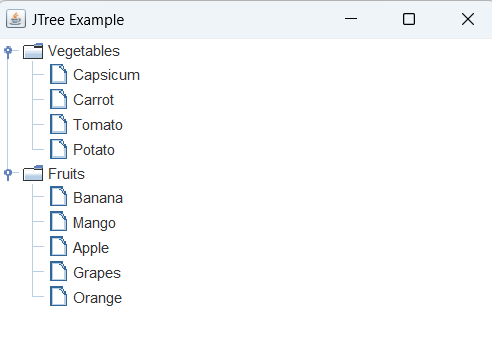
JTREE



**Salida**

**1. Introducción a JTree:**

JTree es un componente de Swing con el que podemos mostrar datos jerárquicos. JTree es un componente bastante complejo. Un JTree tiene un 'nodo raíz' que es el padre superior de todos los nodos del árbol. Un nodo es un elemento en un árbol. Un nodo puede tener muchos nodos secundarios. Estos propios nodos secundarios pueden tener más nodos secundarios. Si un nodo no tiene ningún nodo hijo, se denomina nodo hoja.

El nodo de hoja se muestra con un indicador visual diferente. Los nodos con hijos se muestran con un indicador visual diferente junto con un "controlador" visual que se puede utilizar para expandir o contraer ese nodo. Al expandir un nodo se muestran los elementos secundarios y al contraerlos se ocultan.

**2. Desarrollando un JTree simple:**

Intentemos ahora construir un JTree simple. Digamos que queremos mostrar la lista de verduras y frutas de forma jerárquica.

El nodo está representado en Swing API como TreeNode, que es una interfaz. La interfaz MutableTreeNode extiende esta interfaz que representa un nodo mutable. Swing API proporciona una implementación de esta interfaz en forma de clase DefaultMutableTreeNode.

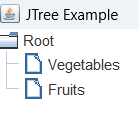
Usaremos la clase DefaultMutableTreeNode para representar nuestro nodo. Esta clase se proporciona en la API Swing y podemos usarla para representar nuestros nodos. Esta clase tiene un práctico método add() que toma una instancia de MutableTreeNode.

Entonces, primero crearemos el nodo raíz. Y luego podemos agregar nodos de forma recursiva a esa raíz. Comencemos con una raíz simple con sólo 2 nodos:

|  |  |
| --- | --- |
|  | **package** net.codejava.swing;  **import** javax.swing.JFrame;  **import** javax.swing.JTree;  **import** javax.swing.SwingUtilities;  **import** javax.swing.tree.DefaultMutableTreeNode;  **public** **class** TreeExample **extends** JFrame  {  **private** JTree tree;  **public** TreeExample()      {          //create the root node          DefaultMutableTreeNode root = **new** DefaultMutableTreeNode("Root");          //create the child nodes          DefaultMutableTreeNode vegetableNode = **new** DefaultMutableTreeNode("Vegetables");          DefaultMutableTreeNode fruitNode = **new** DefaultMutableTreeNode("Fruits");          //add the child nodes to the root node          root.add(vegetableNode);          root.add(fruitNode);            //create the tree by passing in the root node          tree = **new** JTree(root);          add(tree);    **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  **this**.setTitle("JTree Example");  **this**.pack();  **this**.setVisible(**true**);      }    **public** **static** **void** main(String[] args)      {          SwingUtilities.invokeLater(**new** Runnable() {              @Override  **public** **void** run() {  **new** TreeExample();              }          });      }  } |

Como se explicó anteriormente, creamos un nodo raíz y le agregamos nodos secundarios. Tenga en cuenta que, cuando creamos la instancia de JTree, simplemente pasamos la instancia del nodo raíz. Esto se debe a que el nodo raíz contiene toda la información, incluidos los hijos.

Veamos como quedaría nuestro ejemplito:



**Salida inicial con 2 hijos**

El resultado muestra el nodo raíz y los dos hijos. Observe la diferencia en el icono que se muestra antes del texto. Los nodos que tienen hijos se indican con un icono de carpeta y los nodos hoja se muestran con un icono diferente.

**3. Añadiendo más hijos a JTree**

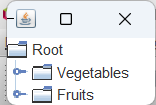
Agreguemos ahora más hijos al nodo de verduras y frutas. Esto es muy sencillo. Solo necesitamos agregar instancias de DefaultMutableTreeNode a la instancia de vegetalNodo y fruta, como:

|  |  |
| --- | --- |
|  | DefaultMutableTreeNode vegetableNode = **new** DefaultMutableTreeNode("Vegetables");  vegetableNode.add(**new** DefaultMutableTreeNode("Capsicum"));  vegetableNode.add(**new** DefaultMutableTreeNode("Carrot"));  vegetableNode.add(**new** DefaultMutableTreeNode("Tomato"));  vegetableNode.add(**new** DefaultMutableTreeNode("Potato"));  DefaultMutableTreeNode fruitNode = **new** DefaultMutableTreeNode("Fruits");  fruitNode.add(**new** DefaultMutableTreeNode("Banana"));  fruitNode.add(**new** DefaultMutableTreeNode("Mango"));  fruitNode.add(**new** DefaultMutableTreeNode("Apple"));  fruitNode.add(**new** DefaultMutableTreeNode("Grapes"));  fruitNode.add(**new** DefaultMutableTreeNode("Orange")); |

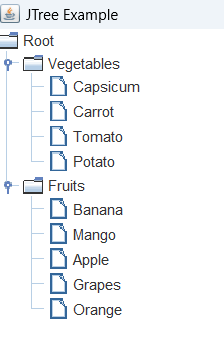
Cuando ejecutamos el programa obtenemos la siguiente salida:

**Más hijos añadidos**

Podemos ver que el ícono que se muestra para los nodos 'Verduras' y 'Frutas' ha cambiado de un ícono de hoja a un ícono de carpeta. Esto indica que estos nodos ahora tienen hijos. Además, observe el icono del controlador (el que parece una llave) que se muestra junto a estos nodos. Se puede hacer clic en ellos para expandir los nodos.



Ahora ejecutemos el programa nuevamente y hagamos clic en ambos identificadores para expandirlos. Obtenemos el siguiente resultado:

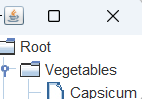


**Árbol expandido**

Una vez que expandimos los nodos, podemos ver todos los elementos del árbol mostrados en una bonita estructura jerárquica. También tenga en cuenta que el controlador se muestra con un icono indicador diferente.

**4. Customizando el Display de JTree:**

Intentemos ahora personalizar la visualización de un JTree. Ahora ejecutemos el programa nuevamente y hagamos clic en las llaves. Luego, cambie el tamaño del marco para usar menos altura. Obtendremos el siguiente resultado:



**Tree sin Scrollbars**

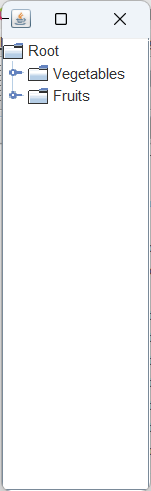
Como podemos ver, cuando se cambia el tamaño del marco, los elementos quedan ocultos. En cambio, lo que queremos es una barra de desplazamiento y permitir al usuario desplazarse hacia arriba y hacia abajo para ver los datos completos del árbol.

**5. Añadiendo un Scrollpane para JTree**

Hacer esto es muy sencillo. Como muchos otros componentes de Swing, sólo necesitamos envolver nuestro árbol dentro de un JScrollPane y agregarlo al marco. En lugar de agregar el árbol al marco, agregamos el panel de desplazamiento, como:

add(new JScrollPane(tree));

Ahora ejecutemos el programa nuevamente, hagamos clic en los íconos del controlador para expandir los nodos y cambiar el tamaño del marco. Obtendremos un resultado como este:



**Tree con Scrollbars**

Podemos ver que ahora se agrega una barra de desplazamiento y podemos desplazarnos para ver los nodos. Ahora, intenta colapsar uno de los nodos. Podemos ver que la barra de desplazamiento desaparece. La barra de desplazamiento aparece cuando tiene elementos más allá del área de visualización.

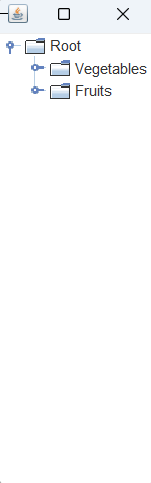
**6. Mostrando identificadores de raíz para JTree:**

Cuando vuelves a observar de cerca el resultado, vemos que los nodos 'Verduras' y 'Frutas' tienen el icono de 'controlador', pero el nodo 'Raíz' no. El nodo raíz es el padre de todos los nodos, por lo que también sería posible expandirlo y contraerlo. Si hace doble clic en el nodo raíz, puede contraerlo y expandirlo. Sin embargo, esto no es muy conveniente ni coherente con el resto del árbol.

Entonces, intentemos ahora mostrar el identificador del nodo raíz. Esto se puede hacer con una simple llamada API como:

tree.setShowsRootHandles(true);

Ejecutemos ahora el programa nuevamente. Podemos ver que se muestra el identificador raíz:



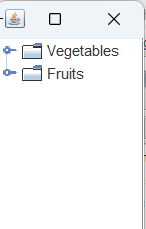
**Tree con Root Handles Displayed**

**7. Ocultando el nodo Root de JTree**

El nodo raíz es el más alto de la jerarquía. Sin embargo, es posible que en algunos casos no sea necesario mostrar el nodo raíz. Por ejemplo, hemos estado usando un nodo raíz que se muestra con el texto "Raíz". Esto no es muy útil. Entonces, ¿qué pasa si queremos ocultarlo? Esto es posible con una simple llamada API:

tree.setRootVisible(false);

Ejecutemos ahora el programa. Obtenemos lo siguiente cuando se expanden los árboles:



**Tree con nodo Root oculto**

Tenga en cuenta que aún hacemos la llamada a tree.setShowsRootHandles(true) ya que de lo contrario no veremos los identificadores raíz para los nodos 'Verduras' y 'Frutas'.

**8. Cambiando los iconos de un JTree**

JTree usa diferentes íconos para representar nodos hoja y nodos con hijos como hemos visto arriba. ¿Qué pasa si necesitamos proporcionar nuestros propios íconos para este propósito? Es muy posible. Necesitamos hacer uso del renderizador para hacer esto.

**Representación de JTree:**

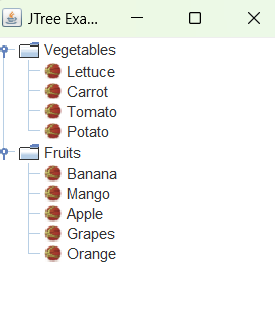
JTree delega la visualización de sus elementos a los renderizadores. De forma predeterminada, se crea automáticamente un renderizador para que un JTree muestre todos sus elementos. La representación está representada por una interfaz llamada TreeCellRenderer. La API Swing proporciona una implementación predeterminada de esta interfaz conocida como DefaultTreeCellRenderer. Esta clase tiene algunos métodos útiles que podemos usar para personalizar la visualización.

**Usando DefaultTreeCellRenderer:**

Crearemos una instancia de la clase DefaultTreeCellRenderer y usaremos el método setLeafIcon() para configurar el ícono para todos nuestros nodos hijo. Este método toma una instancia de la clase ImageIcon. La clase ImageIcon se utiliza para manejar la visualización de imágenes como ya hemos visto. Entonces, primero crearemos una instancia de ImageIcon y luego la usaremos con el renderizador:

|  |  |
| --- | --- |
|  | Icon leafIcon = new ImageIcon("/Imagenes/bola.png");  DefaultTreeCellRenderer renderer = new DefaultTreeCellRenderer();  renderer.setLeafIcon(leafIcon); |

La primera línea utiliza el mecanismo estándar para cargar imágenes donde la imagen es parte de la aplicación (empaquetada dentro del jar). Luego creamos una instancia de DefaultTreeCellRenderer y llamamos al método setLeafIcon() pasando esta instancia de ImageIcon. Ahora ejecutemos el programa y expandamos los nodos para ver la imagen que se utiliza para los nodos hoja.



**Árbol con imagen propia para los nodos hojas (Sin hijos)**

**9. Establecer controladores de eventos para JTree**

Intentemos ahora desarrollar controladores de eventos para el árbol. Conocer el nodo actualmente seleccionado será uno de los eventos más útiles de conocer. El manejo de eventos en JTree es muy similar al de otros componentes de Swing. Registramos un detector de eventos y recibimos una notificación cuando ocurre el evento.

**Desarrollando un SelectionListener:**

Necesitamos agregar un TreeSelectionListener para escuchar los eventos de selección. Esta es una interfaz definida en Swing API y necesitamos implementar el método valueChanged(). La fuente de la selección se pasa como parámetro a este método.

La selección en JTree es manejada por una clase llamada TreeSelectionModel. Entonces, JTree delega todo el trabajo relacionado con la selección a esta clase.

Primero agregaremos un JLabel que se mostrará en la parte inferior. Siempre que se seleccione un nodo en el árbol, mostraremos la ruta del nodo seleccionado.

Primero declaremos una variable de instancia JLabel:

private JLabel selectedLabel;

Luego la agregaremos a la parte inferior del marco:

selectedLabel = new JLabel();

add(selectedLabel, BorderLayout.SOUTH);

Luego agregaremos el listener de selección de la siguiente manera:

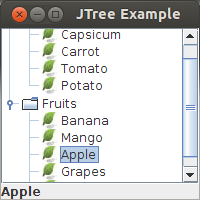
|  |  |
| --- | --- |
|  | tree.getSelectionModel().addTreeSelectionListener(**new** TreeSelectionListener() {      @Override  **public** **void** valueChanged(TreeSelectionEvent e) {          selectedLabel.setText(e.getPath().toString());      }  }); |

Agregamos un listener de selección al modelo de selección de árbol. TreeSelectionListener tiene un método que implementamos. Usamos el origen del evento para invocar el método getPath() y configurarlo en la etiqueta que agregamos anteriormente.

Un uso aún mejor del oyente de selección sería obtener el nodo seleccionado y utilizarlo. Para obtener el nodo seleccionado, podemos utilizar el método getLastSelectedPathComponent() de JTree. Este método devuelve el nodo seleccionado. Luego podemos invocar el método getUserObject() en la clase DefaultMutableTreeNode que devuelve el objeto real que agregamos. Dado que este método devuelve una instancia de Objeto, debemos llamar al método toString() y agregarlo a la etiqueta. Podemos modificar el código de la siguiente manera:

|  |  |
| --- | --- |
|  | tree.getSelectionModel().addTreeSelectionListener(**new** TreeSelectionListener() {      @Override  **public** **void** valueChanged(TreeSelectionEvent e) {          DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();          selectedLabel.setText(selectedNode.getUserObject().toString());      }  }); |

Cuando ejecutamos el programa y seleccionamos un nodo hoja, obtenemos el siguiente resultado:

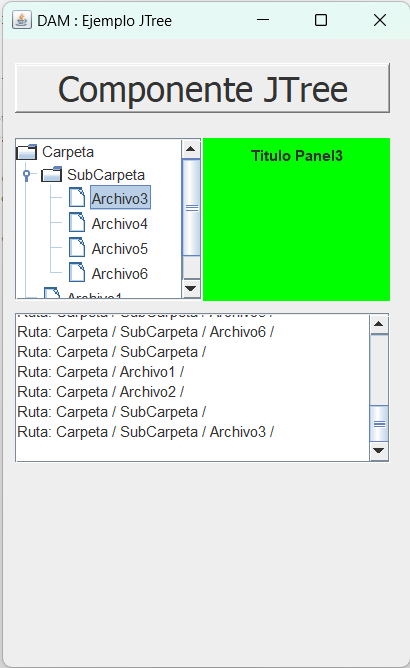


**Listener de selección para obtener el nodo seleccionado**

|  |  |
| --- | --- |
|  | **package** net.codejava.swing;  **import** java.awt.BorderLayout;  **import** javax.swing.ImageIcon;  **import** javax.swing.JFrame;  **import** javax.swing.JLabel;  **import** javax.swing.JScrollPane;  **import** javax.swing.JTree;  **import** javax.swing.SwingUtilities;  **import** javax.swing.event.TreeSelectionEvent;  **import** javax.swing.event.TreeSelectionListener;  **import** javax.swing.tree.DefaultMutableTreeNode;  **import** javax.swing.tree.DefaultTreeCellRenderer;  /\*\*   \* JTree basic tutorial and example   \* @author wwww.codejava.net   \*/  **public** **class** TreeExample **extends** JFrame  {  **private** JTree tree;  **private** JLabel selectedLabel;    **public** TreeExample()      {          //create the root node          DefaultMutableTreeNode root = **new** DefaultMutableTreeNode("Root");          //create the child nodes          DefaultMutableTreeNode vegetableNode = **new** DefaultMutableTreeNode("Vegetables");          vegetableNode.add(**new** DefaultMutableTreeNode("Capsicum"));          vegetableNode.add(**new** DefaultMutableTreeNode("Carrot"));          vegetableNode.add(**new** DefaultMutableTreeNode("Tomato"));          vegetableNode.add(**new** DefaultMutableTreeNode("Potato"));            DefaultMutableTreeNode fruitNode = **new** DefaultMutableTreeNode("Fruits");          fruitNode.add(**new** DefaultMutableTreeNode("Banana"));          fruitNode.add(**new** DefaultMutableTreeNode("Mango"));          fruitNode.add(**new** DefaultMutableTreeNode("Apple"));          fruitNode.add(**new** DefaultMutableTreeNode("Grapes"));          fruitNode.add(**new** DefaultMutableTreeNode("Orange"));          //add the child nodes to the root node          root.add(vegetableNode);          root.add(fruitNode);            //create the tree by passing in the root node          tree = **new** JTree(root);          Icon leafIcon = new ImageIcon("Imagenes/bola.png");  tree.setEditable(true);  DefaultTreeCellRenderer renderer2 = new DefaultTreeCellRenderer();  renderer2.setLeafIcon(leafIcon);  tree.setCellRenderer(renderer2);          tree.setShowsRootHandles(**true**);          tree.setRootVisible(**false**);          add(**new** JScrollPane(tree));            selectedLabel = **new** JLabel();          add(selectedLabel, BorderLayout.SOUTH);          tree.getSelectionModel().addTreeSelectionListener(**new** TreeSelectionListener() {              @Override  **public** **void** valueChanged(TreeSelectionEvent e) {                  DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();                  selectedLabel.setText(selectedNode.getUserObject().toString());              }          });    **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  **this**.setTitle("JTree Example");  **this**.setSize(200, 200);  **this**.setVisible(**true**);      }    **public** **static** **void** main(String[] args)      {          SwingUtilities.invokeLater(**new** Runnable() {              @Override  **public** **void** run() {  **new** TreeExample();              }          });      }  } |

EJERCICIOS JTREE

1)



\*En los nodos hoja (sin hijos) cambiar el icono por defecto a uno que os guste.

2) JTree + JTable + PopUpMenu