreeModel, 206
TreeModelEvent, 206
TreePath, 210
TreeSelectionEvent, 209
TreeSelectionListener, 209
TreeSelectionModel, 206, 207
TreeSet, 638
type, 449

U

UIComponet, 753
UIManager, 24
undo, 99, 115
UndoableEditEvent, 35
UndoableEditListener, 35
UndoManager, 99, 113, 114
UNIQUE, 277
update, 469, 498, 608, 857
UPDATE, 278
URL, 85, 118, 444, 474
  de la carpeta, 475
URLEncoder, 529
url-pattern, 523, 719
USENET, 437, 439
uso de FocusListener, 47
uso de KeyListener, 44

V

validación usando AJAX, 803
validadores, 256, 734
validar un campo de texto, 50
validate, 257
validateDoubleRange, 736
validateLength, 736
validateLongRange, 736
Validator, 257
validatorMessage, 736
valor de la celda, 200

value, 451
Value Object, 852
valueChanged, 209
valueChanged, JList, 151
valueIsAdjusting, 397
variables, 593
  definidas en un ámbito, 606
Vector, 184
ventana, 1
  centrar, 226
  típica de Windows, 1
ventanas, comunicación entre, 187
VetoableChangeListener, 35
vista, 32, 95

W

WADL, 694
war, 552
web, 439
web.xml, 522, 567
web-app, 523
WEB-INF, 510
WebMethod, anotación, 658
WebService, 657
  anotación, 657
WebServiceRef, 706
welcome-file, 523
width, 447
Window, 47
WindowAdapter, 22, 23
windowClosing, 9, 22
WindowEvent, 35
WindowListener, 35
World Wide Web, 437
WSDL, 676
www, 437, 439

X

x, xml, 604
XHTML, 461, 785
XML, 460, 676, 793
  Spring, 841
XMLHTTP, 788
XMLHttpRequest, 788
XmlWebApplicationContext, 843

Ra-Ma®

Del mismo autor

• Curso de programación con **PASCAL**	ISBN: 978-84-86381-36-3 224 págs.
• Curso de programación **GW BASIC/BASICA**	ISBN: 978-84-86381-87-5 320 págs.
• Manual para **TURBO BASIC** Guía del programador	ISBN: 978-84-86381-43-1 444 págs.
• Manual para **Quick C 2** Guía del programador	ISBN: 978-84-86381-65-3 540 págs.
• Manual para **Quick BASIC 4.5** Guía del programador	ISBN: 978-84-86381-74-5 496 págs.
• Curso de programación **Microsoft COBOL**	ISBN: 978-84-7897-001-8 480 págs.
• Enciclopedia del lenguaje **C**	ISBN: 978-84-7897-053-7 888 págs.
• Curso de programación **QBASIC y MS-DOS 5**	ISBN: 978-84-7897-059-9 384 págs.
• Curso de programación **RM/COBOL-85**	ISBN: 978-84-7897-070-4 396 págs.
• El abecé de **MS-DOS 6**	ISBN: 978-84-7897-114-5 224 págs.
• Microsoft **Visual C ++** (ver. 1.5x de 16 bits) Aplicaciones para Windows	ISBN: 978-84-7897-180-0 846 págs. + 2 disquetes
• Microsoft **Visual C ++** Aplicaciones para Win32 (2.ª edición)	ISBN: 978-84-7897-561-7 792 págs. + disquete
• Microsoft **Visual C ++** Programación avanzada en Win32	ISBN: 978-84-7897-344-6 888 págs. + CD-ROM
• **Visual Basic 6** Curso de programación (2.ª edición)	ISBN: 978-84-7897-357-6 528 págs. + disquete
• Enciclopedia de Microsoft **Visual Basic 6**	ISBN: 978-84-7897-386-6 1.072 págs. + CD-ROM
• El lenguaje de programación **Java**	ISBN: 978-84-7897-485-6 320 págs. + CD-ROM
• El lenguaje de programación **C#**	ISBN: 978-84-7897-500-6 320 págs. + CD-ROM

Ra-Ma®

Del mismo autor

• El lenguaje de programación **Visual Basic.NET**	ISBN: 978-84-7897-525-9 464 págs. + CD-ROM
• **Java 2** Lenguaje y aplicaciones	ISBN: 978-84-7897-745-1 392 págs. + CD-ROM
• Programación orientada a objetos con **C ++** (4.ª edición)	ISBN: 978-84-7897-761-1 648 págs. + CD-ROM
• **C/C++** Curso de programación (3.ª edición)	ISBN: 978-84-7897-762-8 708 págs. + CD-ROM
• **Microsoft C#** Lenguaje y aplicaciones (2.ª edición)	ISBN: 978-84-7897-813-7 520 págs. + CD-ROM
• **Java 2**. Interfaces gráficas y aplicaciones para Internet (3.ª edición)	ISBN: 978-84-7897-859-5 718 págs. + CD-ROM
• **Aplicaciones .Net multiplataforma** (Proyecto Mono)	ISBN: 978-84-7897-880-9 212 págs. + CD-ROM
• Enciclopedia del lenguaje **C ++** (2.ª edición)	ISBN: 978-84-7897-915-8 902 págs. + CD-ROM
• **Microsoft Visual Basic .NET** Lenguaje y aplicaciones (3.ª edición)	ISBN: 978-84-9964-020-4 520 págs. + CD-ROM
• **Java 2** Curso de programación (4.ª edición)	ISBN: 978-84-9964-032-7 820 págs. + CD-ROM
• **Microsoft C#** Curso de programación (2.ª edición)	ISBN: 978-84-9964-068-6 850 págs. + CD-ROM
• **Visual C#**. Interfaces gráficas y aplicaciones para Internet con WPF, WCF y Silverlight	ISBN: 978-84-9964-203-1 956 págs. + CD-ROM
• **Visual Basic**. Interfaces gráficas y aplicaciones para Internet con WPF, WCF y Silverlight	ISBN: 978-84-9964-204-8 938 págs. + CD-ROM
• Enciclopedia de Microsoft **Visual C#**. Interfaces gráficas y aplicaciones para Internet con Windows Forms y ASP.NET (4.ª edición)	ISBN: 978-84-7897-986-8 1.145 págs. + CD-ROM
• Enciclopedia de Microsoft **Visual Basic**. Interfaces gráficas y aplicaciones para Internet con Windows Forms y ASP.NET (3.ª edición)	ISBN: 978-84-7897-987-5 1.125 págs. + CD-ROM

INSTALACIÓN

Para instalar el software necesario siga las recomendaciones dadas en el apéndice B de este libro.

SOBRE LOS EJEMPLOS DEL LIBRO

El material adicional de este libro, con las aplicaciones desarrolladas y el software para reproducirlas, puede descargarlo desde *http://www.fjceballos.es* (sección *Mis publicaciones* > *Java* > *Material adicional*) o desde *http://www.ra-ma.com* (en la página correspondiente al libro). La descarga consiste en un fichero ZIP con una contraseña que encontrará en el libro.

LICENCIA

Los paquetes de software a los que se hace referencia en el material adicional del libro es propiedad de las firmas que los representan (Oracle, Apache Software Foundation, MySQL Software, etc.). La inclusión en este libro se debe a su gentileza y es totalmente gratuita y con la finalidad de apoyar el aprendizaje del software correspondiente. Para obtener más información y actualizaciones, visite las direcciones indicadas en dicho software.

Al realizar el proceso de instalación, haga el favor de consultar el acuerdo de licencia para cada uno de los productos.

WEB DEL AUTOR: http://www.fjceballos.es

En esta web podrá echar una ojeada a mis publicaciones más recientes y acceder a la descarga del software necesario para el estudio de esta obra, así como a otros recursos.

Java™

Interfaces gráficas y aplicaciones para Internet

4.ª EDICIÓN

Hasta hace pocos años Java solo nos traía a la mente una taza de café, objeto que se ha convertido en su logotipo, seguramente por las muchas que se tomaron sus creadores. Sin embargo, hoy en día, cualquiera que haya tenido contacto con una página web tiene otro concepto, y sabe que Java es un lenguaje de programación orientado a objetos, introducido por Sun Microsystems, actualmente soportado por Oracle, cuyas características lo convierten en el producto ideal para desarrollar programas para la Web.

A modo de resumen, Java le permitirá crear programas para su uso personal, para su grupo de trabajo, para una empresa, aplicaciones distribuidas a través de Internet, aplicaciones de bases de datos, páginas web, servicios web y otras muchas cosas.

En este libro se explica cómo crear aplicaciones que muestren una interfaz gráfica al usuario, se estudian los enlaces a datos, SQL y el acceso a bases de datos (JDBC), la persistencia de los datos, el desarrollo de aplicaciones Java EE, el diseño de clientes web (HTML, *applets*), el desarrollo de *servlets*, el diseño de aplicaciones web con JSP, JSTL y JDBC, los servicios web XML y RESTFUL, los marcos de trabajo *JavaServer Faces* (JSF) y *Spring* para crear aplicaciones de tres o más capas para la web, las tecnologías para incluir AJAX en una aplicación web... todo ello con el fin de que diseñe aplicaciones para Internet.

Java: Interfaces gráficas y aplicaciones para Internet es un libro totalmente actualizado a las nuevas características de JDK 8/Java EE 7, con ejemplos claros y sencillos, fáciles de entender, que ilustran el diseño de interfaces gráficas, de páginas web y de aplicaciones con acceso a bases de datos para Internet utilizando JSF, persistencia y AJAX.

Podrá descargarse de **www.ra-ma.es**, en la página web correspondiente al libro, un CD-ROM con los ejemplos realizados, con los apéndices, así como con los URL del software necesario para que el lector pueda reproducirlos durante el estudio.

9 788499 645223

ra-ma.es

Ra-Ma®