Windows Forms kullanarak Graphical User Interface-2

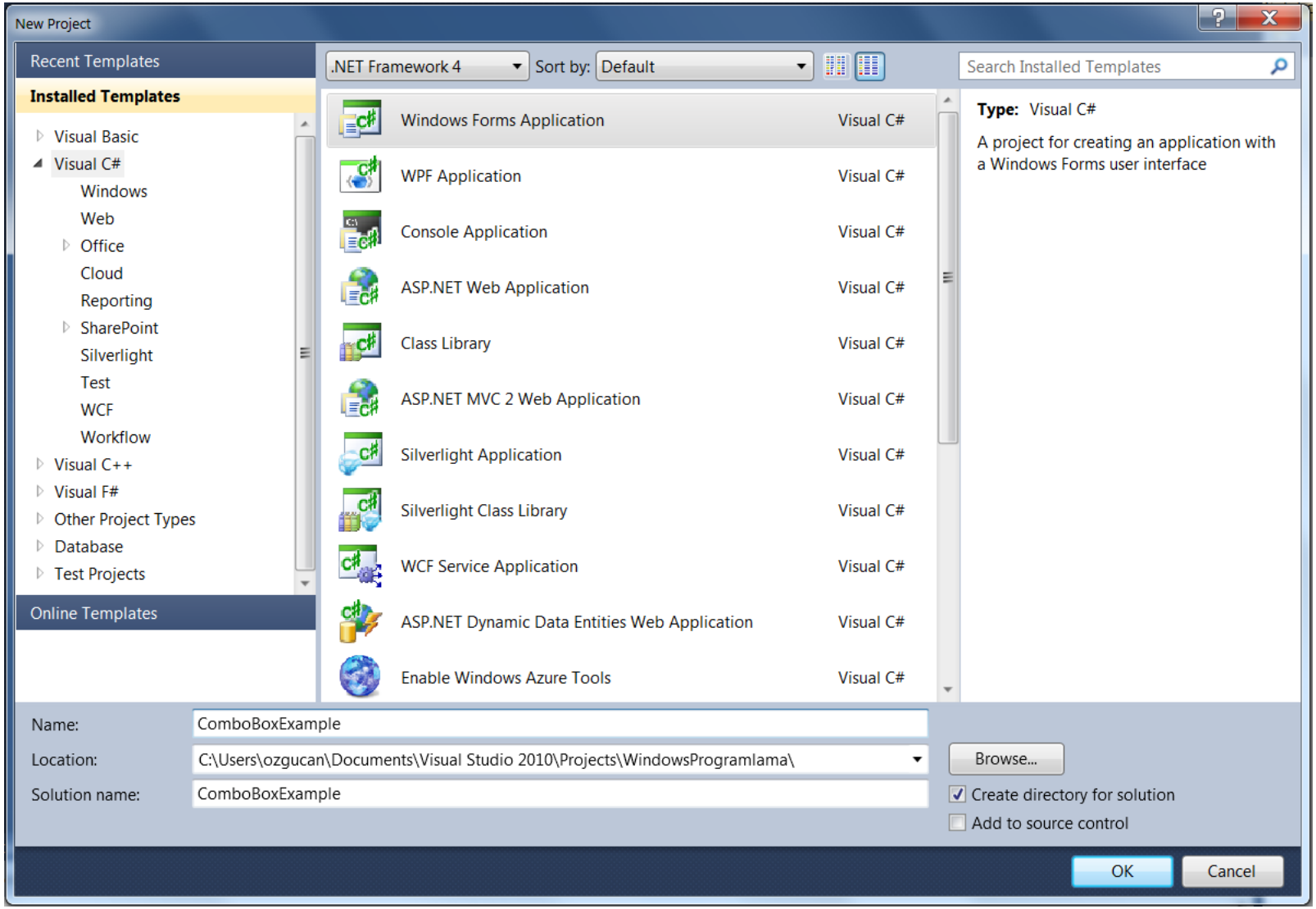
Yrd. Doç. Dr. Özgü Can

ComboBox

- **`TextBox`** ve **drop down list** özelliklerini birleştirir.
- Listede ki öğelerden birini seçmeye izin verir.
- Bir kerede listelenecek **max. öğe sayısı** için:
  - **`MaxDropDownItems`**
- **`ComboBox`**, GUI'lerde yerden kazanım sağlar.

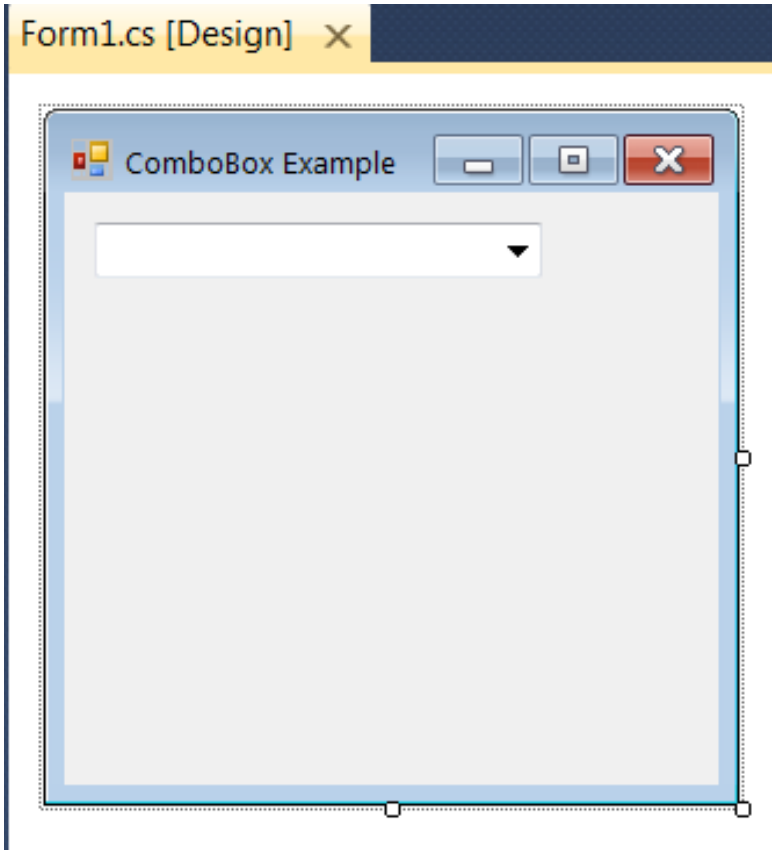
ComboBox properties and an event	Description
Common Properties	
`DropDownStyle`	Determines the type of `ComboBox`. Value `Simple` means that the text portion is editable and the list portion is always visible. Value `DropDown` (the default) means that the text portion is editable but the user must click an arrow button to see the list portion. Value `DropDownList` means that the text portion is not editable and the user must click the arrow button to see the list portion.
`Items`	The collection of items in the `ComboBox` control.
`MaxDropDownItems`	Specifies the maximum number of items (between `1` and `100`) that the drop-down list can display. If the number of items exceeds the maximum number of items to display, a scrollbar appears.
`SelectedIndex`	Returns the index of the selected item, or `-1` if none are selected.
`SelectedItem`	Returns a reference to the selected item.
`Sorted`	Indicates whether items are sorted alphabetically. Setting this property's value to `true` sorts the items. The default is `false`.
Common Event	
`SelectedIndexChanged`	Generated when the selected index changes (such as when a different item is selected). This is the default event when control is double clicked in the designer.

Örnek Uygulama - ComboBox



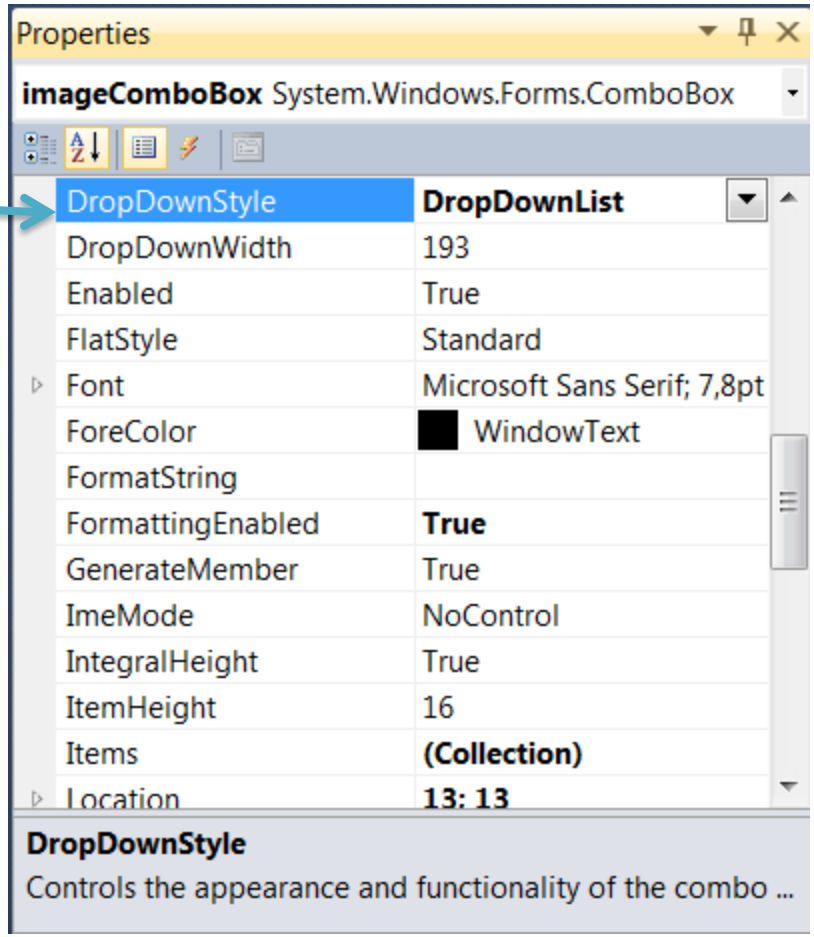
Örnek Uygulama - ComboBox

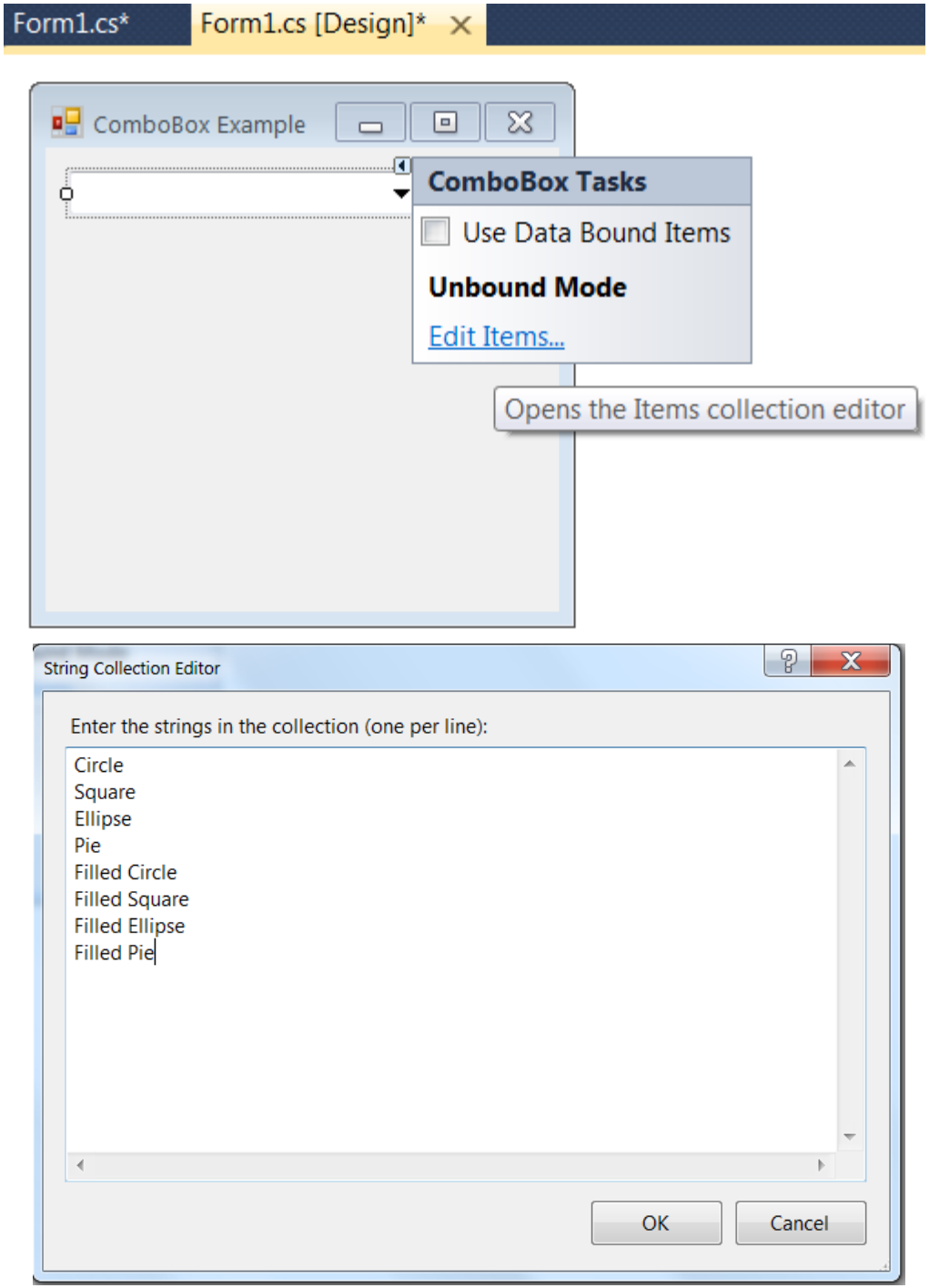
- ComboBox
  - **Name = imageComboBox**



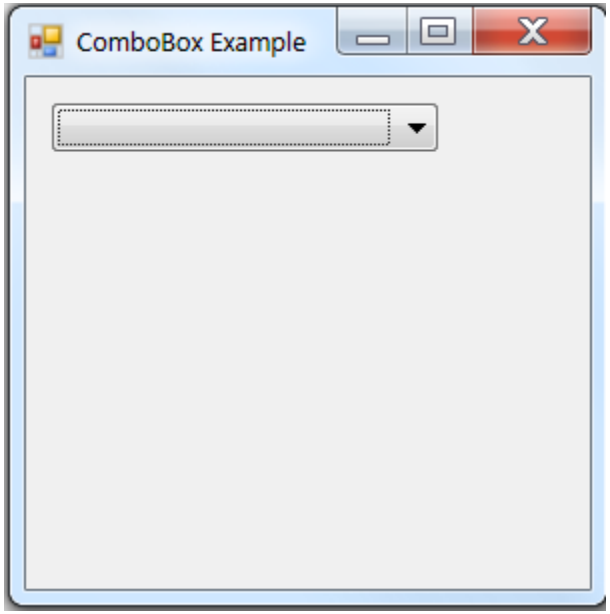
Örnek Uygulama - ComboBox

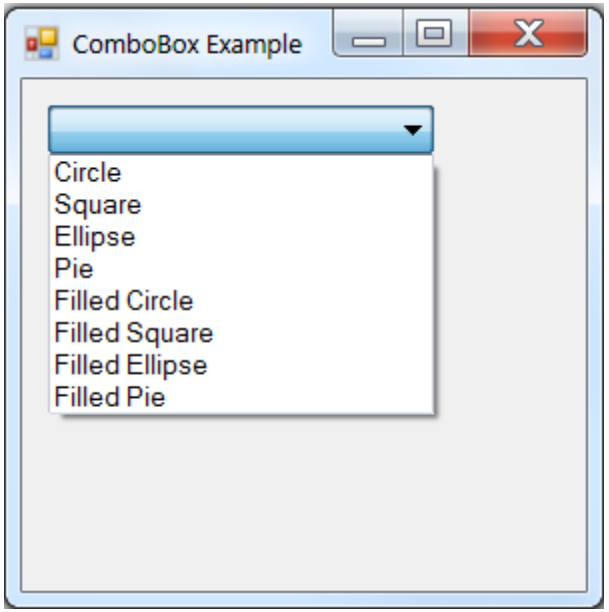
- **ComboBox**'ın düzenlenmesi özelliğini kaldırmak için:

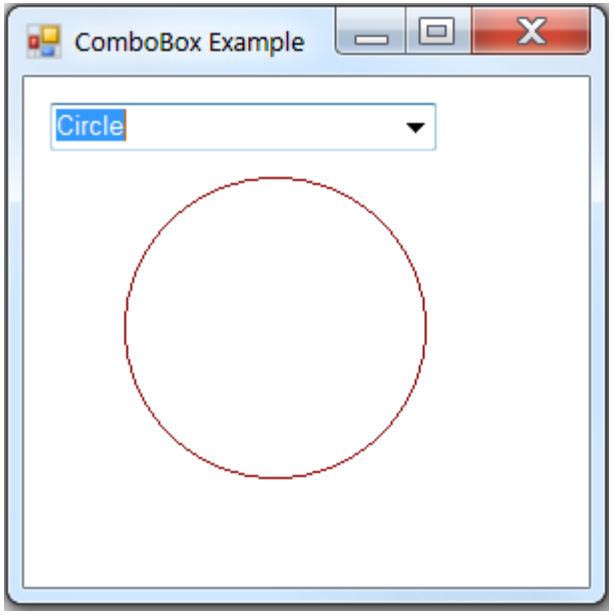


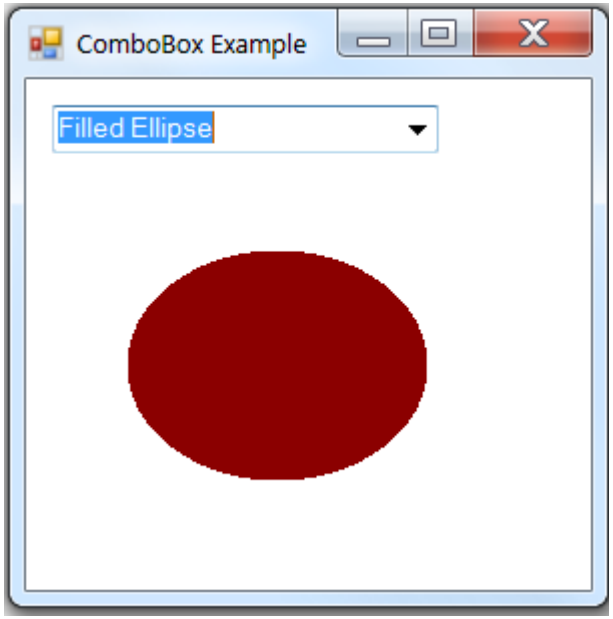

Form1.cs*
Form1.cs [Design]*
ComboBox Example
ComboBox Tasks
Use Data Bound Items
Unbound Mode
Edit Items...
Opens the Items collection editor
String Collection Editor
Enter the strings in the collection (one per line):
Circle
Square
Ellipse
Pie
Filled Circle
Filled Square
Filled Ellipse
Filled Pie
OK
Cancel

```
private void imageComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    // create graphics object, Pen and SolidBrush
    Graphics myGraphics = base.CreateGraphics();

    // create Pen using color DarkRed
    Pen myPen = new Pen(Color.DarkRed);

    // create SolidBrush using color DarkRed
    SolidBrush mySolidBrush = new SolidBrush(Color.DarkRed);

    // clear drawing area setting it to color white
    myGraphics.Clear(Color.White);

        // find index, draw proper shape
    switch ( imageComboBox.SelectedIndex )
    {
       case 0: // case Circle is selected
          myGraphics.DrawEllipse(myPen, 50, 50, 150, 150);
          break;
       case 1: // case Rectangle is selected
          myGraphics.DrawRectangle(myPen, 50, 50, 150, 150);
          break;
       case 2: // case Ellipse is selected
          myGraphics.DrawEllipse(myPen, 50, 85, 150, 115);
          break;
       case 3: // case Pie is selected
          myGraphics.DrawPie(myPen, 50, 50, 150, 150, 0, 45);
          break;
       case 4: // case Filled Circle is selected
          myGraphics.FillEllipse(mySolidBrush, 50, 50, 150, 150);
          break;
       case 5: // case Filled Rectangle is selected
          myGraphics.FillRectangle(mySolidBrush, 50, 50, 150, 150);
          break;
       case 6: // case Filled Ellipse is selected
          myGraphics.FillEllipse(mySolidBrush, 50, 85, 150, 115);
          break;
       case 7: // case Filled Pie is selected
          myGraphics.FillPie(mySolidBrush, 50, 50, 150, 150, 0, 45);
          break;
    } // end switch

    myGraphics.Dispose(); // release the Graphics object
 }
```


ComboBox Example


ComboBox Example
Circle
Square
Ellipse
Pie
Filled Circle
Filled Square
Filled Ellipse
Filled Pie


ComboBox Example
Circle


ComboBox Example
Filled Ellipse

TreeView

- **Düğümlerin *(Node)*** hiyerarşik olarak ağaç yapısında görüntülenmesini sağlar.
- Düğümler değer tutan nesnelerdir ve diğer düğümlere atıf (*refer*) yaparlar.
- **Üst düğüm *(parent node)*, alt düğümler *(child node)*** içerebilir.
- Alt düğümlerde başka düğümlerin üst düğümü olabilir.

TreeView

- Aynı üst düğüme sahip olan alt düğümler **kardeş düğümler (*sibling nodes*)** olarak adlandırılır.
- Ağaç (*tree*), hiyerarşik düzendeki düğümlerden oluşur.
- Ağacın ilk üst düğümü **root** olarak adlandırılır.
  - **`TreeView`**'da birden fazla *root* olabilir.

TreeView

- **`TreeView`**, hiyerarşik bilginin (ör: dosya yapısı) görüntülenmesinde kullanışlı bir kontroldür.
- **`TreeNode`** sınıfının bir örneğidir.

TreeView

TreeView properties and an event	Description
Common Properties	
CheckBoxes	Indicates whether CheckBoxes appear next to nodes. A value of true displays CheckBoxes. The default value is false.
ImageList	Specifies an ImageList object containing the node icons. An **ImageList** object is a collection that contains Image objects.
Nodes	Returns the collection of TreeNodes in the control as a TreeNodeCollection. It contains methods Add (adds a TreeNode object), Clear (deletes the entire collection) and Remove (deletes a specific node). Removing a parent node deletes all of its children.
SelectedNode	The selected node.
*Common Event (Event arguments **TreeViewEventArgs**)*	
AfterSelect	Generated after selected node changes. This is the default event when the control is double clicked in the designer.

TreeNode

TreeNode properties and methods	Description
Common Properties	
`Checked`	Indicates whether the `TreeNode` is checked (`CheckBoxes` property must be set to `true` in the parent `TreeView`).
`FirstNode`	Specifies the first node in the `Nodes` collection (i.e., the first child in the tree).
`FullPath`	Indicates the path of the node, starting at the root of the tree.
`ImageIndex`	Specifies the index in the `TreeView`'s `ImageList` of the image shown when the node is deselected.
`LastNode`	Specifies the last node in the `Nodes` collection (i.e., the last child in the tree).
`NextNode`	Next sibling node.
`Nodes`	Collection of `TreeNodes` contained in the current node (i.e., all the children of the current node). It contains methods `Add` (adds a `TreeNode` object), `Clear` (deletes the entire collection) and `Remove` (deletes a specific node). Removing a parent node deletes all of its children.
`PrevNode`	Previous sibling node.
`SelectedImageIndex`	Specifies the index in the `TreeView`'s `ImageList` of the image to use when the node is selected.
`Text`	Specifies the `TreeNode`'s text.
Common Methods	
`Collapse`	Collapses a node.
`Expand`	Expands a node.
`ExpandAll`	Expands all the children of a node.
`GetNodeCount`	Returns the number of child nodes.

TreeView

- ***Root düğümü ekleme:***

```
myTreeView.Nodes.Add(new TreeNode(rootLabel))
```

  - **`myTreeView`** → düğümlerin eklendiği **`TreeView`**
  - **`rootLabel`** → root'un adı

TreeView

- *Root'a alt düğüm eklenmesi:*

```
myTreeView.Nodes.Add[myIndex].Nodes.Add(new TreeNode(childLabel))
```

  - **`myIndex`** → root'un indeksi
  - **`childLabel`** → alt düğümün adı

Örnek Uygulama - TreeView

New Project

Recent Templates

Installed Templates

- Visual Basic
- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Visual C++
- Visual F#
- Other Project Types
- Database
- Test Projects

Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
WCF Service Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
Enable Windows Azure Tools	Visual C#

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: TreeViewExample

Location: C:\Users\ozgucan\Documents\Visual Studio 2010\Projects\WindowsProgramlama\

Browse...

Solution name: TreeViewExample

☑ Create directory for solution

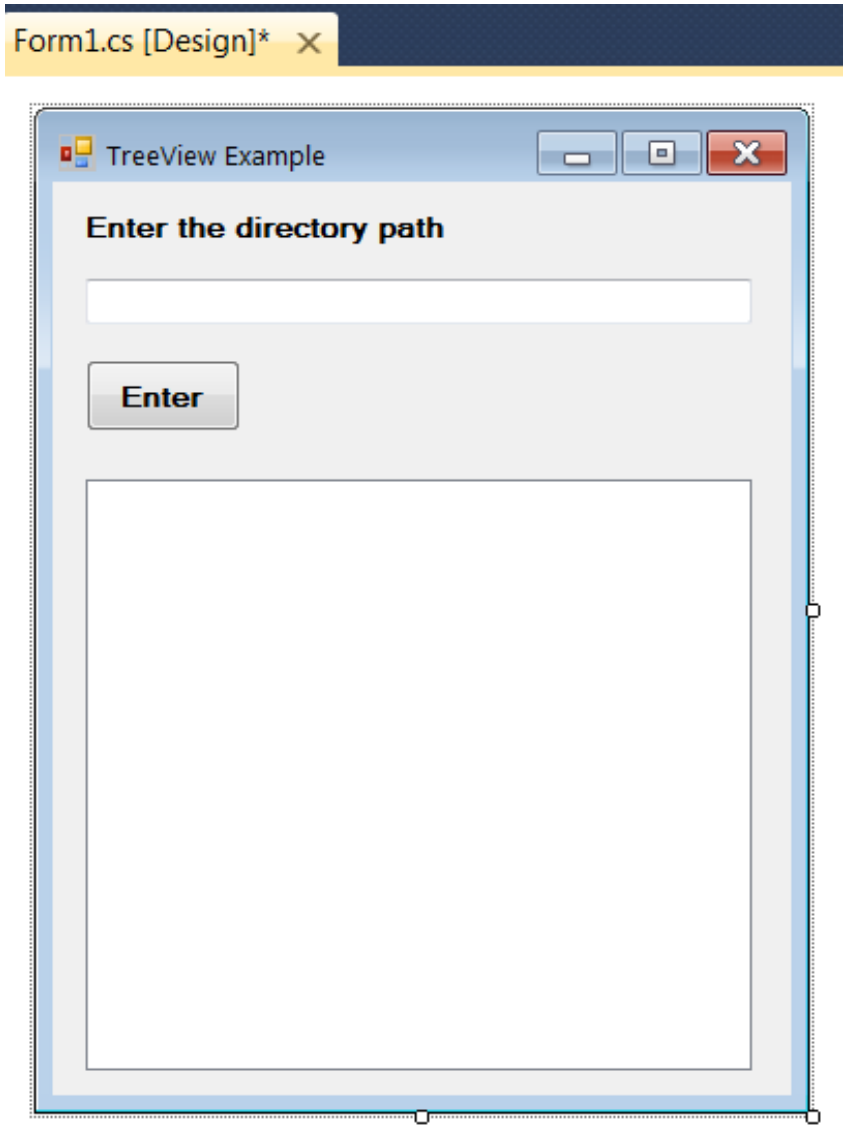
☐ Add to source control

OK | Cancel

Örnek Uygulama - TreeView

- Label
  - Name = inputLabel
  - Text = Enter the directory path
- TextBox
  - Name = inputTextBox
- Button
  - Name = enterButton
  - Text = Enter
- TreeView
  - Name = directoryTreeView

Örnek Uygulama - TreeView



Örnek Uygulama - TreeView

```
public partial class Form1 : Form
{
    string substringDirectory; // store last part of full path name

    public Form1()
    {
        InitializeComponent();
    }
```

```
// populate current node with subdirectories
public void PopulateTreeView( string directoryValue, TreeNode parentNode)
{
    // array stores all subdirectories in the directory
    string[] directoryArray = Directory.GetDirectories(directoryValue);

    // populate current node with subdirectories
    try
    {
        // check to see if any subdirectories are present
        if (directoryArray.Length != 0)
        {
            // for every subdirectory, create new TreeNode,
            // add as a child of current node and recursively
            // populate child nodes with subdirectories
            foreach (string directory in directoryArray)
            {
                // obtain last part of path name from the full path
                // name by calling the GetFileNameWithoutExtension
                // method of class Path
                substringDirectory = Path.GetFileNameWithoutExtension(directory);

                // create TreeNode for current directory
                TreeNode myNode = new TreeNode(substringDirectory);

                // add current directory node to parent node
                parentNode.Nodes.Add(myNode);

                // recursively populate every subdirectory
                PopulateTreeView(directory, myNode);
            } // end foreach
        } //end if
    }

    catch (UnauthorizedAccessException)
    {
        parentNode.Nodes.Add("Access denied");
    }
}
```

```
// populate current node with subdirectories
public void PopulateTreeView(
   string directoryValue, TreeNode parentNode)
{
    // array stores all subdirectories in the directory
    string[] directoryArray =
        Directory.GetDirectories(directoryValue);
```

```
// array stores all subdirectories in the directory
tring[] directoryArray =
 Directory.GetDirectories(directoryValue);
```

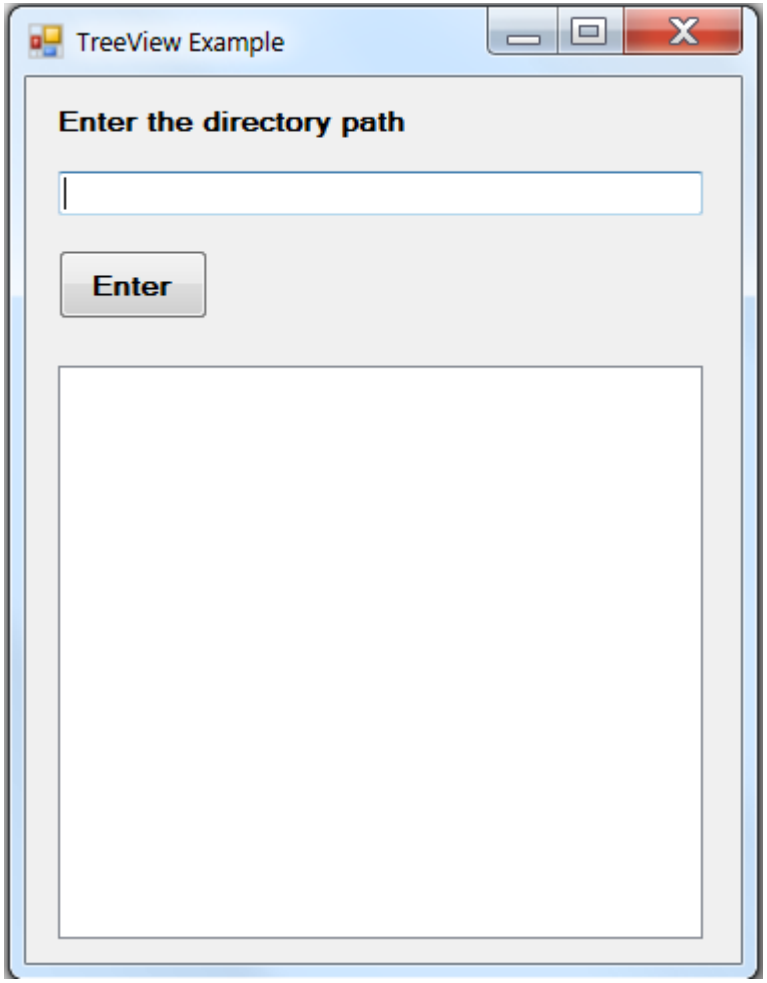
View Designer
Resolve
Refactor
Organize Usings
Create Unit Tests...
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Go To Definition F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T
Breakpoint
Run To Cursor Ctrl+F10
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Outlining

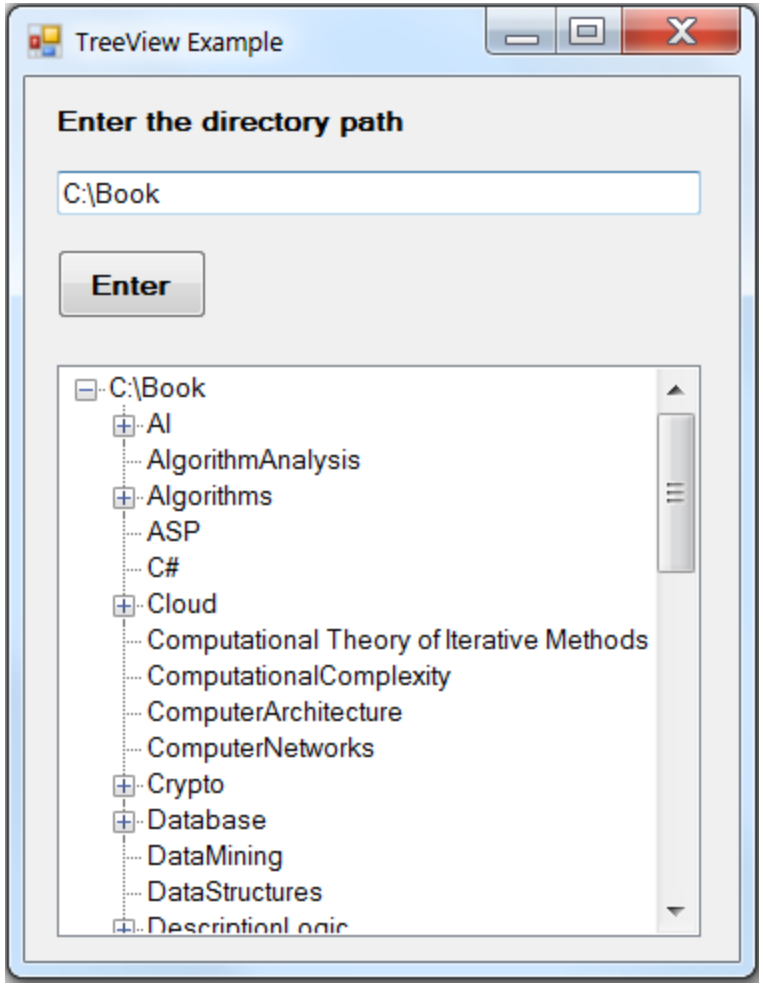
using System.IO;
System.IO.Directory

Örnek Uygulama - TreeView

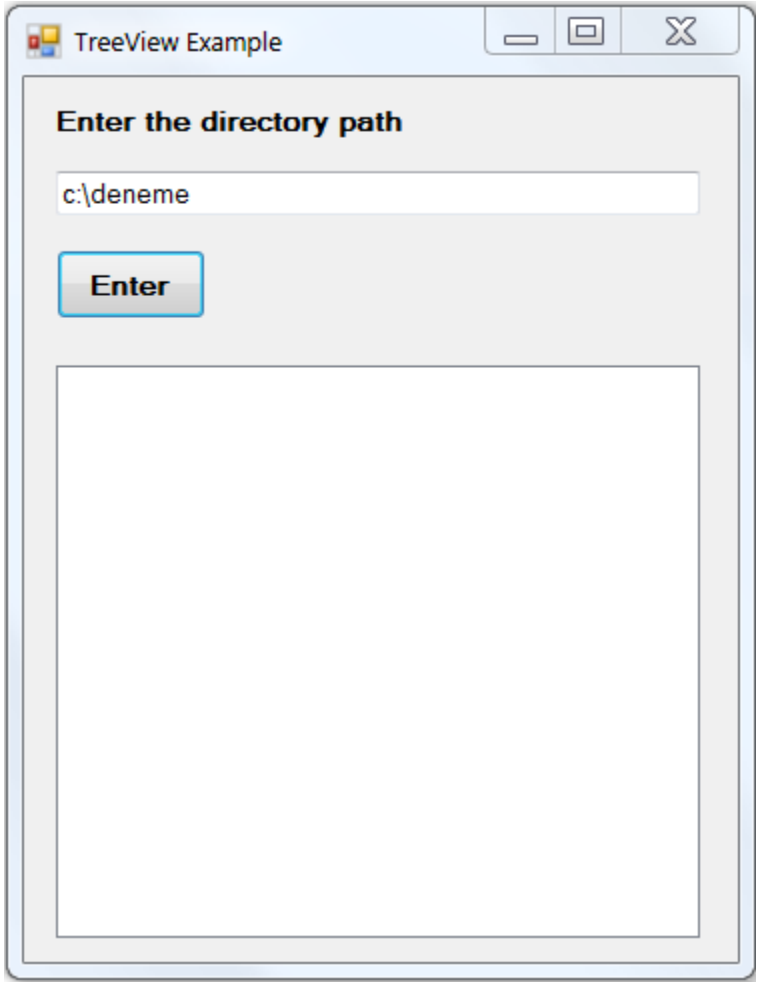
```
private void enterButton_Click(object sender, EventArgs e)
{
    // clear all nodes
    directoryTreeView.Nodes.Clear();

    // check if the directory entered by user exists
    // if it does then fill in the TreeView,
    // if not display error MessageBox
    if (Directory.Exists(inputTextBox.Text))
    {
        // add full path name to directoryTreeView
        directoryTreeView.Nodes.Add(inputTextBox.Text);

        // insert subfolders
        PopulateTreeView( inputTextBox.Text, directoryTreeView.Nodes[0]);
    } // end if
    // display error MessageBox if directory not found
    else
        MessageBox.Show(inputTextBox.Text + " could not be found.",
           "Directory Not Found", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

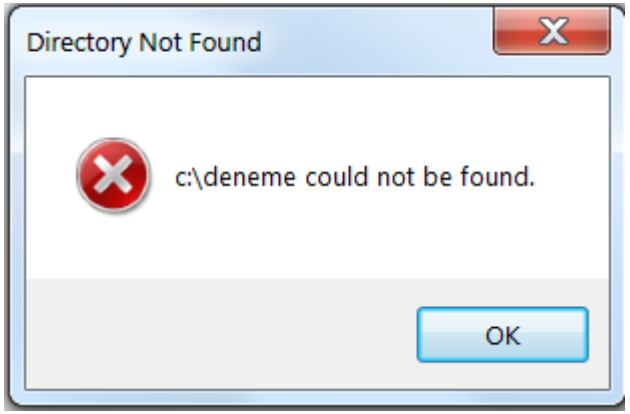
Örnek Uygulama - TreeView





Örnek Uygulama - TreeView





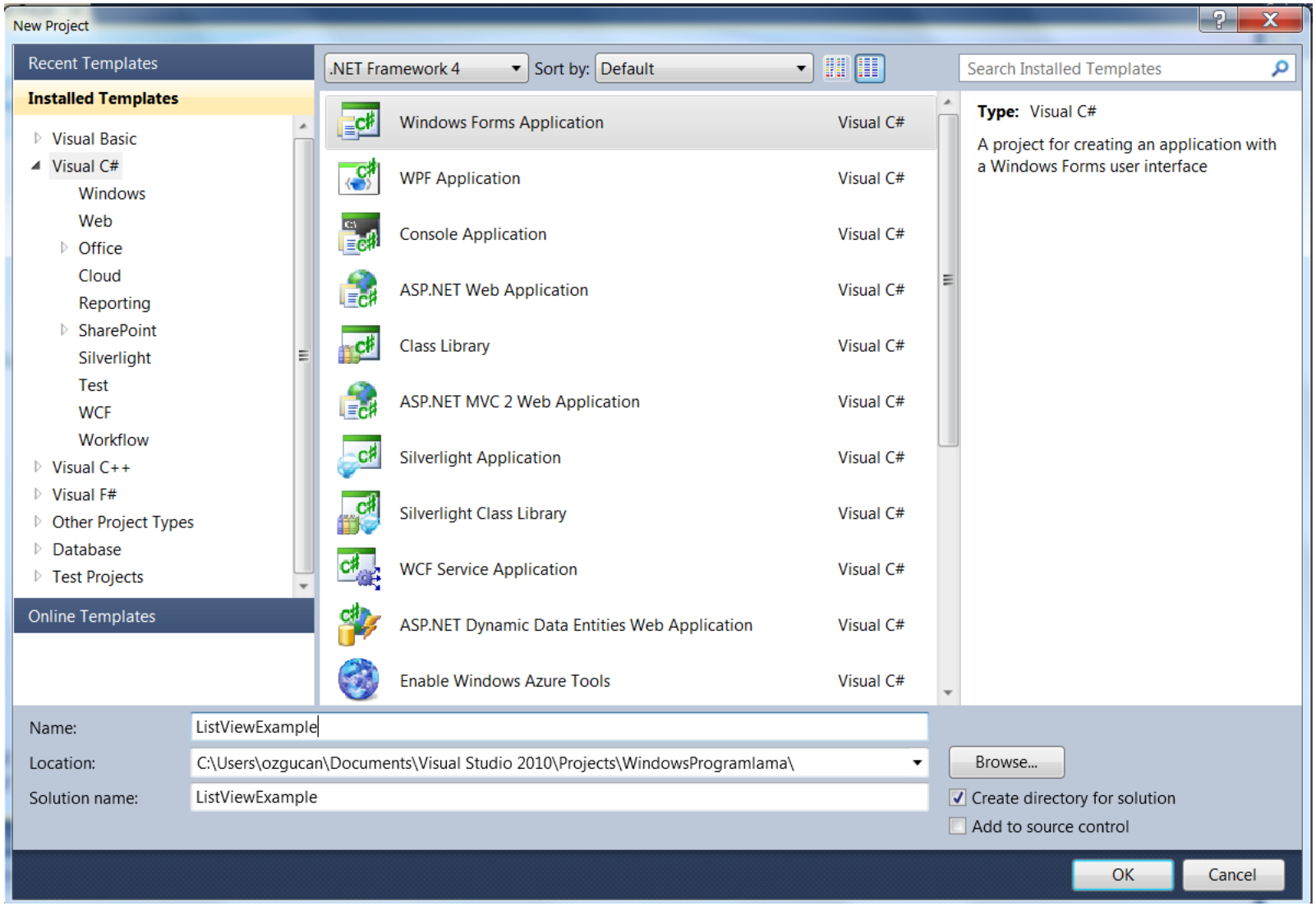
ListView

- **`ListBox`**'a benzer.
  - **`ListView`**, **`ListBox`**'a göre daha çok işlem gerçekleştirir.
- Kullanıcının seçim yapabileceği liste öğelerini görüntüler.
  - Öğeleri farklı formatlarda görüntüleyebilir.

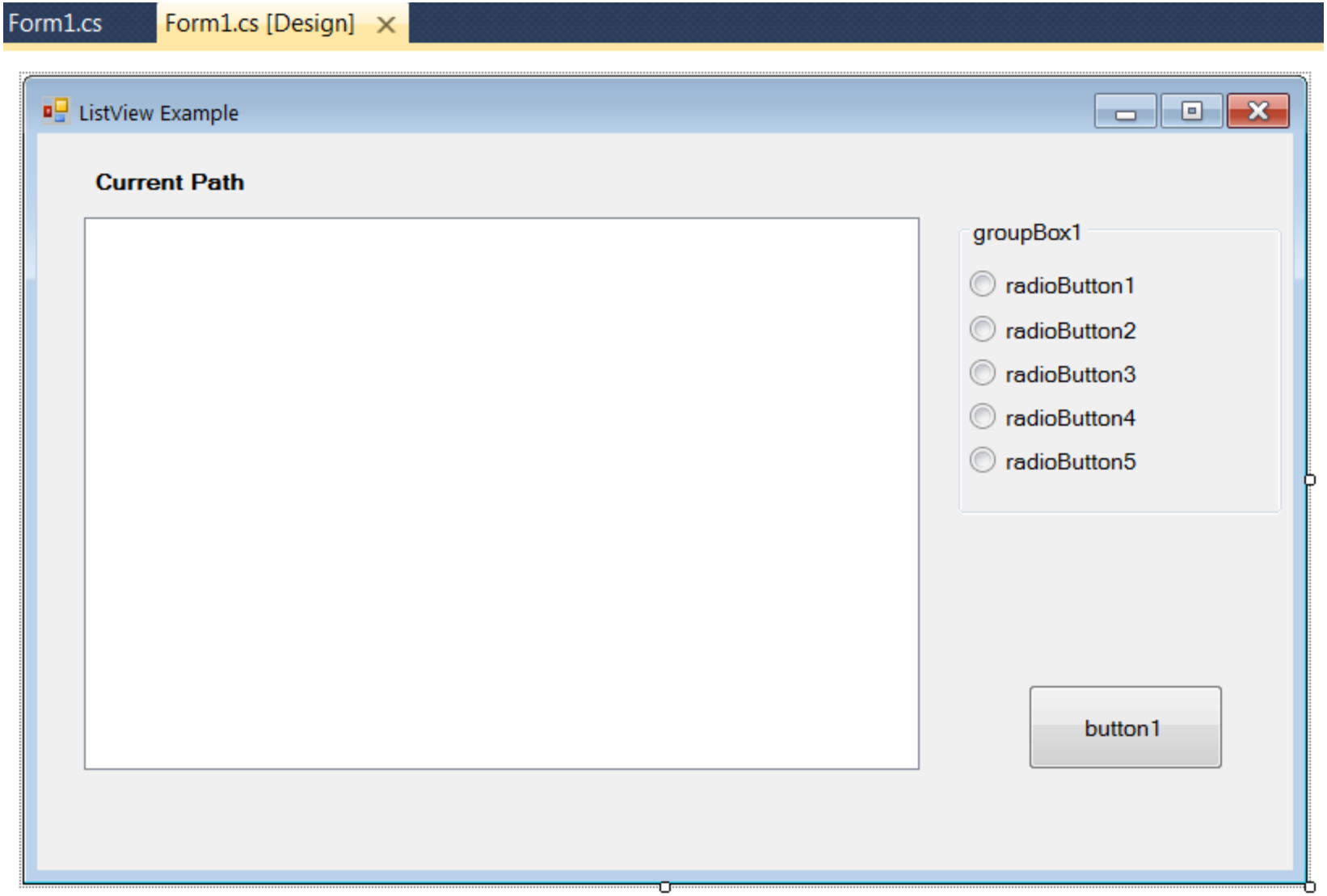
ListView

ListView properties and events	Description
Common Properties	
`Activation`	Determines how the user activates an item. This property takes a value in the `ItemActivation` enumeration. Possible values are `OneClick` (single-click activation), `TwoClick` (double-click activation, item changes color when selected) and `Standard` (the default; double-click activation, item does not change color).
`CheckBoxes`	Indicates whether items appear with `CheckBoxes`. `true` displays `CheckBoxes`. The default is `false`.
`LargeImageList`	Specifies the `ImageList` containing large icons for display.
`Items`	Returns the collection of `ListViewItems` in the control.
`MultiSelect`	Determines whether multiple selection is allowed. The default is `true`, which enables multiple selection.
`SelectedItems`	Returns the collection of selected items as a `ListView.SelectedListViewItemCollection`.
`SmallImageList`	Specifies the `ImageList` containing small icons for display.
`View`	Determines appearance of `ListViewItems`. Possible values are `LargeIcon` (the default; large icon displayed, items can be in multiple columns), `SmallIcon` (small icon displayed, items can be in multiple columns), `List` (small icons displayed, items appear in a single column), `Details` (like `List`, but multiple columns of information can be displayed per item) and `Tile` (large icons displayed, information provided to right of icon; valid only in Windows XP or later).
Common Events	
`Click`	Generated when an item is clicked. This is the default event.
`ItemActivate`	Generated when an item in the `ListView` is activated (clicked or double clicked). Does not contain the specifics of which item is activated.

Örnek Uygulama - ListView



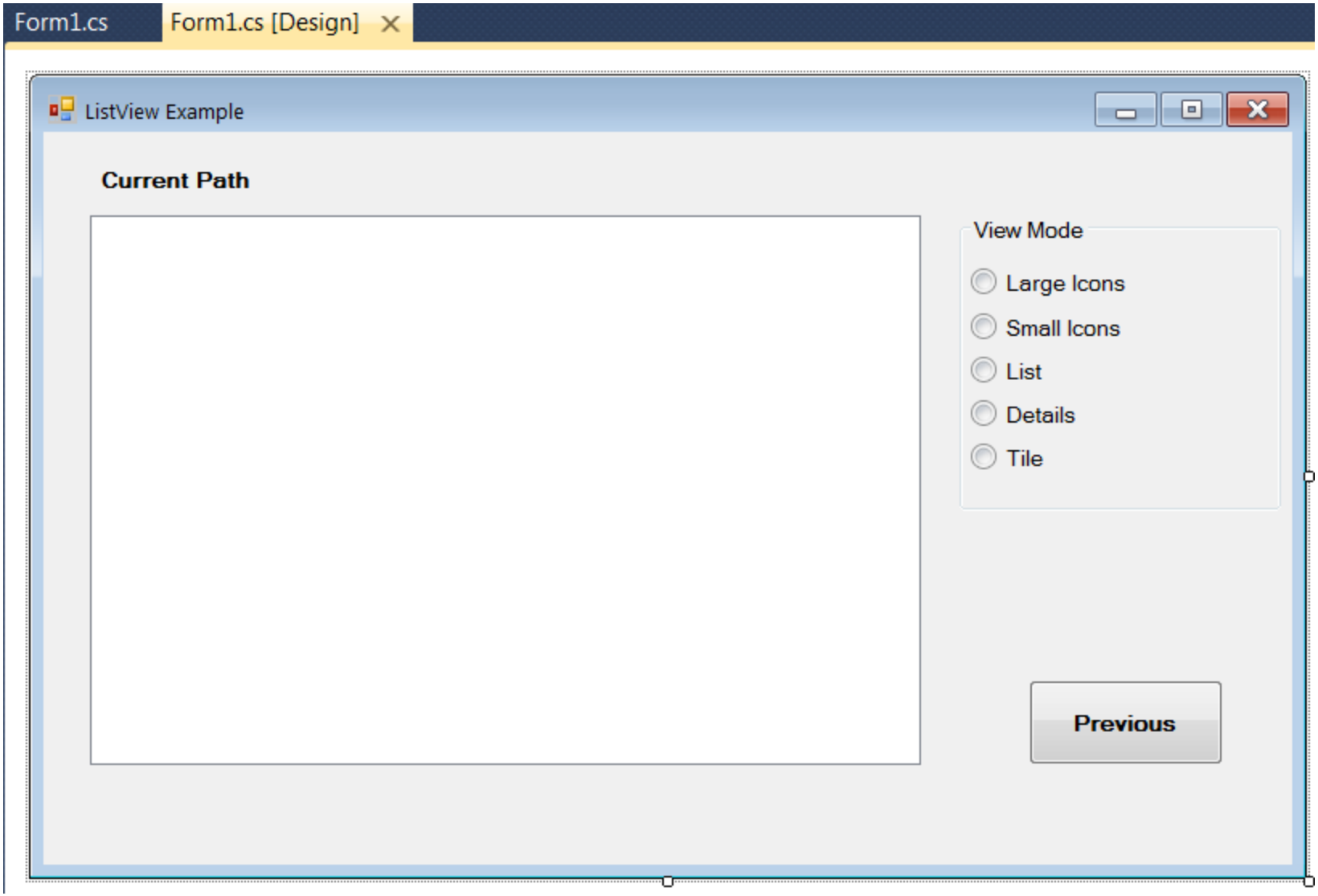
Örnek Uygulama - ListView



Örnek Uygulama - ListView

- **Label**
  - Name = currentPathLabel
  - Text = Current Path
- **ListView**
  - Name = directoryListView
- **Button**
  - Name = previousButton
  - Text = Previous
- **GroupBox**
  - Name = viewModeGroupBox
  - Text = View Mode
- 5 **RadioButton**
  1. Name = largeRadioButton
     Text = Large Icons
  2. Name = smallRadioButton
     Text = Small Icons
  3. Name = listRadioButton
     Text = List
  4. Name = detailsRadioButton
     Text = Details
     Checked = True
  5. Name = tileRadioButton
     Text = Tile

Örnek Uygulama - ListView



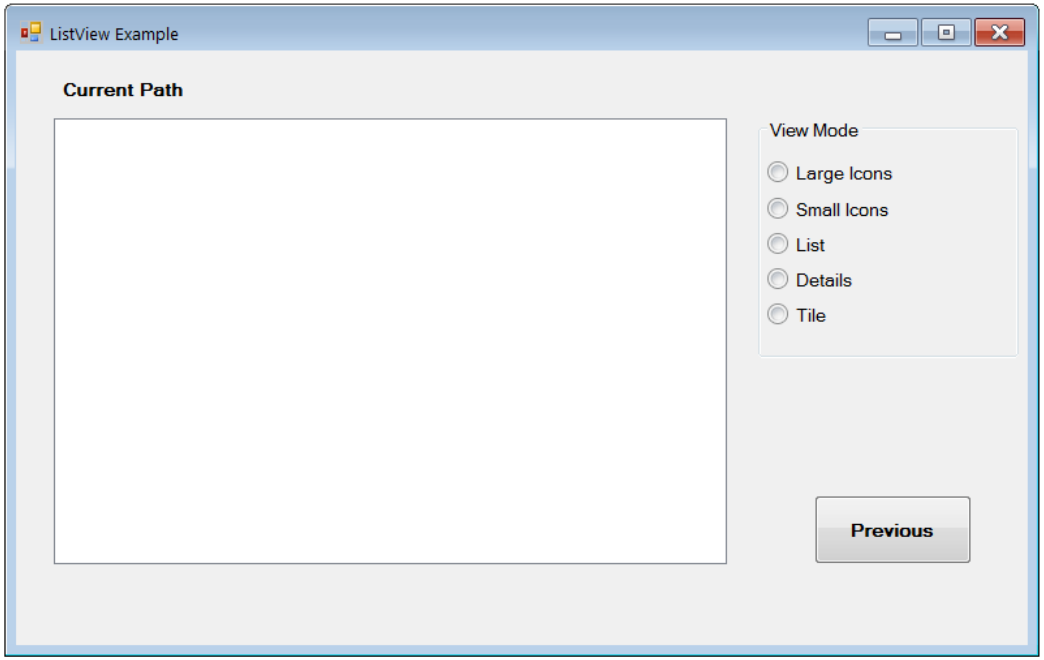
Örnek Uygulama - ListView

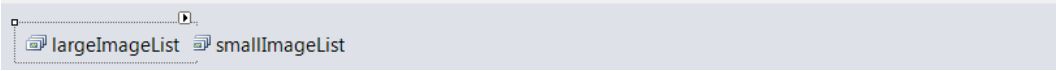
- 2 **ImageList**

  1. `Name = largeImageList`

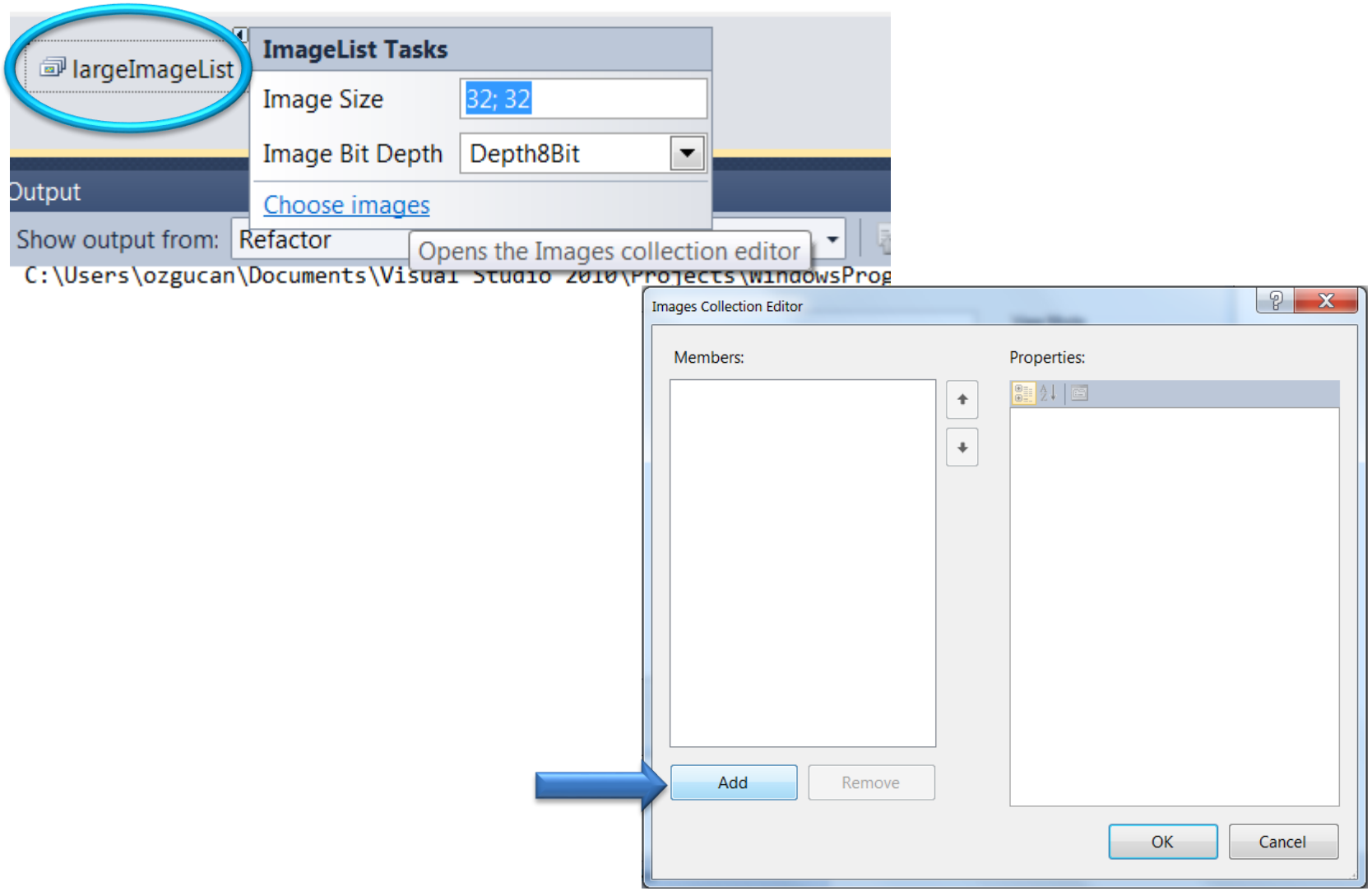
     `ImageSize = 32; 32`

  2. `Name = smallImageList`

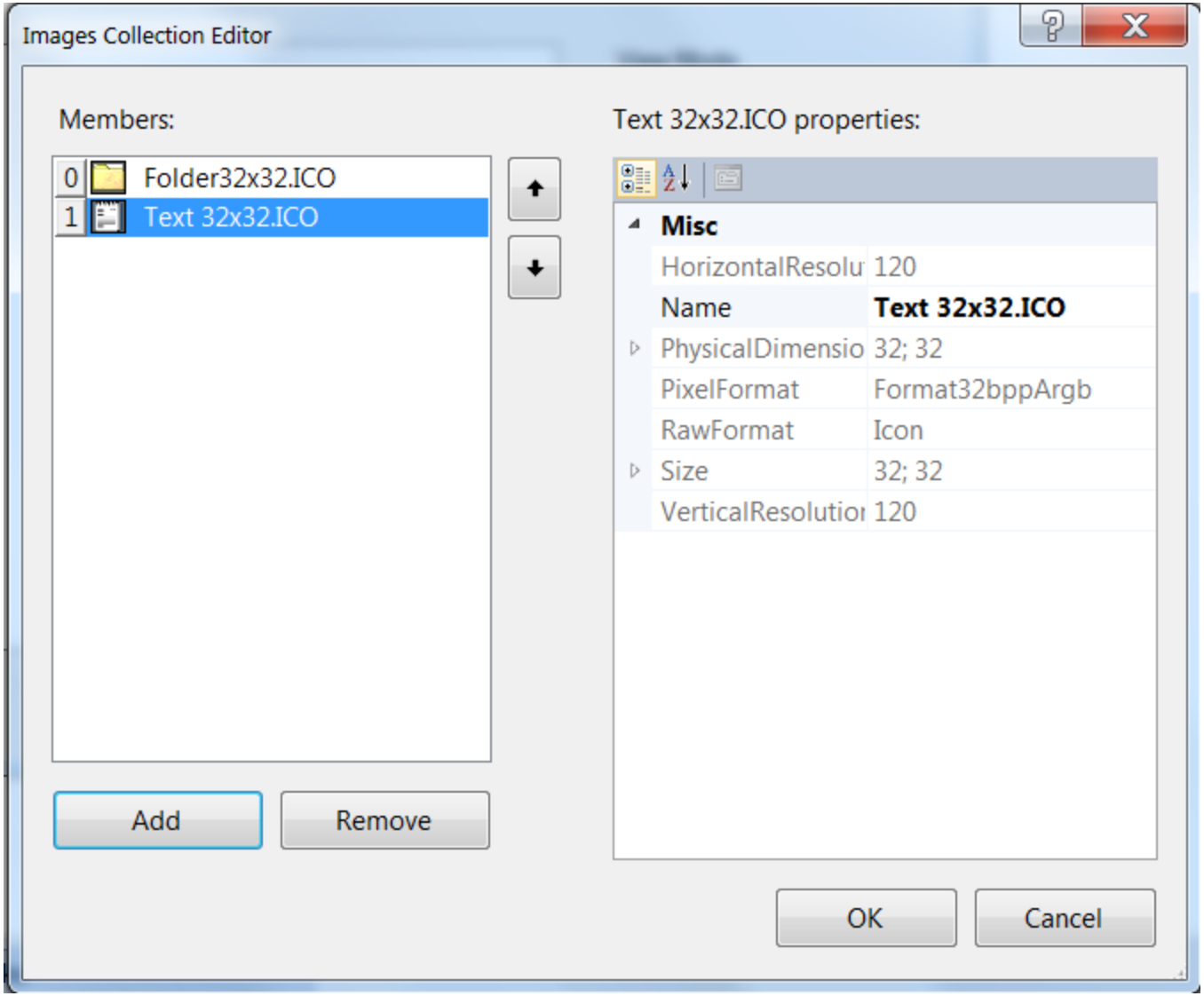




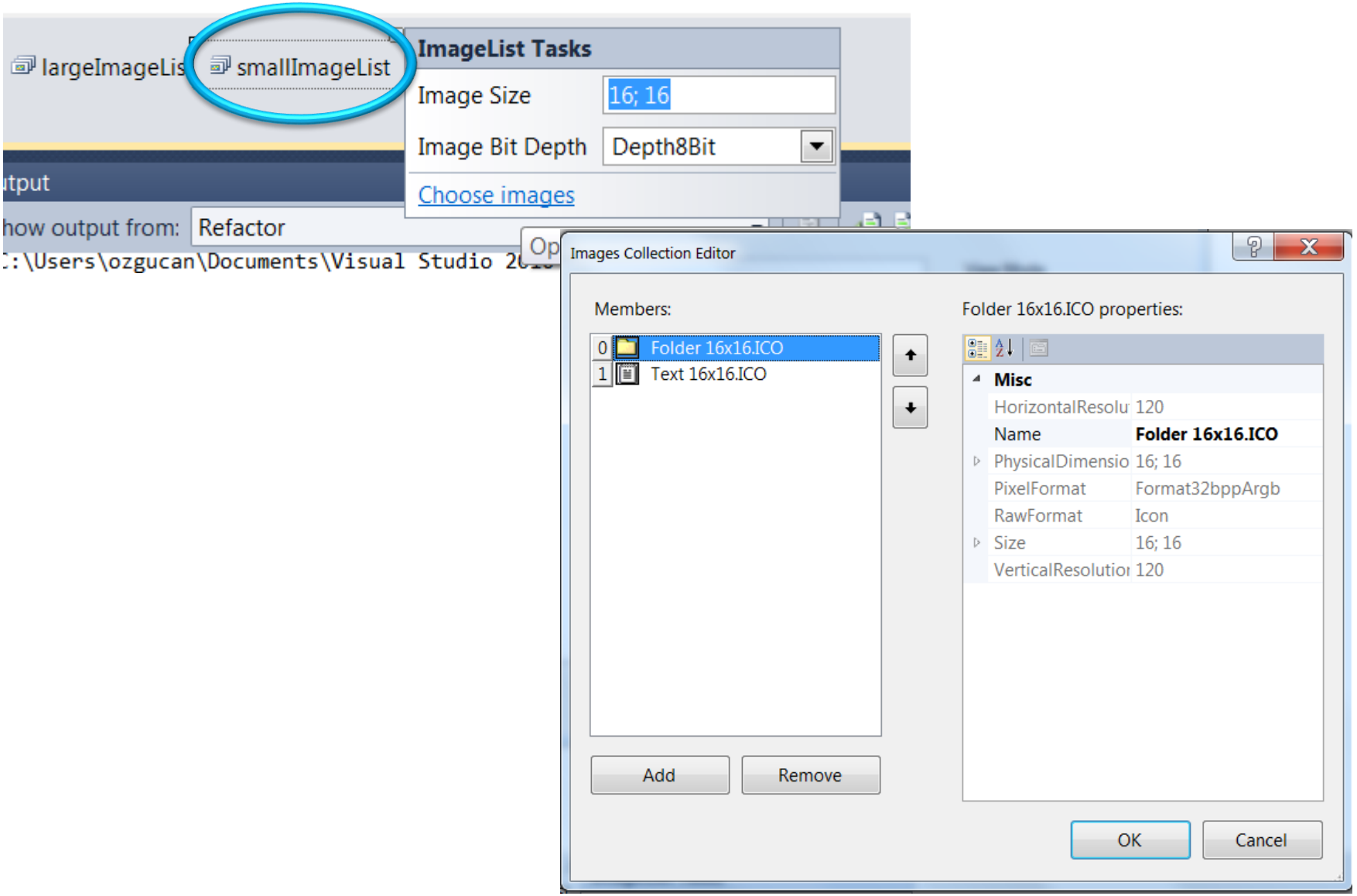
Örnek Uygulama - ListView



Örnek Uygulama - ListView

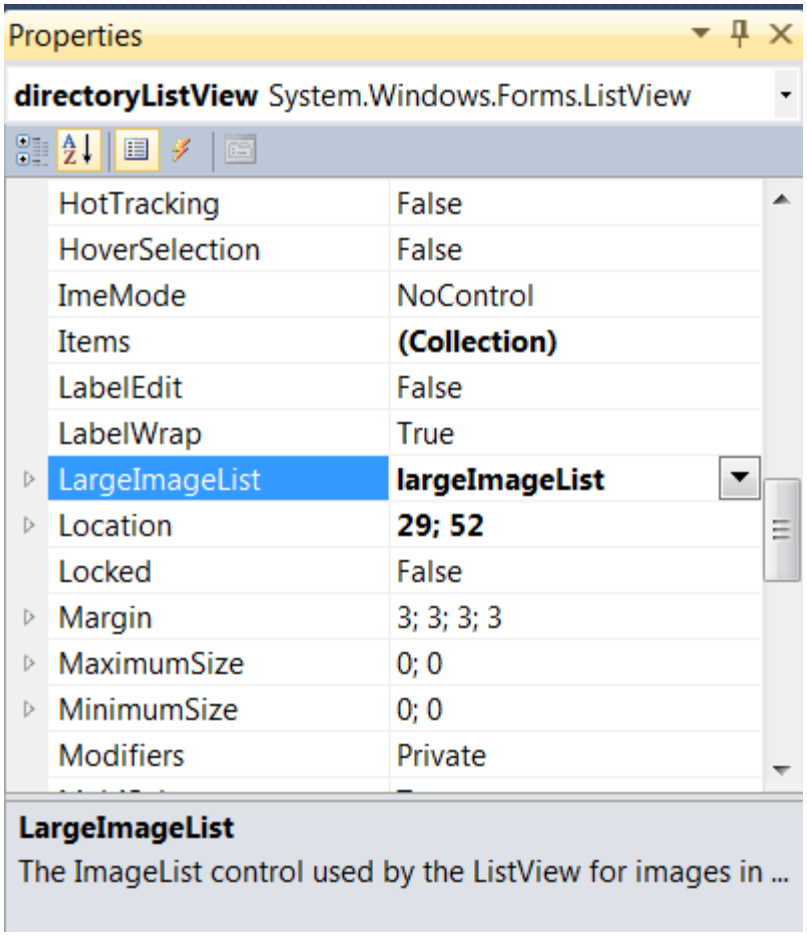


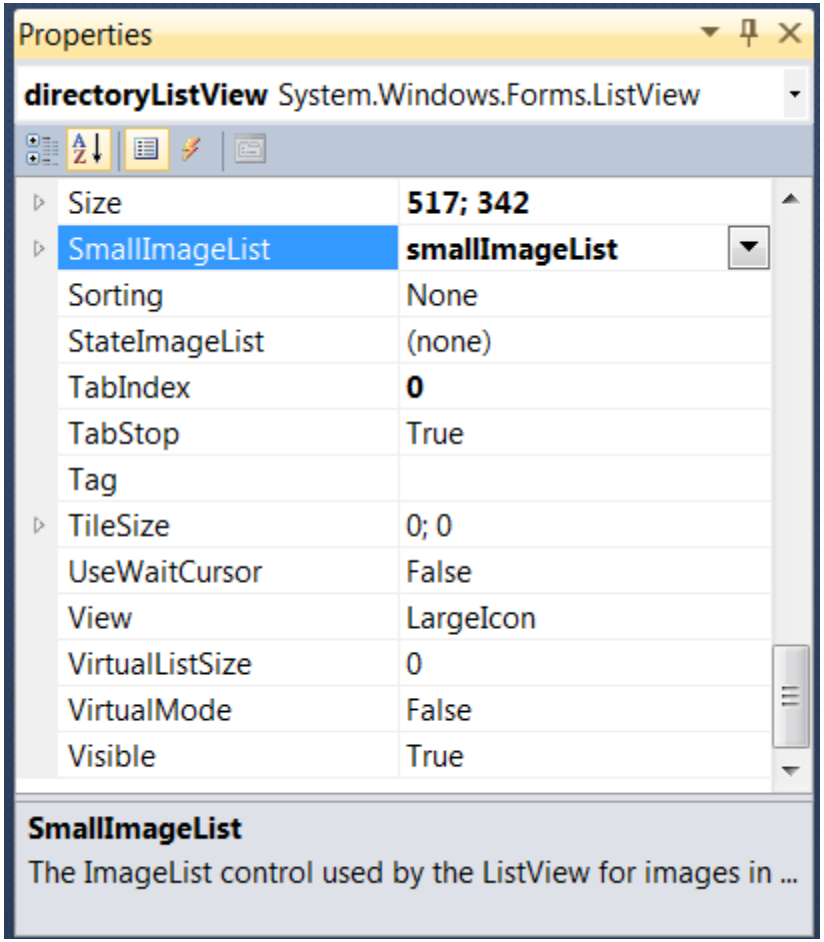
Örnek Uygulama - ListView



Örnek Uygulama - ListView

- **`ListView`** için;

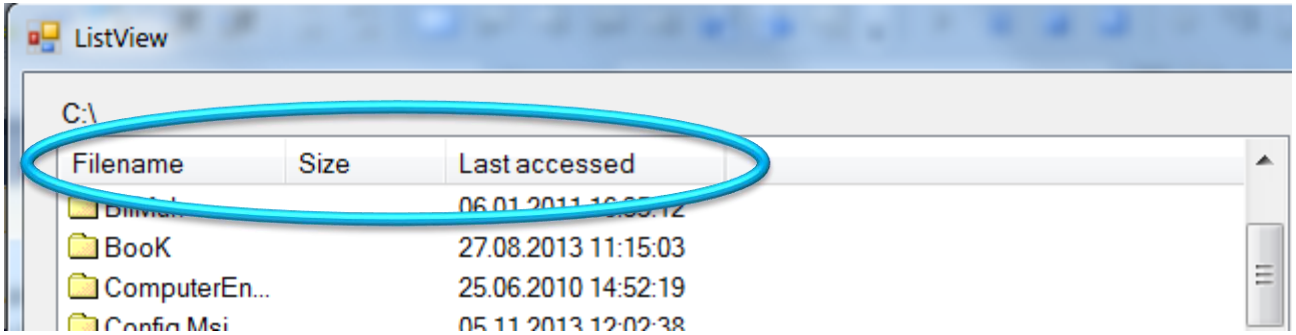




Örnek Uygulama - ListView

- **Previous**'a basıldığından daha önce gezilen klasörleri listeleyebilmek için bir alana ihtiyaç vardır.
- Klasörlerin yolunu (*path*) saklamak için;
  - **StringCollection** kullanılmaktadır.

```
public partial class Form1 : Form
{
    private System.Collections.Specialized.StringCollection folderCol;

    public Form1()
    {
        InitializeComponent();
    }
```



***ListView**'da sütun başlıklarının görüntülenmesi için yazılacak metot:*

```
private void CreateHeadersAndFillListView()
{
    ColumnHeader colHead;

    // First header
    colHead = new ColumnHeader();
    colHead.Text = "Filename";
    directoryListView.Columns.Add(colHead); // Insert the header

    // Second header
    colHead = new ColumnHeader();
    colHead.Text = "Size";
    directoryListView.Columns.Add(colHead); // Insert the header

    // Third header
    colHead = new ColumnHeader();
    colHead.Text = "Last accessed";
    directoryListView.Columns.Add(colHead); // Insert the header
}
```

Örnek Uygulama - ListView

`ListView`'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot:

```
private void PaintListView(string root)
{
    try
    {
         // Two local variables that are used to create the items to insert
        ListViewItem lvi;
        ListViewItem.ListViewSubItem lvsi;

        // If there's no root folder, we can't insert anything.
        if (string.IsNullOrEmpty(root))
        return;

        // Get information about the root folder.
        DirectoryInfo dir = new DirectoryInfo(root);

        // Retrieve the files and folders from the root folder.
        DirectoryInfo[] dirs = dir.GetDirectories(); // Folders
        FileInfo[] files = dir.GetFiles();           // Files
```

.cs [Design]*

orm1 | View(string root)

```
        colHead.Text
        directoryList

        // Third head
        colHead = new
        colHead.Text
        directoryList
    }

    private void Pair
    {
        try
        {
            // Two
            ListView
            ListView

            // If the
            if (strin
            return;

            // Get in
            DirectoryInfo dir = new DirectoryInfo(root);

            // Retrieve the files and folders from the root folder.
            DirectoryInfo[] dirs = dir.GetDirectories(); // Folders
            FileInfo[] files = dir.GetFiles();           // Files
```

the header

e the items to insert

anything.

- View Designer
- Resolve ▸
  - using System.IO;
  - System.IO.DirectoryInfo
- Refactor ▸
- Generate ▸
- Organize Usings ▸
- Create Unit Tests...
- Insert Snippet... Ctrl+K, Ctrl+X
- Surround With... Ctrl+K, Ctrl+S
- Go To Definition F12
- Find All References Shift+F12
- View Call Hierarchy Ctrl+K, Ctrl+T
- Breakpoint ▸
- Run To Cursor Ctrl+F10
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Outlining ▸

ListView'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot:

```
private void PaintListView(string root)
{
    try
    {
         // Two local variables that are used to create the items to insert
        ListViewItem lvi;
        ListViewItem.ListViewSubItem lvsi;

        // If there's no root folder, we can't insert anything.
        if (string.IsNullOrEmpty(root))
        return;

        // Get information about the root folder.
        DirectoryInfo dir = new DirectoryInfo(root);

        // Retrieve the files and folders from the root folder.
        DirectoryInfo[] dirs = dir.GetDirectories(); // Folders
        FileInfo[] files = dir.GetFiles();           // Files
```

Örnek Uygulama - ListView

`ListView`'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot (devam):

```
// Clear the ListView. Note that we call the Clear method on the
// Items collection rather than on the ListView itself.
// The Clear method of the ListView remove everything, including column
// headers, and we only want to remove the items from the view.
directoryListView.Items.Clear();

// Set the label with the current path.
currentPathLabel.Text = root;

// Lock the ListView for updates.
directoryListView.BeginUpdate();
```

ListView'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot (devam):

```
// Loop through all folders in the root folder and insert them.
foreach (DirectoryInfo di in dirs)
{
     // Create the main ListViewItem.
    lvi = new ListViewItem();
    lvi.Text = di.Name; // Folder name
    lvi.ImageIndex = 0; // The folder icon has index 0
    lvi.Tag = di.FullName; // Set the tag to the qualified path of the folder

    // Create the two ListViewSubItems.
    lvsi = new ListViewItem.ListViewSubItem();
    lvsi.Text = ""; // Size—a folder has no size and so this column is empty
    lvi.SubItems.Add(lvsi); // Add the subitem to the ListViewItem

    lvsi = new ListViewItem.ListViewSubItem();
    lvsi.Text = di.LastAccessTime.ToString(); // Last accessed column
    lvi.SubItems.Add(lvsi); // Add the subitem to the ListViewItem.

    // Add the ListViewItem to the Items collection of the ListView.
    directoryListView.Items.Add(lvi);
 }
```

ListView'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot (devam):

```
// Loop through all the files in the root folder.
foreach (FileInfo fi in files)
 {
    // Create the main ListViewItem.
    lvi = new ListViewItem();
    lvi.Text = fi.Name; // Filename
    lvi.ImageIndex = 1; // The icon we use to represent a folder has index 1.
    lvi.Tag = fi.FullName; // Set the tag to the qualified path of the file.

    // Create the two subitems.
    lvsi = new ListViewItem.ListViewSubItem();
    lvsi.Text = fi.Length.ToString(); // Length of the file
    lvi.SubItems.Add(lvsi); // Add to the SubItems collection

    lvsi = new ListViewItem.ListViewSubItem();
    lvsi.Text = fi.LastAccessTime.ToString(); // Last Accessed Column
    lvi.SubItems.Add(lvsi); // Add to the SubItems collection

    // Add the item to the Items collection of the ListView.
    directoryListView.Items.Add(lvi);
  }
```

ListView'un ilk yüklemede görüntüleyeceği dosya ve klasörler için ilgili metot (devam):

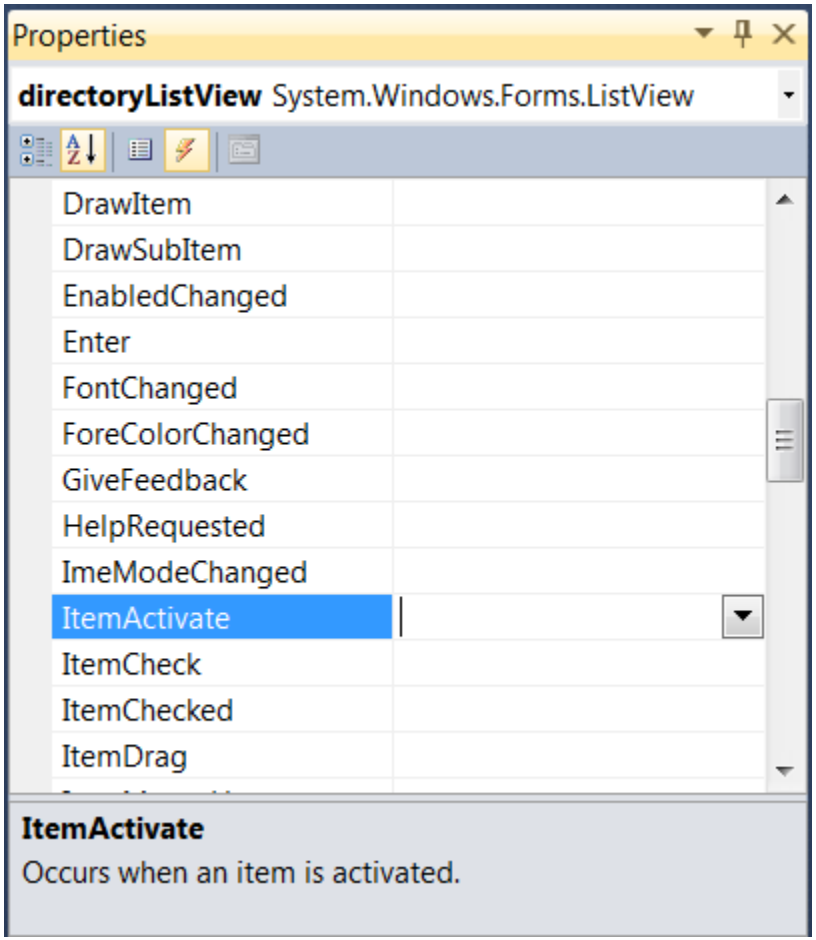
```
        // Unlock the ListView. The items that have been inserted will now be displayed.
        directoryListView.EndUpdate();
    }
    catch (System.Exception err)
    {
        MessageBox.Show("Error: " + err.Message);
    }
}
```

Örnek Uygulama - ListView

`ListView` için

`ItemActivate`

event'inin yaratılması



```
private void directoryListView_ItemActivate(object sender, EventArgs e)
{
    // Cast the sender to a ListView and get the tag of the first selected item.
    System.Windows.Forms.ListView lw = (System.Windows.Forms.ListView)sender;
    string filename = lw.SelectedItems[0].Tag.ToString();

    if (lw.SelectedItems[0].ImageIndex != 0)
    {
        try
        {
            // Attempt to run the file.
            System.Diagnostics.Process.Start(filename);
        }
        catch
        {
            // If the attempt fails we simply exit the method.
            return;
        }
    }
    else
    {
        // Insert the items.
        PaintListView(filename);
        folderCol.Add(filename);
    }
}
```

Örnek Uygulama - ListView

- **`Previous`** button'u için **`Click`** event'i:

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

RadioButton'lar için **Checked_Changed** event'leri

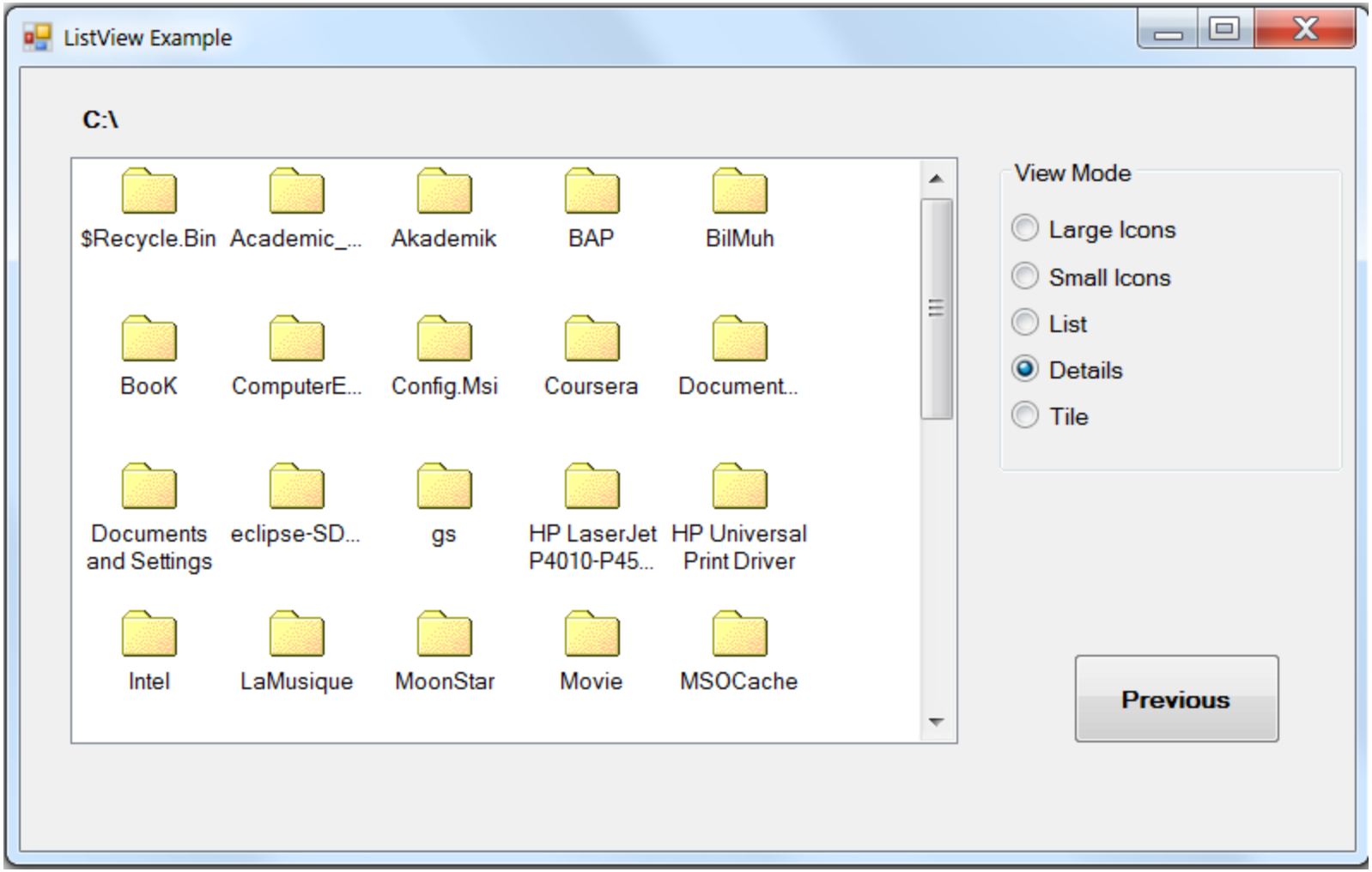
```
private void largeRadioButton_CheckedChanged(object sender, EventArgs e)
{
    RadioButton rdb = (RadioButton)sender;
    if (rdb.Checked)
        this.directoryListView.View = View.LargeIcon;
}

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

private void listRadioButton_CheckedChanged(object sender, EventArgs e)
{
    RadioButton rdb = (RadioButton)sender;
    if (rdb.Checked)
        this.directoryListView.View = View.List;
}
private void detailsRradioButton_CheckedChanged(object sender, EventArgs e)
{
    RadioButton rdb = (RadioButton)sender;
    if (rdb.Checked)
        this.directoryListView.View = View.Details;
}

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

Örnek Uygulama - ListView

`ListView`'in ***Initialize*** edilmesi:

```
public partial class Form1 : Form
{

    private System.Collections.Specialized.StringCollection folderCol;

    public Form1()
    {
        InitializeComponent();

        // Init ListView and folder collection
        folderCol = new System.Collections.Specialized.StringCollection();
        CreateHeadersAndFillListView();
        PaintListView(@"C:\");
        folderCol.Add(@"C:\");
    }
```

Örnek Uygulama - ListView



Örnek Uygulama - ListView