# Limnor Studio – User's Guide

*Resource manager*

## Contents

# 1 Introduction

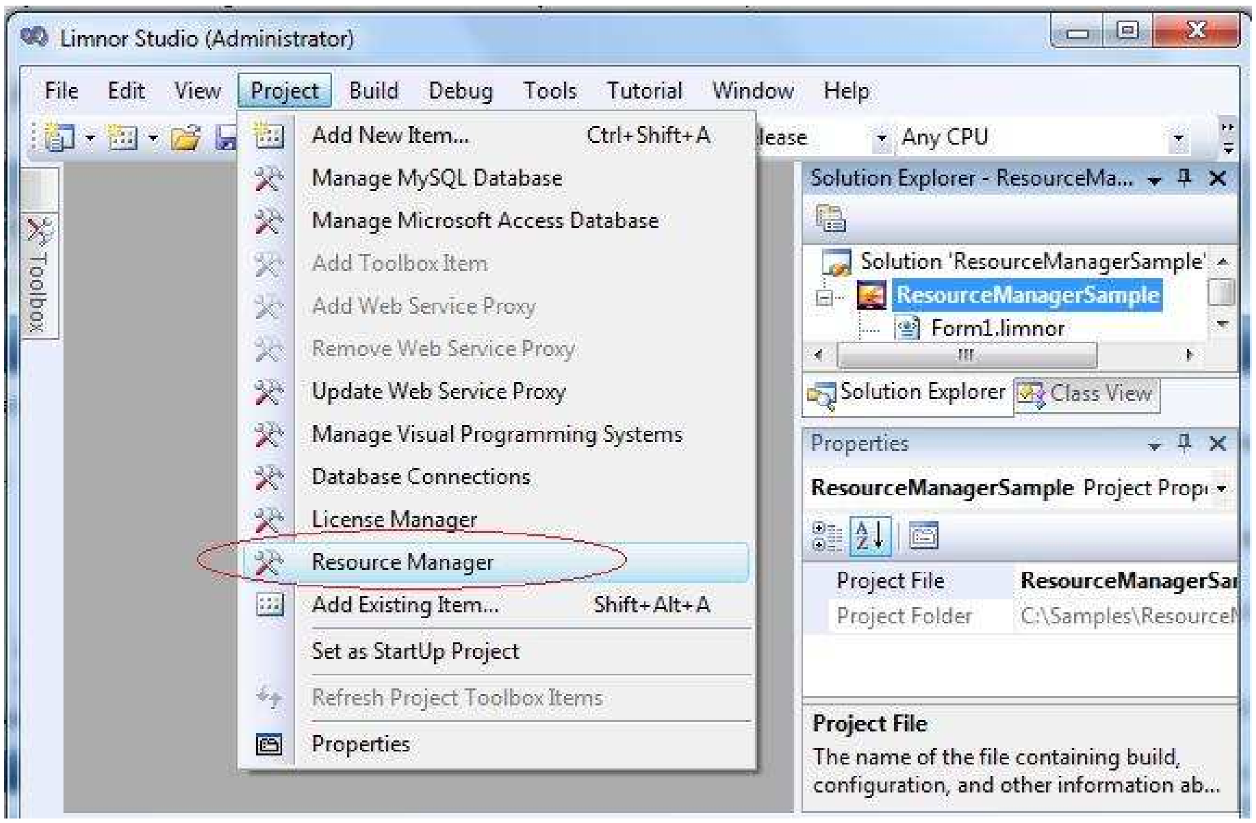
Resource Manager allows embedding resources (text, image, icon and other files) in software. Since the resources are embedded in the software (*.EXE/*.DLL), it is easy to distribute the software to your customers. You do not have to distribute image files, text files, icon files, etc.

Another important usage of the Resource Manager is to help localizing your software. It helps you managing multiple language resources. Multi-language resources are embedded in your software and correct resources are used based on the culture at runtime. Your software may instantly switch language at runtime. For example, a kiosk at an international airport should allow the user to choose language for the user interface.

The Resource Manager may also be used to develop multi-language web pages. See chapter "Multi-Language Web Site" in http://www.limnor.com/support/WebApplicationDevelopment.pdf.

## 1.1 Launch Resource Manager

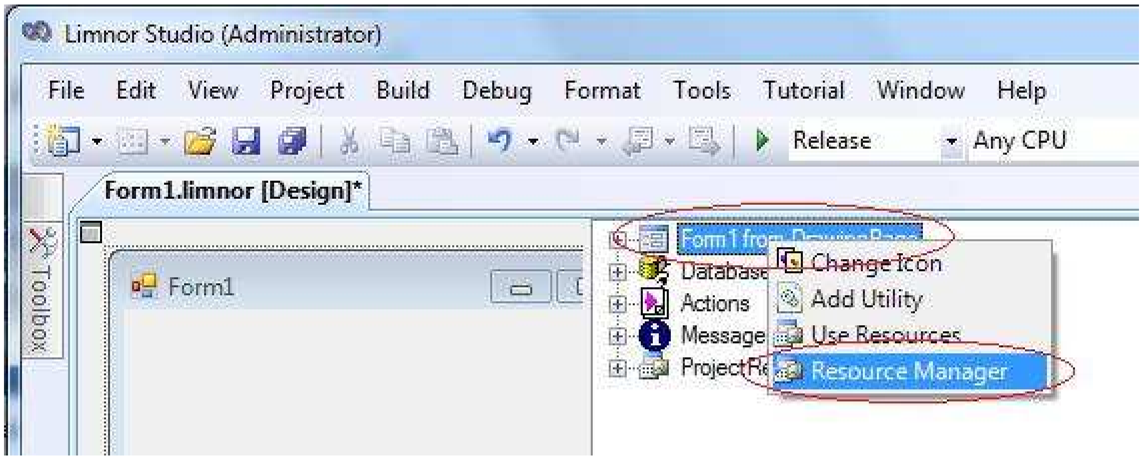
Accessing the Resource Manager via Project menu:


Limnor Studio (Administrator)
File Edit View Project Build Debug Tools Tutorial Window Help
Add New Item... Ctrl+Shift+A
Manage MySQL Database
Manage Microsoft Access Database
Add Toolbox Item
Add Web Service Proxy
Remove Web Service Proxy
Update Web Service Proxy
Manage Visual Programming Systems
Database Connections
License Manager
Resource Manager
Add Existing Item... Shift+Alt+A
Set as StartUp Project
Refresh Project Toolbox Items
Properties
Any CPU
Toolbox
Solution Explorer - ResourceMa...
Solution 'ResourceManagerSample'
ResourceManagerSample
Form1.limnor
Solution Explorer
Class View
Properties
ResourceManagerSample Project Prop
Project File
Project Folder

| | |
|---|---|
| Project File | ResourceManagerSa |
| Project Folder | C:\Samples\Resource |

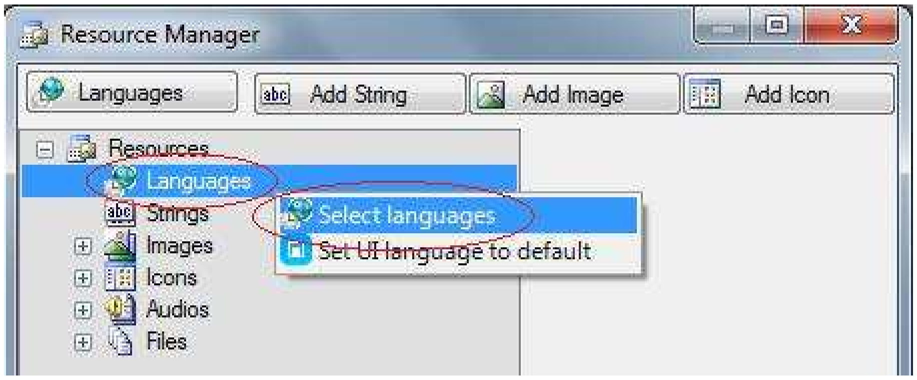
**Project File**
The name of the file containing build, configuration, and other information ab...

Accessing the Resource Manager via Object Explorer:

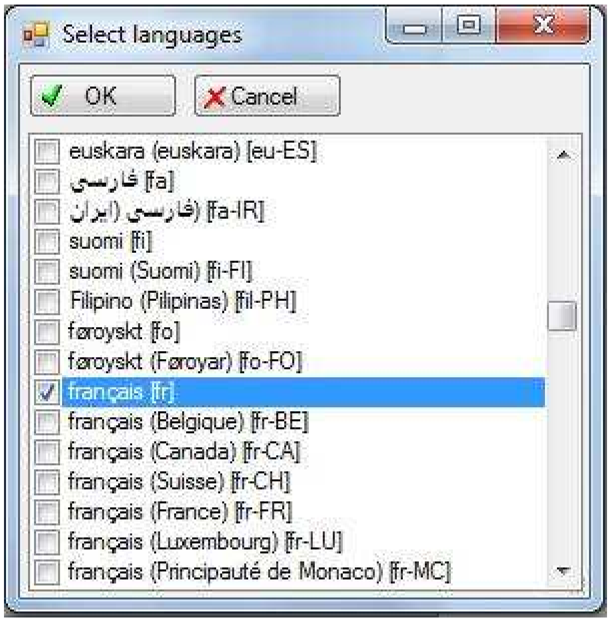


## 1.2 Add supported languages

Right-click “Languages”, choose “Select languages”:

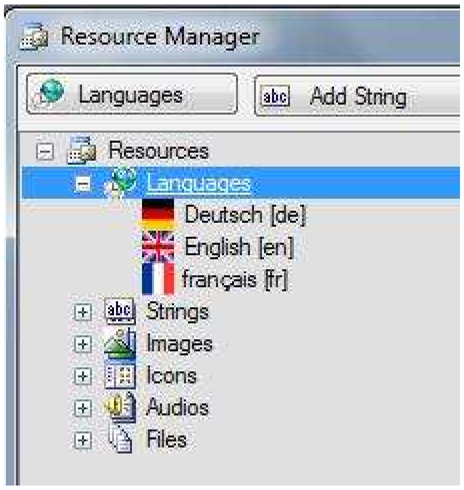


Check all the languages your want your software to support:



- [ ] euskara (euskara) [eu-ES]
- [ ] فارسى [fa]
- [ ] فارسى (ایران) [fa-IR]
- [ ] suomi [fi]
- [ ] suomi (Suomi) [fi-FI]
- [ ] Filipino (Pilipinas) [fil-PH]
- [ ] føroyskt [fo]
- [ ] føroyskt (Føroyar) [fo-FO]
- [x] français [fr]
- [ ] français (Belgique) [fr-BE]
- [ ] français (Canada) [fr-CA]
- [ ] français (Suisse) [fr-CH]
- [ ] français (France) [fr-FR]
- [ ] français (Luxembourg) [fr-LU]
- [ ] français (Principauté de Monaco) [fr-MC]

The checked languages are listed under Languages:

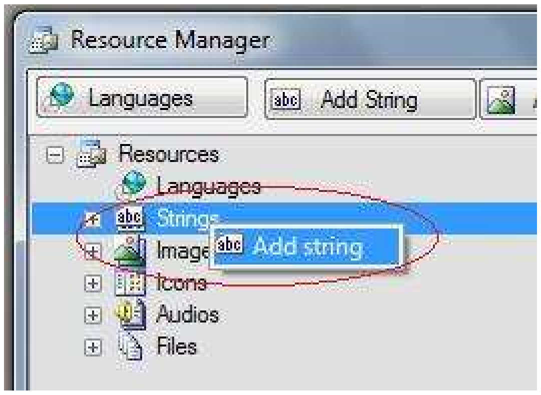


## 2 Create Resources

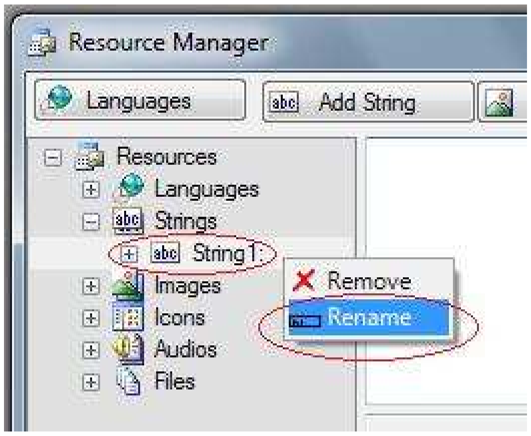
### 2.1 Create String/Text Resources

A string is a sequential list of characters, also called a text.

Right-click on Strings, choose "Add string" to create a new string resource:

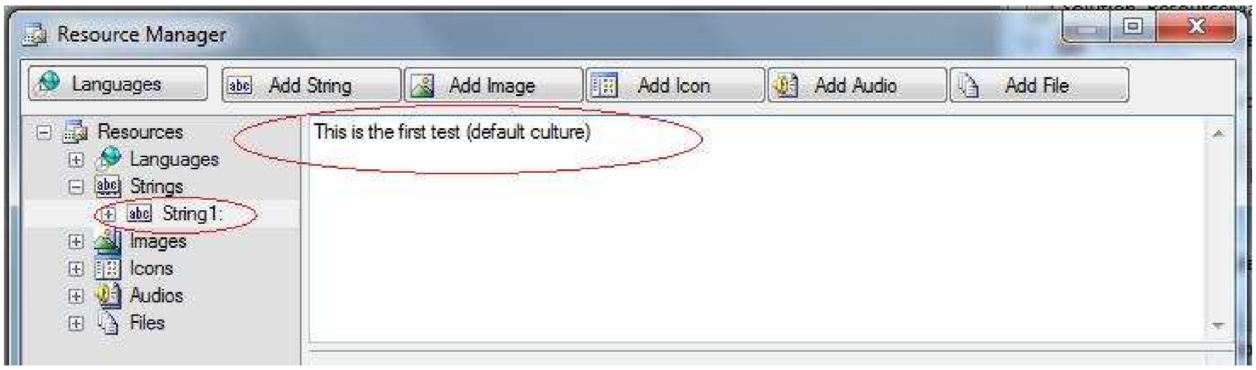


A new string resource is created and named String1. You may right-click it for naming or deleting:



## 2.2 Set text values

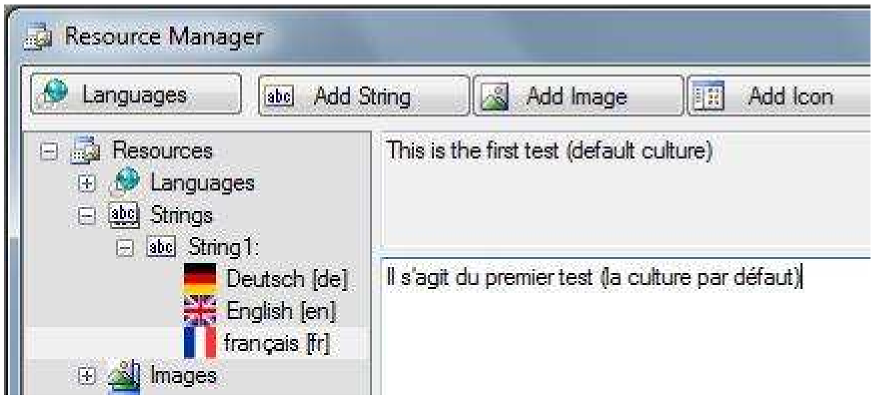
You may type text as the value for this string.



This is the first test (default culture)

Note that this value is for the default culture. If you do not give values for a language then this value will be used for that language.

Select a supported language, type in text in that language:

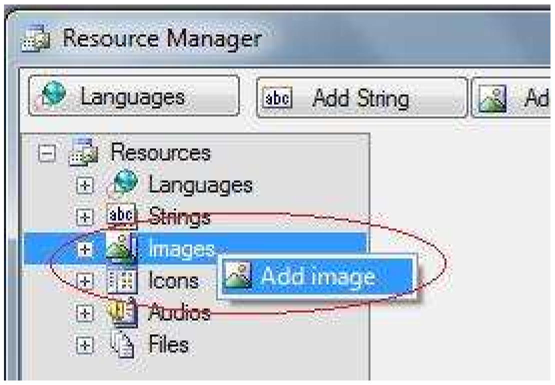


This is the first test (default culture)

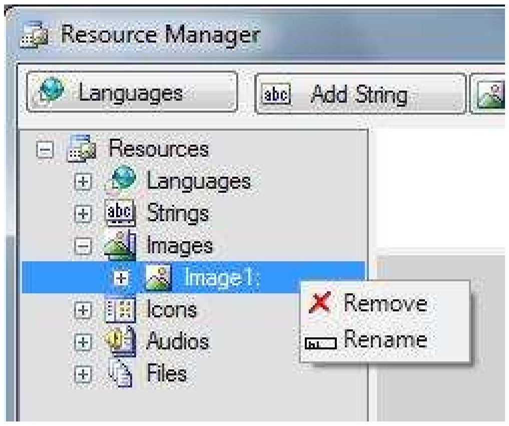
Il s'agit du premier test (la culture par défaut)

## 2.3 Create Image Resources

Right-click Images, choose "Add image":

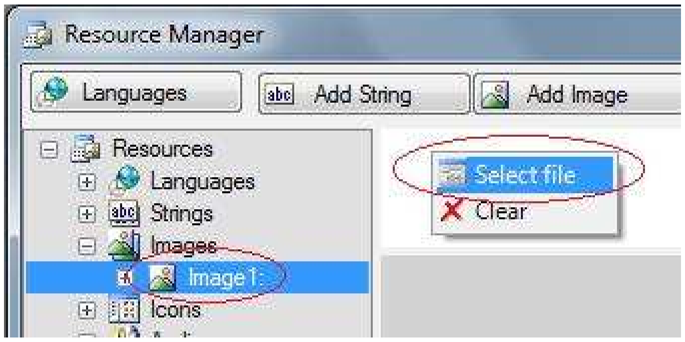


A new Image resource is created. It is named Image1. Right-click it to rename it or delete it:

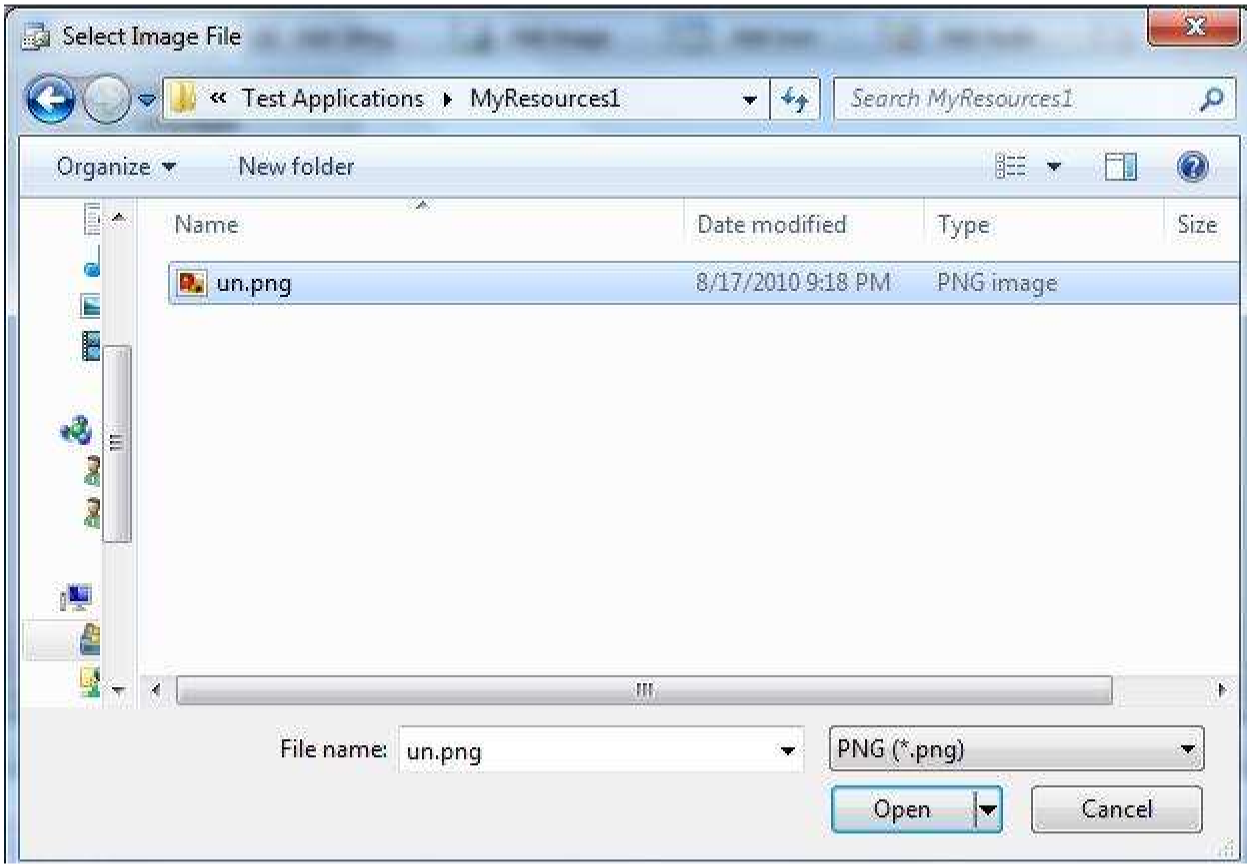


## 2.4 Set Image value

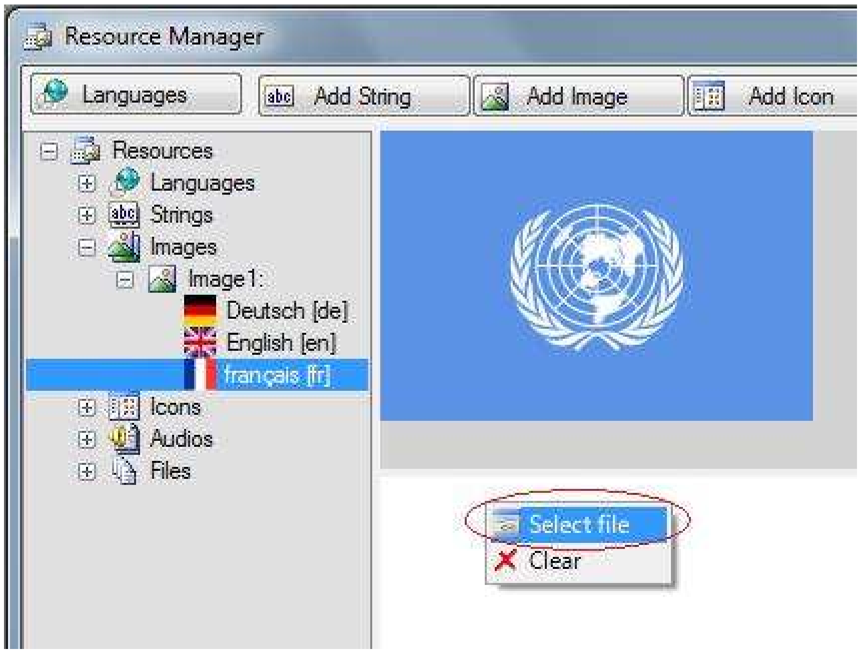
Set default value by selecting the resource name. Right-click the right-pane, choose "Select file"



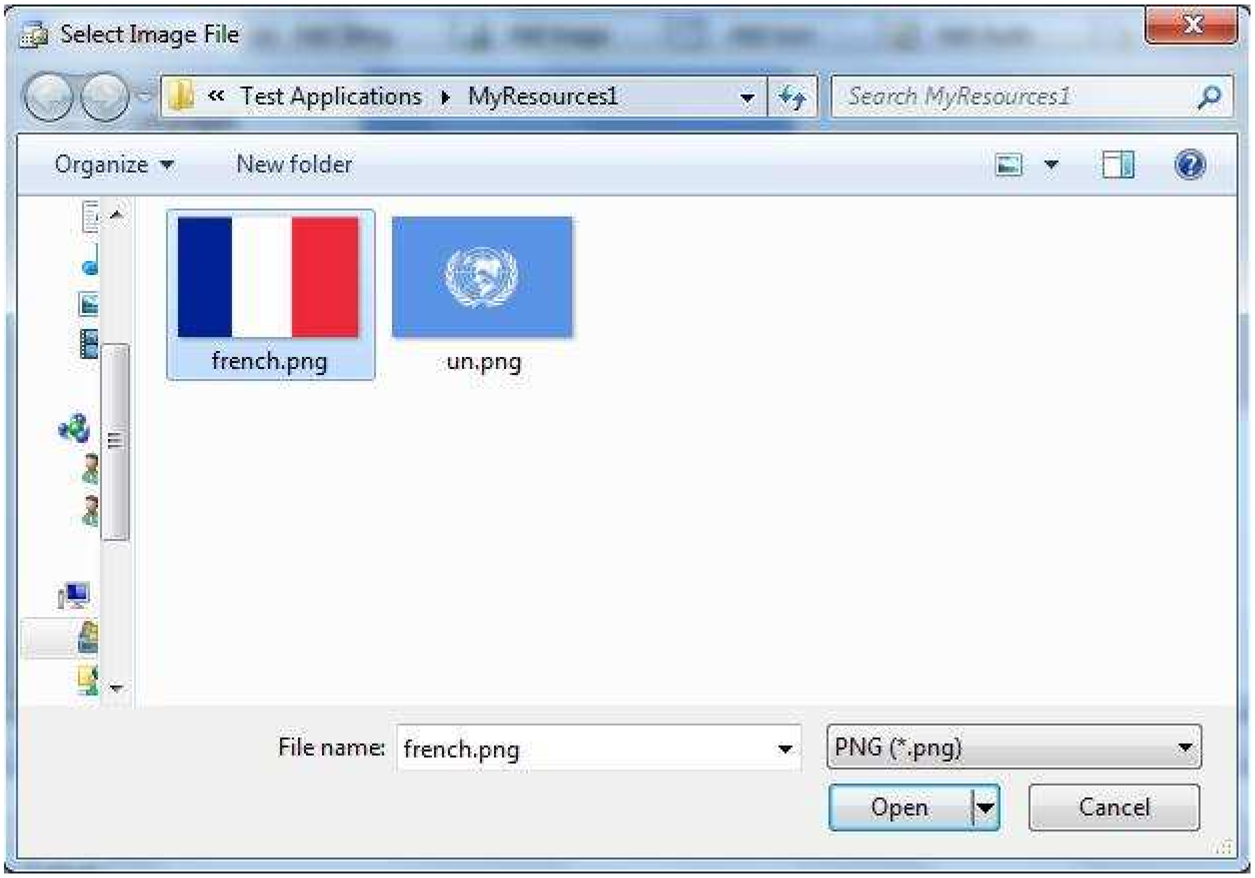
Select the image file you want to use as the default value for this resource:



Select a supported language, right-click on the right pane, choose “Select file”:



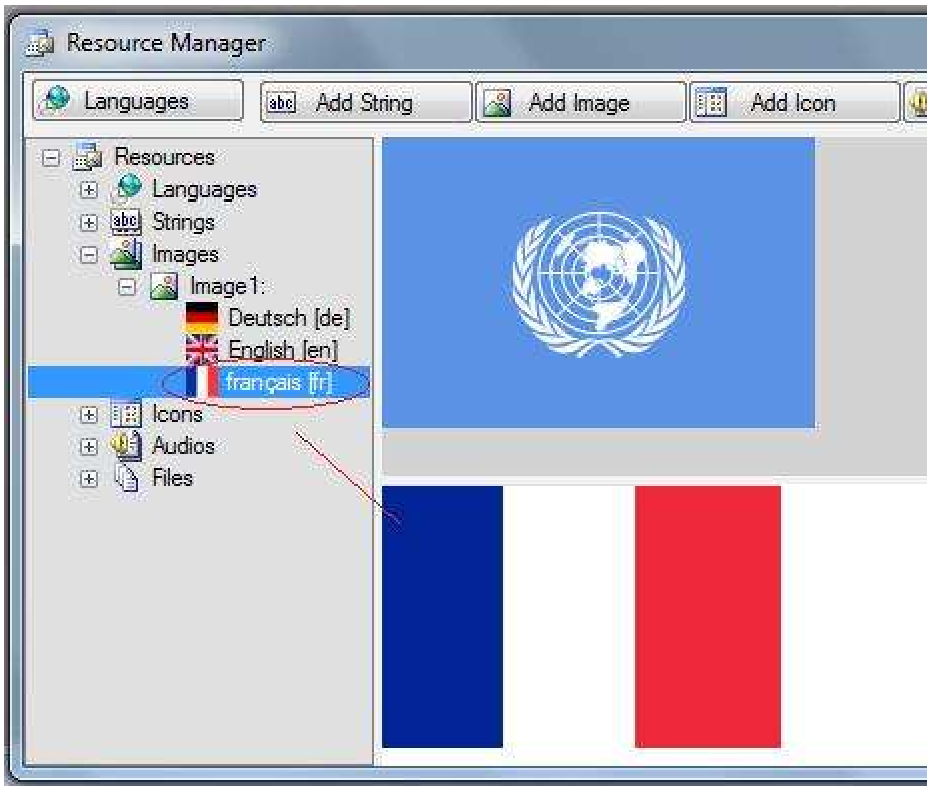
Select the image file for this resource and for this language:



File name: french.png

PNG (*.png)

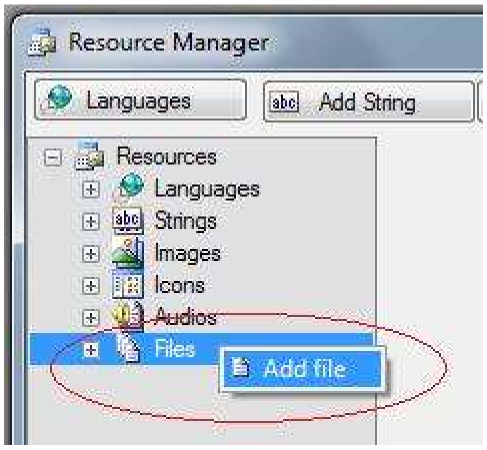
The image appears under the default image:



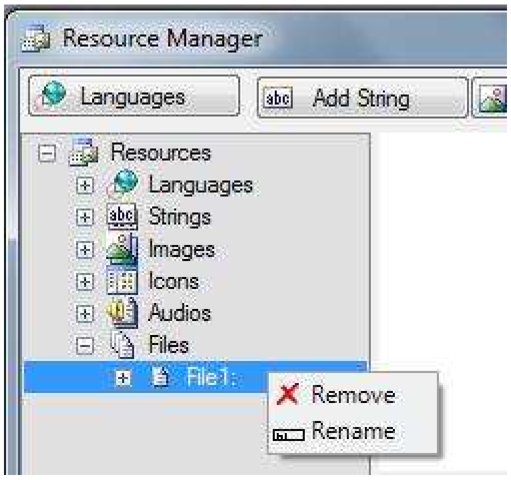
## 2.5 Create File Resources

An entire file can be embedded in your software. For a txt, xml, xsd, html or htm file, the whole file contents are embedded as a string value. Other types of files are embedded as byte arrays.

Right-click Files, choose “Add file”:

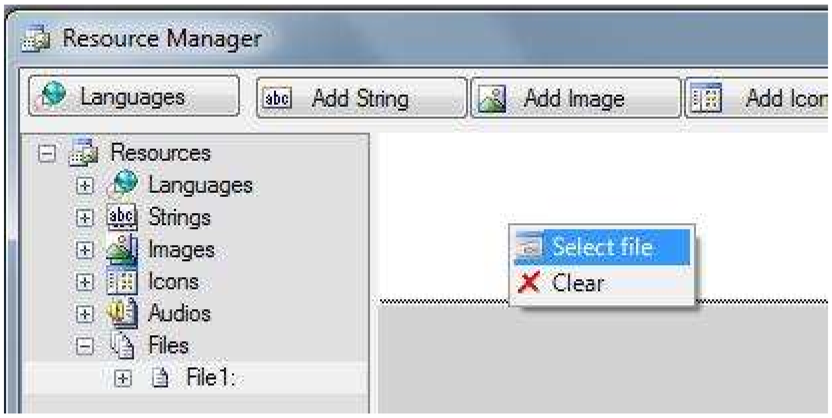


A new File resource is created named File1. Right-click it, you may rename it or delete it.

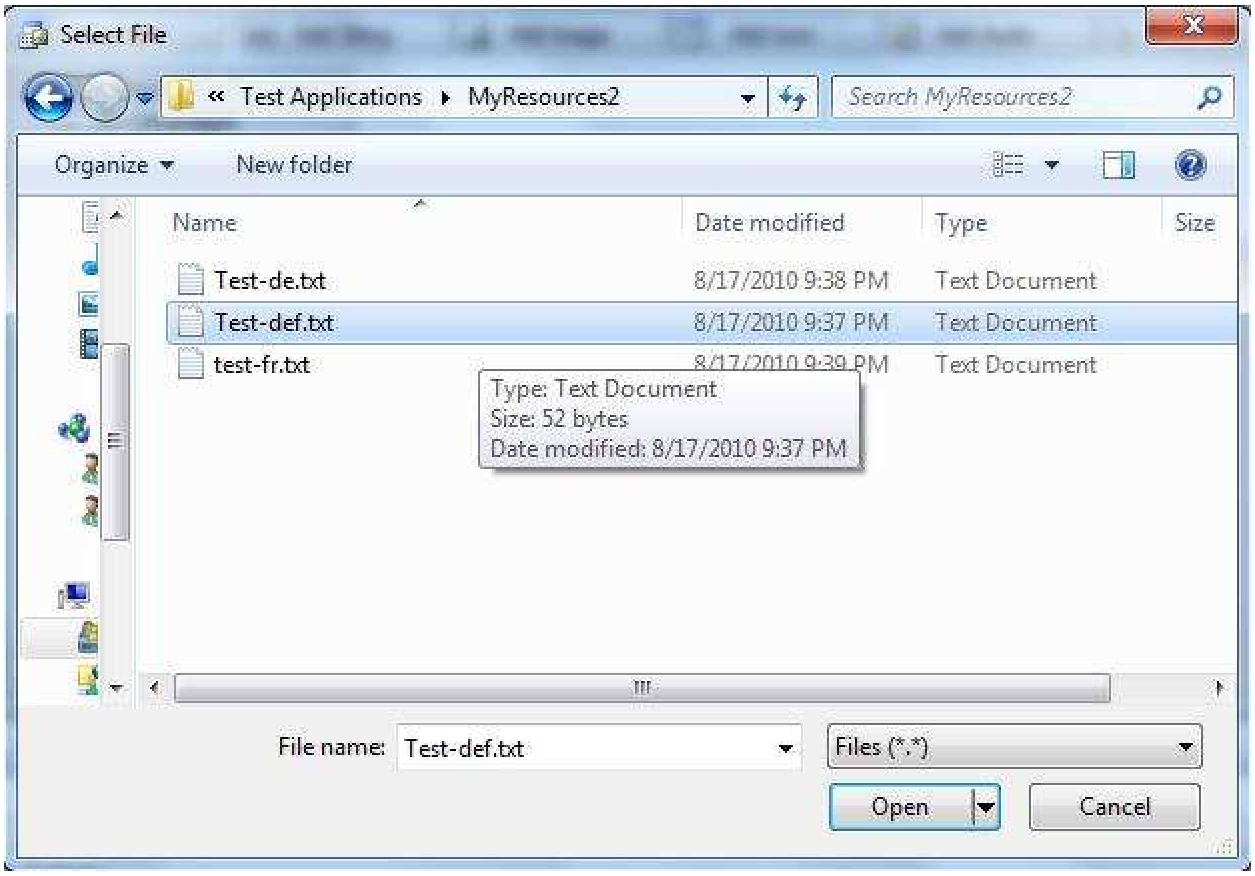


## 2.6 Set file resource values

Set default value by selecting the resource name. Right-click the right-pane, choose “Select file”



Select the file you want to use for the default value of this resource:

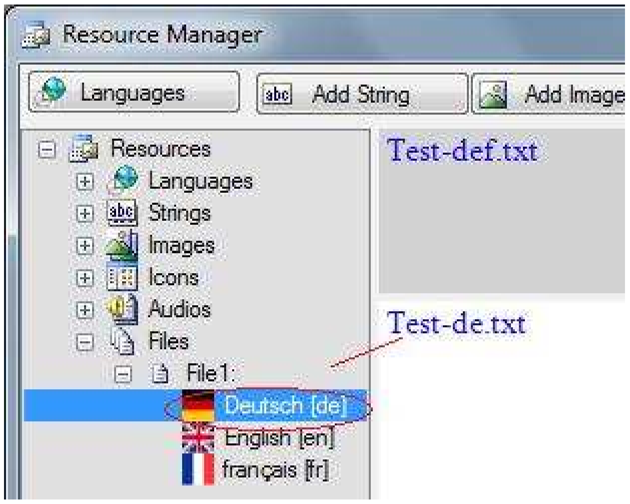


| Name | Date modified | Type |
|---|---|---|
| Test-de.txt | 8/17/2010 9:38 PM | Text Document |
| Test-def.txt | 8/17/2010 9:37 PM | Text Document |
| test-fr.txt | 8/17/2010 9:39 PM | Text Document |

Type: Text Document
Size: 52 bytes
Date modified: 8/17/2010 9:37 PM

File name: Test-def.txt
Files (*.*)

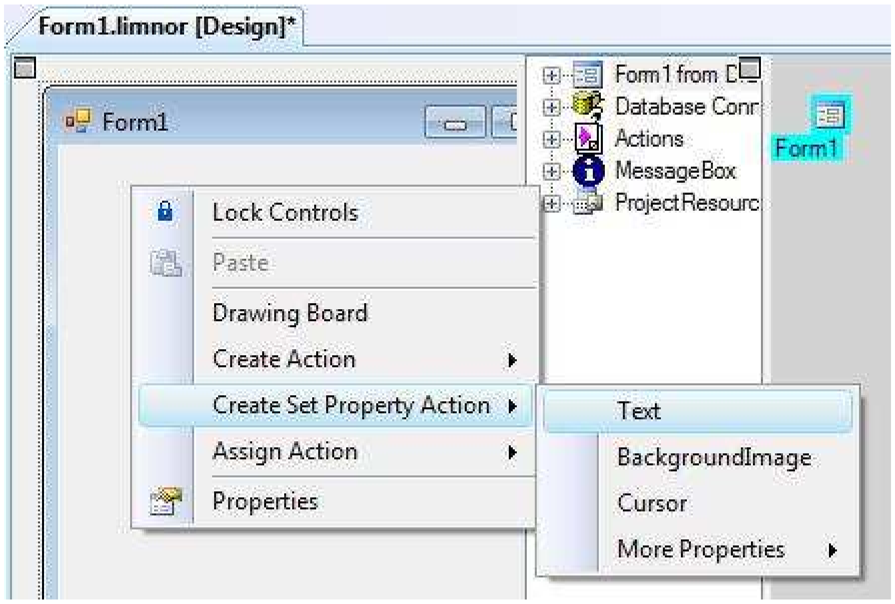
Select each supported language, select file for it:



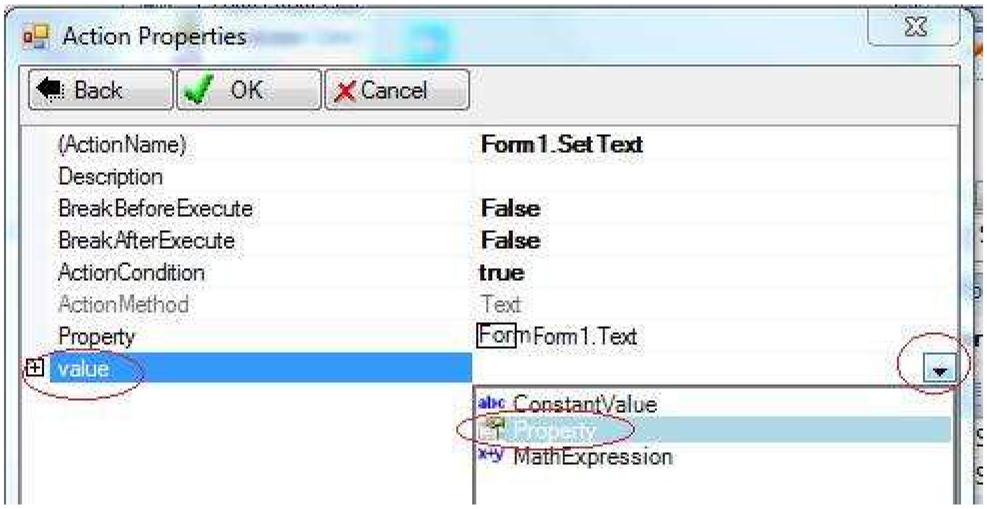
## 3 Use Resources in Actions

Every resource becomes a property which can be used in actions. Let’s create an action to set the title of a form.

Right-click on the form, choose “Create Set Property Action”, choose “Text” property:

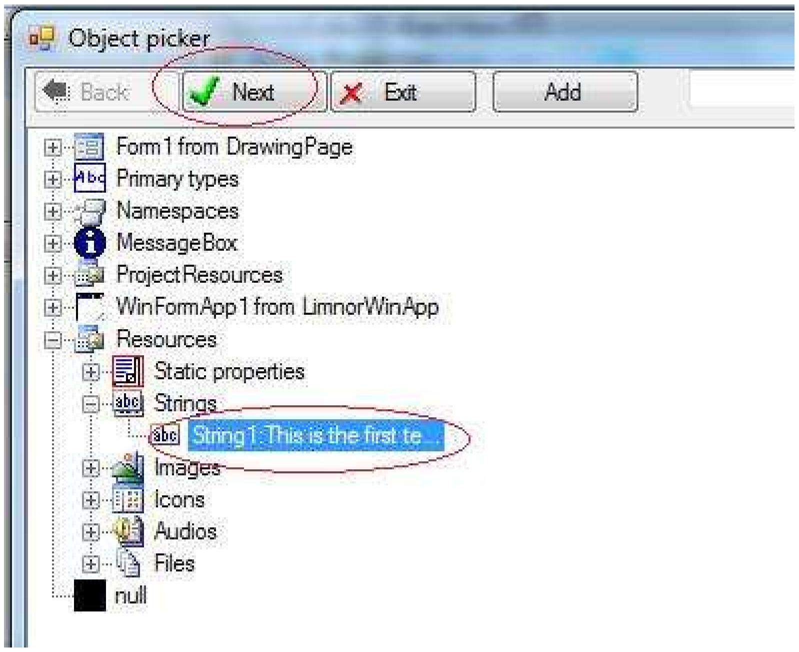


For the "value" of this new action, select "Property" so that we may choose a resource:

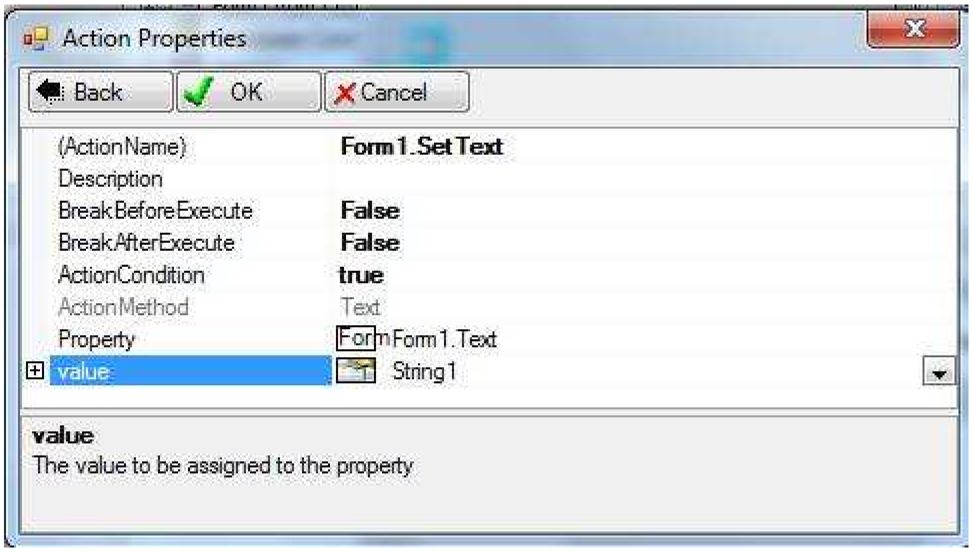


|  |  |
|---|---|
| (ActionName) | Form1.SetText |
| Description |  |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | Text |
| Property | Form Form1.Text |
| value |  |

Under "Resources", select the resource we want to use for this action. For this sample, we choose "String1":



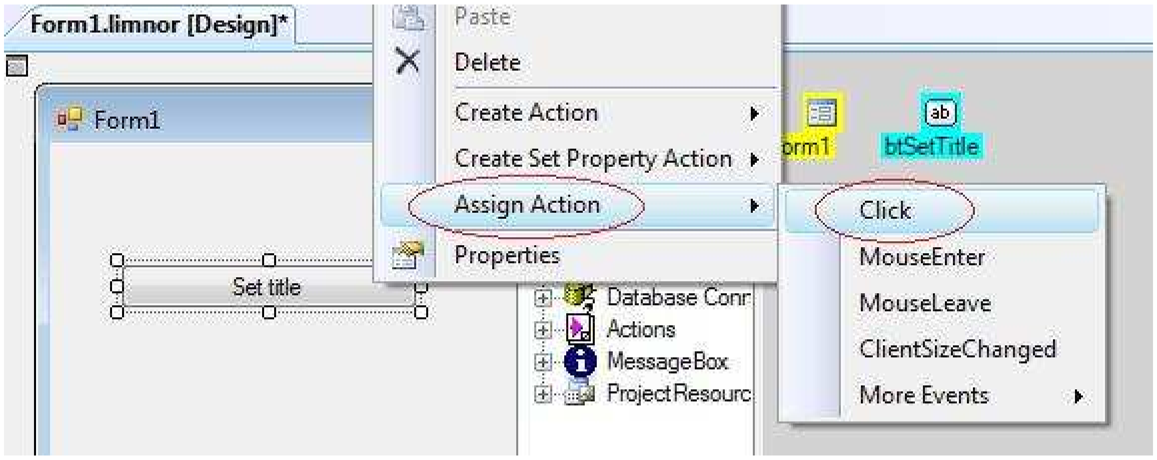
Click OK to finish creating this action:



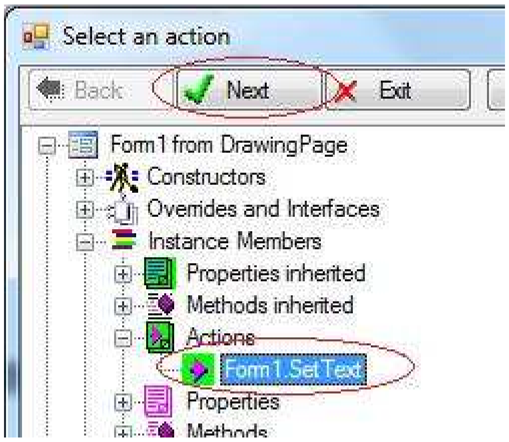
| | |
|---|---|
| (ActionName) | Form1.SetText |
| Description | |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | Text |
| Property | Form Form1.Text |
| value | String1 |

**value**
The value to be assigned to the property

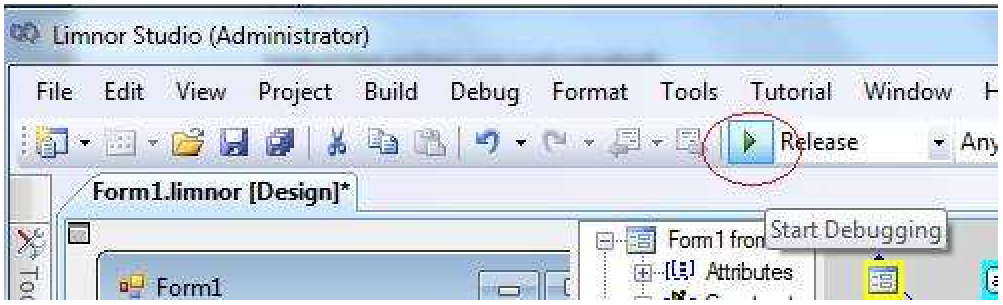
Assign this action to a button by right-clicking the button and choose “Assign action”. Choose “Click” event:



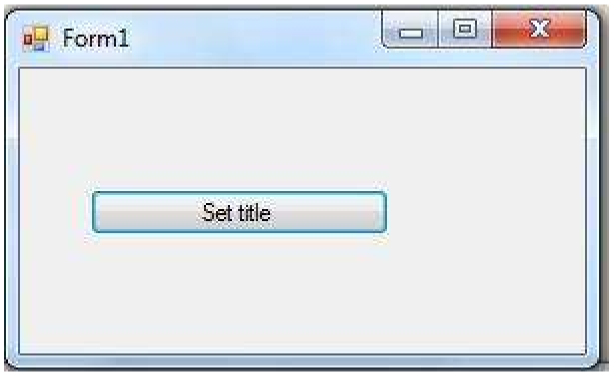
Select the action we just created:



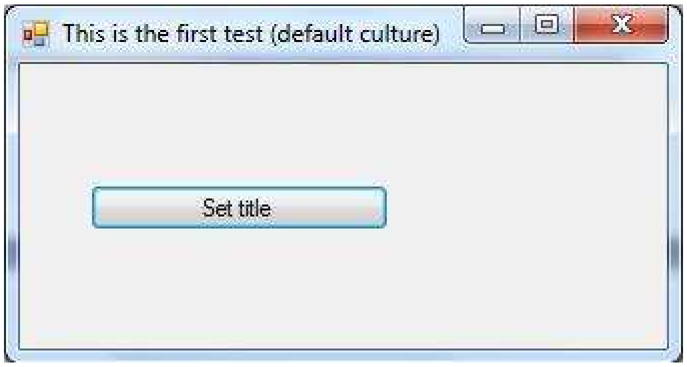
We may test this application now:



The form appears:



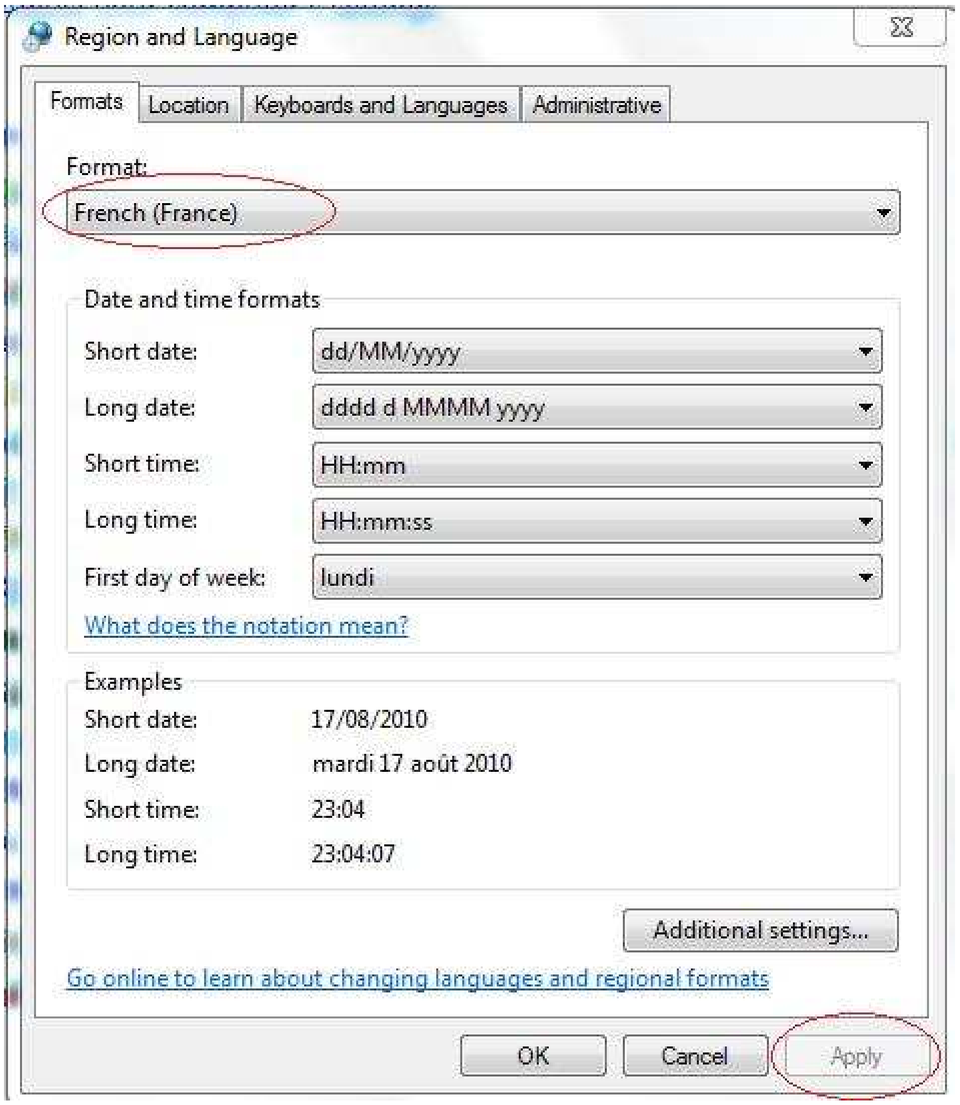
Click the button, the title changes:



We can see the default value for the resource is used.

Now close the application.

Change the computer regional format via Control Panel:


Region and Language

Format: French (France)

Date and time formats

Short date: dd/MM/yyyy

Long date: dddd d MMMM yyyy

Short time: HH:mm

Long time: HH:mm:ss

First day of week: lundi

What does the notation mean?

Examples

Short date: 17/08/2010

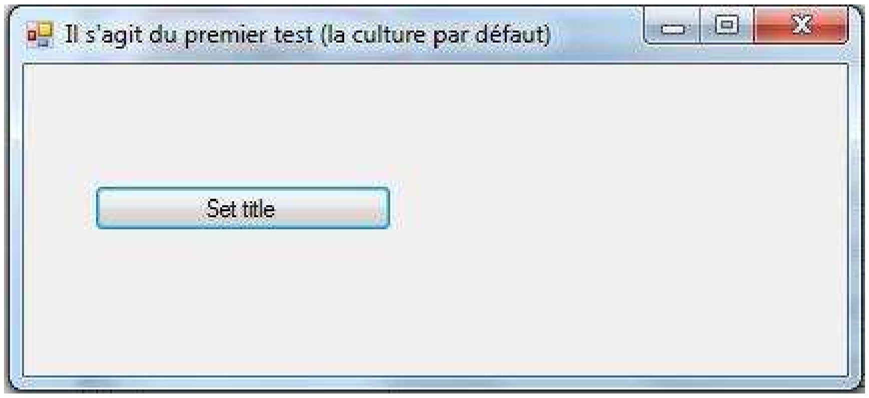
Long date: mardi 17 août 2010

Short time: 23:04

Long time: 23:04:07

Additional settings...
Go online to learn about changing languages and regional formats
OK
Cancel
Apply

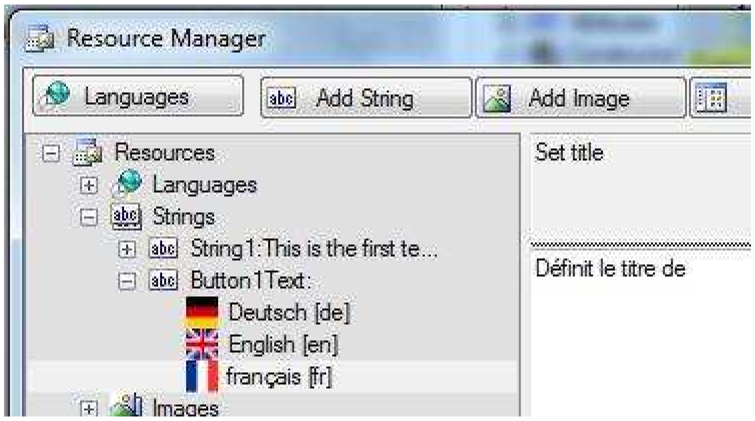
Run the application again. Click the button. We can see now the French resource is used by the action:

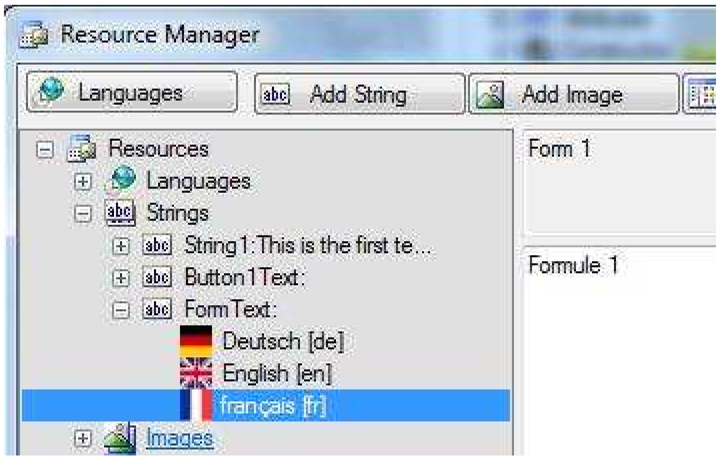

Il s'agit du premier test (la culture par défaut)

## 4 Use Resources in Properties

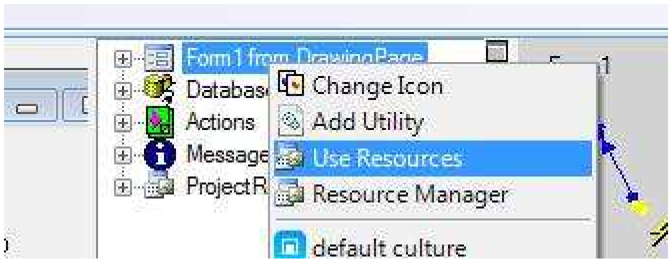
In the above example, when the computer is using French format, the button caption is still in English "Set title". We may map the properties to resources so that the properties may use the correct language according to the computer setting.

In this sample, we map the Text property of the button, the Image property of a PictureBox and the Text property of the Form to resources. We already have an image resource, Image1. We may create two more string resources for the Button Text and Form Text:

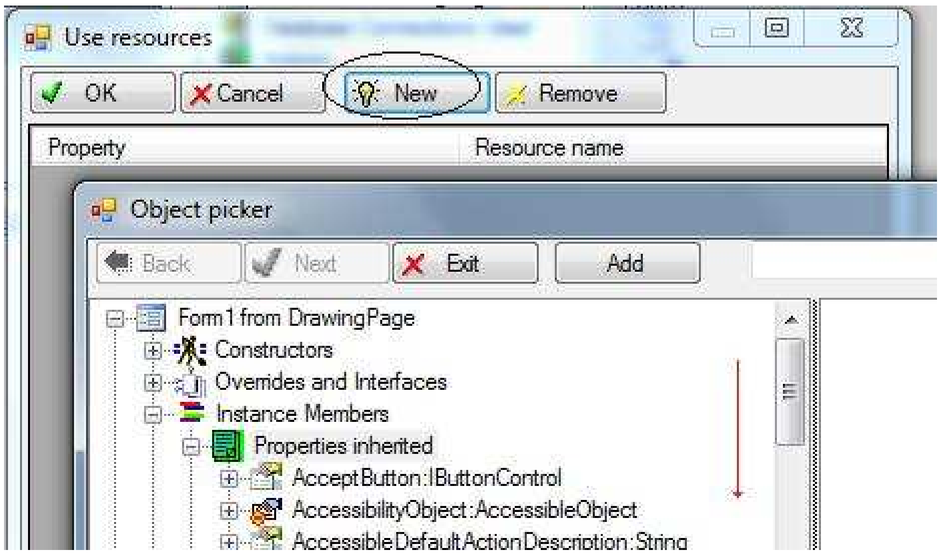




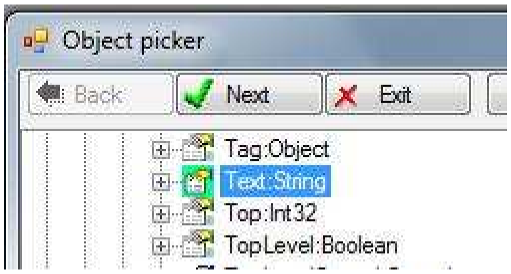
Right-click the root node of the Form, choose "Use Resources":



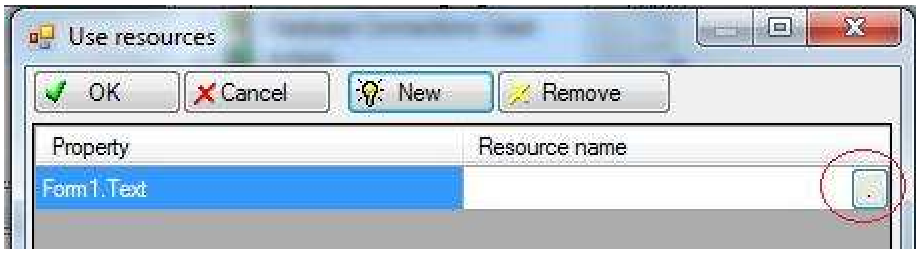
Click New button to create a new property-resource mapping. Scroll down to select the Text property of the Form:



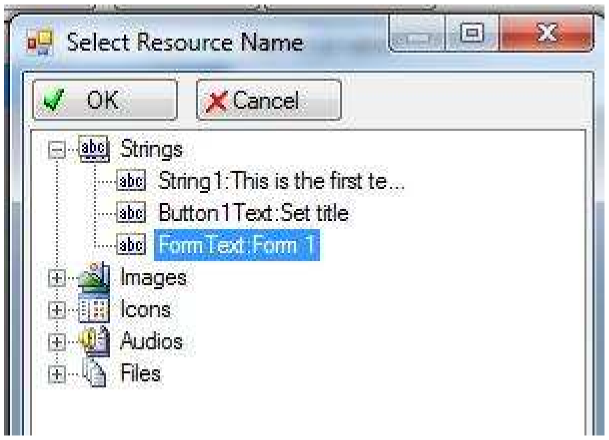
Select the Text property. Click Next:



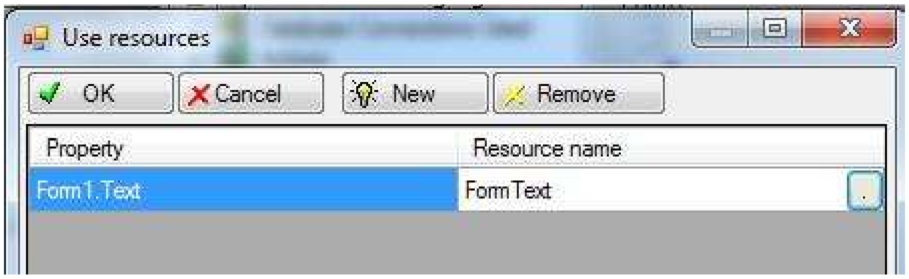
Click the button to select the resource for this property:



Select the resource FormText:

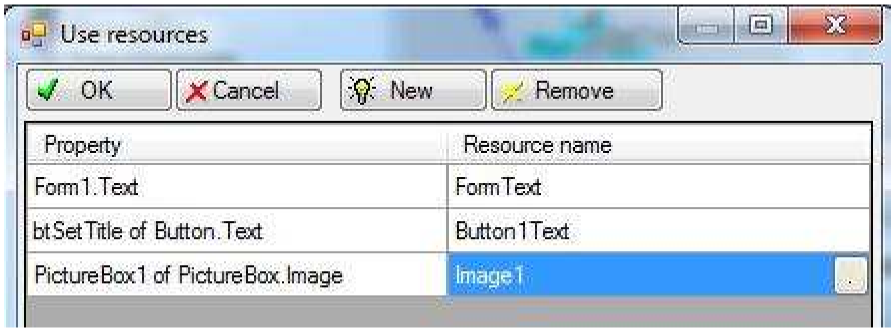


We made the property-resource mapping:



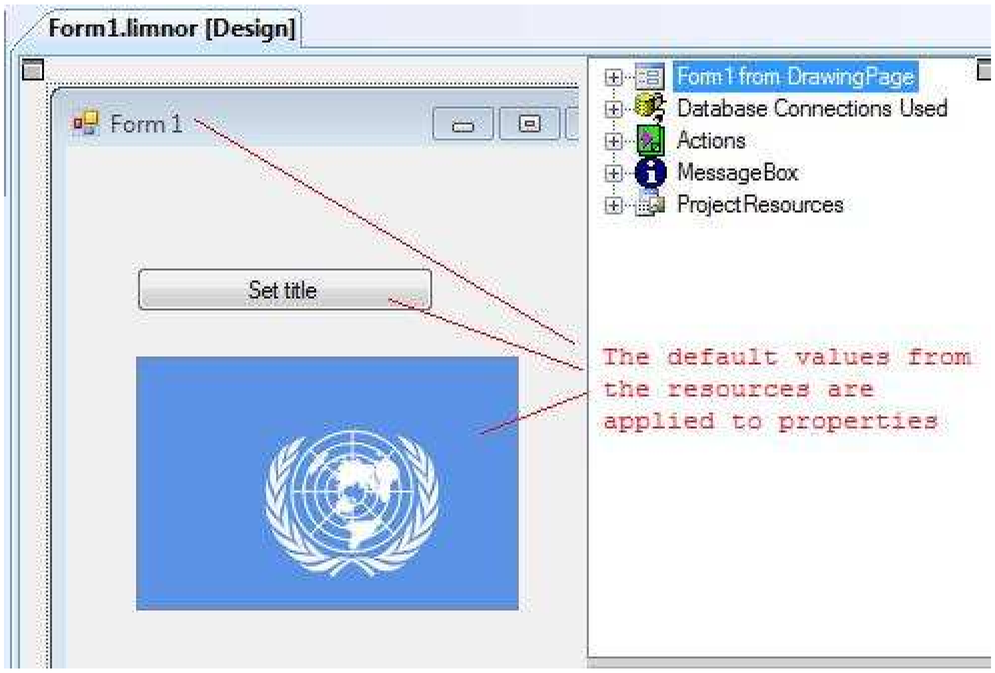
| Property | Resource name |
|---|---|
| Form1.Text | FormText |

We may map Text property of the button to resource Button1Text and Image property of the Picture Box to Image1:

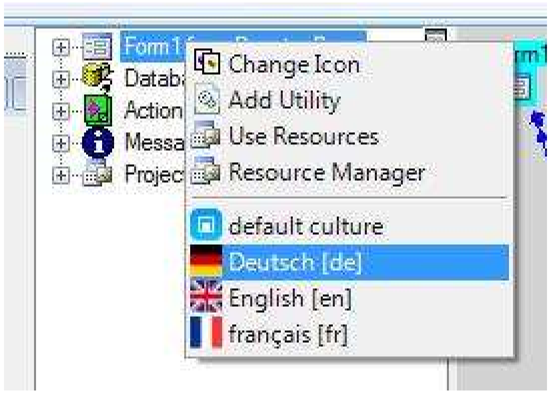


| Property | Resource name |
|---|---|
| Form1.Text | FormText |
| btSetTitle of Button.Text | Button1Text |
| PictureBox1 of PictureBox.Image | Image1 |

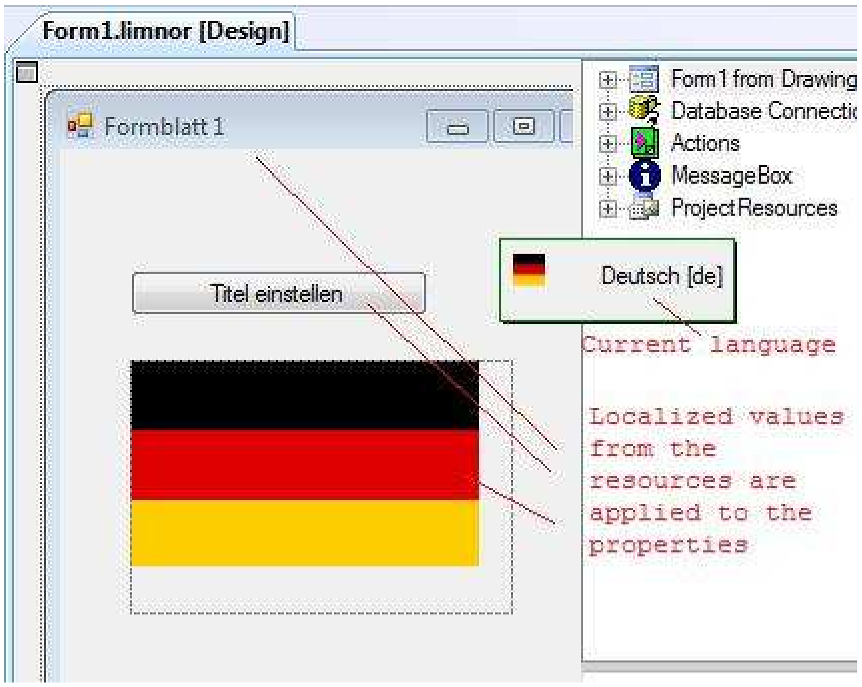
The default values from the resources are applied to the mapped properties:

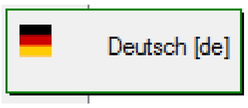


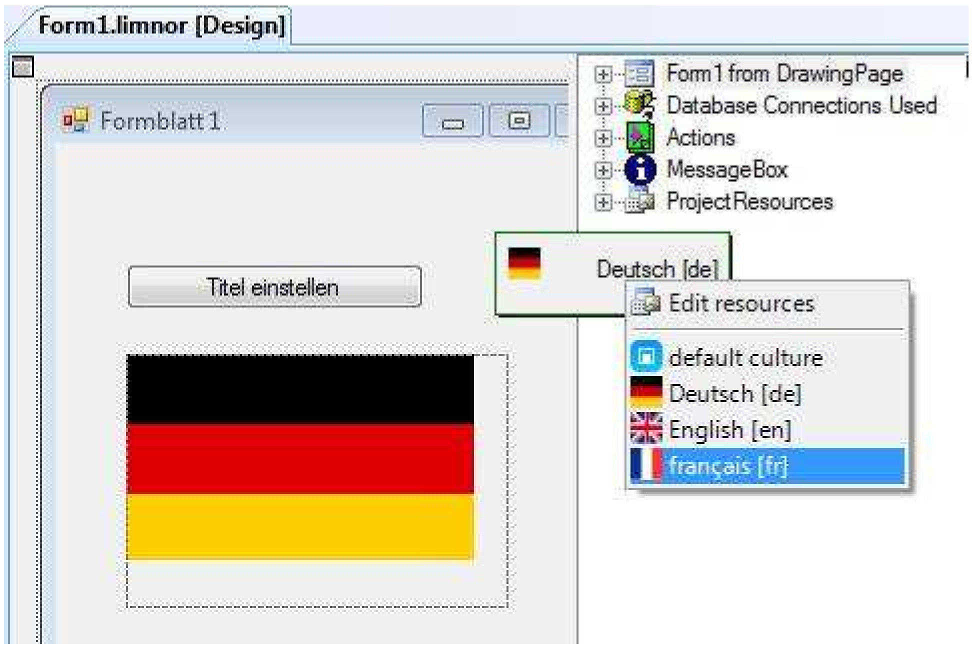
To see the effects of the localization, select a supported language:

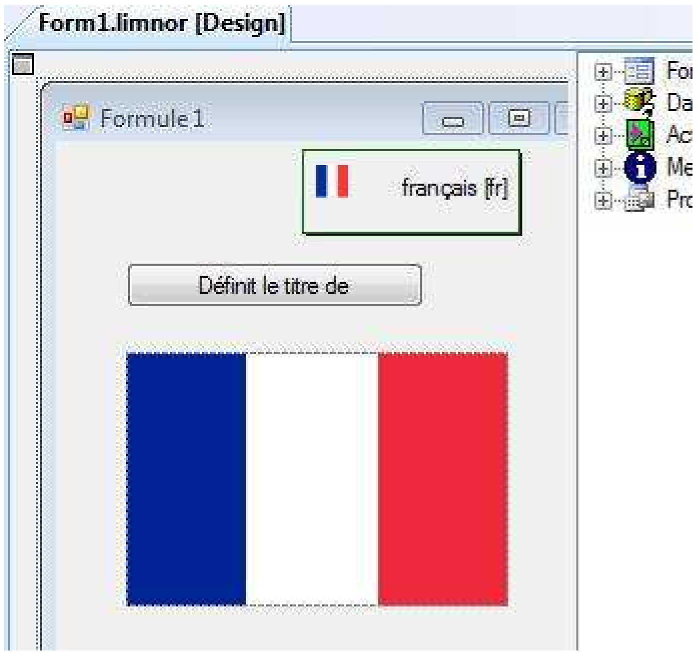


Values for the selected language from the resources are applied to the mapped properties:

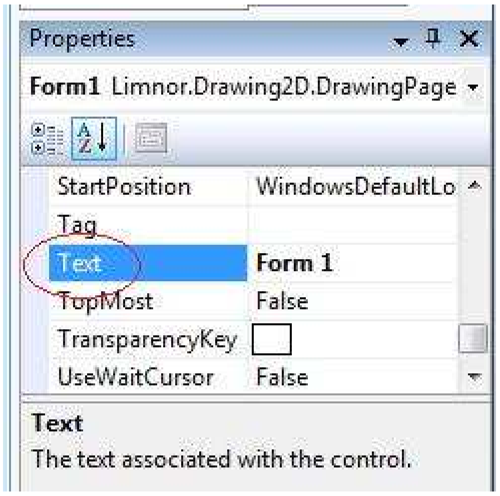


There is a language indictor  indicating which language is selected. You may drag and move it around the screen. You may also right-click it to change language:





Warning: once a property is mapped to a resource, do not modify the property in the PropertyGrid window:

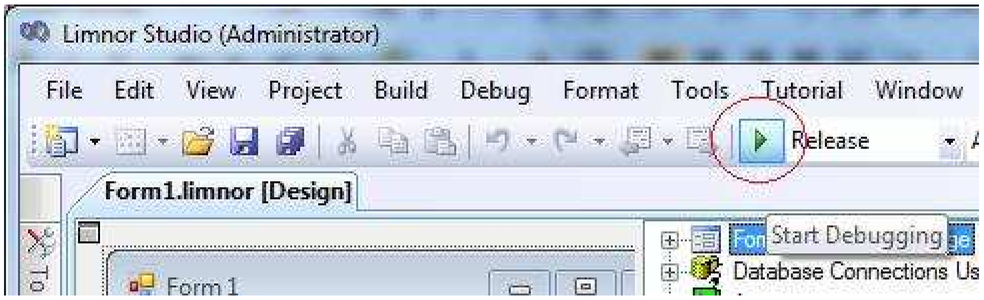


| | |
|---|---|
| StartPosition | WindowsDefaultLo |
| Tag | |
| Text | Form 1 |
| TopMost | False |
| TransparencyKey | |
| UseWaitCursor | False |

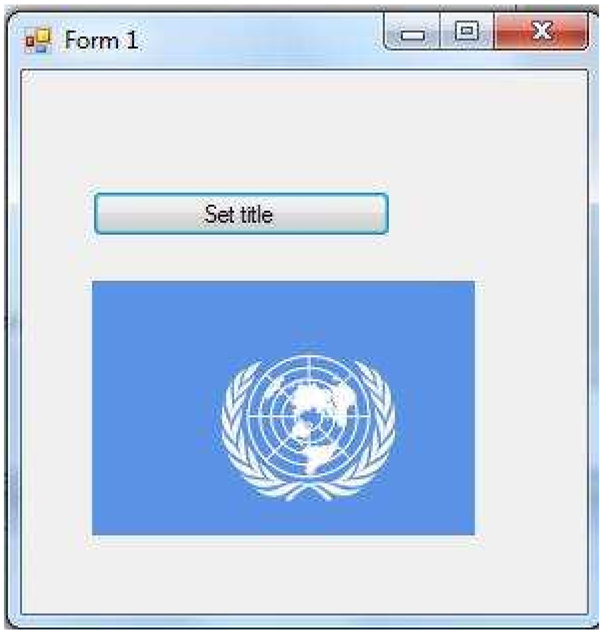
**Text**
The text associated with the control.

Always modify the properties through the Resource Manager.

Let’s test the application:

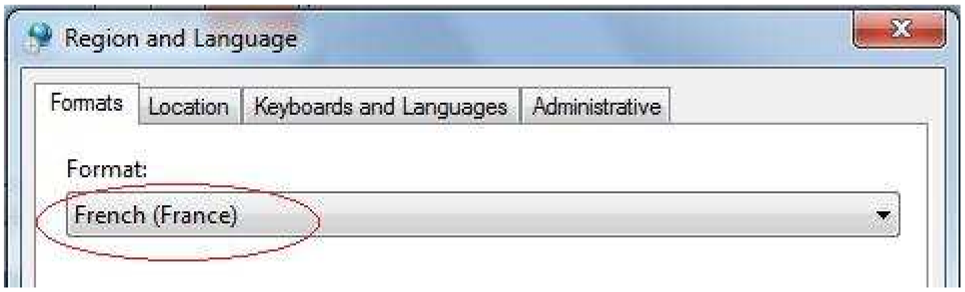

Limnor Studio (Administrator)
File Edit View Project Build Debug Format Tools Tutorial Window
Release
Form1.limnor [Design]
Start Debugging
Database Connections
Form 1

The form appears showing all the default values from the resources:



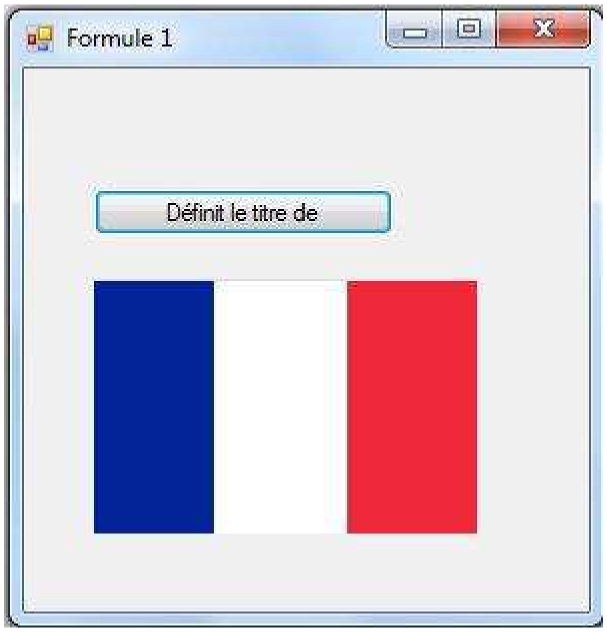
Close this application.

Switch the computer format to French via the Control Panel:

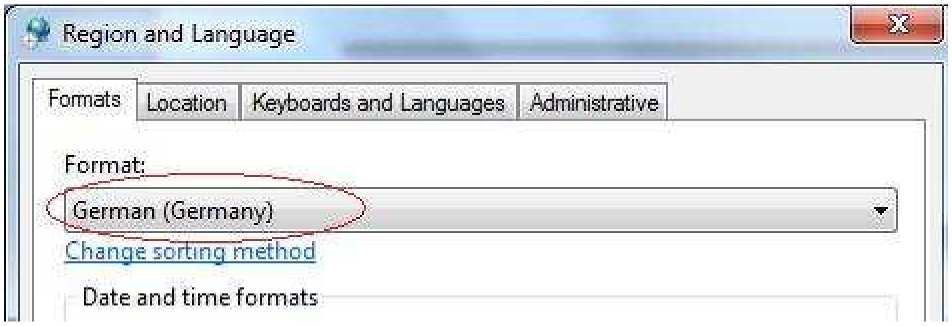


Format: French (France)

Run the application again. This time the French resources are used:

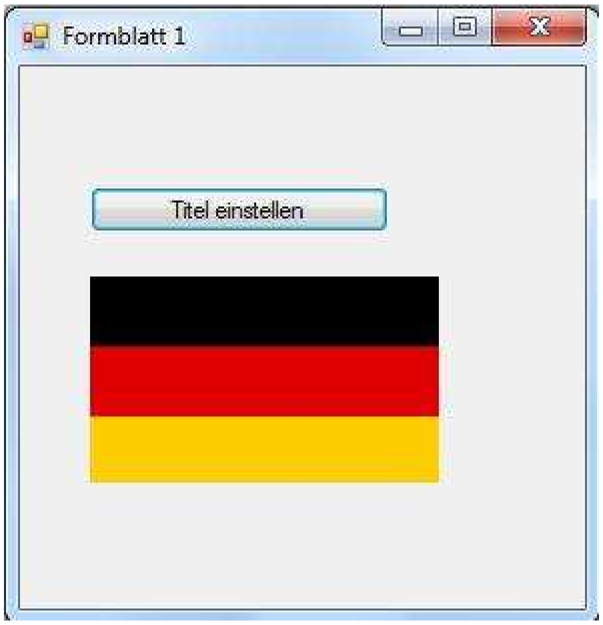


Close this application. Switch the computer format again via the Control Panel:



Format: German (Germany)

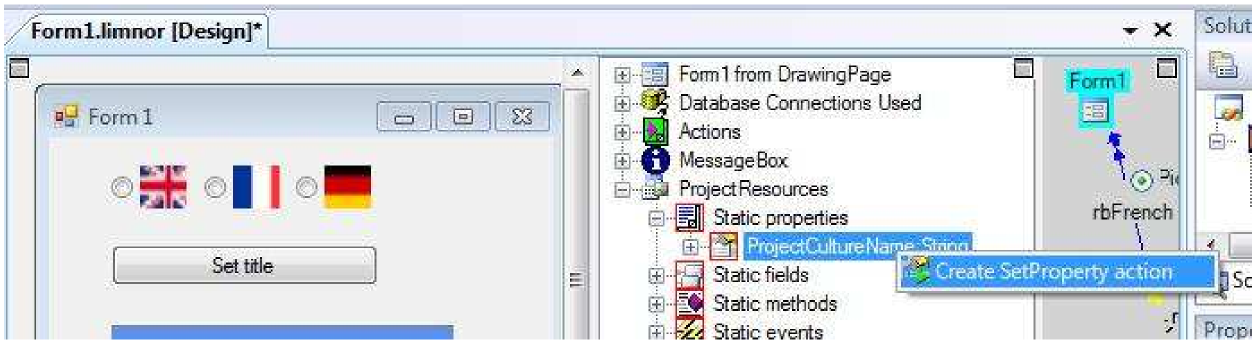
Restart the application. This time the resources for Germany are used:



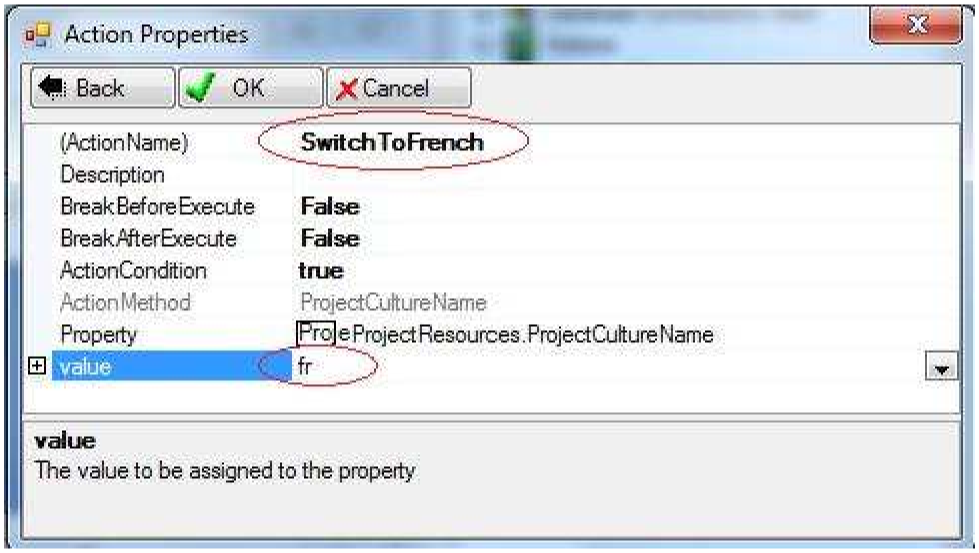
## 5 Switch Languages at Runtime

In the previous examples, the application determines which language to use at the starting of the application according to the computer settings. Here we describe how to switch the language at runtime.

We use a set of radio buttons for the user to switch the languages. The ProjectResources object has a property ProjectCultureName indicating the language to be used. Right-click it and choose “create SetProperty action” to create an action to switch the language:



Name this action as “SwitchToFrench” and set its “value” to “fr”:

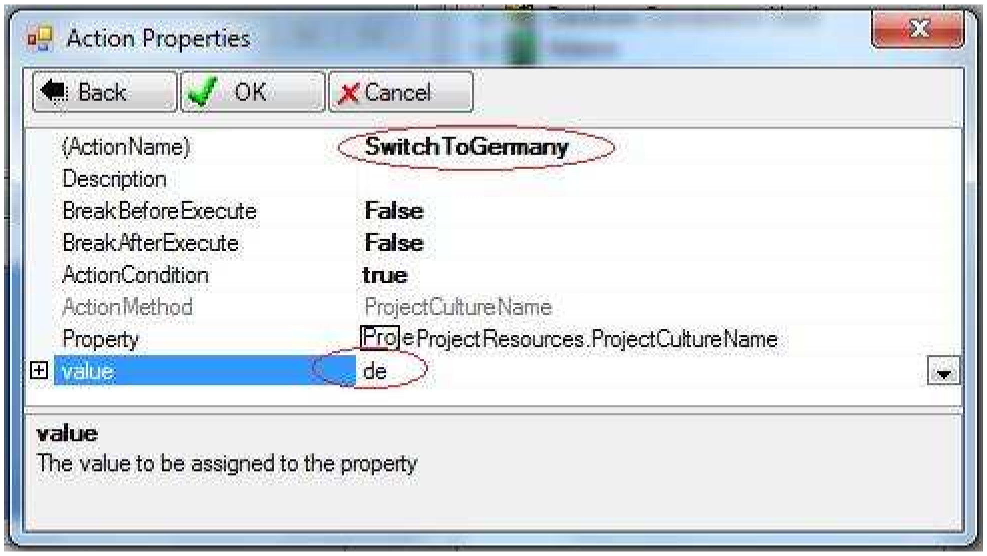


| | |
|---|---|
| (ActionName) | SwitchToFrench |
| Description | |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ProjectCultureName |
| Property | ProjeProjectResources.ProjectCultureName |
| value | fr |

**value**
The value to be assigned to the property

Click OK.

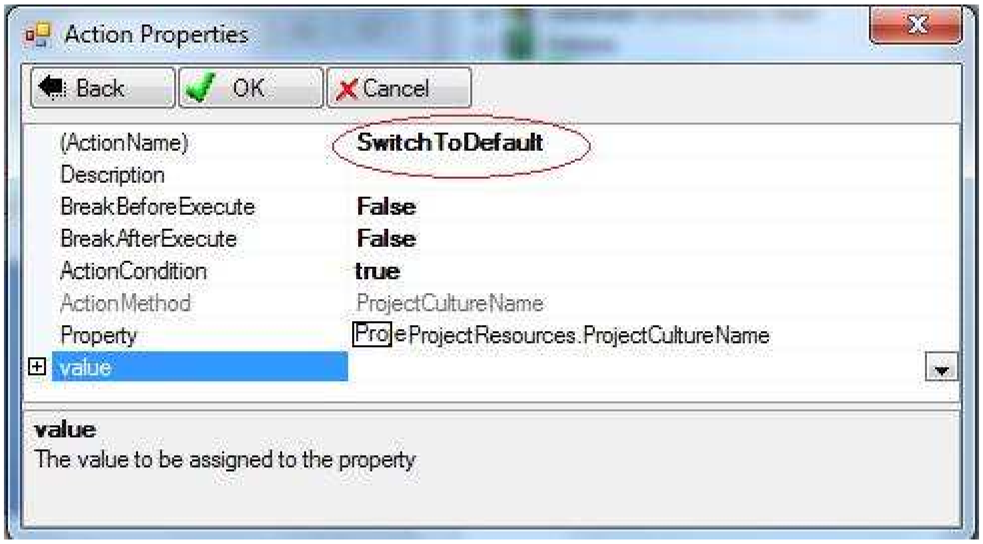
Right-click ProjectCultureName again to create another language switching action:



| | |
|---|---|
| (ActionName) | SwitchToGermany |
| Description | |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ProjectCultureName |
| Property | ProjeProjectResources.ProjectCultureName |
| value | de |

**value**
The value to be assigned to the property

And another action to switch to default language:

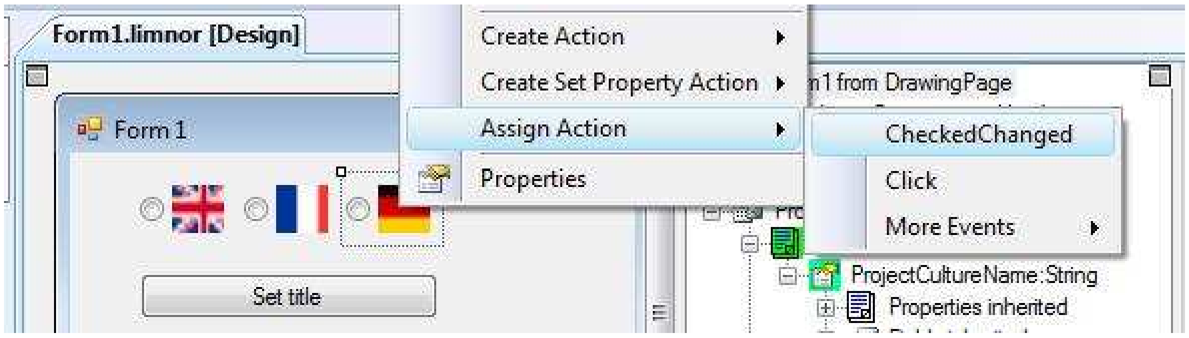


| | |
|---|---|
| (ActionName) | SwitchToDefault |
| Description | |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ProjectCultureName |
| Property | ProjectResources.ProjectCultureName |
| value | |

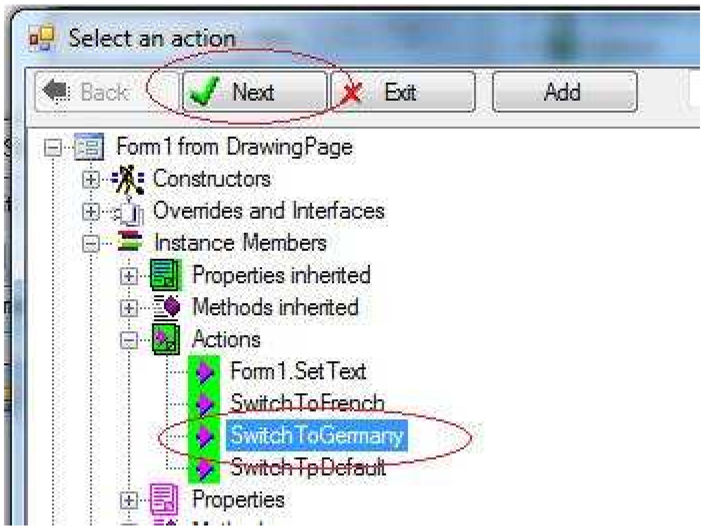
**value**
The value to be assigned to the property

Now we need to assign the actions to the radio buttons.

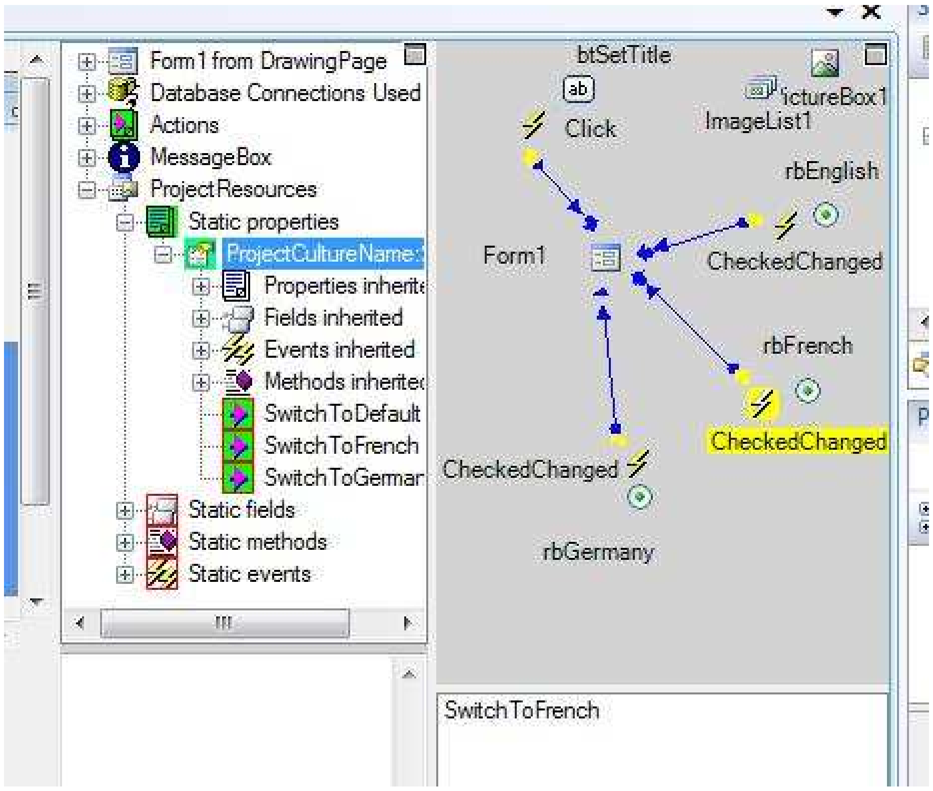
Right-click a radio button, choose "Assign action". Choose event "CheckedChanged":



Select the action for the radio button and click Next:

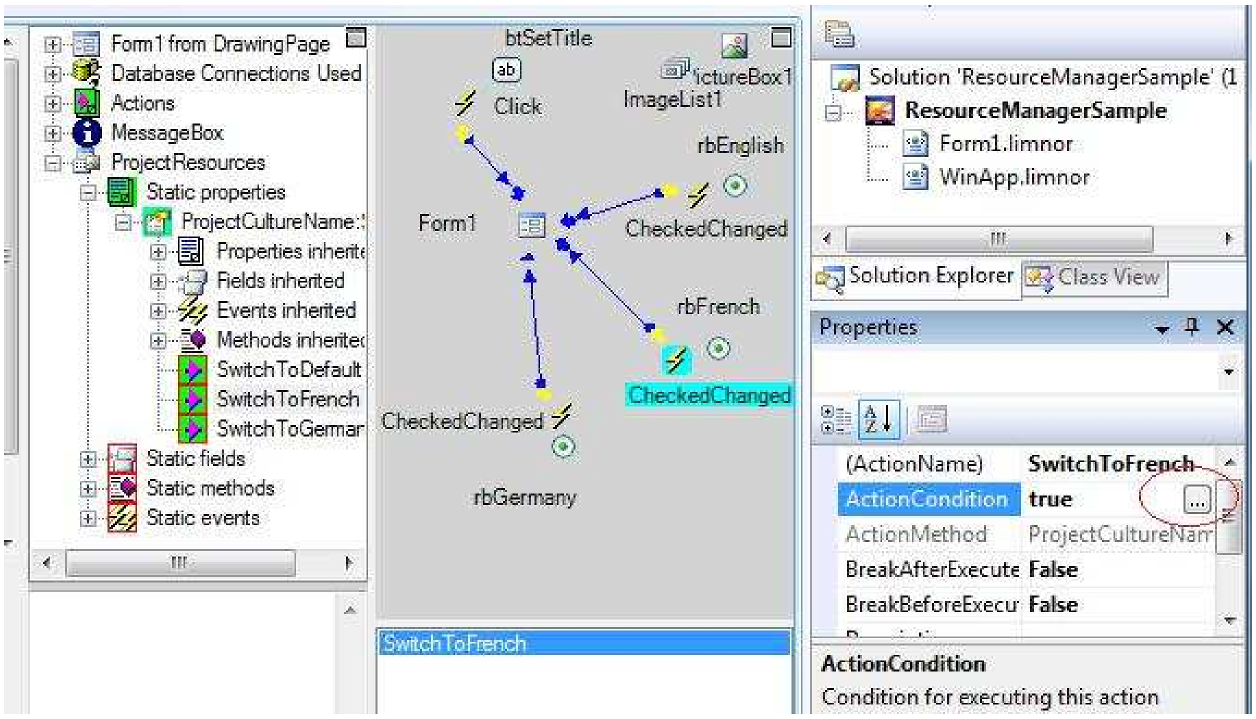


In this way, we assign each language-switching action to one radio button:



Before we test it, we realize there is a bug in the programming. The event CheckedChanged not only happens when a radio button becomes checked. It also happens when a radio button becomes unchecked. When a radio button becomes unchecked we do not want the action to execute. We may set an ActionCondition to each action to make it execute only when the radio button for the action is checked.

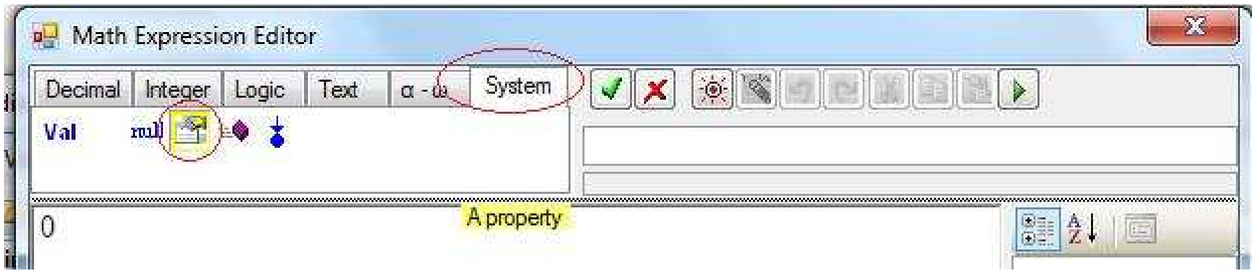
To modify an action, select the action, and change its properties through the property window. In our case, click [...] to change the ActionCondition:



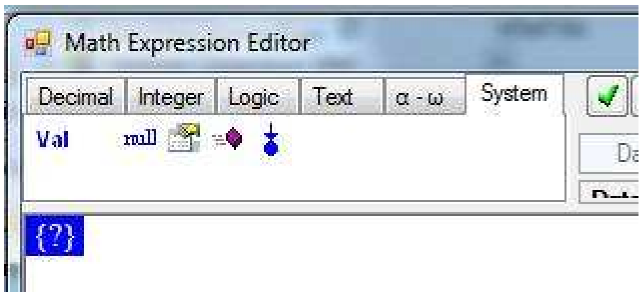
| | |
|---|---|
| (ActionName) | SwitchToFrench |
| ActionCondition | true |
| ActionMethod | ProjectCultureNam |
| BreakAfterExecute | False |
| BreakBeforeExecu | False |

ActionCondition
Condition for executing this action

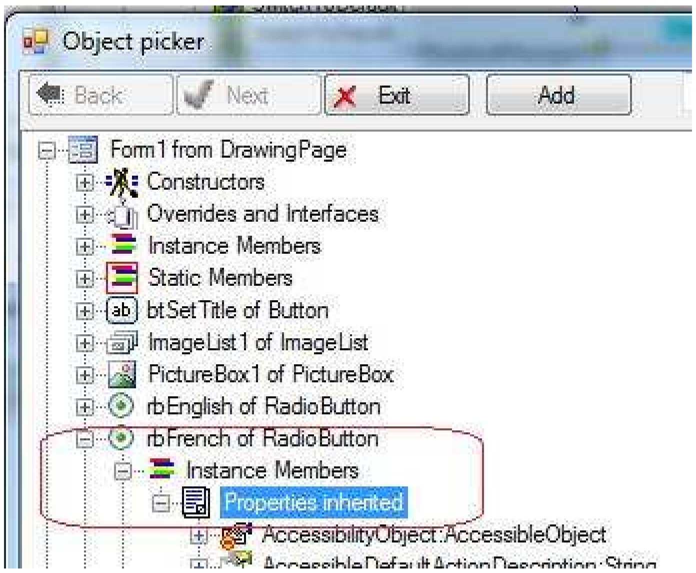
Click the Property icon so that we may use the Checked property of the radio button:



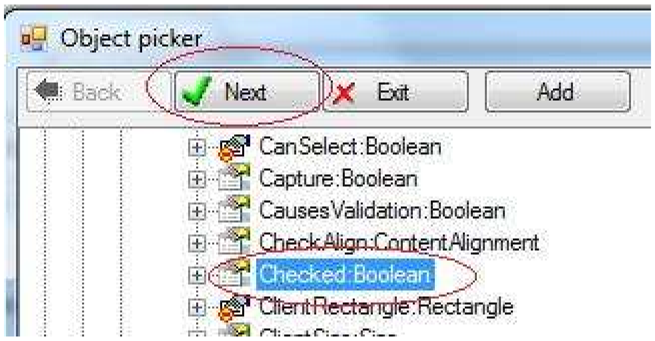
Double-click {?} to select a property:



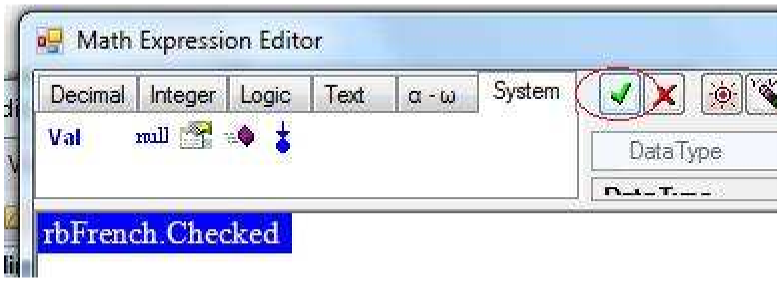
Expand "Properties inherited" node of the radio button to find its Checked property:



Select Checked property and click Next:



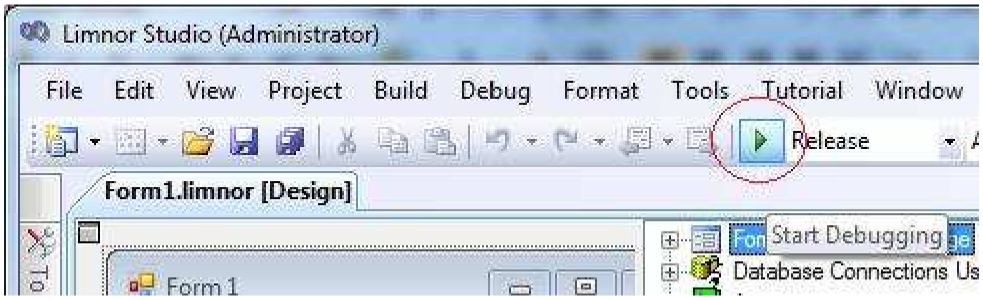
This is the condition we need for the action:



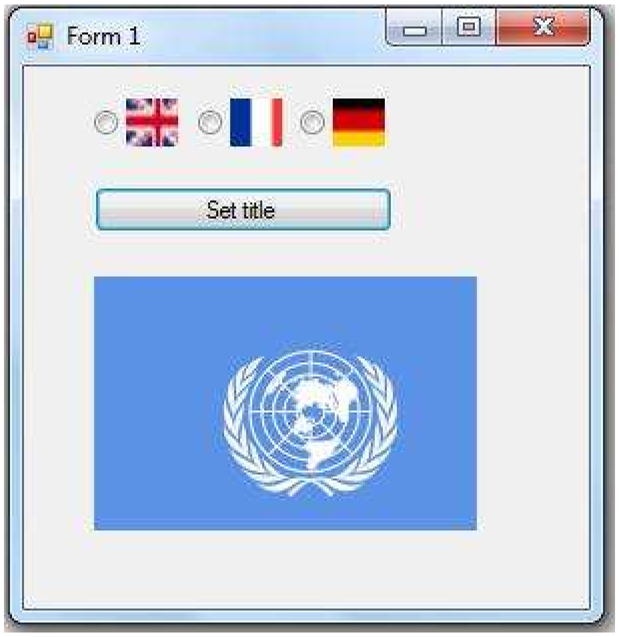
```
rbFrench.Checked
```

Do the same for the other two actions.

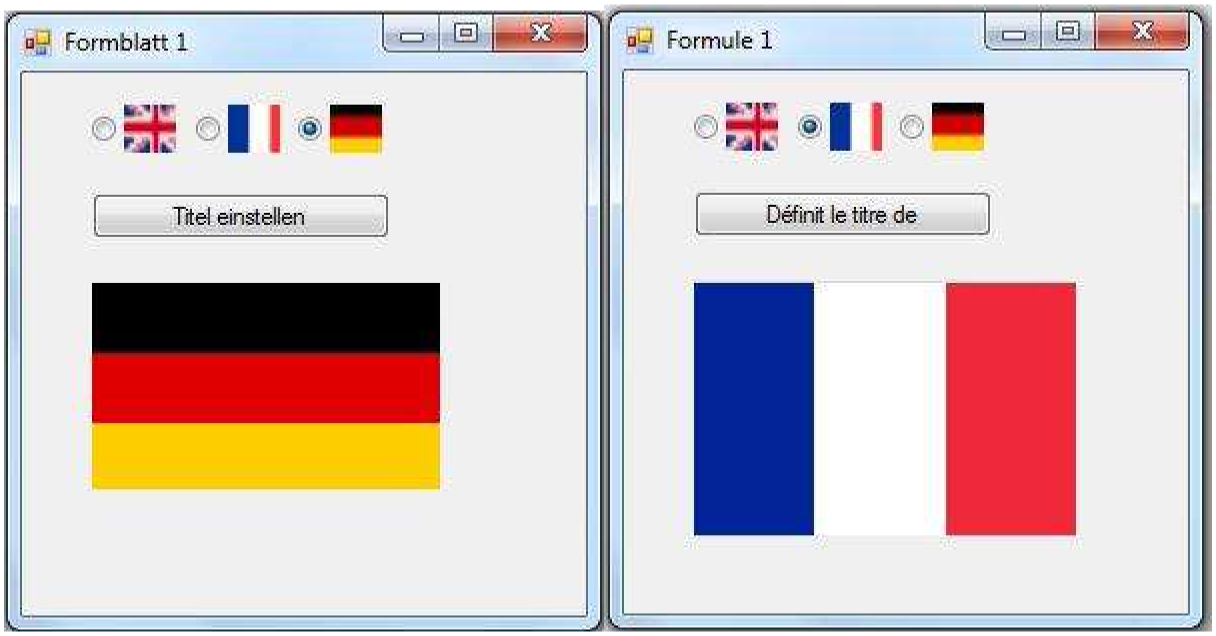
Now we may test this application:



The form appears:

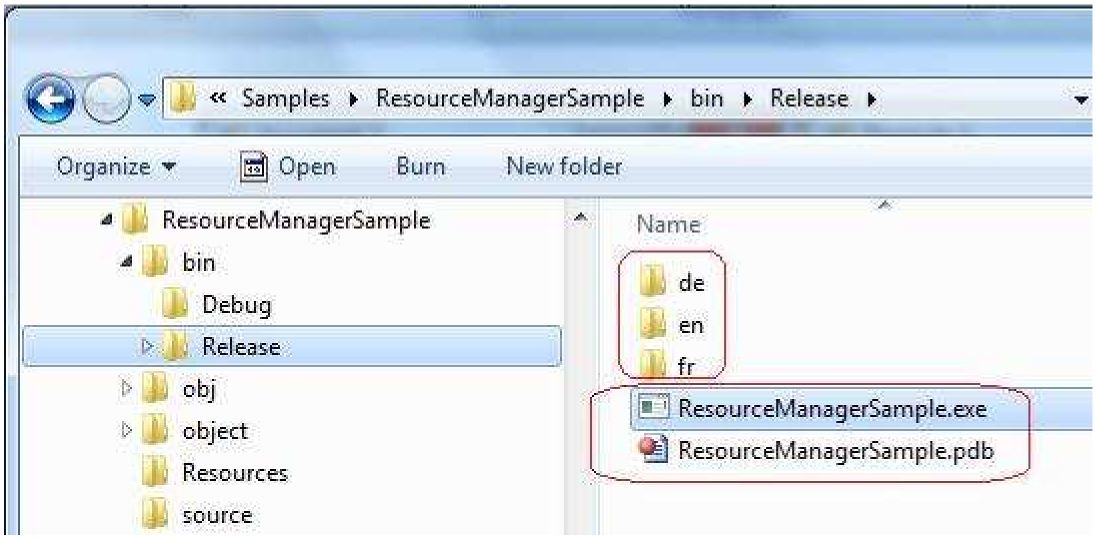


Click a radio button, the language for the application is switched to the selected language:

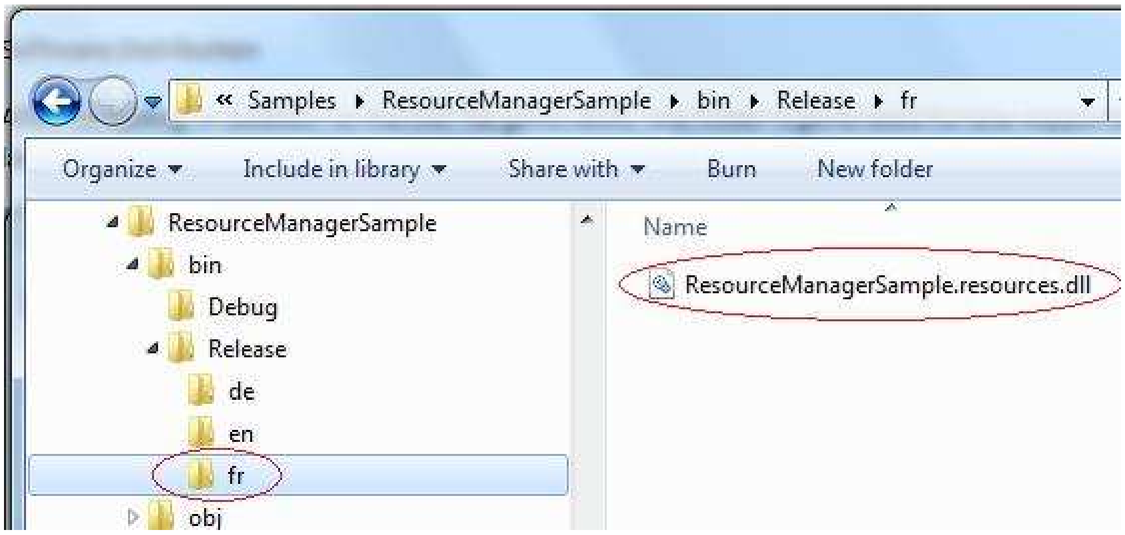


## 6 Software Distribution

After compiling, in addition to the EXE file generated, one folder is generated for one supported language:


Samples ▸ ResourceManagerSample ▸ bin ▸ Release ▸
Organize
Open
Burn
New folder
ResourceManagerSample
bin
Debug
Release
obj
object
Resources
source
Name
de
en
fr
ResourceManagerSample.exe
ResourceManagerSample.pdb

Within each language folder is a DLL file. The resources for that language are embedded inside that DLL:



When you distribute your software to your customers, all the language folders also need to be distributed.

## 7 Use Culture Information

We saw examples of various ways of using resources and mapping resources with properties. That may not be enough in handling culture changes. Some objects may need to be notified of the culture change. The ProjectResources object has following members related to culture changes:

- Property **ProjectCultureName** – This is the language culture name for the current culture, such as en-US, en-CA, fr-FR. See http://msdn.microsoft.com/en-us/library/ee825488(v=cs.20).aspx
- Property **ProjectCulture** – This is the current culture information corresponding to ProjectCultureName. See http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo(v=vs.90).aspx
- Event **ProjectCultureChanged** – It occurs on changing property ProjectCultureName.

Typically, at the event ProjectCultureChanged, you provide a culture-aware object with culture information from ProjectCulture.

We use an example to show such usages. Mr. Thomas Duwe created a culture aware datetime picker control. See http://www.codeproject.com/Articles/45684/Culture-Aware-Month-Calendar-and-Datepicker

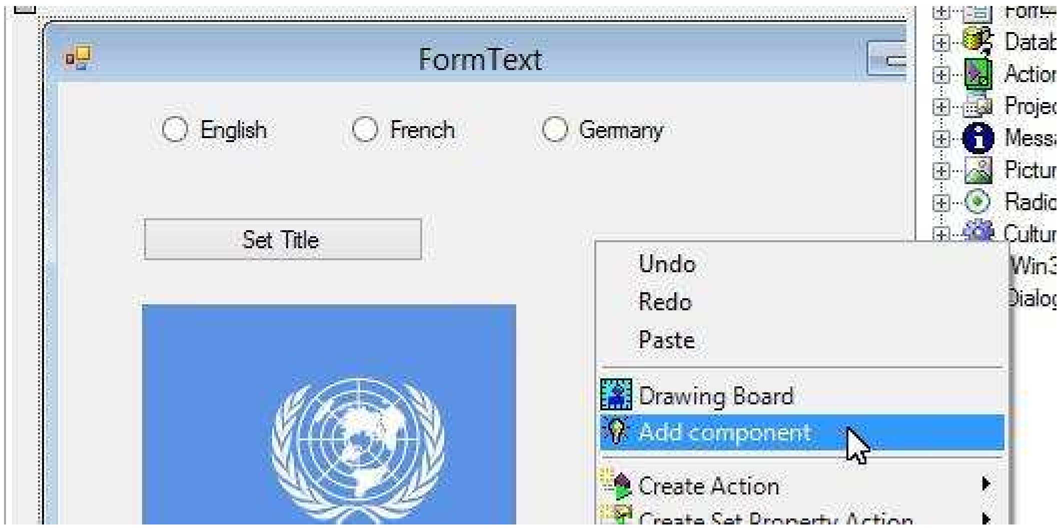
This datetime picker made by Mr. Duwe has a property named Culture. Changing this property will change the control UI. To automatically change the Culture property of the datatime picker, we may handle event ProjectCultureChanged and set the Culture to property ProjectCulture. Below we show how it is done.

## 7.1 Use library

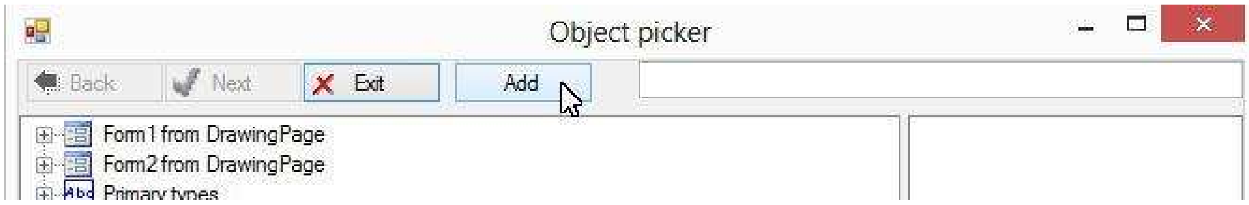
The datetime picker made by Mr. Duwe can be downloaded from http://www.codeproject.com/Articles/45684/Culture-Aware-Month-Calendar-and-Datepicker. It provides C# source code for generating library file MonthCalendarControl.dll. For your convenience we compiled it. You may download a zip file containing the compiled DLL file from http://www.limnor.com/support/datetimepicker.zip. The zip file contains MonthCalendarControl.dll. Unzip it to your computer for your Limnor Studio projects to use.

For more information about using libraries, see http://www.limnor.com/support/HowToLoadTypes.pdf. But in this sample, we are not using the process described in that PDF file. We use a simple way.

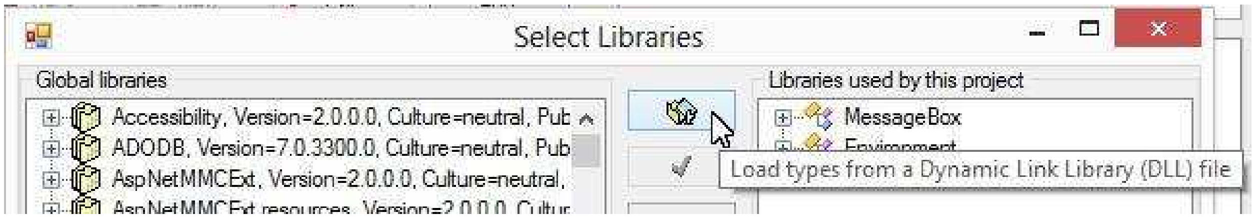
Right-click the form; choose "Add component":



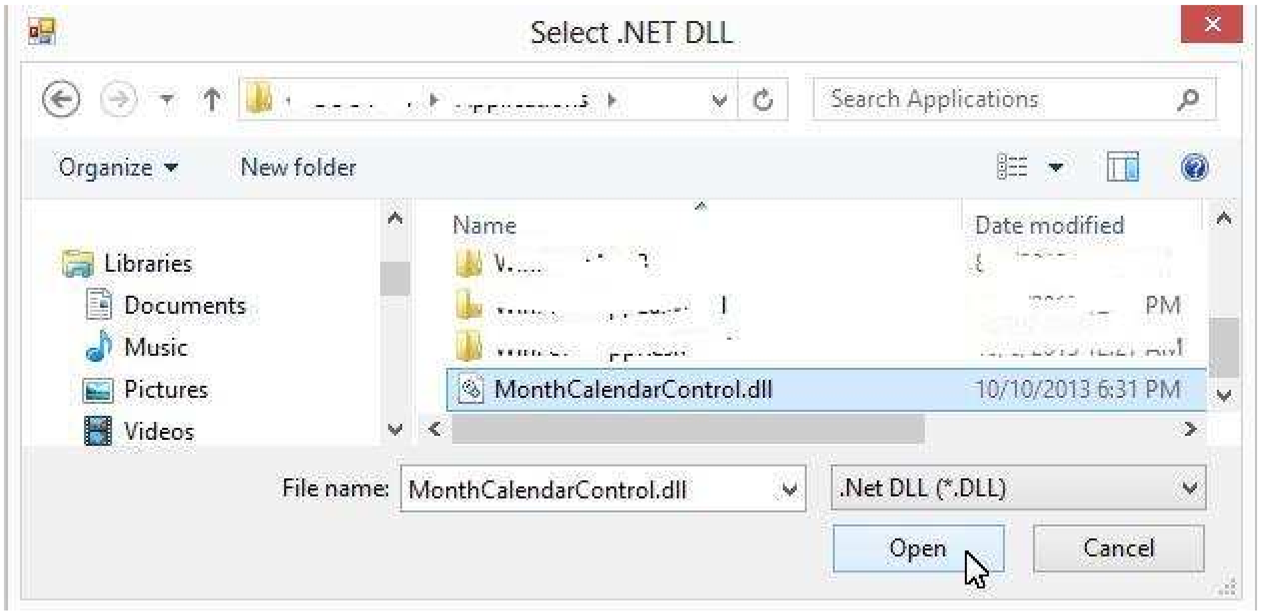
Click Add button because the DatePickr control is not in the tree view:



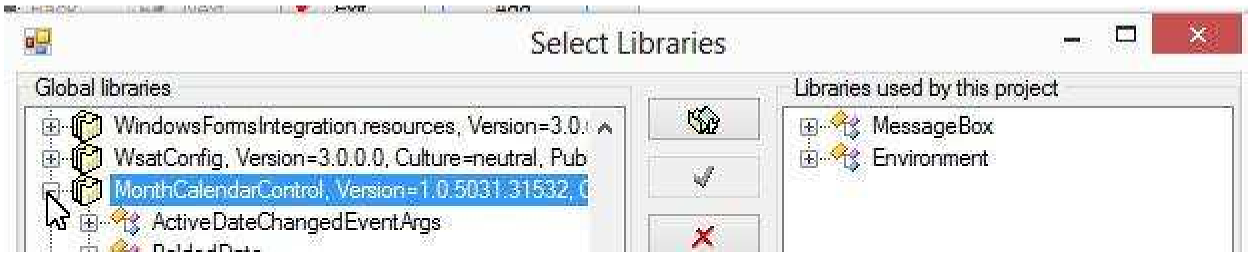
Click the folder button to select the DLL file we want to use:


Select Libraries
Global libraries
Accessibility, Version=2.0.0.0, Culture=neutral, Pub
ADODB, Version=7.0.3300.0, Culture=neutral, Pub
AspNetMMCExt, Version=2.0.0.0, Culture=neutral,
Libraries used by this project
MessageBox
Load types from a Dynamic Link Library (DLL) file

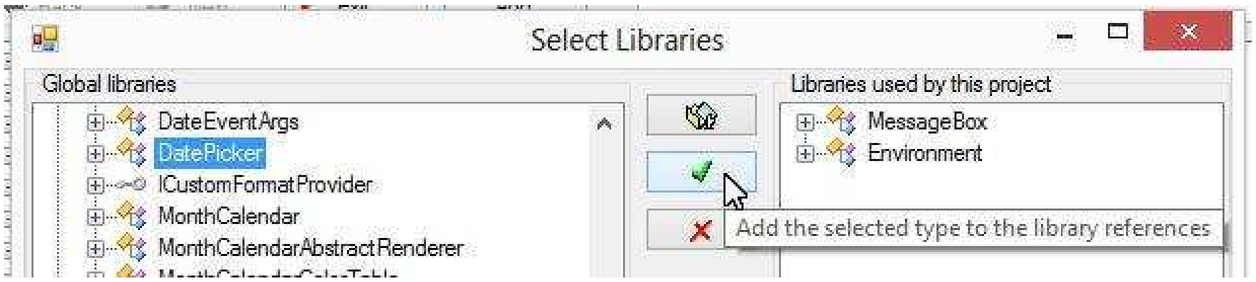
Select MonthCalendarControl.DLL:



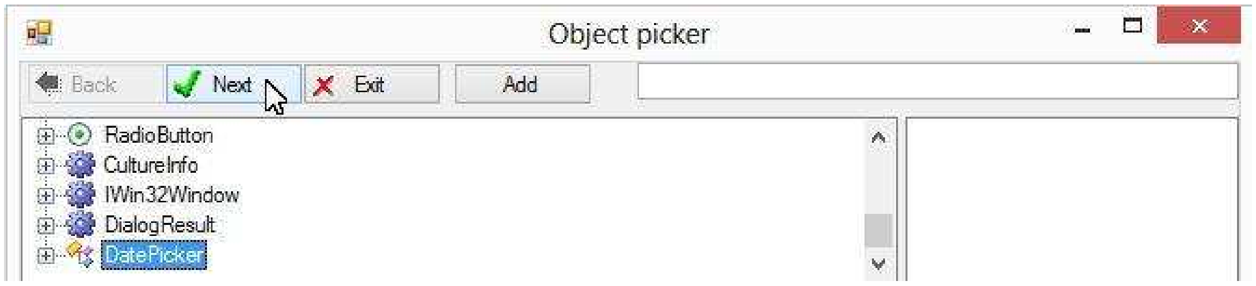
The DLL is loaded at the end of the tree view. Expand it to see the classes within it:


Select Libraries
Global libraries
WindowsFormsIntegration.resources, Version=3.0.
WsatConfig, Version=3.0.0.0, Culture=neutral, Pub
MonthCalendarControl, Version=1.0.5031.31532, C
ActiveDateChangedEventArgs
Libraries used by this project
MessageBox
Environment

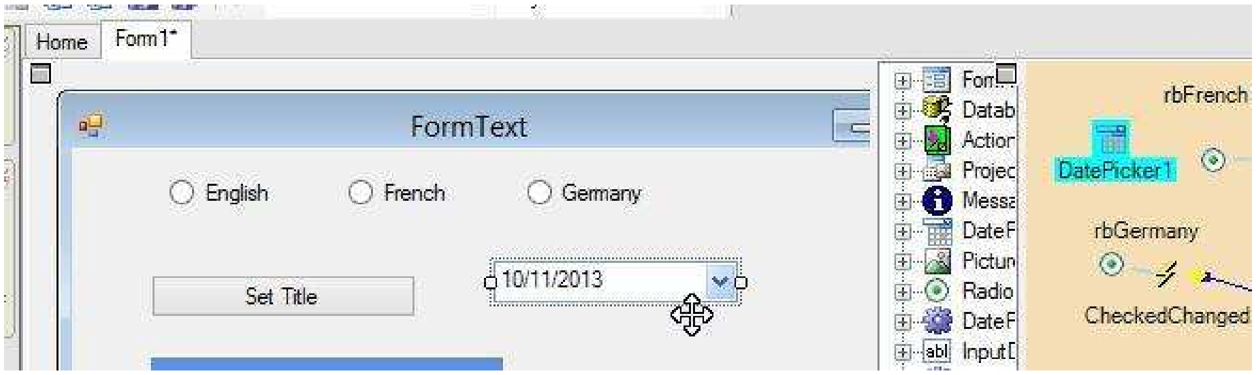
Select DatePicker and click OK button (green check button):



The class DatePicker appears in the first dialogue box. Select DatePicker and click Next:

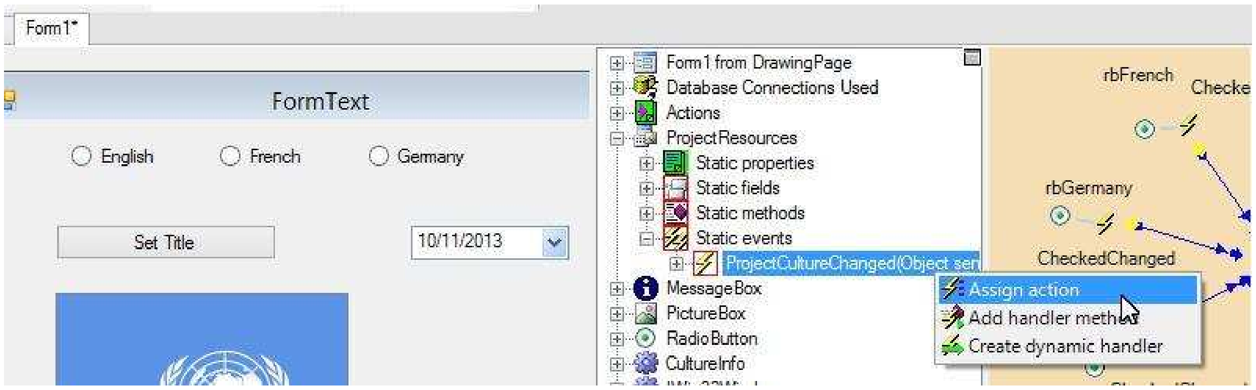


A DatePicker control appears on the form:

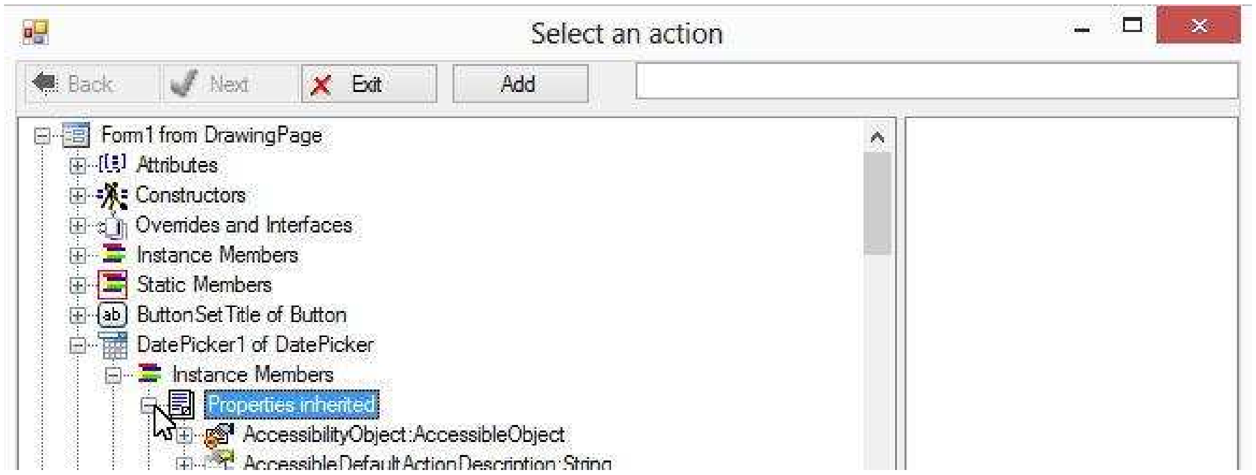


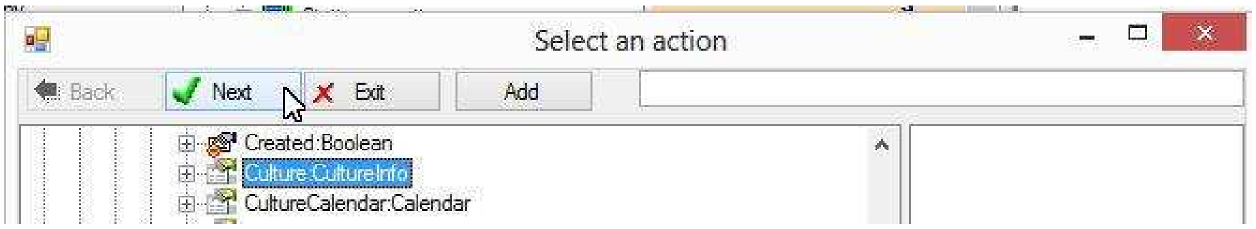
## 7.2 Handle ProjectCultureChanged

Right-click ProjectCultureChanged; choose “Assign action”:

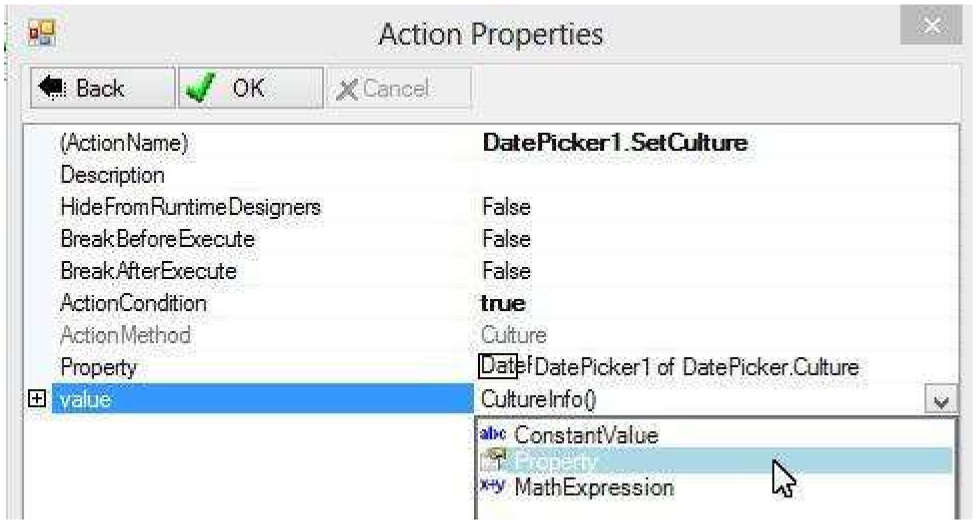


Select Culture property of DatePicker1:

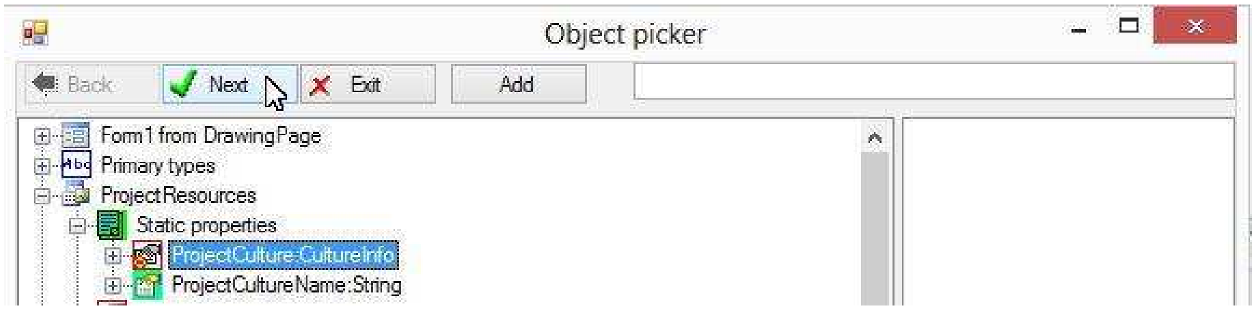




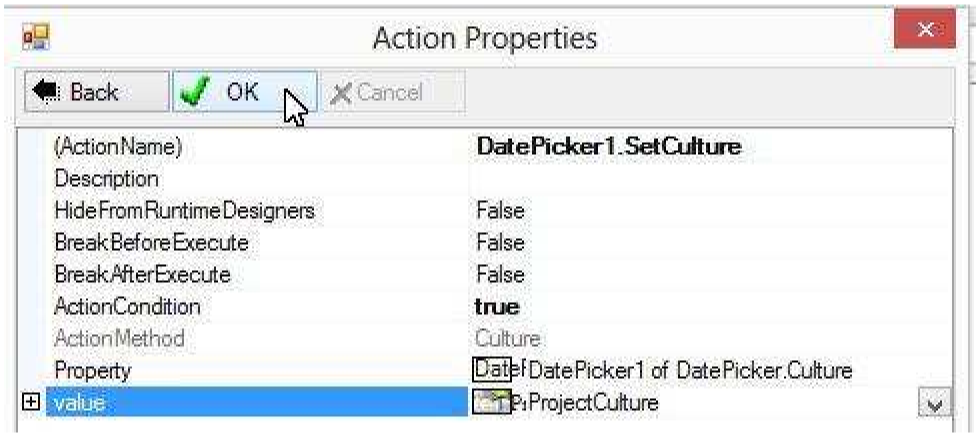
Choose ProjectCulture for the "value":


Action Properties

| | |
|---|---|
| (ActionName) | **DatePicker1.SetCulture** |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | **true** |
| ActionMethod | Culture |
| Property | DatePicker1 of DatePicker.Culture |
| value | CultureInfo() |

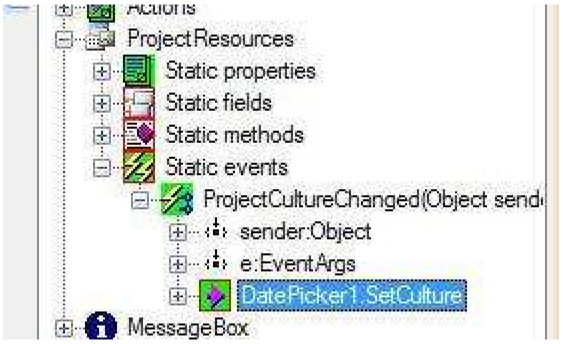


Click OK to finish creating the action:


Action Properties

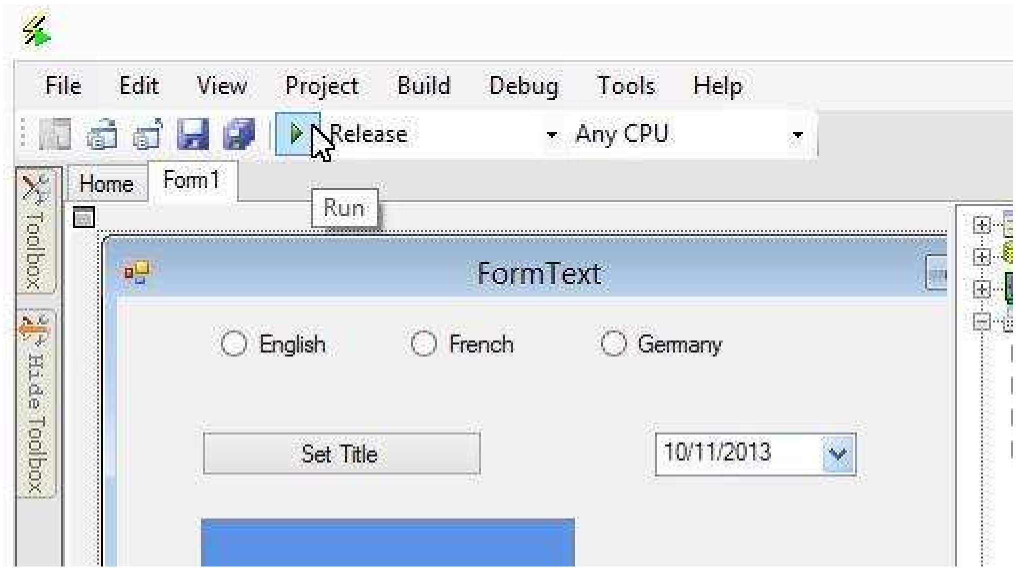
| | |
|---|---|
| (ActionName) | **DatePicker1.SetCulture** |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | **true** |
| ActionMethod | Culture |
| Property | DatePicker1 of DatePicker.Culture |
| value | ProjectCulture |

The action is created and appears under the event:

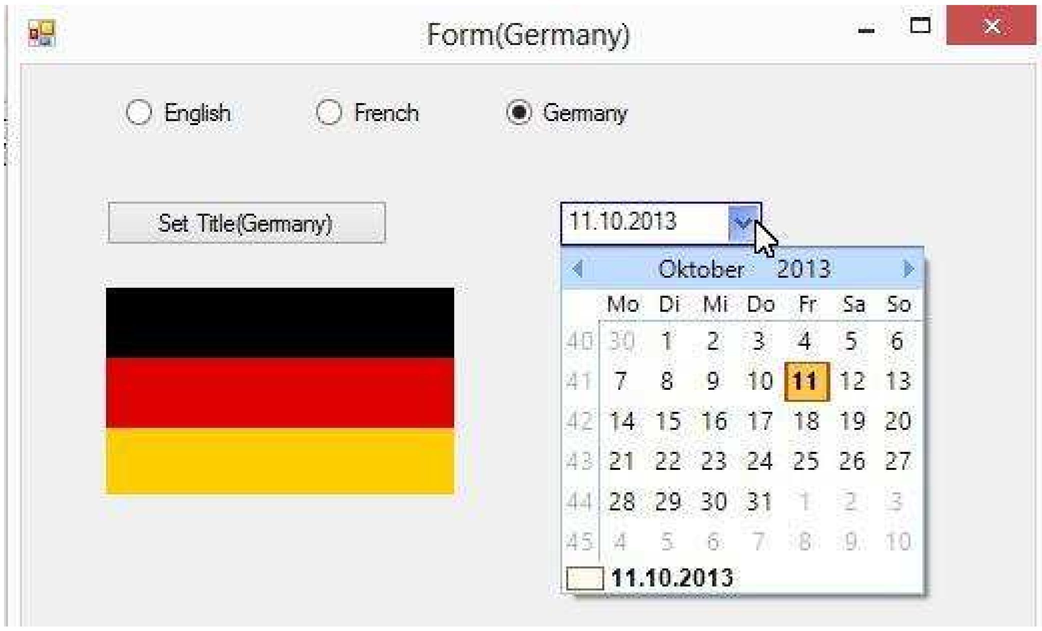


## 7.3 Test

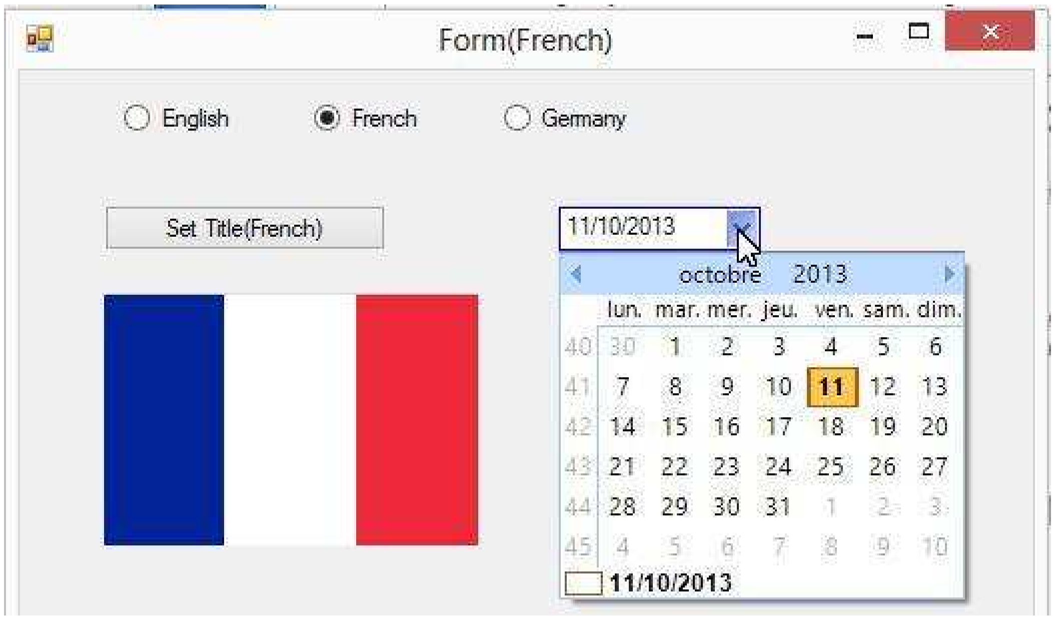
We may test it now:



The form appears. Select Germany. Note that the datetime picker is in Germany.



Click French. The datetime picker is in French:



Note that for a date-time localization you cannot use neural culture. For example, you cannot use “en”, you must use “en-CA”, “en-US”, etc. You may use “fr-FR”, but cannot use “fr”.

## 8 Feedback

Please send your feedback to support@limnor.com.