Programming Skills C#/.NET

Windows Forms
Classic graphical user interface.
Convert temperatures back and forth.
Model-View-Controller design pattern.
Reuse model classes.

Pattern: Model-View-Controller

Model:

represents the state of an application and provides algorithm to change the state.

View:

handles interaction with the user, *usually* observes the model.

Controller:

reacts to view's events and sends data from the view to the model's algorithms. Might send results back to view.

[SmallTalk]

Pattern: Observer-Observable

Observer:

registers at observable, is informed of state changes.

Observable:

is connected to one or more observers, sends each state change to all current observers.

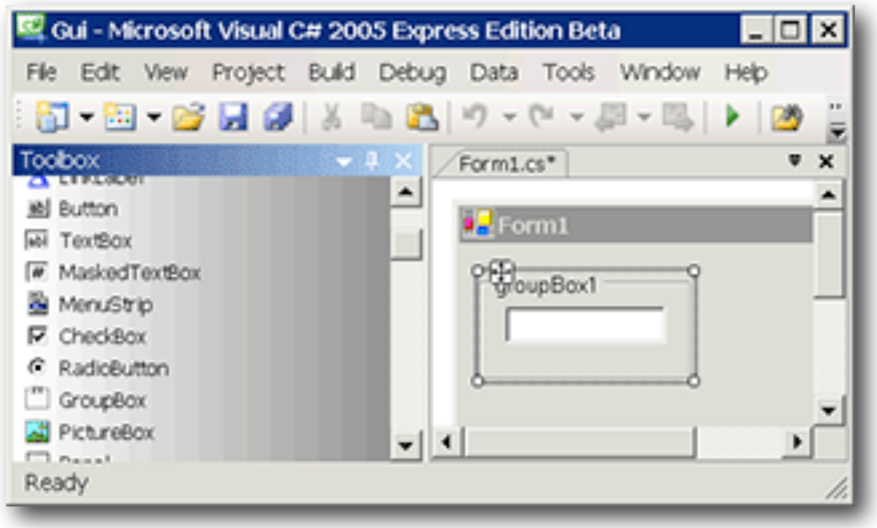
`[java.util]`

View

VS generates a `partial class` (eventually XAML) to create and configure a view.

Create a **Windows Application** project in VS.

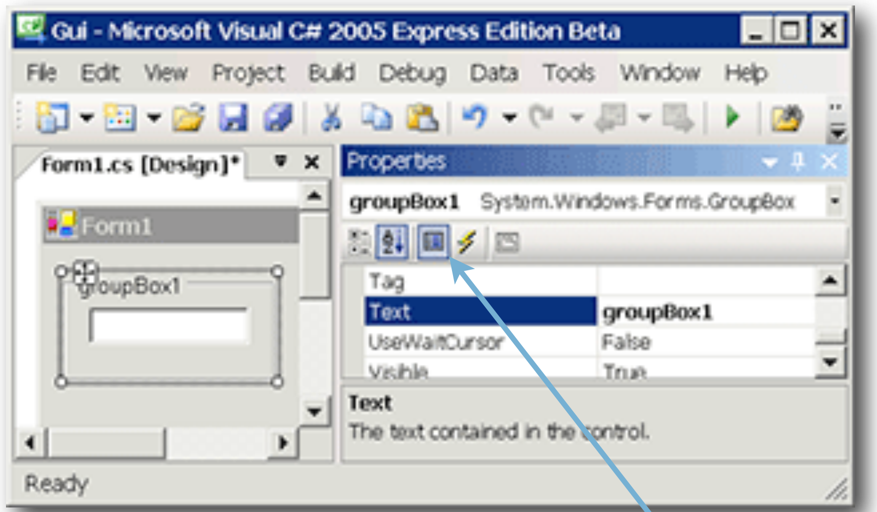
Open **View/Toolbox** and create two `GroupBox`es, each containing one `TextBox`:


Gui - Microsoft Visual C# 2005 Express Edition Beta
File Edit View Project Build Debug Data Tools Window Help
Toolbox
Button
TextBox
MaskedTextBox
MenuStrip
CheckBox
RadioButton
GroupBox
PictureBox
Form1.cs*
Form1
groupBox1
Ready

Properties

View/Properties Window, select a visual object and change properties such as the **Appearance** of the `Text` of the `GroupBox` or the **Design (Name)** of the `TextBox`.

In **Layout** set `Anchor` to **Top,Left,Right** so that the boxes stay at the top of the form and grow horizontally with it.


Gui - Microsoft Visual C# 2005 Express Edition Beta
File Edit View Project Build Debug Data Tools Window Help
Form1.cs [Design]*
Form1
groupBox1
Properties
Ready

groupBox1 System.Windows.Forms.GroupBox

Tag	
Text	groupBox1
UseWaitCursor	False
Visible	True

Text
The text contained in the control.

properties/events

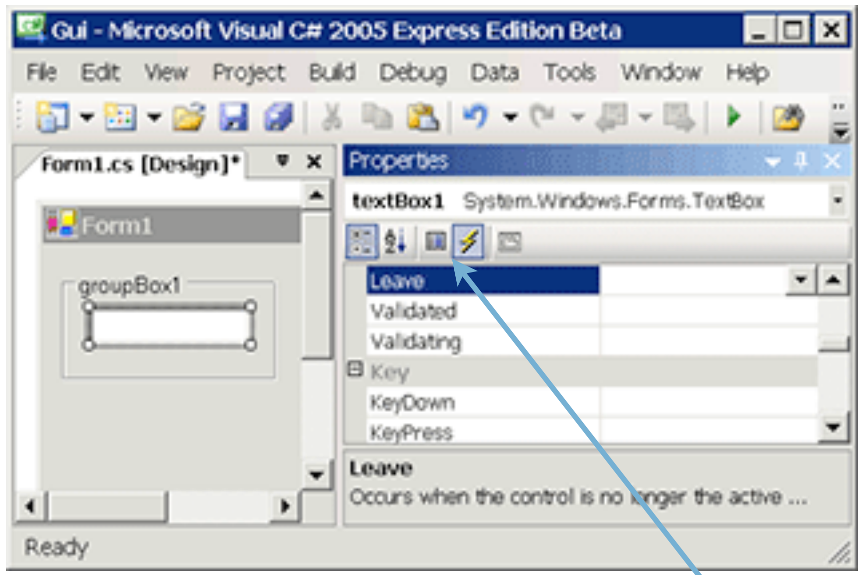
Events

Select a `TextBox` and select or enter method names to handle events such as `KeyPress` or `Leave`.

Method headers are generated as needed.

A double-click creates and connects a handler for the most likely event.

Clear the method name for `TextChanged`.


Gui - Microsoft Visual C# 2005 Express Edition Beta
File Edit View Project Build Debug Data Tools Window Help
Form1.cs [Design]*
Form1
groupBox1
Properties
textBox1 System.Windows.Forms.TextBox
Leave
Validated
Validating
Key
KeyDown
KeyPress
Ready

Leave
Occurs when the control is no longer the active ...

properties/events

Controller

VS creates empty methods for all new names entered to handle events and connects them to the view objects.

The method bodies need to be filled in.

Unfortunately, the `Form` is subclassed to hold the methods; the methods access the model.

- Very tightly coupled MVC architecture.
- View+controller is hardly reusable.

Events

`celsius, fahrenheit`	`Enter`	to speed up typing, if box acquires focus select all text in it.
	`Leave`	to avoid confusion, if box loses focus make sure values in boxes correspond.
	`KeyPress`	*return* requests conversion; to speed up typing, select all text in box.
	`MouseDown`	actually selects all.

Edit: Controller

`TextBox.KeyPress`: *return* requests conversion; to speed up typing, select all text in box.

```
private void box_KeyPress(object sender,
    KeyPressEventArgs e) {
  flag = false;
  switch (e.KeyChar) {
  case '\n': case '\r':
    post((TextBox)sender);
    ((TextBox)sender).SelectAll();
    break;
  }
}
```

C#

`if (condition)`
`  statement`
	selection based on `bool` values.

`else`
`  statement`
	optional `else` part.

`condition`
`  ? expression`
`  : expression`
	conditional evaluation.
	matching types for values required.

`switch {`
`case constant: ...`
	selection based on integer and string values.

`  ...`
`  break;`
	mandatory termination.

`...`
`default:`
`  ...`
`  goto case constant;`
`}`	

`'c' '\n'`	character constants, act as integer values.
`(type)value`	explicit conversion (or unboxing).

Edit: Controller

`post()` runs input through a model and back to a view:

```
private IFunction c2f, f2c;

private void post(TextBox from) {
  var model = from == celsius ? c2f : f2c;
  var to = from == celsius ? fahrenheit : celsius;

  to.Text = model.Y(double.Parse(from.Text)).ToString();
}
```

Edit: Construction

The constructor should be changed to receive the form and group titles and the model to make the view/controller more reusable.

```
public Gui (string title, string fromName, string toName,
    IReversibleFunction model) {
  // ...
  Text = title;
  groupBox1.Text = fromName;
  groupBox2.Text = toName;
  c2f = model;
  f2c = model.inverse();
}
```

Edit: Startup

The main program is changed to pass construction parameters:

```
static void Main() {
  Application.EnableVisualStyles();
  Application.Run(new Gui("Temperatures",
    "Celsius", "Fahrenheit",
    new ReversibleLinearFunction(9.0/5.0, 32.0)));
}
```

Command Line Compilation

VS collects all files below the `bin\` directory.

A command line build can use modules:

```
> mkdir lib
> copy ..\oop\DegF.exe lib
> copy ..\oop\*.netmodule lib
> copy ..\java\DegC.exe lib
> copy ..\java\*.dll lib
> csc /lib:lib /r:DegF;DegC;IReversibleConversion
    /r:ReversibleLinearConversion Gui.cs
```

Configuration

`Gui.exe.config` describes where the other assemblies for `Gui.exe` can be found:

```
<configuration>
  <runtime>
    <assemblyBinding
        xmlns='urn:schemas-microsoft-com:asm.v1'>
      <probing privatePath='lib'/>
    </assemblyBinding>
  </runtime>
</configuration>
```

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/gngrfNETFrameworkConfigurationFileSchema.asp