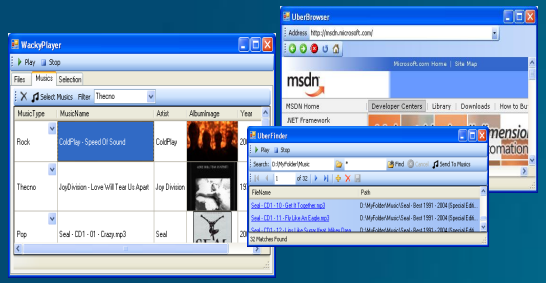
Microsoft .net Framework Windows® Forms

Windows Forms e Data Binding

Fernando Miguel Carvalho
Centro de Cálculo
Engineering Superior Institute of Lisbon, DEETC
mcarvalho@cc.isel.ipl.pt

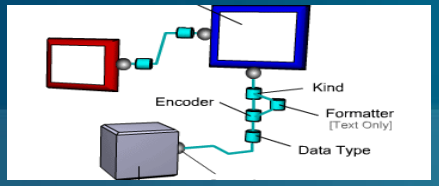
De La Rue –
Currency and Authentication Solutions

Sumário



- Novas Funcionalidades

- *BackgroundWorker*

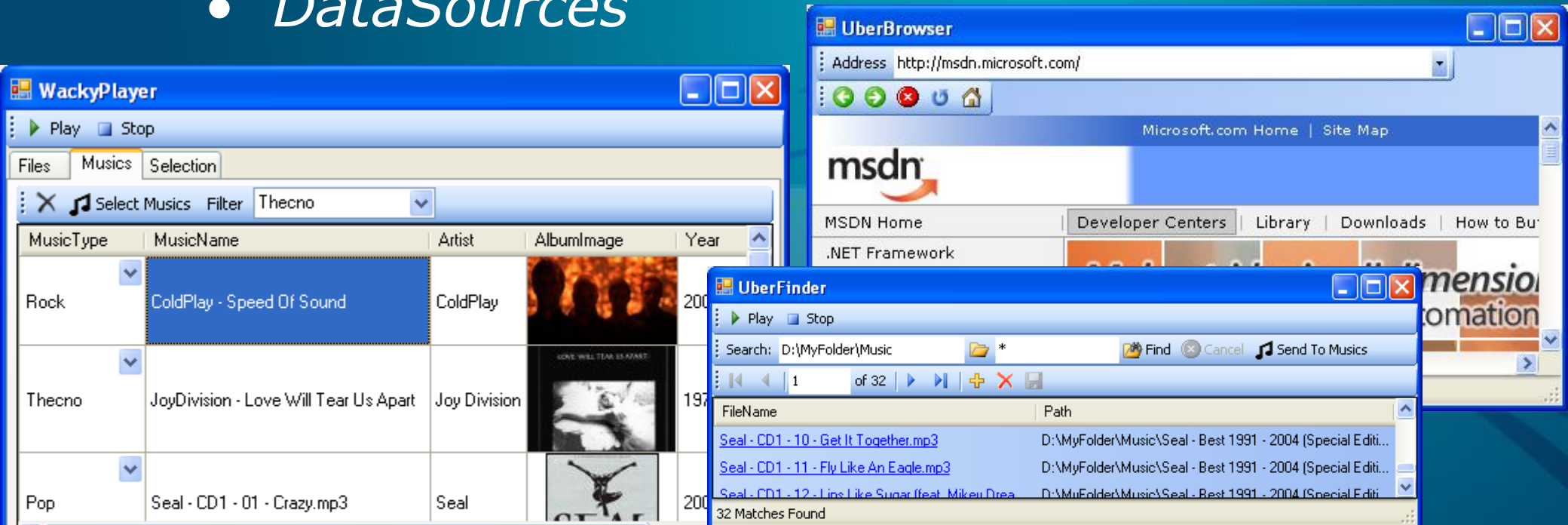


- Novo *Data-Binding*

- Algumas notas

Novas Funcionalidades

- Novos Controlos
- Novo projecto *WindowsApplication*
- *Resources* and *Settings Designer*
- *DataSources*



Novos Controlos

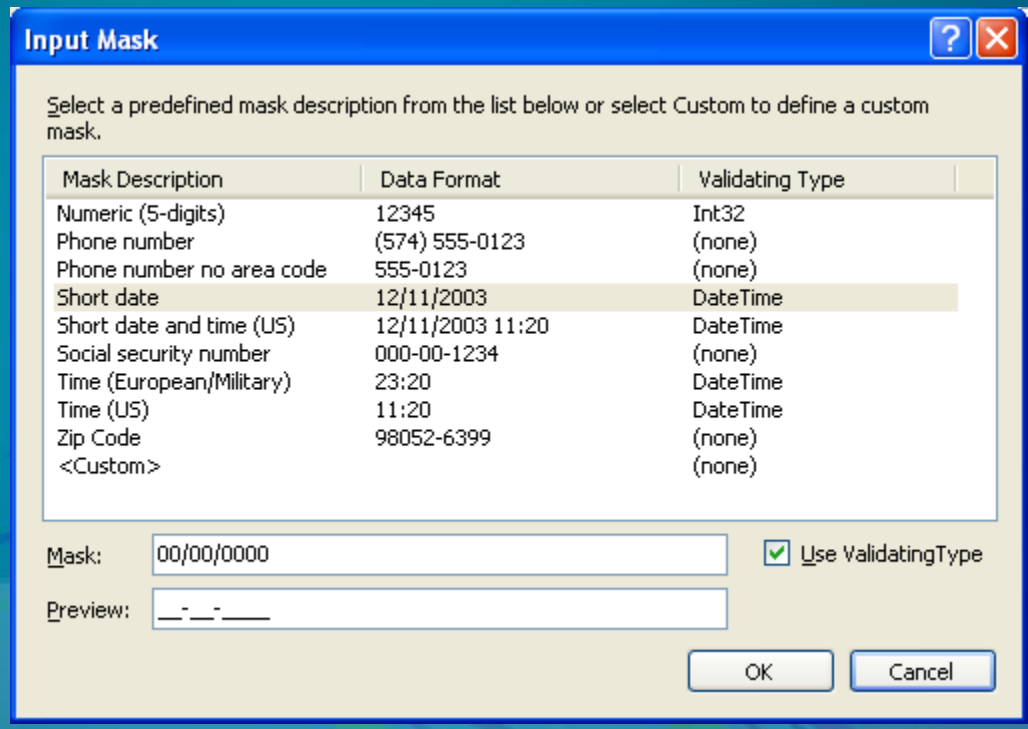
DataGridView
BindingNavigator
ComboBox e TextBox: suporte para *autocompletion*;
Strip Controls:

- ToolStripContainer
- MenuStrip
- ToolStrip
- StatusStrip

MaskedTextBox: validação do texto;
SoundPlayer

FlowLayoutPanel
TableLayoutPanel

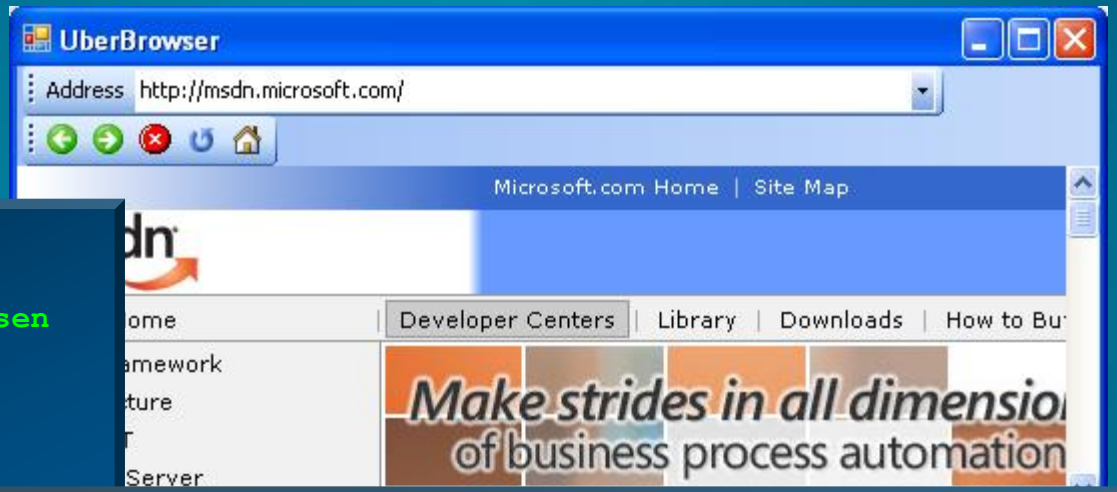
SplitContainer
WebBrowser
ListView – suporte para 3 novas funcionalidades: *tile view*, *grouping* e *drag-and-drop*;



UberBrowser Demo ...

Controlos:

- **WebBrowser;**
- **ToolStripContainer** tem um **ToolStripPanel,** que tem uma **ToolStrip;**
- **StatusStrip.**



- **ComboBox:**

```
enum AutoCompleteMode {
  None = 0x0,          // No autocompletion
  AutoSuggest = 0x1,   // Possible matches chosen
                       // from drop-down list
  AutoAppend = 0x2,    // Possible matches
                       // appended to text
                       // while typing
  AutoSuggestAppend = 0x3 // AutoSuggest
                          // AutoAppend c
}
```

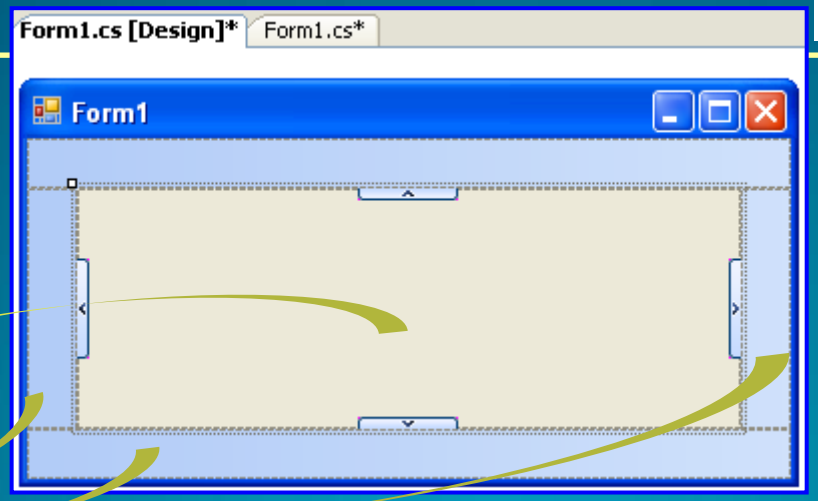
```
enum AutoCompleteSource {
  FileSystem = 0x1,          // File system
  HistoryList = 0x2,         // All URLs from History list
  RecentlyUsedList = 0x4,    // All URLs from Recently Used list
  AllURL = 0x6,              // HistoryList + RecentlyUsedList
  AllSystemSources = 0x7,    // FileSystem + AllURL
  CustomSource = 0x40,       // AutoCompleteCustomSource
  None = 0x80                // No source
}
```

ToolStrips

MenuStrip
ToolStrip
StatusStrip

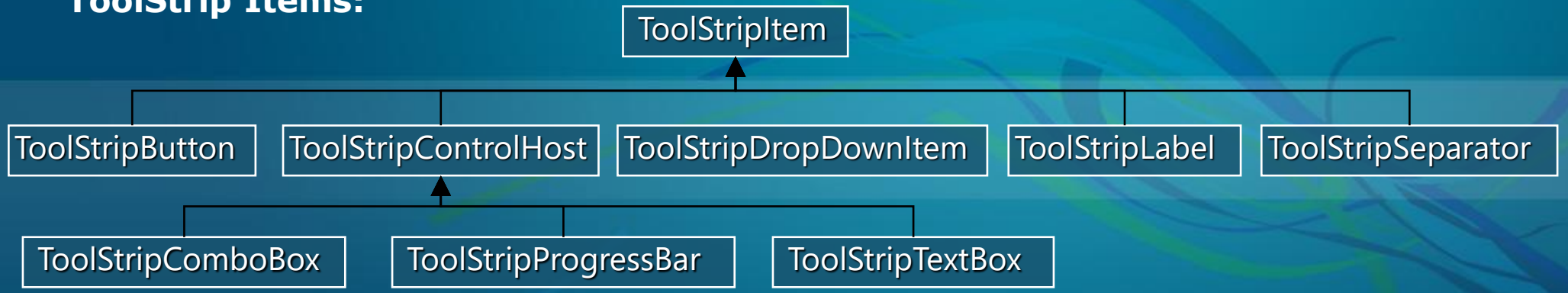
ToolStripContainer:

- **ToolStripContentPanel**
- **ToolStripPanel**

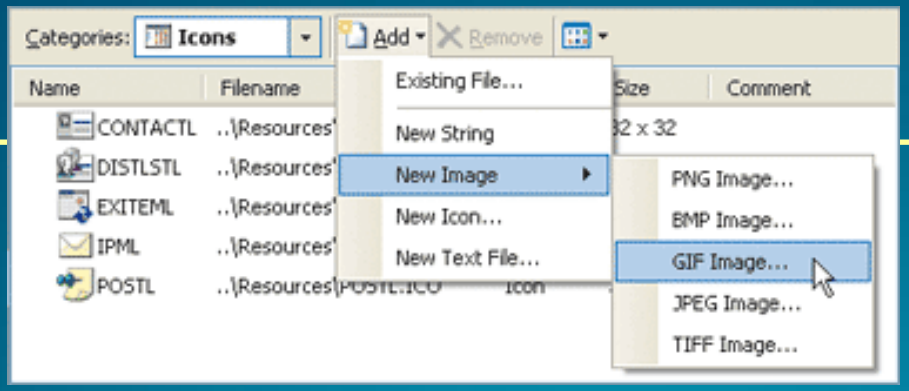



```
toolStripContainer1.TopToolStripPanel.Controls.Add(this.toolStrip1);
```

ToolStrip Items:



Resources



Microsoft .net Framework Windows® Forms

Novas características:

- ***Resources Designer;***
- *Drag and Drop*;
- *Strongly typed*;
- Várias categorias incluindo: ***Strings***, ***Images***, ***Icons***, **ficheiros de texto e audio**.

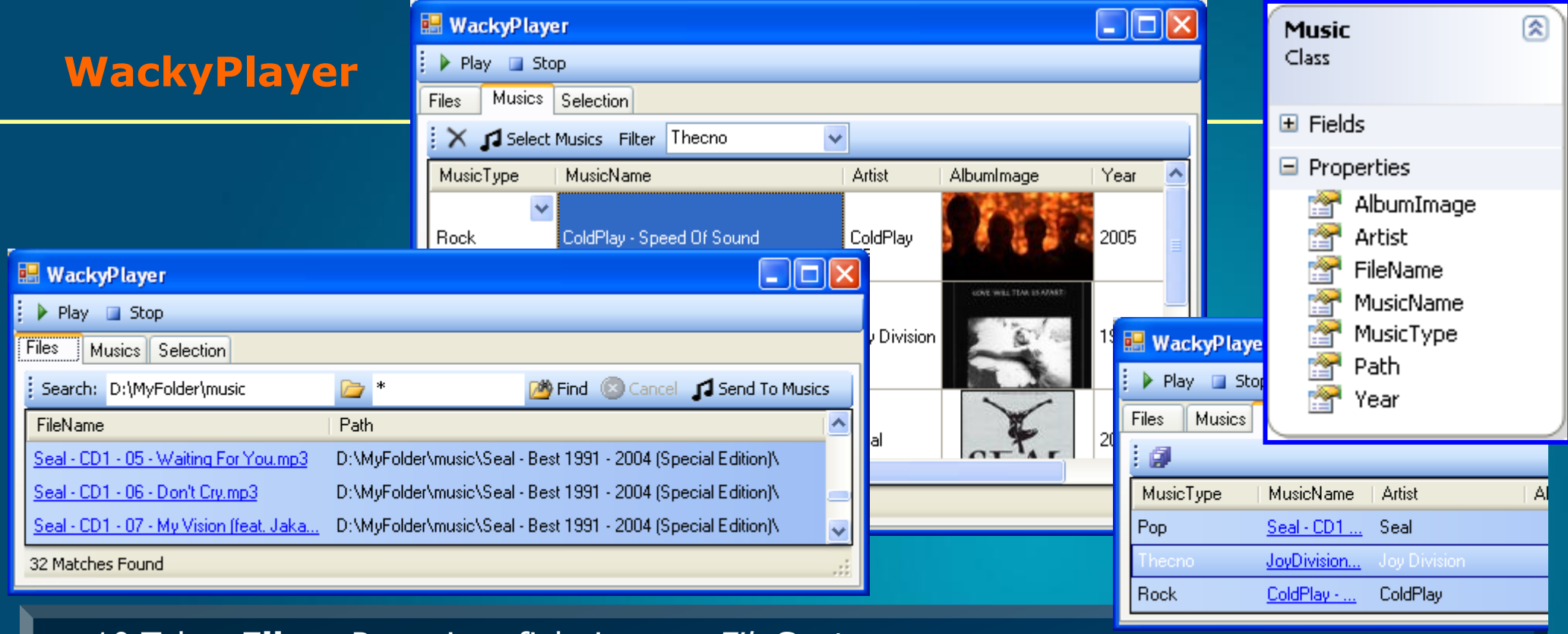
Obter uma Imagem dos *resources* na versão anterior:

```
// Load image resource from project resource file
Assembly assem = Assembly.GetExecutingAssembly();
ResourceManager resman = new ResourceManager("MyApp.Resource1", assem);
Image myImage = (Image) resman.GetObject("MyImage");
```

A mesma operação no VS 2005:

```
// Load strongly typed string resource
Image myImage = Configuration.Resources.MyImage;
```

WackyPlayer



- 1º Tab – **Files**: Pesquisar ficheiros no *FileSystem*;
  Seleccionar ficheiros de música (mp3, wma, etc) e enviá-los para ***Musics Tab***;

- 2º Tab – **Musics**: Classificar as musicas por *MusicType*; Filtrar por *MusicType*; Completar os campos da música e associar uma imagem com *Drag and Drop*;
  Seleccionar musicas e enviá-las para o ***Selection Tab***;

- 3º Tab – **Selection**: Copiar os ficheiros de músicas seleccionadas para uma directoria.

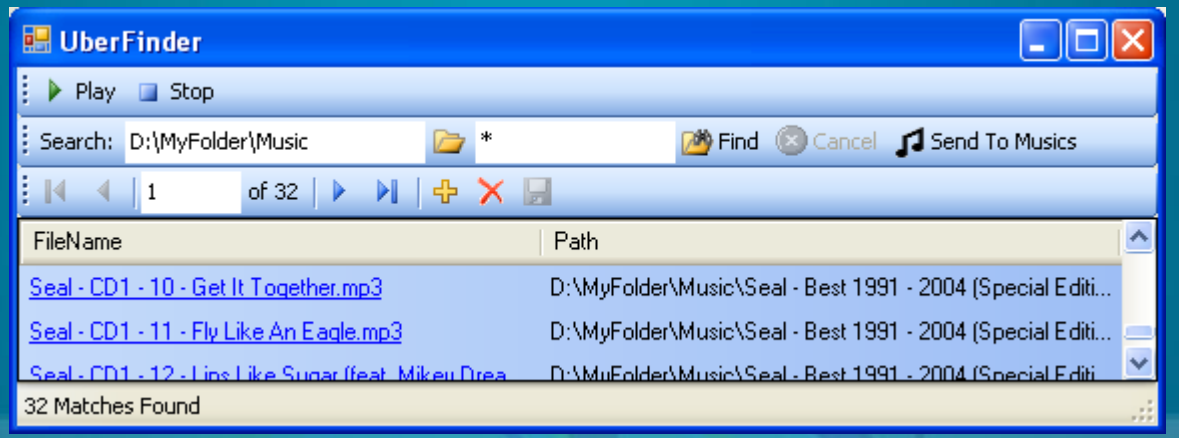
Em qualquer *tab* fazer *play* de uma música.

UberFinder Demo...

Settings
DataSource
SoundPlayer (só funciona para ficheiros wav)
➔ Alternativa: Controlo **AxWindowsMediaPlayer** do namespace **AxWMPLib**.
(propriedades *uiMode* "invisible" e *Visible* "false")

FolderBrowserDialog
ToolTip

StatusStrip
ToolStripProgressBar
BackgroundWorker

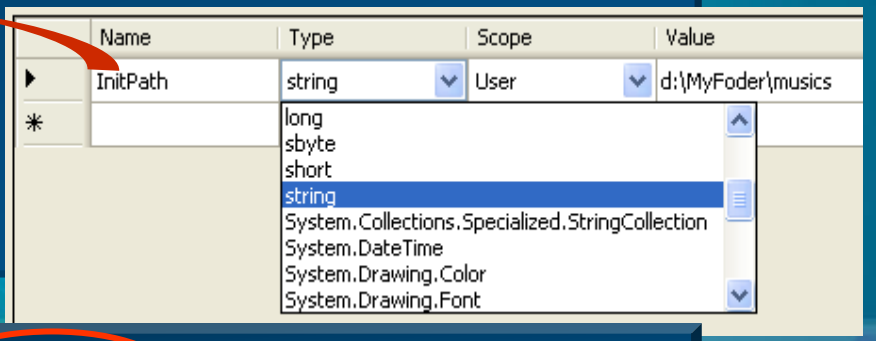


Settings

O *Settings Designer* gera:

- **uma classe** `Settings` (localizada no *namespace* `ProjectName.Properties`),
- com **uma propriedade para cada *setting*** especificado e do **respectivo tipo**.

```
namespace ProjectName.Properties {
internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase {
  private static Settings defaultInstance = ((Settings)
    (global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));
  public static Settings Default {
    get {return defaultInstance;}
  }
 public string InitPath {
    get {return ((string)(this["InitPath"]));}
    set {this["InitPath"] = value;}
 }
```




```
txtPath.Text = Properties.Settings.Default.InitPath;
```

app.config | FrmWackyPlayer.cs | Form1.cs | Form1.cs [Design]

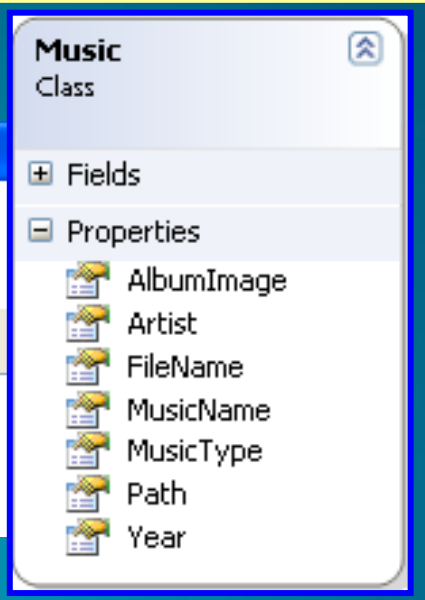
```
<userSettings>
    <UberFinder.Properties.Settings>
        <setting name="InitPath" serializeAs="String">
            <value>d:\MyFoder\musics</value>
        </setting>
    </UberFinder.Properties.Settings>
</userSettings>
```

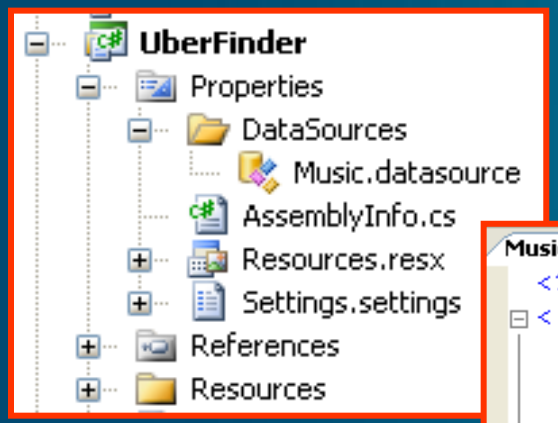
DataSource

Representa os dados disponíveis para uma aplicação.

As fontes podem ser:

- Base de Dados;
- *WebServices*;
- Colecções de objectos.

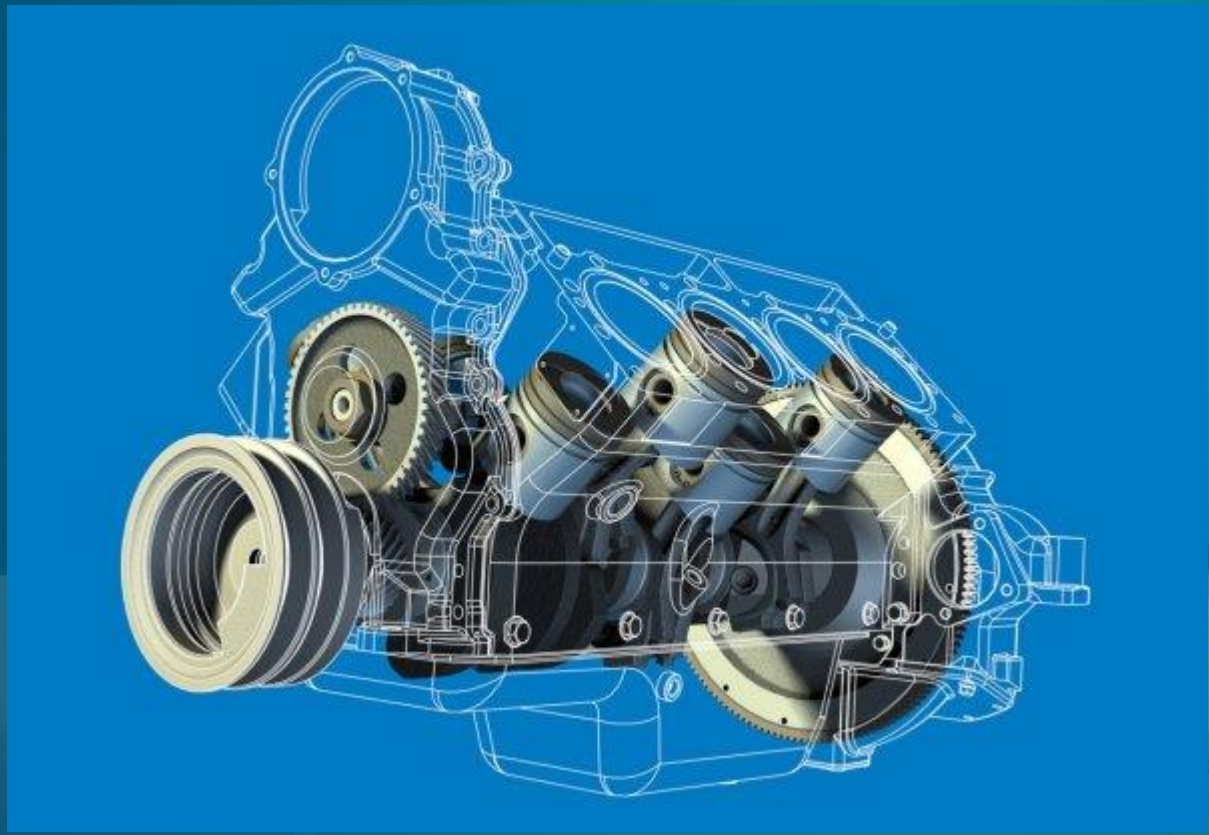




Music.datasource* | Form1.cs [Design]* | Form1.cs*

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    This file is automatically generated by Visual Studio .Net. It is
    used to store generic object data source configuration information.
    Renaming the file extension or editing the content of this file may
    cause the file to be unrecognizable by the program.
-->
<GenericObjectDataSource
  DisplayName="Music" Version="1.0" xmlns="urn:schemas-microsoft-com:xml-msdatasource">
    <TypeInfo>
      UberFinder.Music, UberFinder, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
    </TypeInfo>
</GenericObjectDataSource>
```

BackgroundWorker



FinderWorker

Problema: Não bloquear a aplicação durante o processo de pesquisa;
Solução: Executar a pesquisa numa *thread* separada;

Problema: A *thread* de pesquisa de ficheiros não pode fazer actualizações à UI;
Solução: Executar a pesquisa numa *thread* separada;

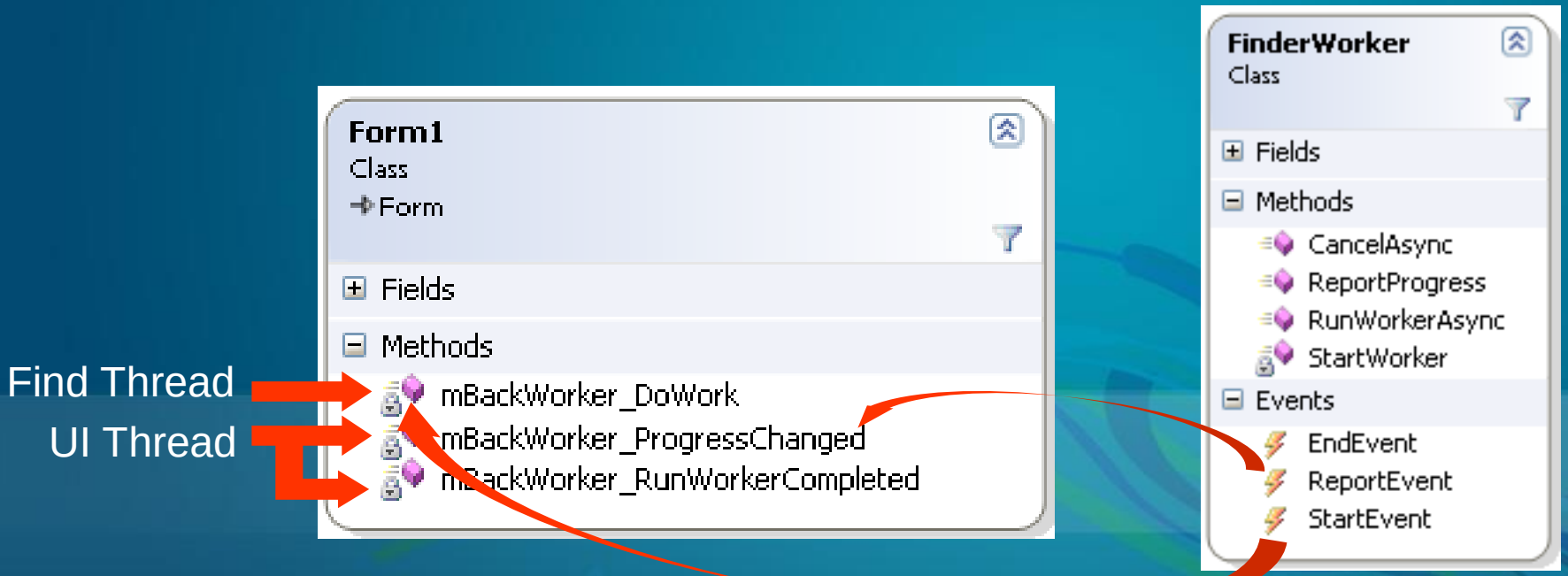
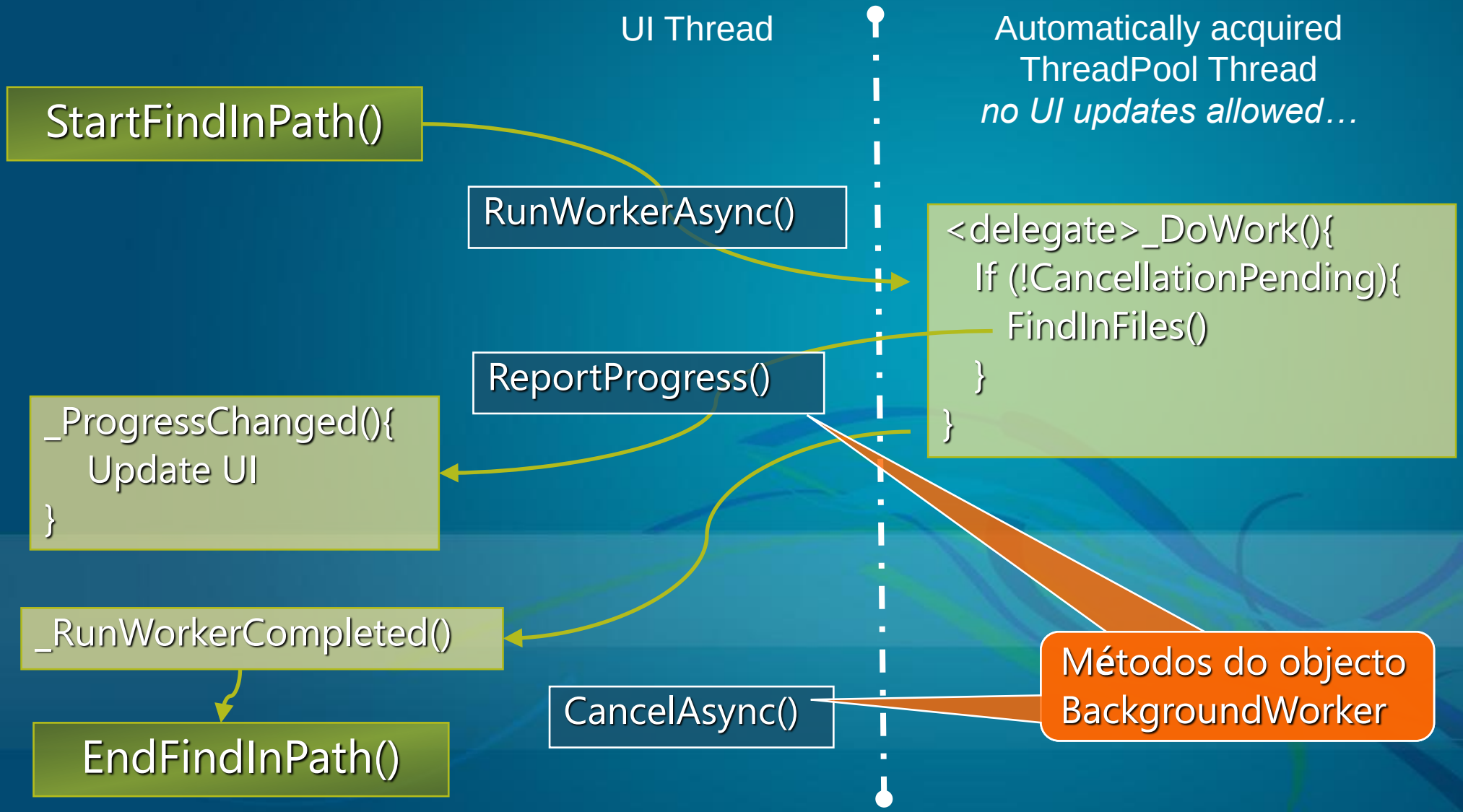
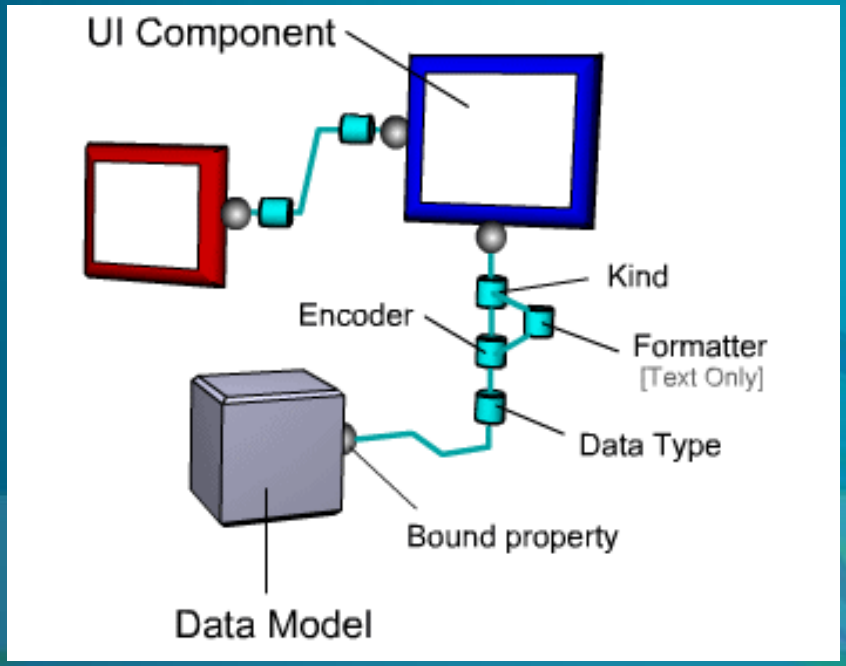


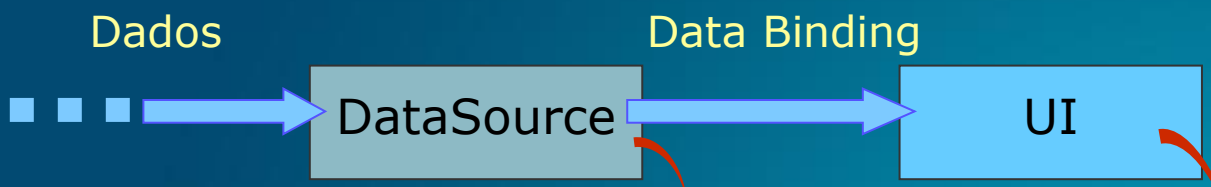
Diagram: BackgroundWorker



Data Binding



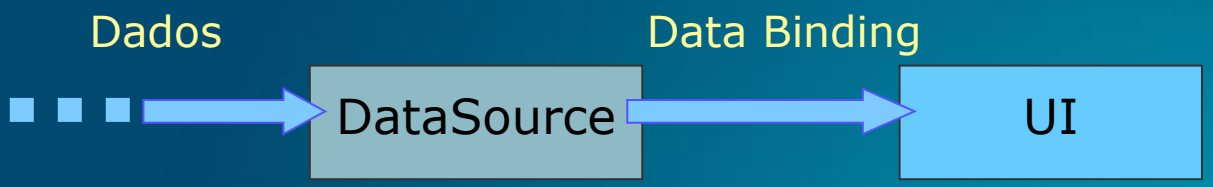
DataBinding



Representa os dados disponíveis para uma aplicação.

- **Details View**: field-style;
- **GridView** (DataGrid na versão1.1): tabular-style;
  (em relação à DataGrid só não tem a ligação a dados Oracle)
  - Tem **6 built-ins** de tipos de colunas contra 2 da versão antiga.
    - DataGridViewButtonColumn
    - DataGridViewCheckBoxColumn
    - DataGridViewComboBoxColumn
    - DataGridViewImageColumn
    - DataGridViewLinkColumn
    - DataGridViewTextBoxColumn

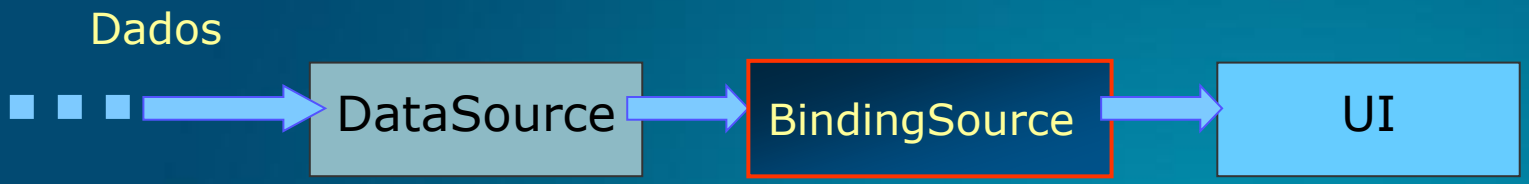
DataBinding...



Na versão 1.x...

- ***DataSources:***
  - Não suporta *custom types*.
- ***Data binding* para `IList`:**
  - O ***Windows Forms Designer*** não suporta a criação automática de colunas.
  - Não suporta as funcionalidades base de **edição** e **navegação** (necessário implementar várias interfaces adicionais).

BindingSource (Abordagem *Model View Control*)



Uma interface consistente para acesso a diferentes tipos de ***data sources***.

Objectivos:

- **Indirecção entre a UI e os dados**, que disponibiliza serviços para: *currency manager*, edição, notificação de alterações, entre outros;
- Actua como um ***strongly typed data source***;
- Disponibiliza um mecanismo de inspecção das propriedades dos **data sources** em *design time*.

Currency management: disponibiliza métodos, propriedades e eventos para controlar programaticamente a navegação nos elementos do data source.

Data Binding Steps...

(os passos não são todos obrigatórios)

1. Criar o **Data Source**.
   Esta informação apenas será usada pelo VS em design time.

2. Adicionar um **Binding Source** através do Designer.
   Definir o **Binding Source** para o **Data Source** anterior.

```
bindingSource1.DataSource = typeof(<Namespace>.<Class>);
```

3. Definir uma **GridView** para o **BindingSource** anterior.

```
dataGridView1.DataSource = bindingSource1;
```

4. Popular o **BindingSource** e consequentemente a **GridView**:

```
bindingSource1.DataSource = <list: IBindingList ou IList, IListSource, ITypedList>;
```

Data Binding Steps...

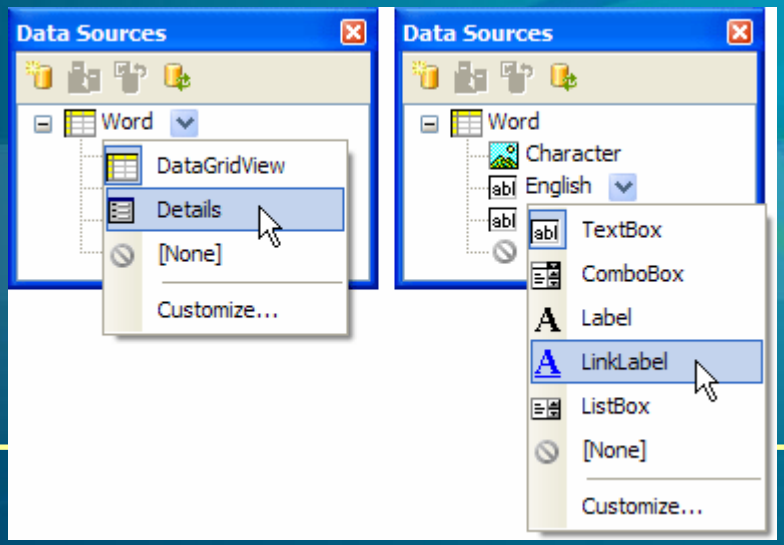
Alternativas:

1. Criar um **BindingSource** programaticamente ➔ Não é actualizado no ***Forms Designer***.

2. Criar a **GridView** e fazer o *binding* para um tipo em *design time,* cria **automaticamente um BindingSource** e ***DataSource***.

3. "Arrastar" um **data source** para um form.

   O **data source** pode ser configurado para gerar automaticamente uma vista do tipo *tabular style* (DataGridView) ou *details style*.



BindingList<T>

```
// BindingSource creates a BindingList<Item> internally
bs.DataSource = typeof(Item);
```

BindingList<T>:

- permite ao **BindingSource** converter um tipo numa **lista *strongly typed***;
- implementação genérica da interface **IBindingList**.

IBindingList: interface mínima requerida pelo **BindingSource** para suportar edição, ordenação, pesquisa e notificação de actualizações.

IList: não disponibiliza ao **BindingSource** edição, ordenação, pesquisa e notificação de actualizações (suficiente para controlos não editáveis como a ComboBox ou ListBox).

IBindingList

Microsoft .net Framework Windows® Forms

```
public interface IBindingList : IList, ICollection, IEnumerable {
  // Editing members
  bool AllowNew { get; } // The default is true
  bool AllowEdit { get; } // The default depends on the underlying type contained in the list.
  bool AllowRemove { get; } // The default is true
  Object AddNew(); // Adds and returns a new list data

  // Sorting members
  bool SupportsSorting { get; } // The default is false in BindingList
  bool IsSorted { get; } // Is the list data source sorted?
  PropertyDescriptor SortProperty { get; } // Current sort column
  ListSortDirection SortDirection { get; } // Current sort direction
  void ApplySort(PropertyDescriptor property, ListSortDirection direction);//Sort list by column/direction
  void RemoveSort(); // Revert to an unsorted state

  // Searching members
  bool SupportsSearching { get; } // The default is false in BindingList
  int Find(PropertyDescriptor property,  object key); // Find a list data item whose specified property
                                                      // matches the provided key value
  // Indexing members
  void AddIndex(PropertyDescriptor property); // Add index to desired column
  void RemoveIndex(PropertyDescriptor property); // Remove index from desired column

  // Change notification members
  bool SupportsChangeNotificationCore { get; } // Supported?
  event ListChangedEventHandler ListChanged; // Broadcast list change
```

Sorting…

Problema:

- Nenhuma implementação de **IBindingList** está sob a obrigação de implementar a interface completa;
- **BindingList<T>** não implementa as funcionalidades de **ordenação e pesquisa.**

Razões:

- Dificuldade de implementar uma "**ordenação genérica genuína**".
- Diferentes tipos têm diferentes propriedades, que podem não ser ordenáveis ou pesquisáveis.

Core Members ...

BindingList<T> tem um conjunto de propriedades e métodos, com o mesmo nome do correspondente membro em **IBindingList** acrescido do sufixo "***Core***".

```
public class BindingList<T> : IBindingList, ... {
  // Core sort methods
  protected virtual void ApplySortCore(PropertyDescriptor property, ListSortDirection direction);
  protected virtual void RemoveSortCore();

  // Core sort properties
  protected virtual bool SupportsSortingCore { get; }
  protected virtual bool IsSortedCore { get; }
  protected virtual ListSortDirection SortDirectionCore { get; }
  protected virtual PropertyDescriptor SortPropertyCore { get; }
}
```

Cada membro do **IBindingList** é um wrapper para o correspondente membro "***Core***" de **BindingList<T>**.

```
public class BindingList<T> : IBindingList, ... {
  public bool SupportsSorting {
    get { return this.SupportsSortingCore; }
  }
  protected virtual bool SupportsSortingCore {
    get { return false; }
  }
```

Sorting…

A ordenação é iniciada com o *click* sobre o *header* de uma coluna.
Este processo tem 3 fases:

1. Determinar a ***metada*** para ordenação: **PropertyDescriptor e ListSortDirection;**
2. O **DataGridView** invoca **IBindingList.ApplySort** passando **PropertyDescriptor** e **ListSortDirection** como argumentos;
3. Concluído o **ApplySort**, a **DataGridView** faz o repaint para reflectir a ordenação.

O método **ApplySortCore** tem de escolher o critério de comparação em função do **PropertyDescriptor** e **ListSortDirection** seleccionados.

Como ordenar?

- **BindingList<T>** herda de **Collection<T>;**
- **Collection<T>** tem uma propriedade **Items** que retorna uma referencia para **List<T>**;
- **List<T>** tem um método **Sort** para ordenação.

```
public class List<T> : IList<T>, ... {
   ...
   // Use system-provided IComparer
   public void Sort();
   // Use custom IComparer<T> object
   public void Sort(IComparer<T> comparer);
   // Creates an internal IComparer shim that calls the Comparison<T>
   // delegate to perform the sort
   public void Sort(Comparison<T> comparison);
   // Use custom IComparer<T> object to sort a portion of the list
   public void Sort(int index, int count, IComparer<T> comparer);
   ...
}
```

Implementar IComparer


```
public class PropertyComparer<T>:System.Collections.Generic.IComparer<T> {
  // Members
  private PropertyDescriptor _property;
  private ListSortDirection _direction;

  // Constructor
  public PropertyComparer(PropertyDescriptor property, ListSortDirection direction){...}

  // IComparer<T> interface
  public int Compare(T xValue, T yValue) {...}
  public bool Equals(T xValue, T yValue) {...}
  public int GetHashCode(T obj) {...}
}
```

Obter a propriedade do objecto (xValue ou yValue) por reflexão, baseado no **PropertyDescriptor**:

```
// Get property info by reflection
PropertyInfo propertyInfo = xValue.GetType().GetProperty(_property.Name);

// Get the property from xValue object
Object propertyInfo.GetValue(xValue, null);
```

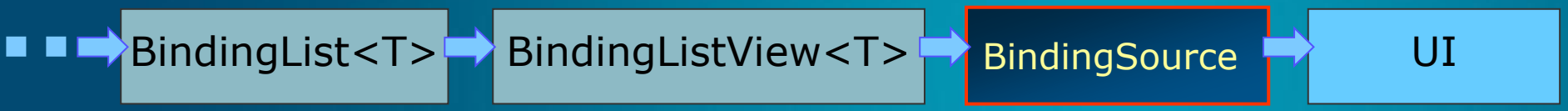
Filtrar... IBindingListView

Microsoft .net Framework Windows® Forms

Suporte para cenários avançados de ordenação e filtragem.

Padrão sugerido para implementação de **IBindingListView**:

- Uma instância de BindingListView<T> tem um BindingList<T> como **data source;**
- O UI control faz binding para BindingListView<T> e não BindingList<T>.




```
public interface IBindingListView : IBindingList, IList, ICollection, IEnumerable {
  string Filter { get; set; }
  ListSortDescriptionCollection SortDescriptions { get; }
  bool SupportsAdvancedSorting { get; }
  bool SupportsFiltering { get; }

  void ApplySort(ListSortDescriptionCollection sorts);
  void RemoveFilter();
```

Drag & Drop

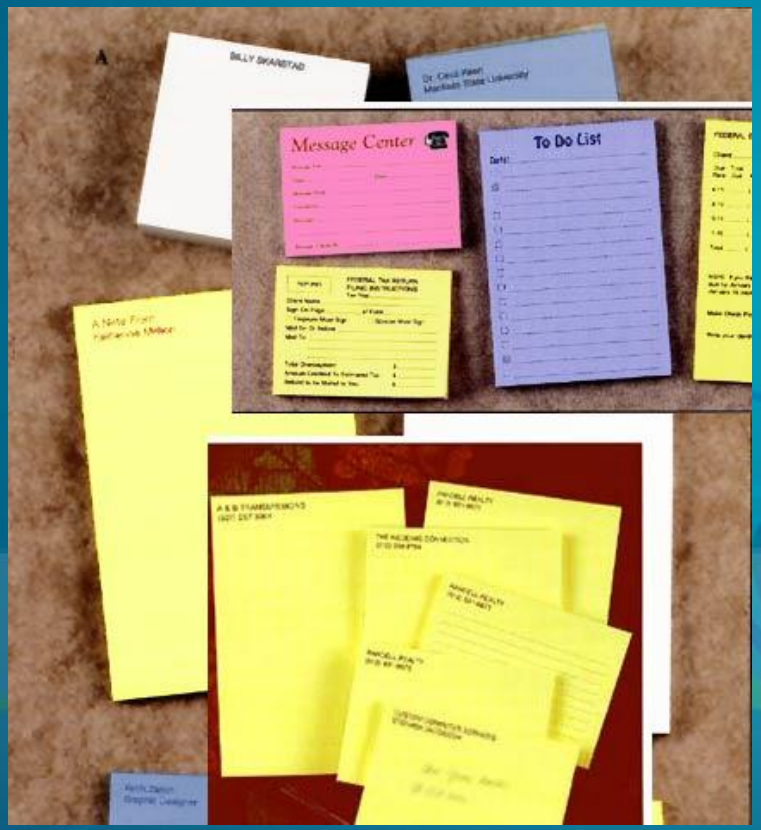
Microsoft .net Framework Windows® Forms

Propriedade AllowDrop da DataGridView.
Tratar os eventos DragOver e DragDrop.

```
private void dataGridView2_DragOver(object sender, DragEventArgs e) {
  //Computes the location of the specified screen point into client coordinates.
  Point point = this.dataGridView2.PointToClient(new Point(e.X, e.Y));
  DataGridView.HitTestInfo hti = this.dataGridView2.HitTest(point.X, point.Y);

  string file = ((string[])e.Data.GetData(DataFormats.FileDrop))[0];
  // Image column?
  if ((this.dataGridView2.Columns[hti.ColumnIndex].Name ==
    "albumImageDataGridViewImageColumn")) {
        e.Effect = DragDropEffects.Copy;
        ...
  }
...
```

Notas



Particularidades de desenho

- **DataGridView – RowHeadersVisible** a false;
- **DataGridView – SelectionMode: FullRowSelect;**
- **DataGridView** - **AllowDrop** a true, para a permitir arrastar um conteúdo (ex: imagem) para uma célula;

- **DataGridViewColumn** – Propriedade **AutoSizeMode** com valor **Fill**, faz com que a coluna seja redimensionada para ocupar toda a DataGrid;

- **ToolStripStatusLabel –** Propriedade **spring**: automatically fills the available space on the StatusStrip

- **BackgroundWorker –** Propriedades
  WorkerReportsProgress e **WorkerSupportsCancellation**

- Em **BindingList** fazer ((List<T>)this.Items).AddRange(mOriginalList);

Barretes...

- **GridView -> InvalidOperationException**

  Se estiver a editar uma linha da GridView quando actualizo a BindingList ou o BindingSource tenho uma excepção InvalidOperationException.

  Esta excepção não tem nada a ver com a descrição que vem no MSDN acerca do metodo Add, que diz que: "*value* differs in type from the existing items in the underlying list".

  Soluções: Fazer previamente o clear da List ou fazer na GridView disable de Adding.

- BackgroundWorker -> CancelAsync() – Atenção à validação do estado da propriedade de **CancellationPending**

- Diferentes Tipos de ComboBox:

  ToolStripComboBox, DataGridViewComboBoxColumn e ComboBox

  ToolStripComboBox *rendered* pela ToolStrip, **não tem data source.**

Bugs e Defeitos...

- A ToolTip em modo baloon não fica a apontar para o parent control. Só no 2º Click é que fica.
- GridView e BindingList – o SortDirection não funciona. A coluna da GridView não apresenta aquela seta a indicar a ordem de ordenação.
- GridView – Alguns problemas com a propriedade "Enable Editing"
- BindingList<T> não é serializável embora no MSDN indique que sim.
- Não existe uma forma expedita de obter os itens seleccionados na GridView/BindingSource.

  Além disto, os itens seleccionados não são *StrongTypes* (tenho que fazer *casting* para o tipo)

Referências

Referências