Windows Forms – Serialization, Dialogs, DataBinding

Contents

[1. Serialization/Deserialization 1](#_Toc449427240)

[1.1. Binary Serialization 1](#_Toc449427241)

[1.2. XML Serialization 2](#_Toc449427242)

[1.3. TextFiles 2](#_Toc449427243)

[2. Dialogs 3](#_Toc449427244)

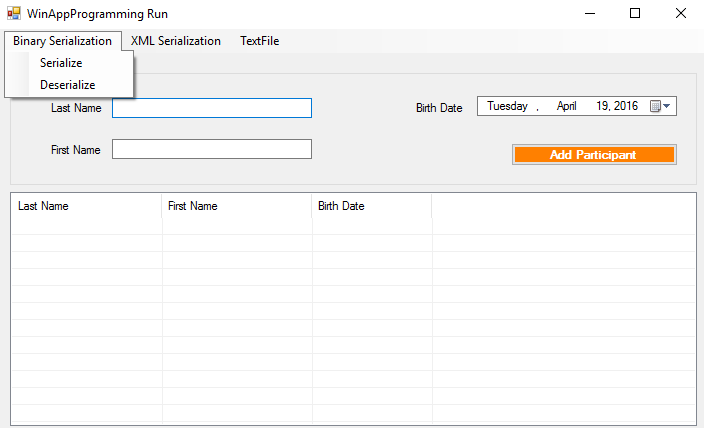
[3. DataBinding 6](#_Toc449427245)

# Serialization/Deserialization

Activity

|  |  |
| --- | --- |
|  | Sample code available at <http://online.ase.ro> – “SerializationBinaryXMLTextFile” Sample |

1. Create a copy of the “ListViewBasicSample” project and name it “SerializationBinaryXMLTextFile”
2. Create the following UI



## Binary Serialization

1. Add a menu for “Binary Serialization” (“Serialize” - btnSerializeBinary, “Deserialize” - btnDeserializeBinary), “XML Serialization” (“Serialize” - btnSerializeXML, “Deserialize” - btnDeserializeXML) and “TextFile” (“Export” – btnExport).
2. Decorate the “Participant” class with the [Serializable] attribute, as follows. Un exception will be thrown otherwise.

|  |
| --- |
| **[**Serializable**]**  **internal** class Participant  **{**  **…………………** |

1. Handle the “Click” event for the “btnSerializeBinary” button as follows

|  |
| --- |
| **private** void btnSerialize\_Click**(object** sender**,** EventArgs e**){**  BinaryFormatter formatter **=** **new** BinaryFormatter**();**  **using** **(**FileStream s **=** File**.**Create**(**"serialized.bin"**))**  formatter**.**Serialize**(**s**,** \_participants**);**  **}** |

1. Remove the **readonly** modifier from the declaration of the “\_participants” attribute in the “MainForm” class. The project will not compile otherwise.
2. Handle the “Click” event for the “btnDeserializeBinary” button as follows

|  |
| --- |
| **private** void btnDeserialize\_Click**(object** sender**,** EventArgs e**){**  BinaryFormatter formatter **=** **new** BinaryFormatter**();**  **using** **(**FileStream s **=** File**.**OpenRead**(**"serialized.bin"**))** **{**  \_participants **=** **(**List**<**Participant**>)**formatter**.**Deserialize**(**s**);**  DisplayParticipants**();**  **}**  **}** |

## XML Serialization

1. Add a parameterless constructor to the “Participant” class. Change the access modifier for the class from “internal” to “public”. Un exception will be thrown otherwise.
2. Handle the “Click” event for the “btnSerializeXML” button as follows.

|  |
| --- |
| XmlSerializer serializer **=** **new** XmlSerializer**(typeof(**List**<**Participant**>));**  **using** **(**StreamWriter writer **=** **new** StreamWriter**(**"SerializedXML.xml"**))**  **{**  serializer**.**Serialize**(**writer**,** \_participants**);**  **}** |

1. Handle the “Click” event for the “btnDeserializeXML” button as follows.

|  |
| --- |
| XmlSerializer serializer **=** **new** XmlSerializer**(typeof(**List**<**Participant**>));**  **using** **(**StreamReader streamReader **=** **new** StreamReader**(**"SerializedXML.xml"**))**  **{**  \_participants **=** **(**List**<**Participant**>)**serializer**.**Deserialize**(**streamReader**);**  DisplayParticipants**();**  **}** |

## TextFiles

1. Handle the “Click” event for the “btnExport” button as follows

|  |
| --- |
| // Create an instance of the open file dialog box.  SaveFileDialog saveFileDialog **=** **new** SaveFileDialog**();**  saveFileDialog**.**Filter **=** "Text File | \*.txt"**;**  saveFileDialog**.**Title **=** "Save as text file"**;**  **if** **(**saveFileDialog**.**ShowDialog**()** **==** DialogResult**.**OK**)**  **{**  //Approach 1  //StreamWriter sw = new StreamWriter(saveFileDialog.FileName);  //try  //{  // sw.WriteLine("LastName,FirstName,BirthDate");  // foreach (var participant in \_participants)  // {  // sw.WriteLine("{0}, {1}, {2}"  // , participant.LastName  // , participant.FirstName  // , participant.BirthDate.ToShortDateString());  // }  //}  //finally  //{  // sw.Dispose();  //}  //2. Approach 2 - recommended  // generates the try{} finally{} in Version 1  **using** **(**StreamWriter sw **=** **new** StreamWriter**(**saveFileDialog**.**FileName**))**  **{**  sw**.**WriteLine**(**"LastName,FirstName,BirthDate"**);**  **foreach** **(**var participant **in** \_participants**)**  **{**  sw**.**WriteLine**(**"{0}, {1}, {2}"  **,** participant**.**LastName  **,** participant**.**FirstName  **,** participant**.**BirthDate**.**ToShortDateString**());**  **}**  **}**  **}** |

Activity

|  |  |
| --- | --- |
|  | Sample code available at <http://online.ase.ro> – “TextFileSample” Sample |

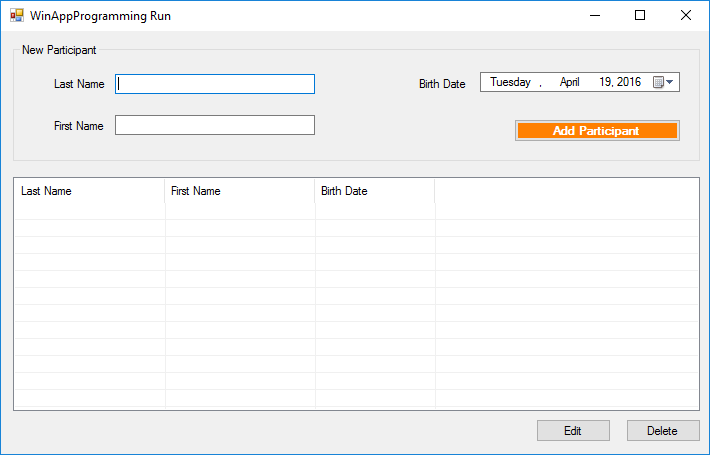
|  |
| --- |
| static void Main**(**string**[]** args**)**  **{**  // Get the directories currently on the C drive.  DirectoryInfo**[]** cDirs **=** **new** DirectoryInfo**(**@"c:\"**).**GetDirectories**();**  // Write each directory name to a file.  **using** **(**StreamWriter sw **=** **new** StreamWriter**(**"CDriveDirs.txt"**))**  **{**  **foreach** **(**DirectoryInfo dir **in** cDirs**)**  **{**  sw**.**WriteLine**(**dir**.**Name**);**  **}**  **}**  // Read and show each line from the file.  string line **=** ""**;**  **using** **(**StreamReader sr **=** **new** StreamReader**(**"CDriveDirs.txt"**))**  **{**  **while** **((**line **=** sr**.**ReadLine**())** **!=** **null)**  **{**  Console**.**WriteLine**(**line**);**  **}**  **}**  **}** |

# Dialogs

Activity

|  |  |
| --- | --- |
|  | Sample code available at <http://online.ase.ro> – “DialogSample” Sample |

1. Create a copy of the “BasicListView” project and name it “DialogSample”
2. Create the following UI



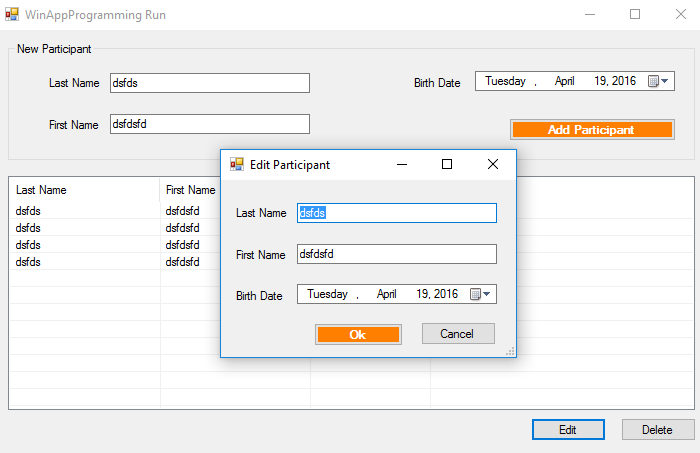
1. Name the “Edit” button “btnEdit” and the “Delete” button “btnDelete”
2. Modify the “DisplayParticipants” method in the “MainForm” class in order to set the “Tag” property for the ListViewItem instances, as shown bellow.

|  |
| --- |
| **public** void DisplayParticipants**()**  **{**  lvParticipants**.**Items**.**Clear**();**  **foreach** **(**Participant participant **in** \_participants**)**  **{**  var listViewItem **=** **new** ListViewItem**(**participant**.**LastName**);**  listViewItem**.**SubItems**.**Add**(**participant**.**FirstName**);**  listViewItem**.**SubItems**.**Add**(**participant**.**BirthDate**.**ToShortDateString**());**    //add this line  listViewItem**.**Tag **=** participant**;**    lvParticipants**.**Items**.**Add**(**listViewItem**);**  **}**  **}** |

1. Handle the “Click” event for the “btnDeletet” button as follows

|  |
| --- |
| **if** **(**lvParticipants**.**SelectedItems**.**Count **==** 0**)**  **{**  MessageBox**.**Show**(**"Choose a participant"**);**  **return;**  **}**  **if** **(**MessageBox**.**Show**(**"Are you sure?"**,** "Delete participant"**,** MessageBoxButtons**.**YesNo**,** MessageBoxIcon**.**Warning**)** **==**  DialogResult**.**Yes**)**  **{**  \_participants**.**Remove**((**Participant**)** lvParticipants**.**SelectedItems**[**0**].**Tag**);**  DisplayParticipants**();**  **}** |

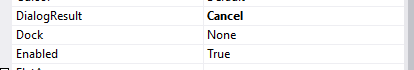
1. Add a new Form to the project and name it “EditForm”
2. Create the following UI



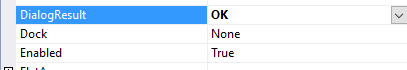
1. Rename the controls as “tbLastName”, “tbFirstName” and “dtpBirthDate”
2. Change the EditForm class, so that it is defined as follow

|  |
| --- |
| #region Attributes  **private** **readonly** Participant \_participant**;**  #endregion  **public** EditForm**(**Participant participant**)**  **{**  \_participant **=** participant**;**  InitializeComponent**();**  **}**  **private** void EditForm\_Load**(object** sender**,** System**.**EventArgs e**)**  **{**  tbLastName**.**Text **=** \_participant**.**LastName**;**  tbFirstName**.**Text **=** \_participant**.**FirstName**;**  dtpBirthDate**.**Value **=** \_participant**.**BirthDate**;**  **}** |

1. Set the DialogResult for the “Cancel” button as “Cancel”



1. Rename the “Ok” button as “btnOk”
2. Set the DialogResult for the “Ok” button as “OK”



1. Handle the “Click” event for the “btnOk” button as follows

|  |
| --- |
| \_participant**.**LastName **=** tbLastName**.**Text**;**  \_participant**.**FirstName **=** tbFirstName**.**Text**;**  \_participant**.**BirthDate **=** dtpBirthDate**.**Value**;** |

1. Handle the “Click” event for the “Edit” button in the “MainForm” as follows:

|  |
| --- |
| **if** **(**lvParticipants**.**SelectedItems**.**Count **==** 0**)**  **{**  MessageBox**.**Show**(**"Choose a participant"**);**  **return;**  **}**  EditForm editForm **=** **new** EditForm**((**Participant**)**lvParticipants**.**SelectedItems**[**0**].**Tag**);**  **if** **(**editForm**.**ShowDialog**()** **==** DialogResult**.**OK**)**  DisplayParticipants**();** |

# DataBinding

**Data binding type:**

|  |  |
| --- | --- |
| Type | Description |
| Simple data binding | The ability of a control to bind to a single data element, such as a value in a column in a dataset table. This is the type of binding typical for controls such as a [TextBox](https://msdn.microsoft.com/en-us/library/system.windows.forms.textbox(v=vs.110).aspx) control or [Label](https://msdn.microsoft.com/en-us/library/system.windows.forms.label(v=vs.110).aspx) control, which are controls that typically only displays a single value. In fact, any property on a control can be bound to a field in a database. |
| Complex data binding | The ability of a control to bind to more than one data element, typically more than one record in a database. Complex binding is also called list-based binding. Examples of controls that support complex binding are the [DataGridView](https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview(v=vs.110).aspx), [ListBox](https://msdn.microsoft.com/en-us/library/system.windows.forms.listbox(v=vs.110).aspx), and [ComboBox](https://msdn.microsoft.com/en-us/library/system.windows.forms.combobox(v=vs.110).aspx) controls. |

**Change notification**

* Ensures that your data source and bound controls always have the most recent data, we must add change notification for data binding. Specifically, we want to ensure that bound controls are notified of changes that were made to their data source, and the data source is notified of changes that were made to the bound properties of a control.

Cases:

* Simple Binding –[INotifyPropertyChanged](https://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(v=vs.110).aspx)
* Complex data binding -[IBindingList](https://msdn.microsoft.com/en-us/library/system.componentmodel.ibindinglist(v=vs.110).aspx)

Activity

|  |  |
| --- | --- |
|  | Sample code available at <http://online.ase.ro> – “DataBindingDialogs” Sample |

1. Create a copy of the “BasicListView” project and name it “DataBindingSample”
2. Replace the “ListView” control with a “DataGrid” control (Name: dgvParticipants)
3. Add a “ViewModel” folder to your project
4. Add the following “MainFormViewModel” class in the “ViewModel” folder

|  |
| --- |
| **internal** class MainFormViewModel **:** INotifyPropertyChanged  **{**  #region Properties  #region LastName  **private** string \_lastName**;**  **public** string LastName **{**  get **{** **return** \_lastName**;** **}**  set  **{**  **if** **(**\_lastName **==** value**)**  **return;**  \_lastName **=** value**;**  //If we use [CallerMemberName] in the OnPropertyChanged method  //OnPropertyChanged();  //If we don't use the [CallerMemberName] in the OnPropertyChanged method  OnPropertyChanged**(**"LastName"**);**  **}**  **}**  #endregion  #region FirstName  **private** string \_firstName**;**  **public** string FirstName  **{**  get **{** **return** \_firstName**;** **}**  set  **{**  **if** **(**\_firstName **==** value**)**  **return;**  \_firstName **=** value**;**  OnPropertyChanged**();**  **}**  **}**  #endregion  #region FirstName  **private** DateTime \_birthDate**;**  **public** DateTime BirthDate  **{**  get **{** **return** \_birthDate**;** **}**  set  **{**  **if** **(**\_birthDate **==** value**)**  **return;**  \_birthDate **=** value**;**  OnPropertyChanged**();**  **}**  **}**  #endregion  **public** BindingList**<**Participant**>** Participants **{** get**;** set**;** **}**  #endregion  **public** MainFormViewModel**()**  **{**  Participants **=** **new** BindingList**<**Participant**>();**  BirthDate **=** DateTime**.**Now**;**  **}**  #region Methods  **public** void AddParticipant**()**  **{**  Participants**.**Add**(new** Participant**(**LastName**,** FirstName**,** BirthDate**));**  LastName **=** FirstName **=** string**.**Empty**;**  BirthDate **=** DateTime**.**Today**;**  **}**  #endregion  #region INotifyPropertyChanged  **public** **event** PropertyChangedEventHandler PropertyChanged**;**  **[**NotifyPropertyChangedInvocator**]**  // [CallerMemberName] - Allows you to obtain the method or property name of the caller to the method. https://msdn.microsoft.com/en-us/library/system.runtime.compilerservices.callermembernameattribute%28v=vs.110%29.aspx  **protected** **virtual** void OnPropertyChanged**([**CallerMemberName**]** string propertyName **=** **null)**  **{**  **if(**PropertyChanged **!=** **null)**  PropertyChanged**.**Invoke**(this,** **new** PropertyChangedEventArgs**(**propertyName**));**  **}**  #endregion  **}** |

1. Update the “MainForm” so that it is defined as follow.

|  |
| --- |
| **public** partial class MainForm **:** Form  **{**  **private** **readonly** MainFormViewModel \_viewModel**;**  **public** MainForm**()**  **{**  InitializeComponent**();**  Load **+=** MainForm\_Load**;**  \_viewModel **=** **new** MainFormViewModel**();**  **}**  **private** void MainForm\_Load**(object** sender**,** EventArgs e**)**  **{**  dgvParticipants**.**DataSource **=** \_viewModel**.**Participants**;**  tbLastName**.**DataBindings**.**Add**(**"Text"**,**\_viewModel**,**"LastName"**,false,**DataSourceUpdateMode**.**OnPropertyChanged**);**  tbFirstName**.**DataBindings**.**Add**(**"Text"**,** \_viewModel**,** "FirstName"**,** **false,** DataSourceUpdateMode**.**OnPropertyChanged**);**  dtpBirthDate**.**DataBindings**.**Add**(**"Value"**,** \_viewModel**,** "BirthDate"**,** **false,** DataSourceUpdateMode**.**OnPropertyChanged**);**  **}**  **private** void btnAdd\_Click**(object** sender**,** EventArgs e**)**  **{**  \_viewModel**.**AddParticipant**();**  **}**  **}** |

|  |  |
| --- | --- |
|  | Further reading about the MVVM pattern: [link](https://msdn.microsoft.com/en-us/library/hh848246.aspx) |