Java 2: Interfaces gráficas y aplicaciones para Internet, 3ª 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: 978-84-7897-859-5
Depósito Legal: M-15950-2008
Autoedición: Fco. Javier Ceballos
Filmación e impresión: Albadalejo, S.L.
Impreso en España
Primera impresión: Abril 2008

APÉNDICE A

ENTORNO DE DESARROLLO INTEGRADO PARA JAVA

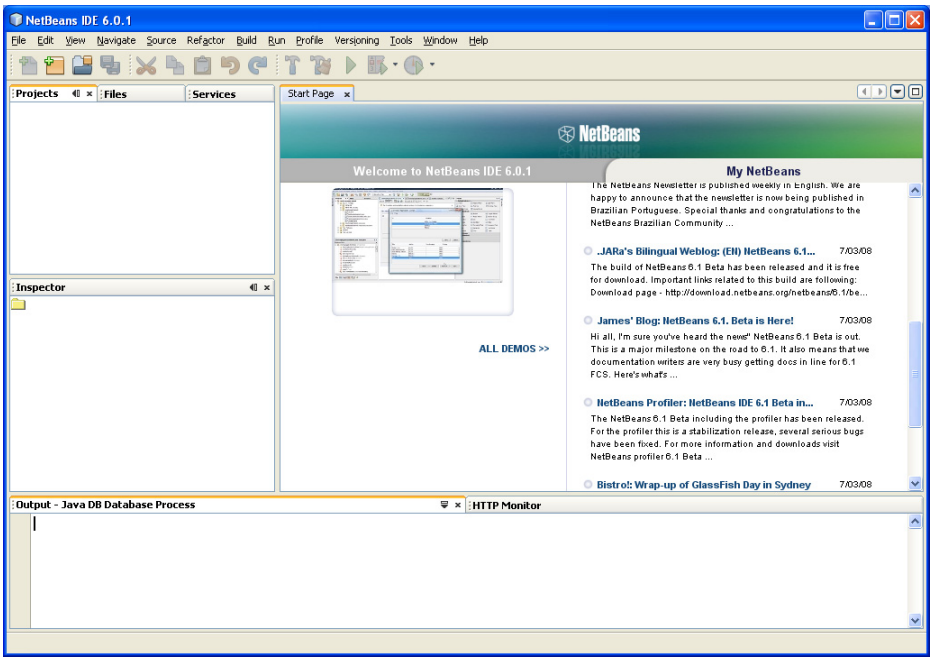
Evidentemente, para poder escribir programas se necesita un entorno de desarrollo Java. *Sun Microsystems*, propietario de Java, proporciona uno de forma gratuita, ***J**2SE **D**evelopment **K**it 6.0* (*JDK 6.0*) para Microsoft Windows, en todas sus versiones, y para Linux. En el apéndice B se explica 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 *Sun Microsystems*. 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 sólo 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:

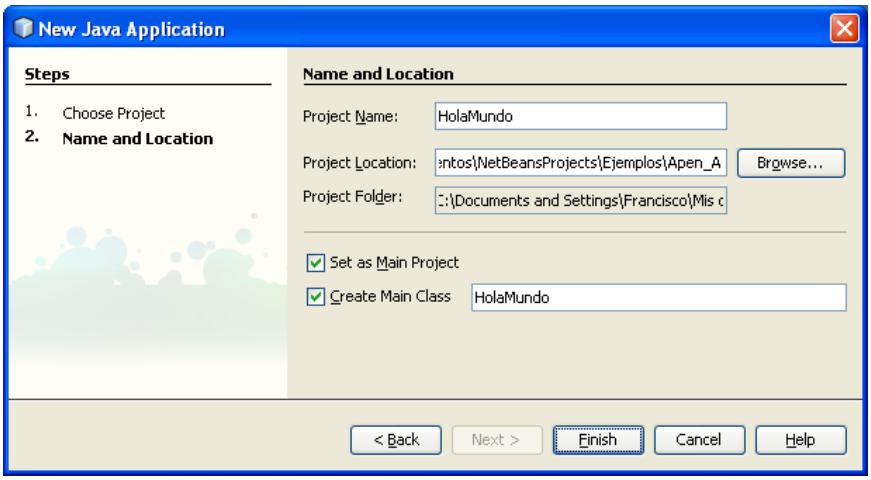
1. Suponiendo que ya se está visualizando el entorno de desarrollo, ejecute la orden *File* > *New Project* (Archivo > Nuevo Proyecto). Se muestra la ventana *New Project*.

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*.

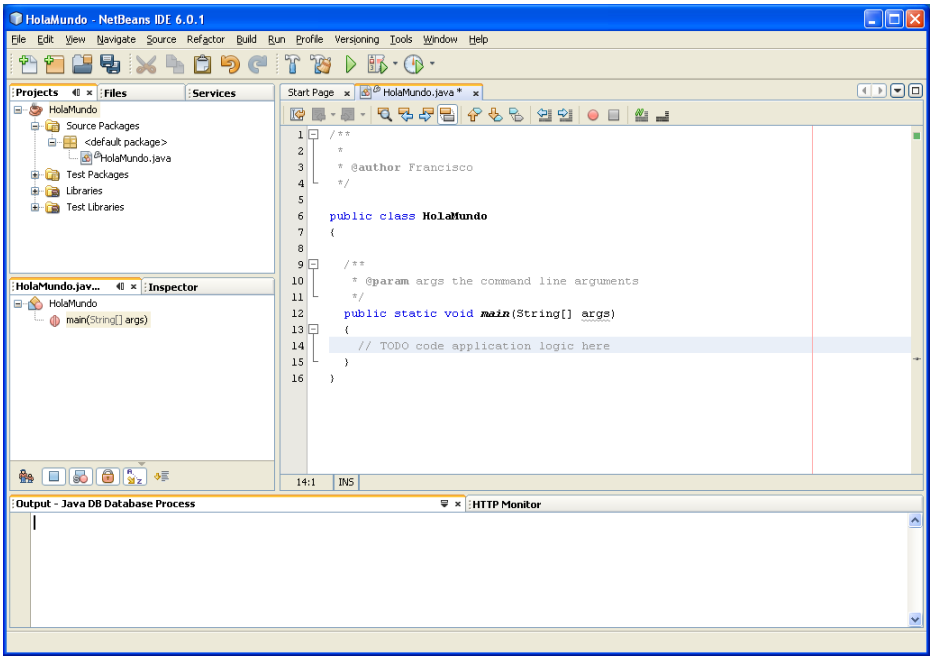
3. Escriba el nombre del proyecto (*Project Name*); en nuestro caso será *HolaMundo* y, a continuación, seleccione la carpeta donde quiere guardarlo.

4. Asegúrese de que las casillas *Set as Main Project* (declararlo como proyecto principal) y *Create Main Class* (crear clase principal) están marcadas.

5. Observe la caja de texto correspondiente al nombre de la clase principal; muestra *holamundo.Main*. Esto significa que la clase principal se llama *Main* y que pertenece al paquete *holamundo*. Asumiremos el paquete por omisión, por lo que en esta caja escribiremos solamente el nombre de la clase; en nuestro caso se llamará *HolaMundo*.



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



El EDI crea la carpeta *Ejemplos\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 panel de proyectos; éste 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 *Build > Build Main Project*, o bien pulse la tecla *F11*. Si la compilación es correcta, puede pasar a ejecutar la aplicación ejecutando la orden *Run > Run Main Project*, o bien pulsando la tecla *F6*; observe el resultado en la ventana *Output*.

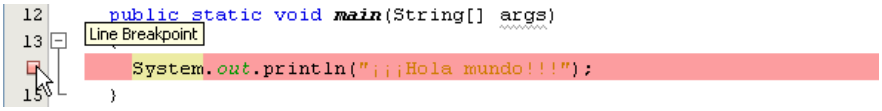
Cuando la aplicación necesite más de un fichero, el proceso es igual de sencillo. Añadir otro fichero a una aplicación, por ejemplo un nuevo fichero que almacene una nueva clase, supone hacer clic con el botón derecho del ratón sobre el nombre del proyecto, elegir la orden *New* y seleccionar del menú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

DEPURAR UNA APLICACIÓN CON NETBEANS

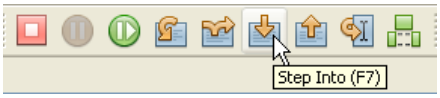
¿Por qué se depura una aplicación? Porque los resultados que estamos obteniendo con la misma no son correctos y no sabemos por qué. El proceso de depuración consiste en ejecutar la aplicación paso a paso, indistintamente por sentencias o por métodos, con el fin de observar el flujo seguido durante su 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 Bre-*

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



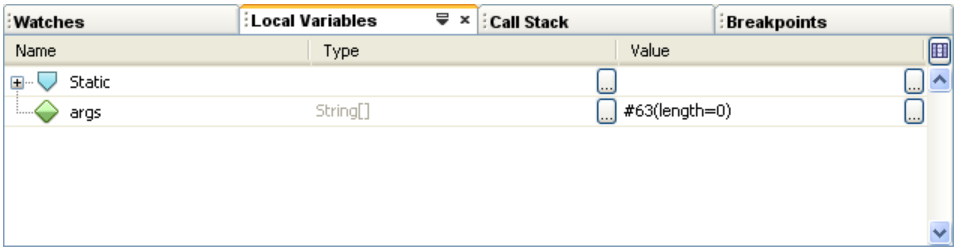
Después, ejecute la orden *Run* > *Debug Main Project*, o bien pulse la tecla *Ctrl+F5* para iniciar la depuración. Continúe la ejecución paso a paso utilizando las órdenes del menú *Run* o los botones correspondientes de la barra de herramientas *Debug* (para saber el significado de cada botón, ponga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



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

- *Debug Main 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 *Watches*, *Local Variables*, etc., del fondo del EDI. Para añadir o quitar ventanas ejecute la orden *Window* > *Debuggin*.

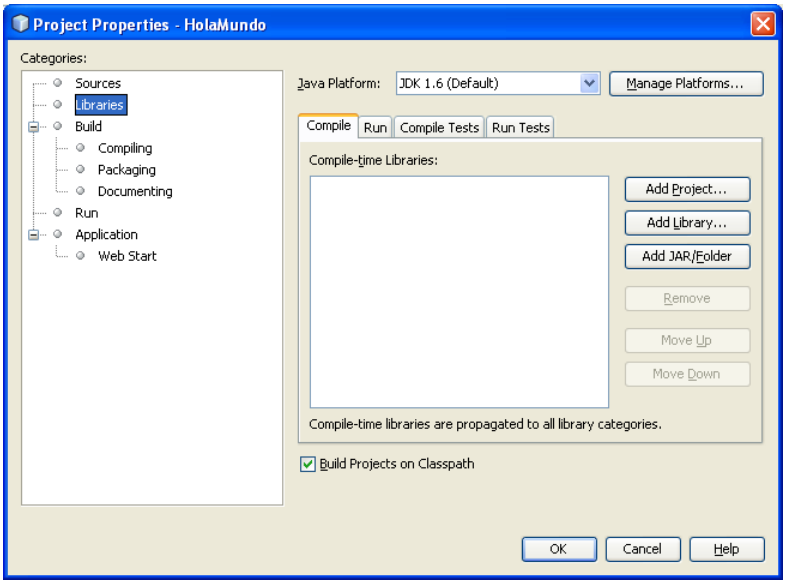


VARIABLE CLASSPATH

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:

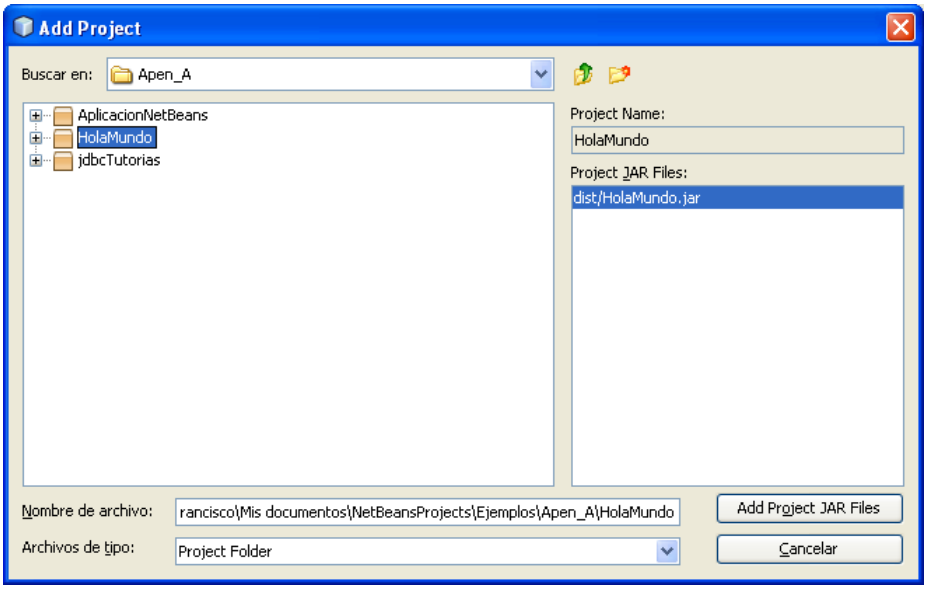
1. Diríjase al panel del proyecto, haga clic con el botón derecho del ratón sobre el nombre del mismo y seleccione la orden *Properties* del menú contextual que se visualiza. Se muestra el diálogo siguiente:



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); mues-

tra 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.

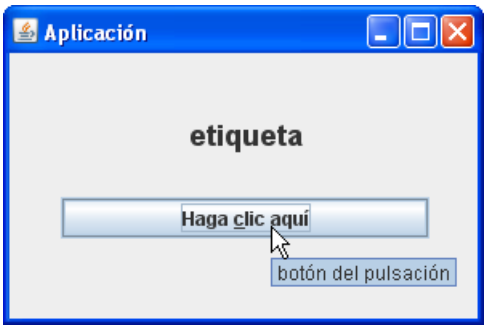
Desde la línea de órdenes esta variable se especificaría análogamente a como indica el ejemplo siguiente:

```
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:

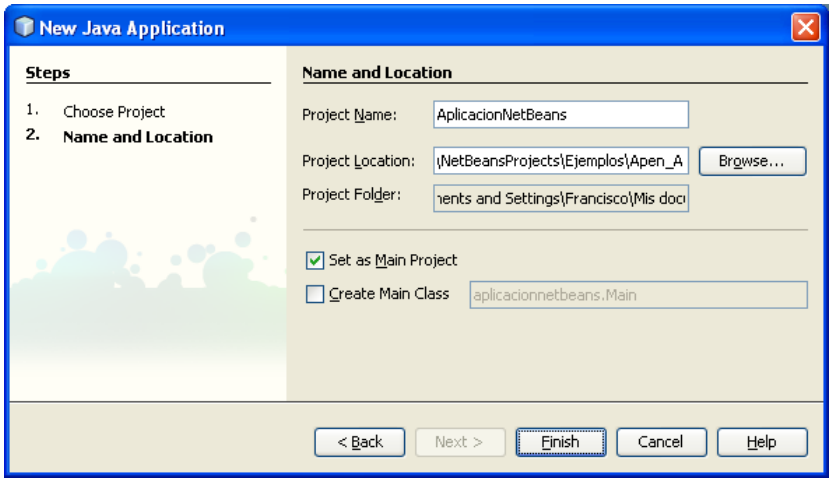
1. Suponiendo que ya está visualizado el entorno de desarrollo, ejecute la orden *File* > *New Project* (Archivo > Nuevo Proyecto). Se muestra la ventana *New Project*.



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.

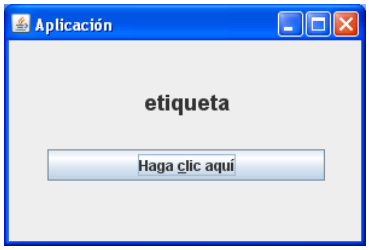


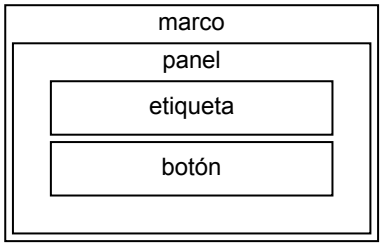
5. Para finalizar, haga clic en el botón *Finish*. El EDI crea la carpeta *Ejemplos\Apen_A\AplicacionNetBeans* en la que guardará el proyecto.

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

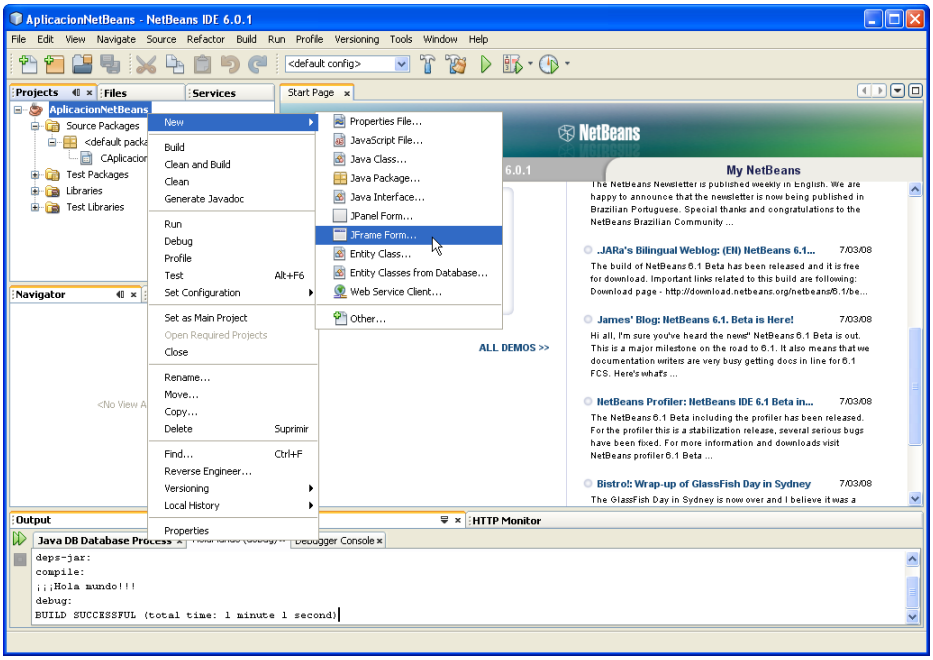
Los contenedores son componentes *Swing* utilizados para ubicar otros componentes. Por ejemplo, la figura siguiente, correspondiente a la interfaz gráfica de la aplicación que queremos desarrollar, explica cómo se acomodan los componentes en uno de estos contenedores.



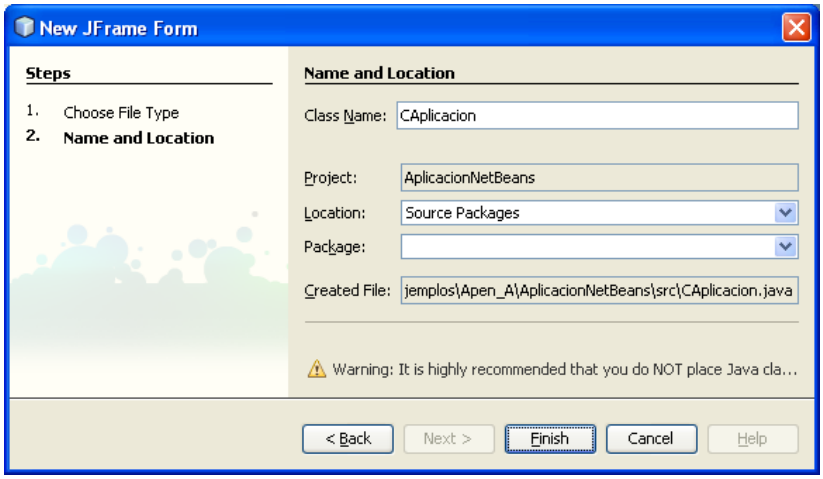


Para agregar componentes a una ventana es aconsejable utilizar un *contenedor* intermedio; esto facilitará la realización de otras operaciones posteriores como, por ejemplo, añadir un borde alrededor de los componentes. Siempre que sea necesario, un contenedor puede contener a otros contenedores, lo 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.

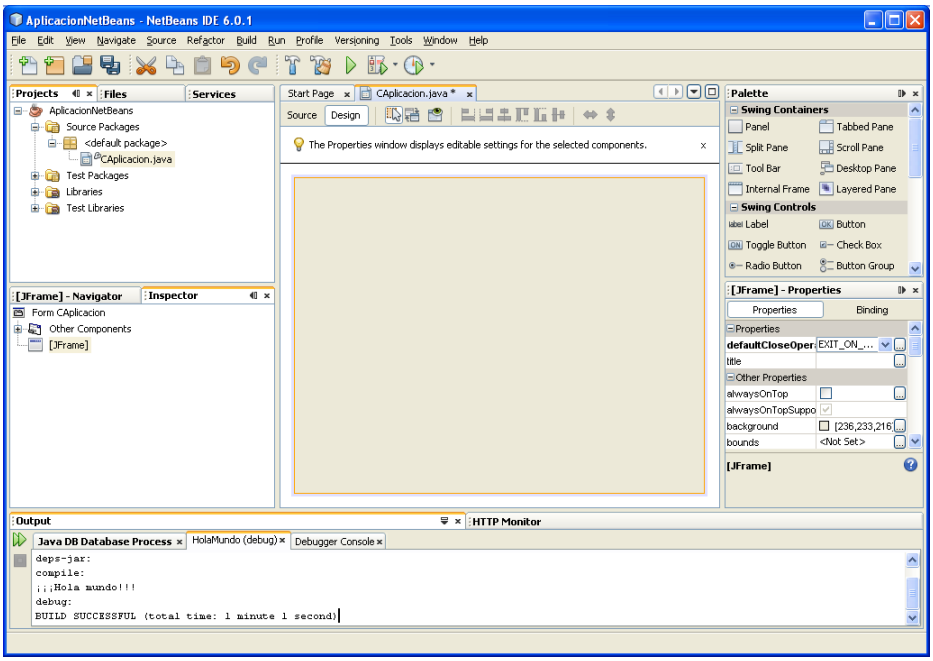
Para añadir una ventana marco (objeto **JFrame**) al proyecto, haga clic sobre el nombre del mismo, utilizando el botón derecho del ratón, y seleccione del menú contextual que se visualiza la orden *New > JFrame Form...*.



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 de la pantalla (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 *Form1*) y el panel de diseño (el que se está mostrando).

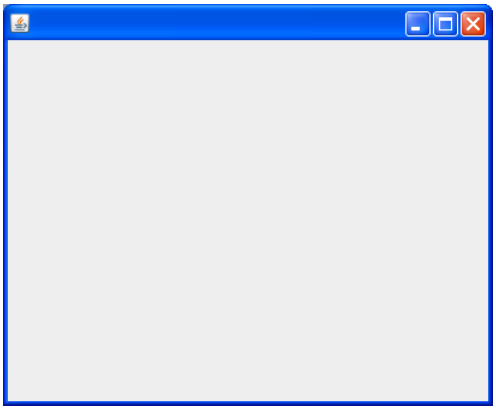
También, a la derecha del formulario, se observan otras dos ventanas: una muestra varias paletas de herramientas (la que se está mostrando es la paleta de componentes *Swing*; si no se muestra ejecute *Windows* > *Palette*) y la otra está mostrando el panel de propiedades con las propiedades del formulario. 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 dos paneles: el supervisor (*Inspector*) y el navegador (*Navigator*). El supervisor permite ver y seleccionar los componentes que forman la interfaz gráfica y el navegador permite navegar por los componentes software de la aplicación (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 Main 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.).



Ésta 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 tanto, 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:

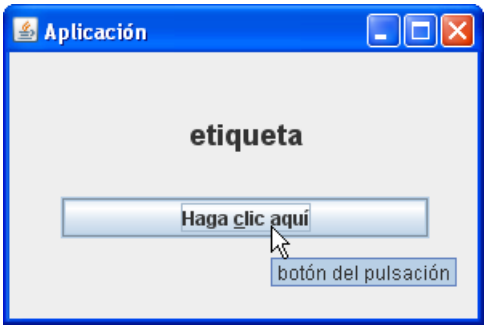
```
public CAplicacion()
{
  initComponents();
  setTitle("Aplicación"); // establecer el título de la ventana
  setSize(300, 200); // fijar el ancho y el alto de la ventana
}
```

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**. 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

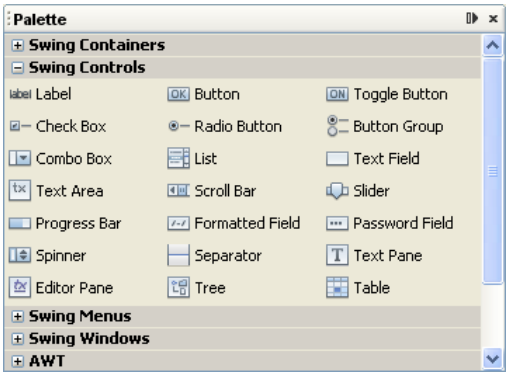
Nuestro objetivo es diseñar una aplicación que muestre una ventana principal con un botón y una etiqueta:



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 los siguientes grupos:

- *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 como ficheros independientes con extensión *.jar*. Se trata de programas Java que

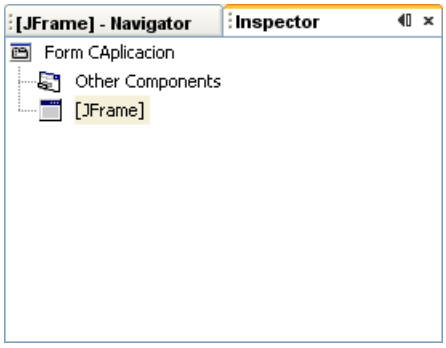
nosotros mismos podemos crear con una funcionalidad específica, con la intención de insertarlos posteriormente en otros programas.

- *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 supervisor de componentes (*Inspector*) 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 (*FreeDesign*) 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. Como *FreeDesign* emplea un modelo de distribución dinámico, siempre que se redimensione el formulario o se modifiquen posiciones la IGU se adaptará para acomodar los cambios sin cambiar las relaciones entre los componentes.

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:

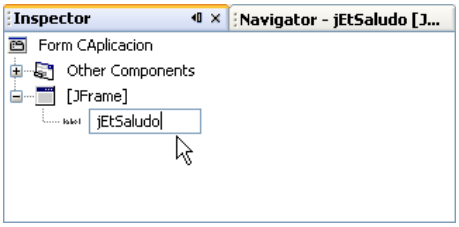
1. Haga clic con el botón derecho del ratón sobre el formulario y seleccione *Set Layout* en el menú contextual.
2. Y, a continuación, haga clic en el administrador de diseño deseado.

Nosotros seguiremos utilizando *FreeDesign* por la automatización que aporta a los formularios para recolocar los componentes cuando dichos formularios cambian de tamaño. Otra opción sería utilizar el componente *Null Layout*.

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*.
2. Haga clic en el componente *JLabel* 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 supervisor de componentes (*Inspector*):



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 supervisor 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.

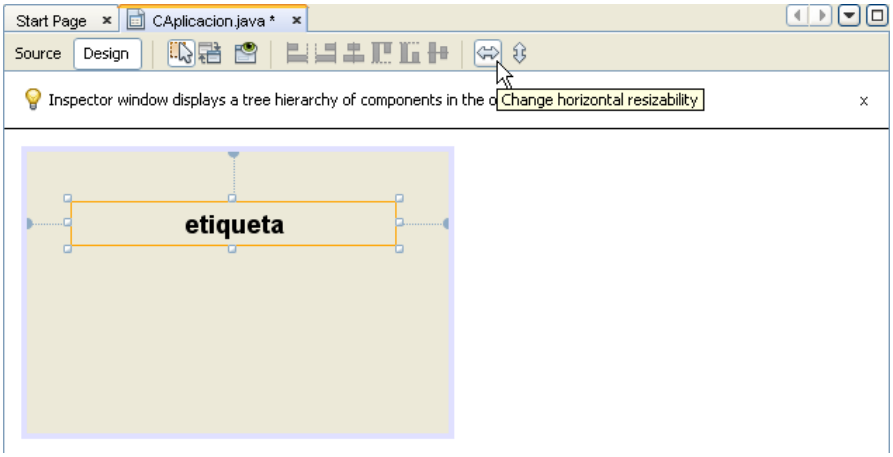
```
private javax.swing.JLabel jEtSaludo;
// ...
jEtSaludo = new javax.swing.JLabel();
// ...
jEtSaludo.setFont(new java.awt.Font("Dialog", 1, 18));
jEtSaludo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jEtSaludo.setText("etiqueta");
// ...
```

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*.
2. Haga clic en el componente *JButton* 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");
// ...
```

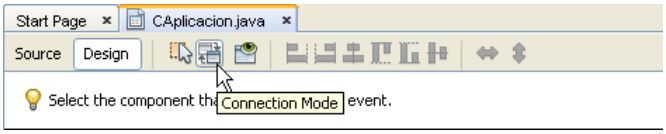
Asignar manejadores de eventos a un objeto

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 recogido 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.

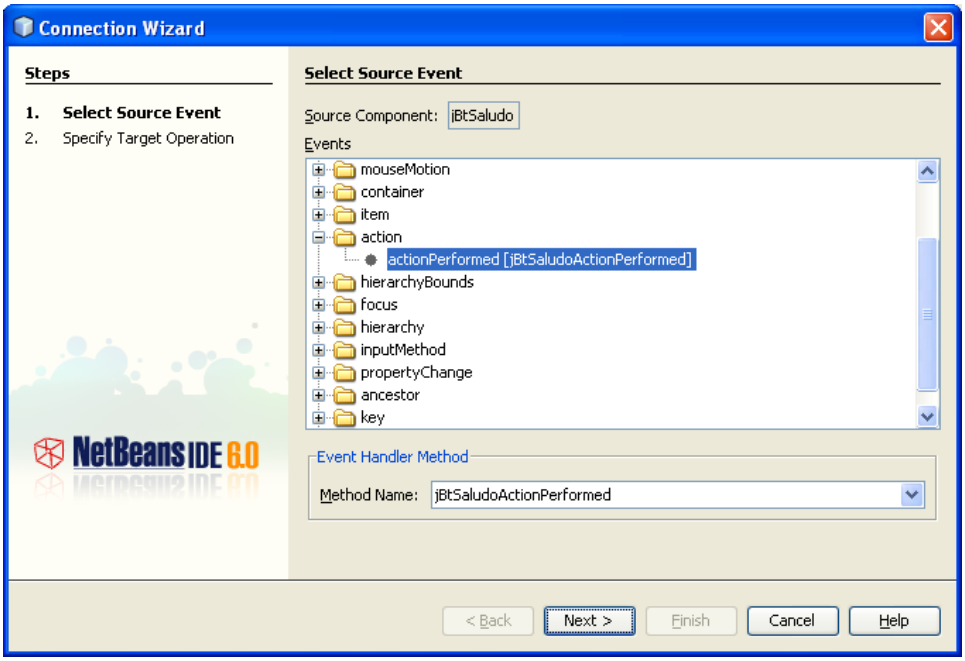
2. Cambie al modo de conexión haciendo clic en el botón *Connection Mode*:

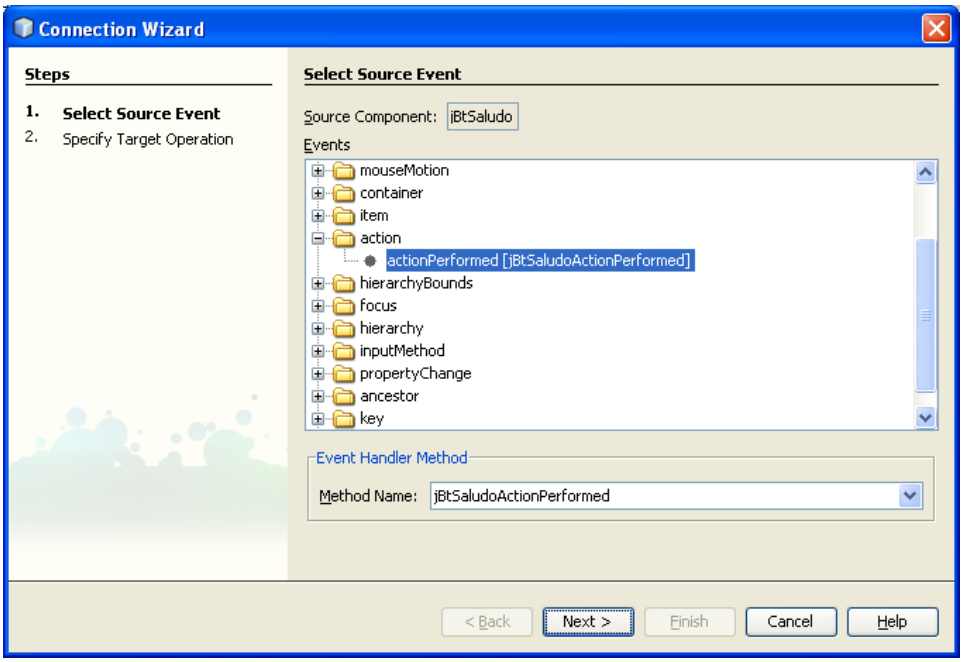


3. A continuación, haga clic primero en el componente sobre el que ocurrirá el evento (en el ejemplo, sobre el botón) y después en el componente sobre el que se realizará la operación (en el ejemplo, sobre la etiqueta). Puede hacer las operaciones indicadas directamente sobre los componentes del formulario,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

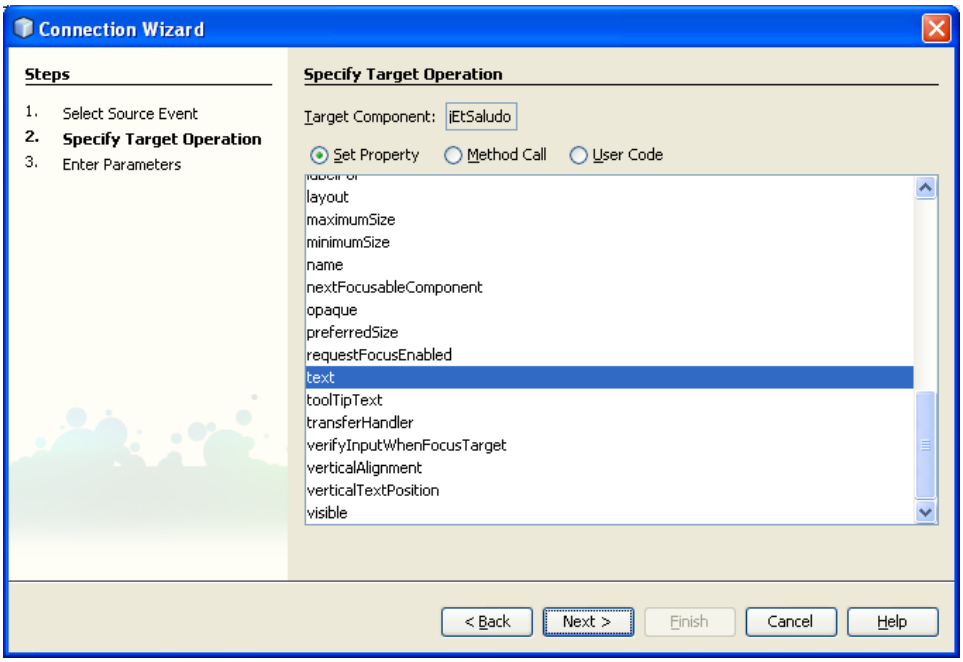
   Cuando haya ejecutado este punto, *NetBeans* abre el asistente que le permitirá realizar la conexión en dos o tres pasos.

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.





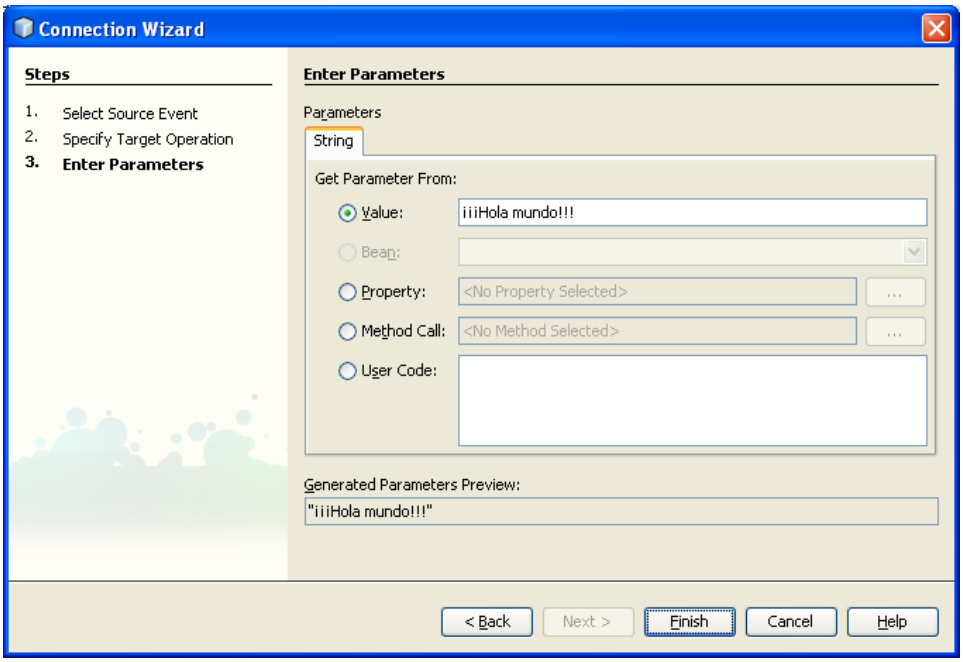
5. Haga clic en el botón *Next* para especificar la operación en el destino (*Specify Target Operation*). En este segundo paso, el diálogo que se muestra permite especificar la operación que tiene que realizarse en el componente seleccionado como destino de la misma (*Set Property > text*):



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.



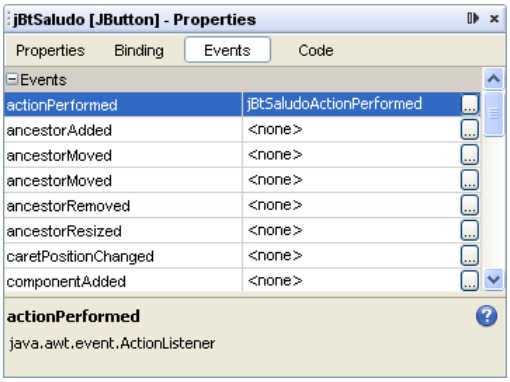
El resultado será que a *CAplicacion* se añade el código siguiente:

```
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:



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:

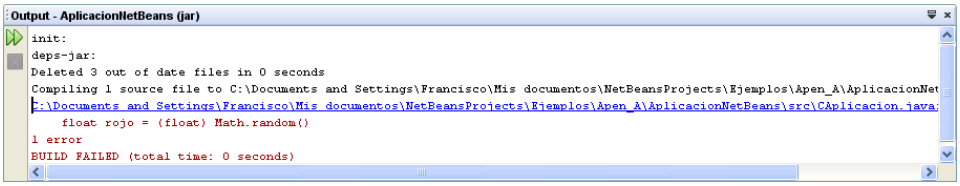
```
private void jBtSaludoActionPerformed(java.awt.event.ActionEvent evt)
{
  float rojo = (float)Math.random();
  float verde = (float)Math.random();
  float azul = (float)Math.random();
  jEtSaludo.setForeground(new java.awt.Color(rojo, verde, azul));
  jEtSaludo.setText("¡¡¡Hola mundo!!!");
}
```

Compilar la aplicación

Una vez finalizada la aplicación, puede compilarla y ejecutarla. Para compilar la aplicación, ejecute la orden *Build Main Project* (*F11*) del menú *Build*.

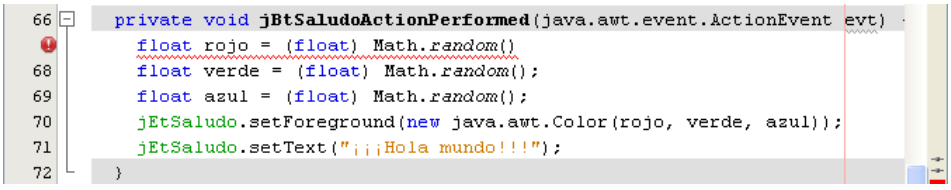
Si la construcción del fichero *.class* resulta satisfactoria, verá en la ventana de salida del compilador el mensaje "*BUILD SUCCESSFUL*". Si hay problemas con la construcción, verá los mensajes de error correspondientes en la misma ventana.

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 punto de inserción sobre ella, le será mostrado un mensaje corto indicando el error cometido:

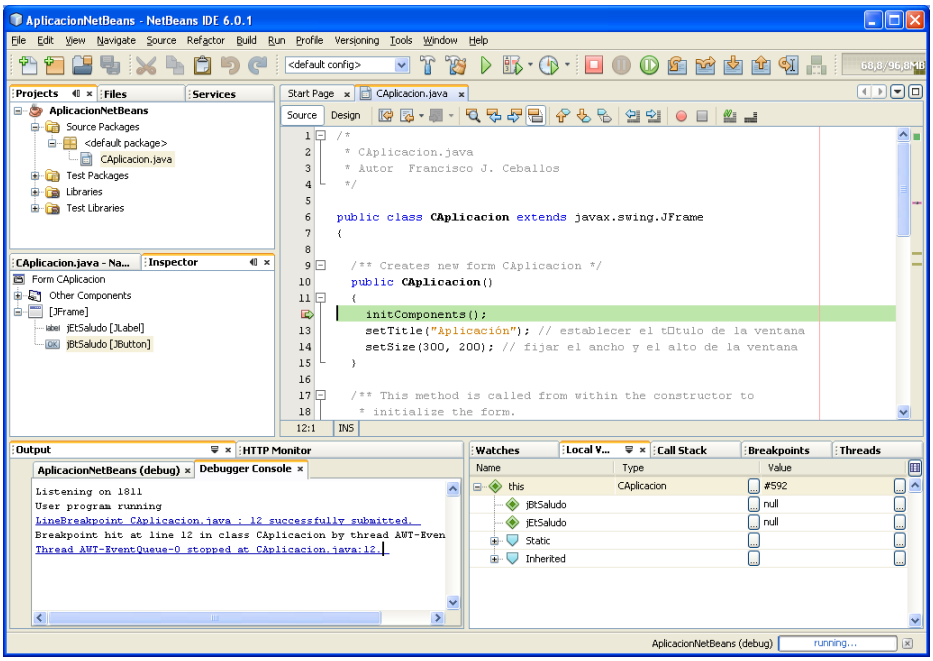


Para ejecutar la aplicación, seleccione la orden *Run Main Project* del menú *Run*. Si no hay errores de ejecución, *NetBeans* mostrará los resultados.

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 Breakpoint* (*Ctrl+F8*) del menú contextual que se visualiza.

2. Ejecute la orden *Debug Main Project* (*Ctrl+F5*) del menú *Run*. 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 *Local Variables*, *Call Stack* y *Watches* que puede observar en la figura anterior.

   La depuración puede también iniciarse ejecutando la orden *Step Into* (*F7*) del menú *Run*. 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 *Run*.

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 *Watches* 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, haga clic sobre la variable utilizando el botón derecho del ratón, ejecute la orden *New Watch* del menú contextual que se visualiza y pulse el botón *OK* de la ventana que se muestra. Para eliminar una variable del panel de inspección, haga clic sobre ella utilizando el botón derecho del ratón y ejecute la orden *Delete* del menú contextual que se visualiza.

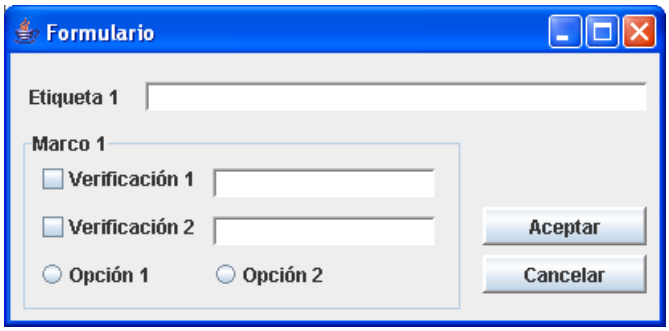
   Así mismo, en el panel *Local Variables* puede ver la lista de variables locales, en el panel *Call Stack* la pila de llamadas y en el panel *Breakpoints* los puntos de parada.

   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 *Run* 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 éste, 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 *Advanced Options*, nodo *Editing – GUI Builder*, panel *Expert*, propiedades *Grid X* y *Grid Y*).

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 supervisor de componentes (*Inspector*) 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; 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, la ventana de diálogo *Abrir* o *Guardar* proporcionada 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:

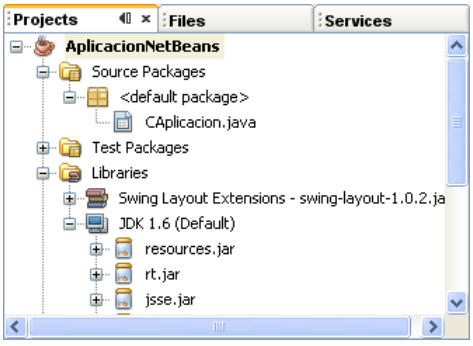
1. Añada un objeto **JDialog** desde la paleta de componentes *Swing*. Una vez añadido, diríjase al supervisor de componentes, haga doble clic sobre el nuevo diálogo añadido y complételo de acuerdo a sus necesidades.

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 la pestaña *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.

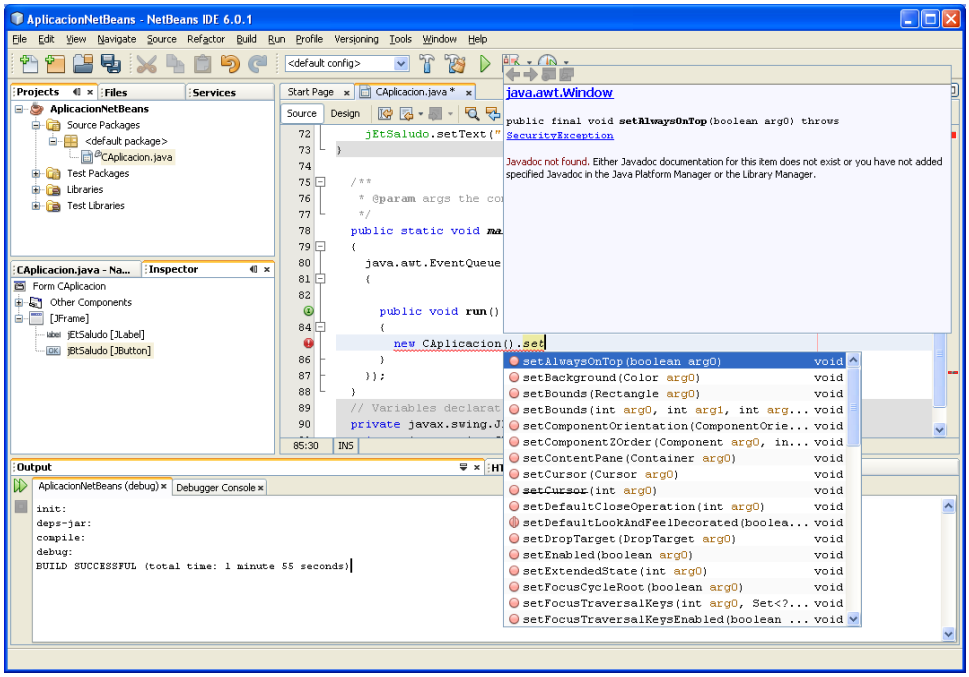


Puede tener cargados varios proyectos simultáneamente y activar en cada instante aquél sobre el que desea trabajar. Para ello, tiene que hacer clic sobre el nombre del proyecto que desea sea el actual y ejecutar la orden *Set as Main Project* del menú contextual que se visualiza.

De forma predeterminada la codificación de los proyectos *NetBeans* es UTF-8. Si utiliza variables o constantes que incluyen vocales acentuadas y otras letras especiales del idioma español, debe especificar que se utilice el juego de caracteres de ISO-8859-1, conocido también como Latín 1. Para ello, 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:

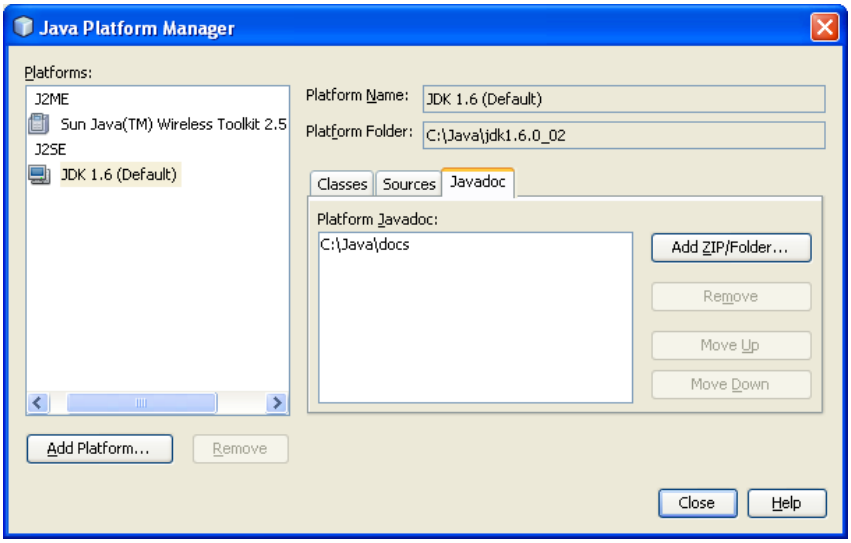


Por ejemplo, según puede verse en la figura anterior, cuando haya escrito *CAplicacion()*. aparecerá una lista de los posibles métodos que pueden ser invocados; seleccione el adecuado y pulse la tecla *Entrar*.

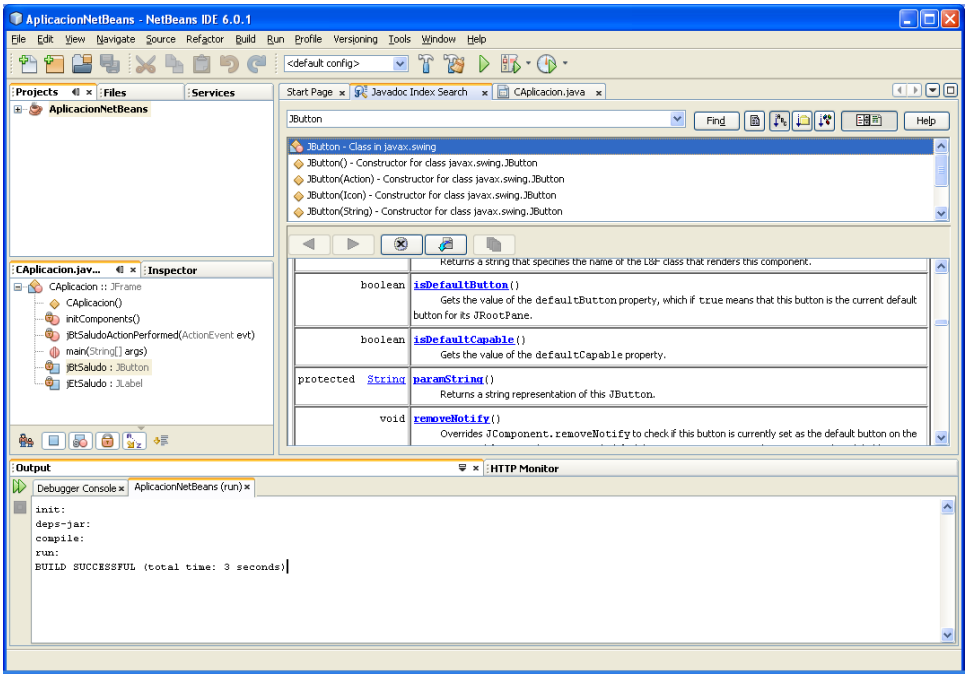
Este tipo de ayuda sólo estará disponible si el valor de la casilla de verificación *Auto Popup Completion Window* tiene el valor *true*. Para verlo, ejecute la orden *Options* del menú *Tools* y haga clic en el botón *Editor > General*.

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.

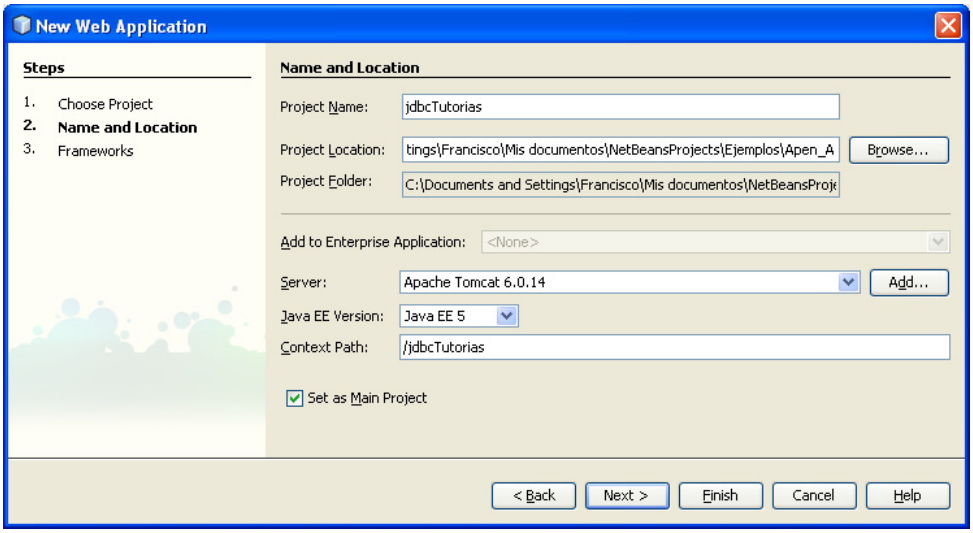
5. Si es necesario, añada una página *html* para desplegarlo. Para ello, seleccione *New* > *Other* > *Other* > *HTML File*.

APLICACIÓN JSP-SERVLET

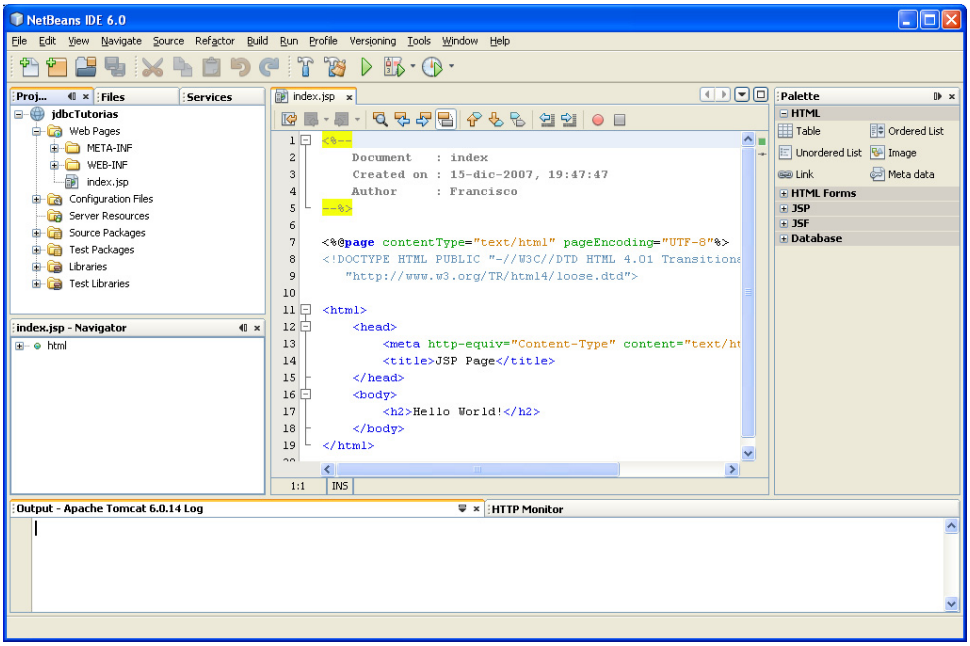
En los capítulos 9 y 10 explicamos cómo desarrollar aplicaciones para Internet utilizando *servlets* y *JSPs*. *NetBeans* también permite este tipo de desarrollos.

Como ejemplo de desarrollo de una aplicación para Internet con *NetBeans*, vamos a reproducir la misma aplicación que desarrollamos en el capítulo 9 que permitía a un alumno concertar una tutoría (*cap09\jdbcTutorias*). Recuerde que esta aplicación estaba formada por el *servlet SvTutorias*, la clase *BDTutorias*, el fichero *tutorias.html* y accedía a la base de datos *bd_tutorias*.

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



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*.

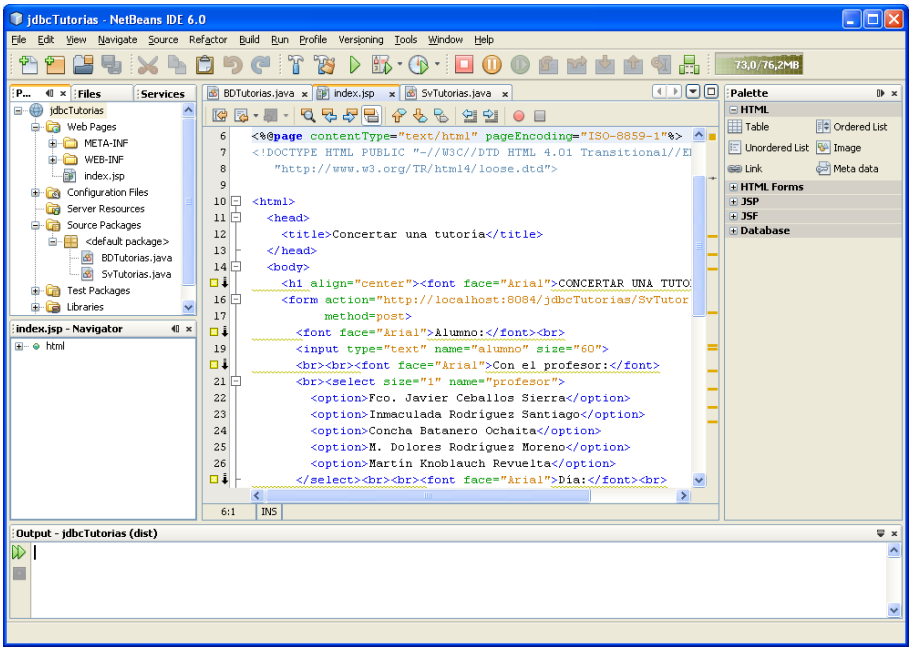
Escriba entre las etiquetas <*html*> y </*html*> del fichero *index.jsp* el contenido del fichero *tutorias.html*. Así, este fichero pasa a llamarse ahora *index.jsp*. Puede también, si lo desea, cambiar su nombre y editar el fichero *web.xml* del nodo *Configuration Files* para hacer referencia al mismo.

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

```
<form action="http://localhost:8084/jdbcTutorias/SvTutorias"
```

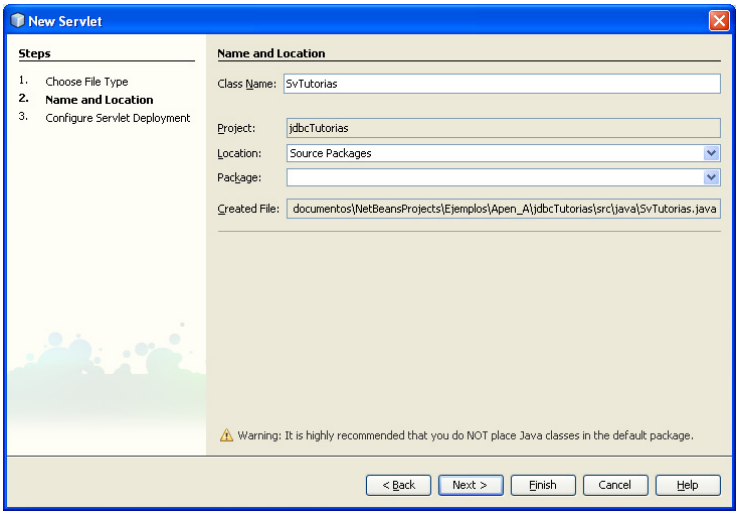
Si utiliza variables o constantes que incluyen vocales acentuadas y otras letras especiales del idioma español, debe especificar que se utilice el juego de caracteres de ISO-8859-1 conocido también como Latín 1:

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
```

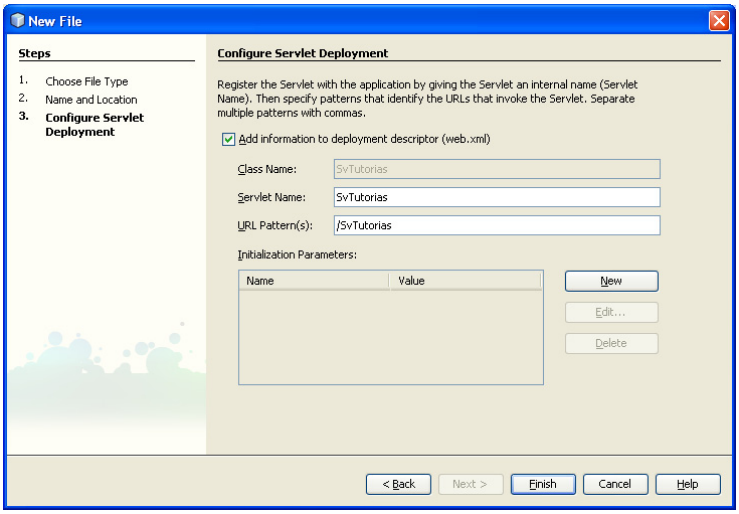


Para añadir a la aplicación un *servlet* o cualquier otro elemento, los pasos siempre son los mismos: clic con el botón derecho del ratón sobre la carpeta donde se va a guardar el elemento, y seleccionar *New* > *elemento a añadir*.

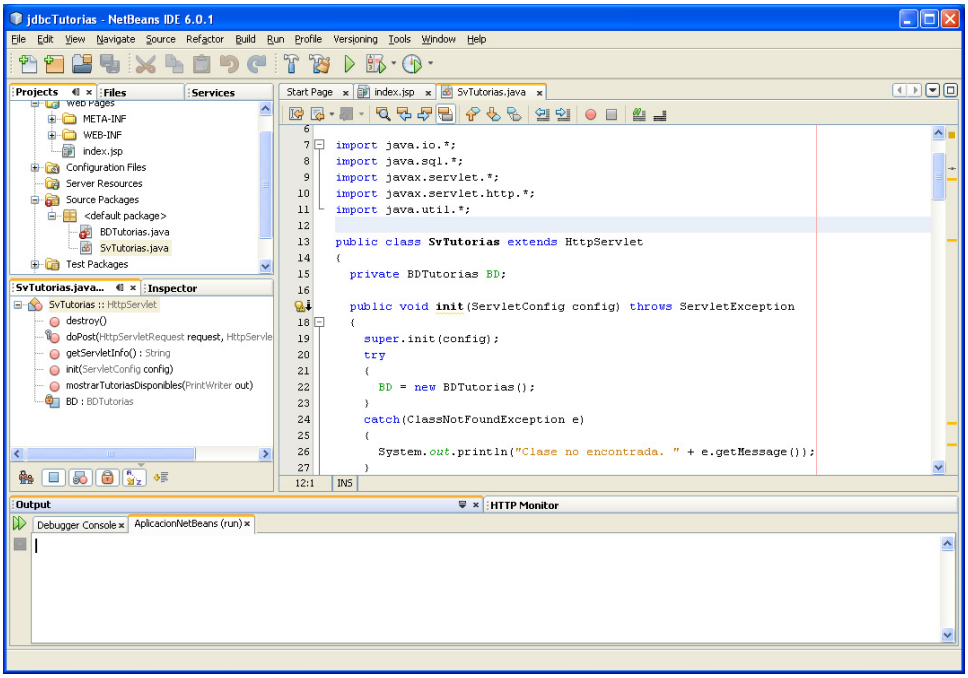
Así, para añadir el *servlet SvTutorias*, haga clic con el botón derecho del ratón sobre el nombre *jdbcTutorias* del proyecto, y seleccione *New* > *Servlet*. En la ventana que se visualiza, introduzca el nombre del fichero *SvTutorias*.



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 que la etiqueta <*welcome-file*> especifica el fichero *index.jsp*.

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 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.

POOL DE CONEXIONES: CONFIGURAR GLASSFISH

1. Iniciar GlassFish.

2. http://localhost/4848.

3. Application Server > JVM Settings > Path Settings > Classpath Suffix > ruta conector java. Por ejemplo: C:\Java\glassfish-v2\lib\mysql-connector-java-5.1.5-bin.jar.

4. Resources > JDBC > Connection Pools > New > Name, Resource Type, Database Vendor. Por ejemplo: MySqlPool, javax.sql.DataSource, MySQL.

5. Resources > JDBC > Connection Pools > MySqlPool > Additional Properties (xxx): editar las propiedades DatabaseName, ServerName, URL). Por ejemplo: bd_tutorias, localhost, jdbc:mysql://localhost:3306/bd_tutorias.

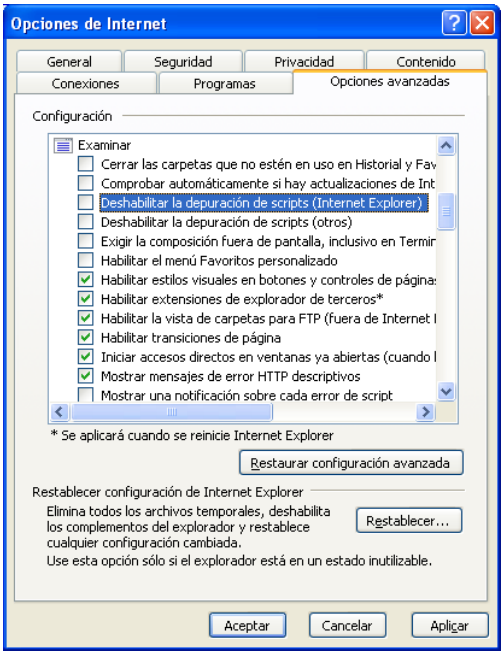
6. Verificar si es correcta la configuración: Resources > JDBC > Connection Pools > MySqlPool > Ping. El resultado tiene que ser: Ping Succeeded.

7. Crear una conexión JDBC: Resources > JDBC > JDBC Resources > New > JNDI Name, Pool Name. Por ejemplo: jdbc/bd_tutorias, MySqlPool.

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.

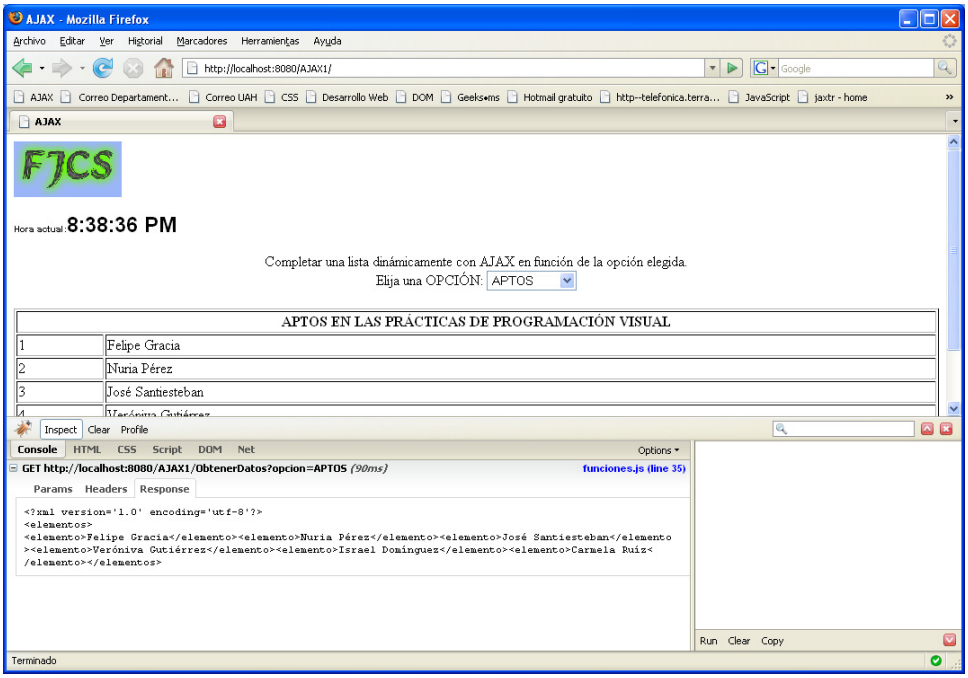
Para poder depurar código JavaScript, el primer paso es habilitar esta opción en el navegador. En el caso de Internet Explorer, seleccione *Herramientas* > *Opciones de Internet* > *Opciones avanzadas* y asegúrese que no esté seleccionada la opción "Deshabilitar la depuración de *scripts*":



Cumplido el requisito anterior, sólo 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/).



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 (*MySQL*), el entorno de desarrollo *NetBeans* o el servidor de aplicaciones *Tomcat*. Este apéndice le indicará de forma breve cómo instalar estos paquetes y cómo ponerlos en marcha.

J2SE 6.0

Java 2 Platform Standard Edition Development Kit 6.0 (abreviadamente, ***J****2SE* ***D****evelopment* ***K****it 6.0* o *JDK 6.0*) 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 J2SE 6.0, siga los pasos indicados a continuación:

1. Ejecute el fichero *jdk-6uxx-windows-i586-p.exe* localizado en el CD que acompaña al libro, o el que haya descargado de Internet.

2. Siga los pasos indicados durante la instalación.

3. Si a continuación desea instalar la documentación, descomprima el fichero *jdk-6-doc.zip* localizado en el CD que acompaña al libro, o el que haya descargado de Internet. Puede instalarla en *jdk1.6.0_xx\docs*.

NetBeans 6.0

NetBeans 6.0 es un entrono 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 (*Apache Tomcat 6.0*) y 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. Ejecute el fichero *netbeans-6.xxx-windows.exe* localizado en el CD que acompaña al libro, o el que haya descargado de *http://www.NetBeans.org*.

2. Realice una instalación personalizada, ya que *Tomcat* no se instala por defecto, y a continuación siga los pasos indicados durante la instalación.

3. Para obtener ayuda acerca de la biblioteca de J2SE 6.0, sólo 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.

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/* o bien, realice la instalación desde el fichero *mysql-*

essential-5.0.xx-win32.msi, suministrado en el CD que acompaña al libro, siguiendo las instrucciones expuestas a continuación:

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 5.0/"
datadir="C:/Archivos de programa/MySQL/MySQL Server 5.0/Data/"
```

Una vez finalizada la instalación, es necesario instalar también el controlador para poder conectarse a *MySQL* vía *JDBC*. Para ello:

1. Descomprima el fichero *mysql-connector-java-5.1.xx.zip* localizado en el CD que acompaña al libro o si lo prefiere, puede descargar la última versión de la dirección de Internet anteriormente indicada.

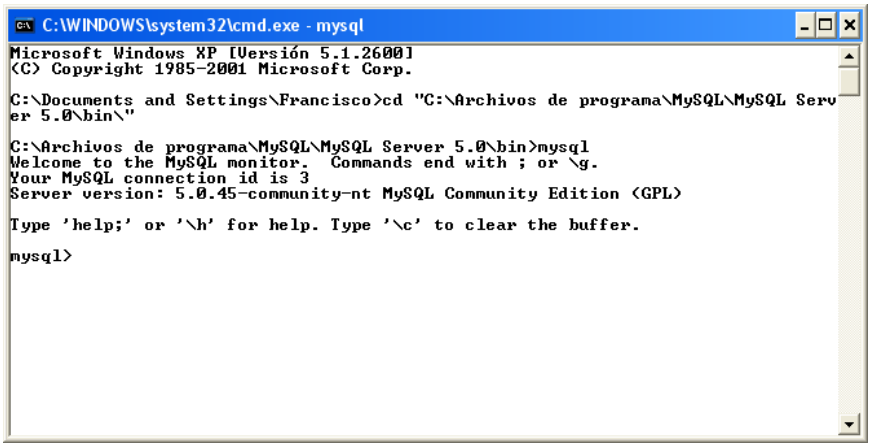
2. Una vez descomprimido, sólo tiene que copiar el fichero *mysql-connector-java-5.1.xx-bin.jar* en la carpeta *.../jre/lib/ext* de la instalación de Java y, cuando lo necesite, establecer la variable *classpath* para que incluya esa ruta. Por ejemplo:

```
set classpath=.;C:\Java\jdk1.6.0_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* > *Rendimiento y mantenimiento* > *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*; éste 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, opcionalmente puede instalar las aplicaciones con interfaz gráfica *MySQL Administrator* y *MySQL Query Browser*, así como *MySQL Migration Toolkit*. Se trata de un software de libre distribución del cual puede obtener versiones actualizadas en la dirección de Internet *http://dev.mysql.com/downloads/*. Para instalar estas utilidades, descárguelas del sitio Web *http://dev.mysql.com/* o bien realice la instalación desde el fichero *mysql-gui-tools-5.0-rxx-win32.msi* suministrado en el CD que acompaña al libro, siguiendo las instrucciones indicadas por el programa de instalación.

El administrador de *MySQL* (*MySQL Administrator*) es un programa para ejecutar operaciones de administración, tales como configuración del servidor *MySQL*, supervisión de su estado y rendimiento, arranque y parada del mismo,

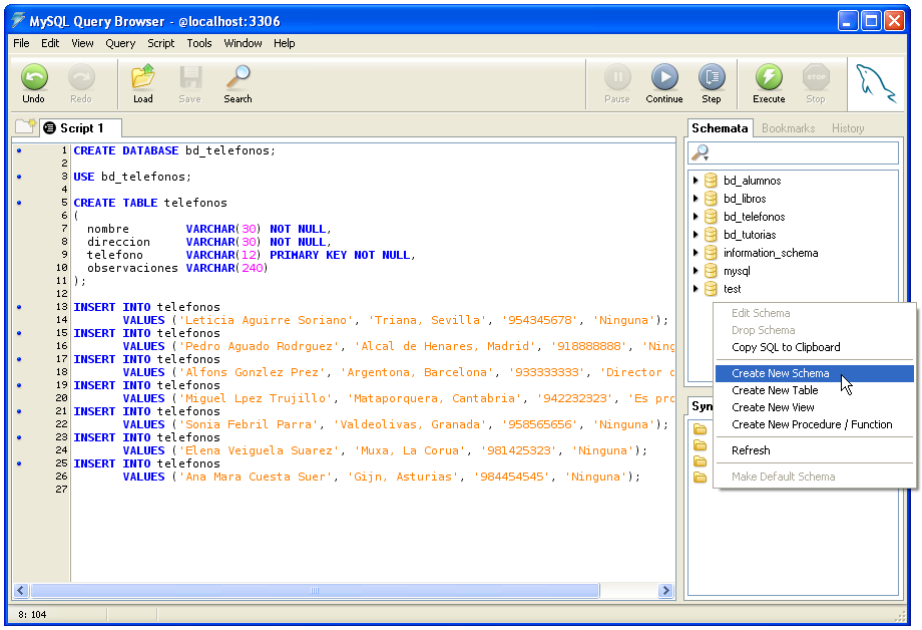
gestión de usuarios y conexiones, realización de copias de seguridad y un buen número de otras tareas administrativas.

El analizador de consultas de *MySQL* (*MySQL Query Browser*) es una utilidad gráfica diseñada para crear, ejecutar y optimizar consultas en un entorno gráfico. Mientras que el administrador de *MySQL* está diseñado para administrar el servidor de base de datos, el analizador de consultas está diseñado para facilitar las consultas y el análisis de los datos almacenados en una base *MySQL*.

La utilidad gráfica para migración de datos, como su nombre indica, permite migrar bases de datos relacionales de otros fabricantes a *MySQL*.

CREAR UNA BASE DE DATOS

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



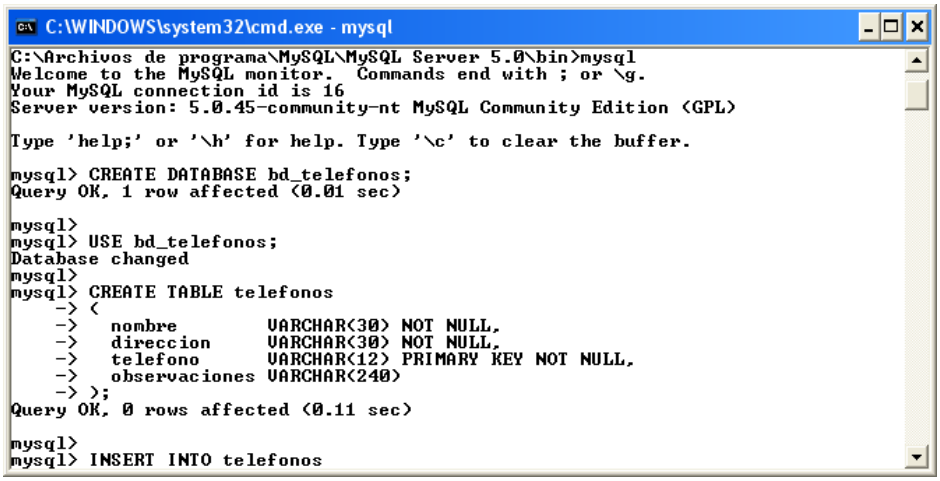
También tenemos la posibilidad de crear la base de datos desde la utilidad *MySQL Query Browser* seleccionando las órdenes adecuadas de sus menús contextuales; esto es, bastaría con hacer clic con el botón derecho del ratón en el panel *Schemata* y ejecutar *Create New Schema* del menú contextual que se

visualiza, o bien ejecutar la orden *Create New Table* para añadir una tabla al esquema que esté seleccionado. En la figura anterior puede apreciar estos detalles.

También, ejecutando la orden *File > New Script Tab* puede abrir un panel para escribir un guión como el que muestra la figura anterior y ejecutarlo haciendo clic en el botón *Execute* de la barra de herramientas.

El guión que muestra la figura anterior crea la base de datos *bd_telefonos*. Dicho guión puede obtenerlo de la carpeta *Cap07* del CD que acompaña al libro.

Si en lugar de emplear la utilidad *MySQL Query Browser* 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 "**>**".

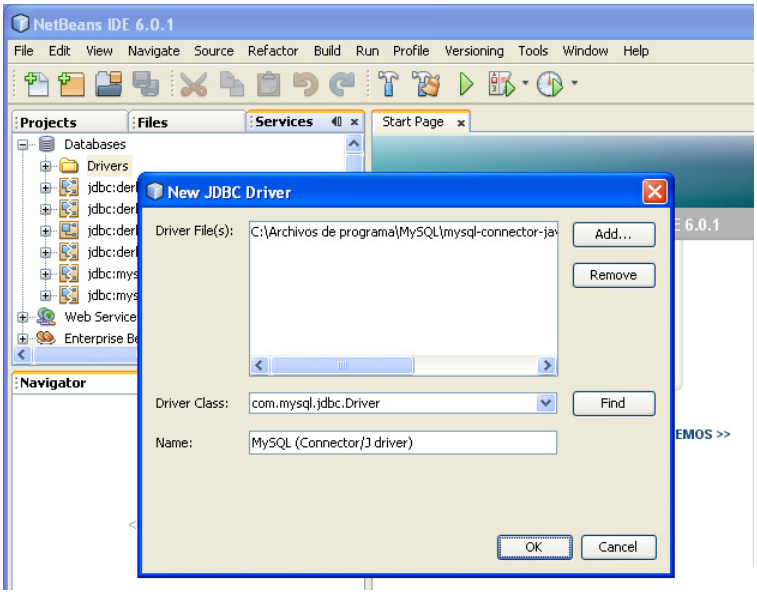


Si utilizamos el EDI *NetBeans 6.0* procederíamos según se expone a continuación. Mostramos el panel *Services* y expandimos el nodo *Databases > Drivers*, hacemos clic con el botón secundario del ratón en *MySQL (Connector/J driver)* y seleccionamos *Connect Using*. Se muestra el diálogo *New Database Connection*; escriba en la caja *Database URL*: `jdbc:mysql:///`; después introduzca el usuario y la contraseña, si procede. 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 *Add 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

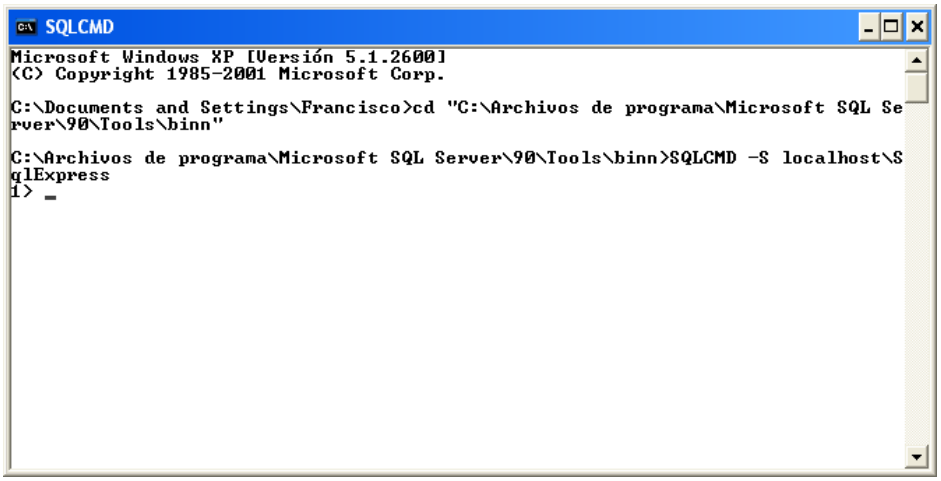
SQL Server 2005 Express es el motor de base de datos gratuito, potente, pero sencillo, que se integra perfectamente con el resto de productos Express. Se trata de una versión aligerada de la nueva generación de SQL Server.

Este producto tiene el mismo motor de base de datos que toda la familia SQL Server 2005 y utiliza el mismo lenguaje SQL.

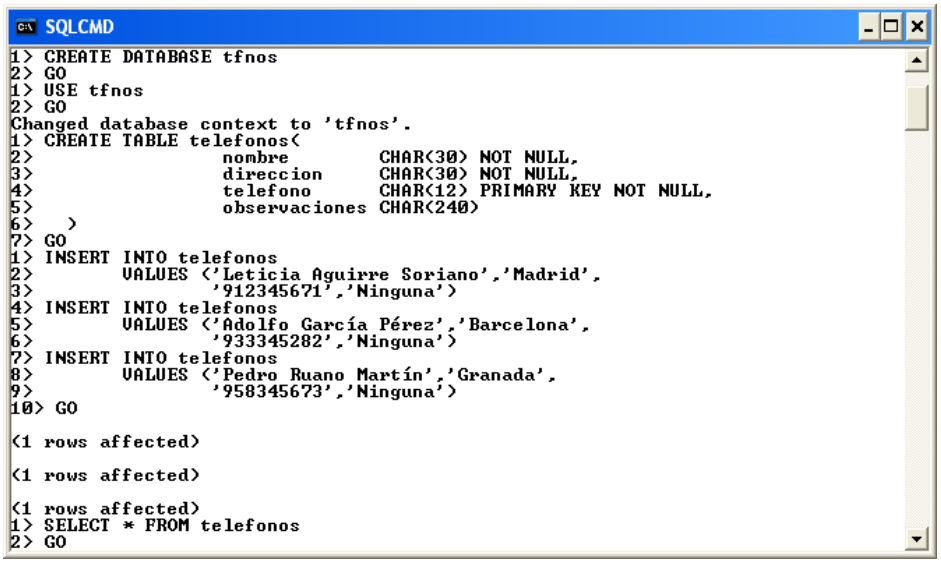
Para crear una base de datos utilizando SQL Server 2005 Express tiene que hacerlo desde la línea de órdenes. Para iniciar la consola que le permita trabajar contra el motor de base de datos SQL Server, localice en su instalación (*C:\Archi-*

vos de programa\Microsoft SQL Server\90\Tools\Binn) el fichero SQLCMD.EXE (o bien SQLCMD90.EXE), cambie a ese directorio y ejecute la orden:

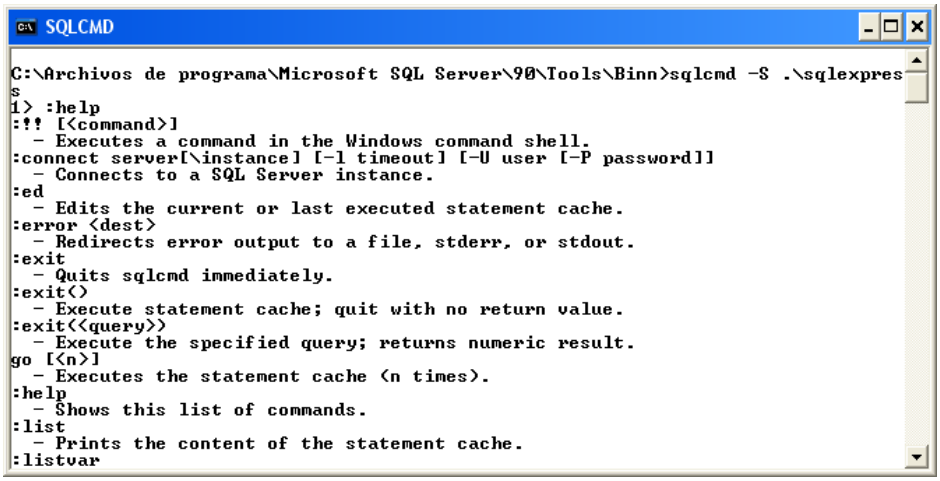
```
SQLCMD -S nombre-del-ordenador\SqlExpress
```



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 (:).



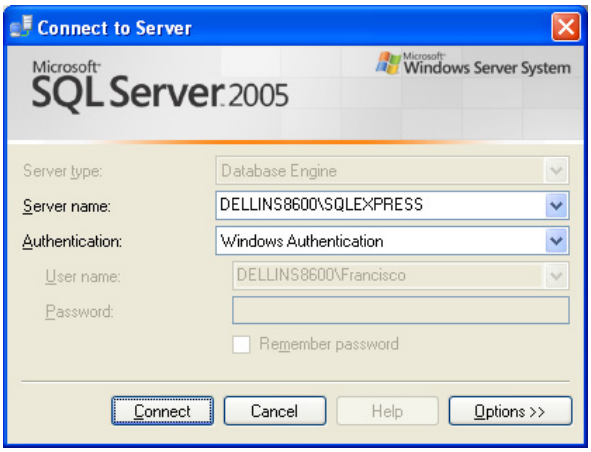
SQL SERVER MANAGEMENT STUDIO EXPRESS

Si instaló SQL Server 2005, habrá comprobado que la única herramienta que proporciona al usuario es *SQL Computer Manager* que sirve para gestionar los servicios básicos de SQL Server y para configurar los protocoles de red. 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 (en el CD del libro se adjunta una copia):

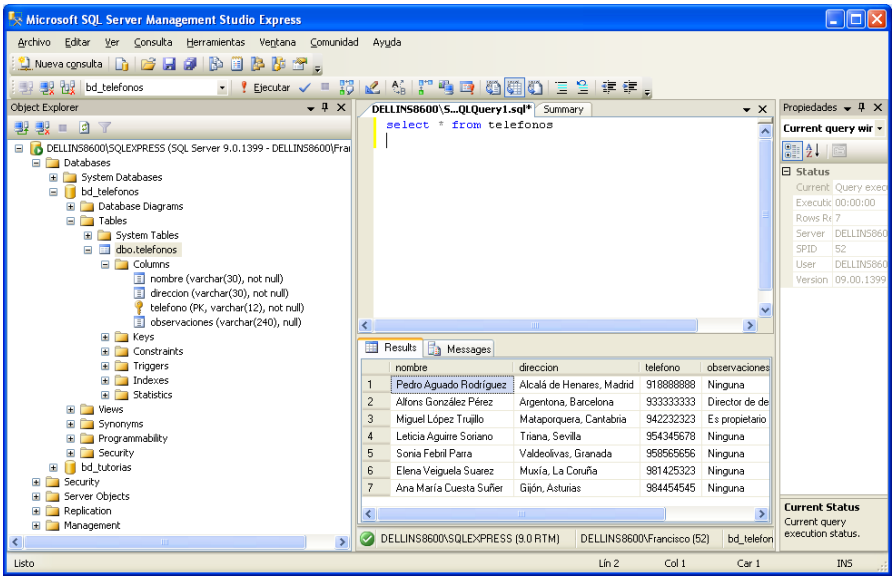
http://www.microsoft.com/downloads

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, 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 sólo 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 6

Tomcat 6.0 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 sólo 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 J2SE 6.0, la instalación sobre Windows de *Tomcat 6.0* es muy sencilla:

1. Ejecute el fichero *apache-tomcat-6.0.xx.exe* localizado en el CD que acompaña al libro, o el 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* sólo 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 el contenido de un fichero JAR.
- *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="FichApplet.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\jdk1.6.0\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 sólo 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	DEC	CAR	DEC	CAR	DEC	CAR
33	!	89	Y	146	'	202	Ê
34	"	90	Z	147	``	203	Ë
35	#	91	[	148	''	204	Ì
36	$	92	\	149	o	205	Í
37	%	93	]	150	–	206	Î
38	&	94		151	—	207	Ï
39	'	96	`	152	▒	208	Ð
40	(	97	a	153	▒	209	Ñ
41	)	98	b	154	▒	210	Ò
42	*	99	c	155	▒	211	Ó
43	+	100	d	156	▒	212	Ô
44	,	101	e	157	▒	213	Õ
45	-	102	f	157	▒	214	Ö
46	.	103	g	159	▒	215	×
47	/	104	h	160		216	Ø
48	0	105	i	161	¡	217	Ù
49	1	106	j	162	¢	218	Ú
50	2	107	k	163	£	219	Û
51	3	108	l	164	¤	220	Ü
52	4	109	m	165	¥	221	Ý
53	5	110	n	166	¦	222	Þ
54	6	111	o	167	§	223	ß
55	7	112	p	168	¨	224	à
56	8	113	q	169	©	225	á
57	9	114	r	170	ª	226	â
58	:	115	s	171	«	227	ã
59	;	116	t	172	¬	228	ä
60	<	117	u	173	-	229	å
61	=	118	v	174	®	230	æ
62	>	119	w	175	¯	231	ç
63	?	120	x	176	°	232	è
64	@	121	y	177	±	233	é
65	A	122	z	178	²	234	ê
66	B	123	{	179	³	235	ë
67	C	124	|	180	´	236	ì
68	D	125	}	181	µ	237	í
69	E	126	~	182	¶	238	î
70	F	127	▒	183	·	239	ï
71	G	128	▒	184	¸	240	ð
72	H	129	▒	185	¹	241	ñ
73	I	130	▒	186	º	242	ò
74	J	131	▒	187	»	243	ó
75	K	132	▒	188	¼	244	ô
76	L	133	▒	189	½	245	õ
77	M	134	▒	190	¾	246	ö
78	N	135	▒	191	¿	247	÷
79	O	136	▒	192	À	248	ø
80	P	137	▒	193	Á	249	ù
81	Q	138	▒	194	Â	250	ú
82	R	139	▒	195	Ã	251	û
83	S	140	▒	196	Ä	252	ü
84	T	141	▒	197	Å	253	ý
85	U	142	▒	198	Æ	254	þ
86	V	143	▒	199	Ç	255	ÿ
87	W	144	▒	200	È		
88	X	145	`	201	É		

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 sólo si tienen el mismo código Unicode.