UNIVERSIDAD TECNOLOGICA NACIONAL

FACULTAD REGIONAL GENERAL PACHECO

TÉCNICO SUPERIOR EN PROGRAMACIÓN

PROGRAMACIÓN III

APUNTE TEORICO

PARTE 2

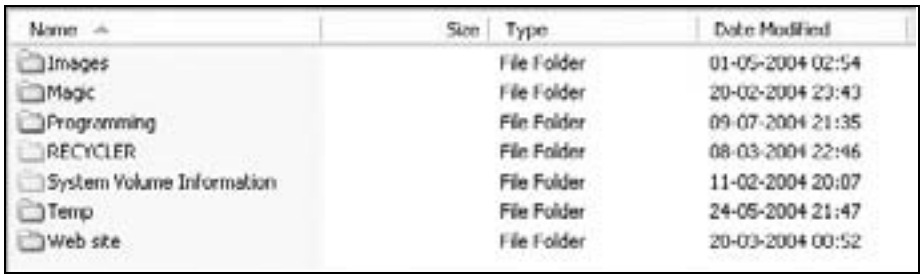
APLICACIÓNES WINDOWS II

EL CONTROL LISTVIEW

El Control ListView se utiliza generalmente para presentar datos, permitiéndole al usuario cierto control de detalle y personalización.

Los datos se pueden mostrar como columnas y filas al igual que una cuadricula o como una sola columna y presentando diferentes iconos. El control ListView es bastante complejo, pero donde mas se lo utiliza es para navegar por las carpetas de una PC.

Veremos algunas de sus propiedades, eventos y métodos más importantes.



Propiedades más usadas en el Control ListView:

PROPIEDADES	DESCRIPCIÓN
Activación	*Controla como un usuario activa un elemento del ListView.*
Alignment	*Controla como los elementos están alineados.*
AllowColumnReorder	*Si se establece en True, permite cambiar el orden de las columnas.*
AutoArrange	*Si se establece en True, permite alinear los elementos en forma automática.*
CheckBoxes	*Si es True, todos los elementos de la lista tendrán un CheckBox a la izq.*
CheckedIndices CheckedItems	*Le da acceso a una colección de los índices y los artículos, respectivamente, que contiene los elementos activados en la lista.*
Columns	*Esta propiedad le da acceso a la colección de columnas a través del cual se pueden agregar o eliminarlos.*
FocusedItem	*Contiene el elemento que tiene el foco en el listview. Si no hay nada seleccionado es nulo.*
FullRowSelect	*True selecciona toda la fila.*
GridLines	*True muestra una línea entre filas y columnas dibujando una cuadricula.*
HeaderStyle	*Controla como se muestran los encabezados de columnas.*

HoverSelection	*True, puede seleccionar un elemento de la lista pasando el Mouse por encima.*
Items	*La colección de Items del ListView.*
LabelEdit	*True, puede editar el contenido de la etiqueta de la primera columna.*
LabelWrap	*True, ajusta la etiqueta en tantas líneas como sea necesario para mostrar todo el texto.*
LargeImageList	*Mantiene el ImageList, que tiene imágenes de gran tamaño. Estas imágenes se pueden utilizar cuando la propiedad View es LargeIcon.*
MultiSelect	*True, permite la selección múltiple.*
Scrollable	*True, muestra el scrollbar.*
SelectedIndices SelectedItems	*Contiene las colecciones que mantienen los índices y los artículos que son seleccionados, respectivamente.*
SmallImageList	*Cuando la propiedad View está SmallIcon, esta propiedad tiene el ImageList que contiene las imágenes utilizadas.*
Sorting	*Permite ordenar los elementos que contiene el ListView. Hay tres los modos posibles: ascendente, descendente, y Ninguno.*
StateImageList	*El ImageList contiene máscaras para las imágenes que se utilizan como sobre impresiones en las imágenes y LargeImageList SmallImageList para representar los estados personalizados.*
View	*El ListView puede mostrar sus elementos en cuatro modos básicos: LargeIcon: Todos los elementos se muestran con un icono grande (32 × 32) y una etiqueta. SmallIcon: Todos los elementos se muestran con un pequeño icono (16×16) y una etiqueta. Lista: Sólo una columna en la pantalla. Esa columna puede contienen un icono y una etiqueta. Detalles: cualquier número de columnas puede ser mostrado. Sólo la primera columna puede contener un icono.azulejo (sólo disponible en Windows XP y las nuevas plataformas de Windows): Muestra un icono grande con una etiqueta y la información subtem a la derecha del icono.*
TopItem	*Devuelve el elemento en la parte superior de la vista de lista.*

Eventos más usados en el Control ListView:

EVENTOS	DESCRIPCIÓN

AfterLabelEdit	*Se dispara cuando una etiqueta fue modificada.*
BeforeLabelEdit	*Se dispara antes de que un usuario modifique una etiqueta.*
ColumnClick	*Se dispara cuando una columna es clickeada.*
ItemActivate	*Se dispara cuando un ítem es seleccionado*

Métodos más usados en el Control ListView:

METODOS	DESCRIPCIÓN
BeginUpdate()	*Le dice al ListView para dejar de dibujar actualizaciones hasta EndUpdate () es llamado. Es útil cuando se está introduciendo muchos elementos a la vez, ya que deja el punto de vista desde el parpadeo, y aumenta significativamente la velocidad.*
Clear()	*Limpia el ListView completamente.*
EndUpdate()	*Se llama después de BeginUpdate.*
EnsureVisible()	*Permite que la lista se deplase para mostrar un ítem y su índice.*
GetItemAt()	*Devuelve el ListViewItem en la posición x, y en la vista de lista.*

ListViewItem

Un elemento en un ListView es siempre una instancia de la clase ListViewItem. El ListViewItem contiene información como el texto y el índice del icono para mostrar. Los Objetos ListViewItem tienen una propiedad SubItems que contiene las instancias de otra clase, ListViewSubItem.

Estos subtemas se muestran si el Control ListView está en detalles o en el modo de mosaico. Cada uno de los subtemas representa una columna en la vista de lista. La principal diferencia entre los subtemas y los elementos principales es que un subtema no puede mostrar un icono.

Para agregar ListViewItems al ListView a través de la colección Items y ListViewSubItems a un ListViewItem a través de la colección de SubItems en el ListView.

Column Header

Para mostrar el encabezado de las columnas por pantalla se agrega al ListView la instancia de la clase ColumnHeader, en fin Column Header proporciona un Titulo para las columnas.

EL CONTROL IMAGELIST

El control ImageList proporciona una colección que se puede utilizar para almacenar las imágenes utilizadas en otros controles sobre su formulario. Puede almacenar imá-

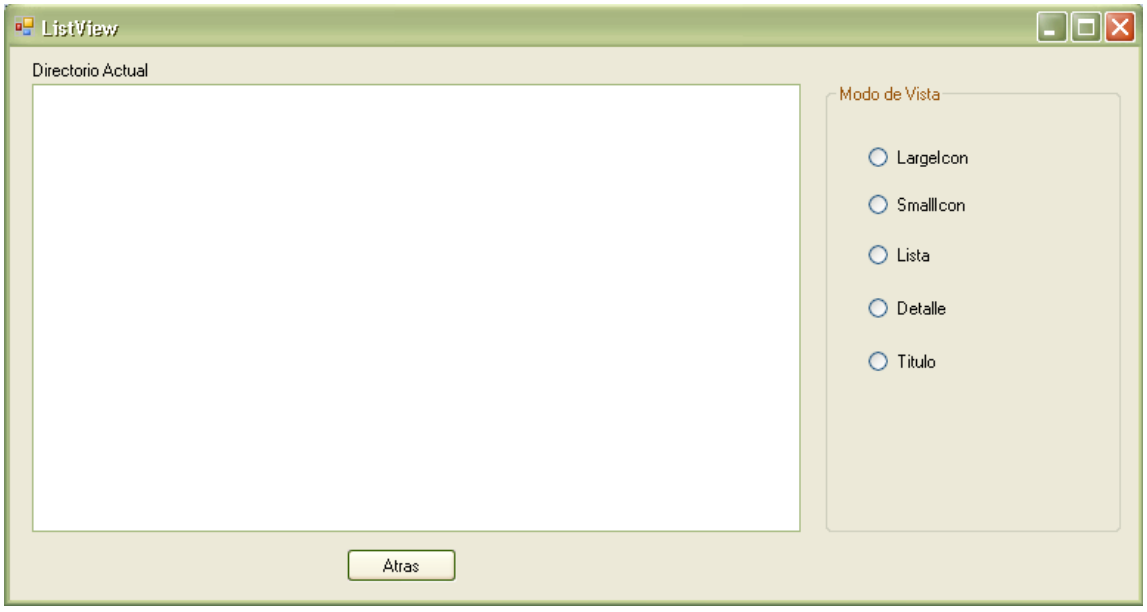
genes de cualquier tamaño en una lista de imágenes, pero dentro de cada control de cada imagen debe ser del mismo tamaño. En el caso de ListView, esto significa que se necesitan dos controles ImageList para mostrar las dos imágenes grandes y pequeñas.

El ImageList no se mostrará en tiempo de ejecución. Cuando lo arrastra a un formulario que está en desarrollo, no se coloca en la misma forma que los demás controles, debajo en una bandeja, que contiene todos los componentes tales. Esta característica interesante es proporcionada para evitar los controles que no forman parte de la interfaz de usuario y evitar la confusión en el Diseñador de formularios. El control se manipula exactamente de la misma manera que cualquier otro control, excepto que no se puede mover en el formulario. Se pueden agregar imágenes al ImageList en tiempo de diseño y en tiempo de ejecución. En tiempo de diseño las imágenes se pueden agregar haciendo clic en el botón derecho de la propiedad imágenes. Con ello se abre un cuadro de diálogo en el que se puede navegar en las imágenes que desea insertar. Si decide añadir las imágenes en tiempo de ejecución, se agregan a través de la colección de imágenes.

PRACTICA 9 (IMAGELIST LISTVIEW)

En la siguiente practica, se crea un Formulario co un ListView y dos ImgeList. El ListView mostrara los archivos y carpetas de la PC. Para no complicar el ejercicio no se mostraran los iconos correctos para las carpetas y archivos, si un icono estándar para cada uno de ellos. Haciendo doble clic en las carpetas, puede navegar por el árbol de carpetas, y un botón para retroceder en el árbol. Además se agregan Cinco RadioButtos que se utilizan para cambiar el modo de la vista del ListView en tiempo de ejecución. Si se hace DobleClick en un archivo, se intentara ejecutarlo.

Generar un Formulario con los controles y diseño que se muestran en la siguiente imagen:



1. Los RadioButton están dentro de un control GroupBox. Cambiar la propiedad AutoSize a false del control Label (Directorio Actual) y luego manualmente cambiar el largo del label al mismo largo que el control ListView.

2. Cambiar la propiedad text de los controles como se muestra en la figura y la propiedad name por el nombre que usted elija.

3. Limpiar la propiedad Text del Label.

4. Agregar dos Controles ImageList haciendo doble click en este control en la caja de herramientas sección componentes y cambie la propiedad name de cada uno por imageListSmall e imageListLarge.

5. Cambiar la Propiedad Imagesize del control imageListLarge a 32; 32...

6. Hacer click en el botón derecho del Mouse sobre el ImageList Large y seleccionar la propiedad Elegir Imágenes, esto abrirá el cuadro de dialogo para agregar las imágenes.

7. Elegir la opción agregar y seleccionar los iconos de carpeta y archivos de texto de formato 32 x32. Estos archivos están disponibles para bajar dentro del mismo sitio.

8. Asegurarse que el icono de carpeta este en la parte superior en el Image listLarge.

9. Repetir los pasos 8 y 9 sobre el imageListSmall para los iconos de formato 16 x 16.

10. Cambiar la propiedad checked a true del optionbutton Detalle.

11. Establecer las siguientes propiedades que aparecen en la tabla para el ListView:

PROPIEDADES	VALOR
LargeImageList	*imageListLarge*
SmallImageList	*imageListSmall*
View	*Details*

12. Agrego una variable String Collection para almacenar la ruta de las carpetas que se exploran, esto sirve para cuando hacemos click en el botón atrás, para saber adonde tenemos que regresar.

```
namespace ListView
{
    public partial class frmListView : Form
    {
        private System.Collections.Specialized.StringCollection carpetacol;
        public frmListView()
        {
            InitializeComponent();
        }
    }
}
```

7. No hemos creado los encabezados de columna en el Diseñador de formularios, así que lo hacemos ahora, utilizando un método llamado CrearCabecerayLlenarListView ()

```
        private void CrearCabecerayLlenarListView()
```

```
        {
            ColumnHeader colCab; //Declaro un objeto ColumnHeader.
            colCab = new ColumnHeader(); //Instancio el objeto ColumnHeader.
            colCab.Text = "Nombre Archivo";
            lvCarpetas.Columns.Add(colCab); // Inserto la Cabecera Nombre Archivo.
            colCab = new ColumnHeader();
            colCab.Text = "Tamaño";
            lvCarpetas.Columns.Add(colCab); // Inserto la Cabecera Tamaño.
            colCab = new ColumnHeader();
            colCab.Text = "Ultima Modificacion";
            lvCarpetas.Columns.Add(colCab); // Inserto la Cabecera Ultima Modificacion.
        }
```

La variable colCab se instancia como nueva se cambia el Text y luego se inserta para agregar una nueva columna o Cabecera.

8. Para finalizar debemos mostrar los archivos y carpetas del disco duro, esto lo hacemos con otro método:

```
        private void DibujarListView(string root)
        {
            try
            {
                ListViewItem lvi;
                ListViewItem.ListViewSubItem lvsi;
                if (string.IsNullOrEmpty(root))
                return;
                DirectoryInfo dir = new DirectoryInfo(root);
                DirectoryInfo[] dirs = dir.GetDirectories();
                FileInfo[] files = dir.GetFiles();
                lvCarpetas.Items.Clear();
                lblActualdir.Text = root;
                lvCarpetas.BeginUpdate();
                foreach (DirectoryInfo di in dirs)
                {
                    lvi = new ListViewItem();
                    lvi.Text = di.Name;
                    lvi.ImageIndex = 0;
                    lvi.Tag = di.FullName;
                    lvsi = new ListViewItem.ListViewSubItem();
                    lvsi.Text = "";
                    lvi.SubItems.Add(lvsi);
                    lvsi = new ListViewItem.ListViewSubItem();
                    lvsi.Text = di.LastAccessTime.ToString();
                    lvi.SubItems.Add(lvsi);
                    lvCarpetas.Items.Add(lvi);
                }
                foreach (FileInfo fi in files)
                {
                    lvi = new ListViewItem();
                    lvi.Text = fi.Name;
                    lvi.ImageIndex = 1;
                    lvi.Tag = fi.FullName;
                    lvsi = new ListViewItem.ListViewSubItem();
                    lvsi.Text = fi.Length.ToString();
                    lvi.SubItems.Add(lvsi);
                    lvsi = new ListViewItem.ListViewSubItem();
                    lvsi.Text = fi.LastAccessTime.ToString();
                    lvi.SubItems.Add(lvsi);
                    lvCarpetas.Items.Add(lvi);
                }
                lvCarpetas.EndUpdate();
```

```
            }
            catch (System.Exception err)
            {
                MessageBox.Show("Error: " + err.Message);
            }
        }
```

Antes del primer bloque foreach, se llama a BeginUpdate () en el control ListView. Recordar que el BeginUpdate () en el ListView señala que el mismo esta en actualización hasta que EndUpdate () es llamado. Si no se llama a este método, cada vez que se dibuje un elemento en el ListView el mismo parpadearía y se haría más lenta su actualización.

Los dos bloques foreach contienen el código para agregar las carpetas y archivos respectivamente. Se comienza por crear una nueva instancia de un ListViewItem y establecer la propiedad Text con el nombre del archivo o la carpeta que se va a insertar. El ImageIndex de ListViewItem se refiere al índice de un elemento en una de las listas de imágenes. Por lo tanto, es importante que los iconos tengan los mismos índices en las dos listas de imágenes. Puede utilizar la propiedad Tag para guardar la ruta completa de las carpetas y archivos, para su uso cuando el usuario hace doble clic en el elemento.

A continuación, cree los dos subtemas. Estos simplemente asignan el texto a mostrar y después se añadió a la colección SubItems de ListViewItem. Por último, el ListViewItem se agrega a la colección Items del ListView. El ListView es lo suficientemente inteligente simplemente ignora los subtemas, si el modo de vista no tiene detalles.

```
// Get information about the root folder.
DirectoryInfo dir = new DirectoryInfo(root);
// Retrieve the files and folders from the root folder.
DirectoryInfo[] dirs = dir.GetDirectories();
FileInfo[] files = dir.GetFiles();
```

Estas líneas utilizan las clases del espacio de nombre System.IO, por lo que para poder acceder a ellas debemos agregar en la parte superior del archivo, el espacio de nombre System.IO con la sentencia using.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
```

Los GetDirectories () es el método del objeto DirectoryInfo que devuelve una colección de objetos que representa las carpetas en el directorio que usted está buscando, y los GetFiles () devuelve una colección de objetos que representan los archivos en el directorio actual. Usted puede recorrer estas colecciones, utilizando la propiedad del objeto Name para devolver el nombre del directorio o archivo, y crear un ListViewItem para mantener esta cadena.

9. Llamar los dos métodos en el constructor del form para inicializar las columnas y cargar los items: Al mismo tiempo crear una instancia de carpetaCol para pasarle el directorio raíz. Correr la aplicación.

```
public frmListView()
{
    InitializeComponent();
    carpetacol = new System.Collections.Specialized.StringCollection();
    CrearCabecerayLlenarListView();
    DibujarListView(@"C:\");
    carpetacol.Add(@"C:\");
}
```

10. Para permitir a los usuarios hacer doble clic en un elemento en el ListView para navegar por las carpetas, es necesario agregar el evento ItemActivate. Seleccione el ListView en el diseñador y haga doble clic en el evento ItemActivate. Agregar el siguiente código.

```
private void lvCarpetas_ItemActivate(object sender, EventArgs e)
{
    System.Windows.Forms.ListView lw = (System.Windows.Forms.ListView)sender;
    string filename = lw.SelectedItems[0].Tag.ToString();
    if (lw.SelectedItems[0].ImageIndex != 0)
    {
        try
        {
            System.Diagnostics.Process.Start(filename);
        }
        catch { return; }
    }
    else
    {
        DibujarListView(filename);
        carpetacol.Add(filename);
    }
}
```

La etiqueta del elemento seleccionado contiene la ruta completa al archivo o carpeta que se hizo doble clic. Usted sabe que la imagen con el índice 0 es una carpeta, por lo que puede determinar si el elemento es un archivo o una carpeta en busca de ese índice. Si se trata de un archivo, a continuación, intenta cargarlo. Si se trata de una carpeta, a continuación, llama al método DibujarListView () con la nueva carpeta y luego añade la carpeta a la colección folderCol.

11. Agregamos el evento Click al Botón Atrás para darle funcionalidad y luego agregamos el siguiente código.

```
private void btnAtras_Click(object sender, EventArgs e)
{
    if (carpetacol.Count > 1)
    {
        DibujarListView(carpetacol[carpetacol.Count - 2].ToString());
        carpetacol.RemoveAt(carpetacol.Count - 1);
    }
    else
        DibujarListView(carpetacol[0].ToString());
}
```

Si hay más de un elemento de la colección carpetacol, entonces usted no está en la raíz del navegador, y llama al método DibujarListView () con la ruta a la carpeta anterior. El último elemento de la colección carpetacol es la carpeta actual, por lo que usted necesita tomar la anteúltima. A continuación, eliminar el último elemento de la colección y

hacer que el nuevo elemento sea último ósea la carpeta actual. Si sólo hay un elemento de la colección carpetacol , a continuación, sólo tiene que llamar al método DibujarList-View () con ese elemento.

12. Ahora la aplicación tiene que ser capaz de cambiar la vista del ListView. Hacer Doble Click en cada RadioButton para agregar el evento CheckedChanged y agregar el código correspondiente a cada uno.

```
        private void rbLarge_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton rdb = (RadioButton)sender;
            if (rdb.Checked)
                this.lvCarpetas.View = View.LargeIcon;
        }

        private void rbSmall_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton rdb = (RadioButton)sender;
            if (rdb.Checked)
                lvCarpetas.View = View.SmallIcon;
        }

        private void rbLista_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton rdb = (RadioButton)sender;
            if (rdb.Checked)
                lvCarpetas.View = View.List;
        }

        private void rbDetalle_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton rdb = (RadioButton)sender;
            if (rdb.Checked)
                lvCarpetas.View = View.Details;
        }

        private void rbTitulo_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton rdb = (RadioButton)sender;
            if (rdb.Checked)
                lvCarpetas.View = View.Tile;
        }
```

Se chequea la propiedad Checked del radioButton si es true se cambia a la vista deseada.