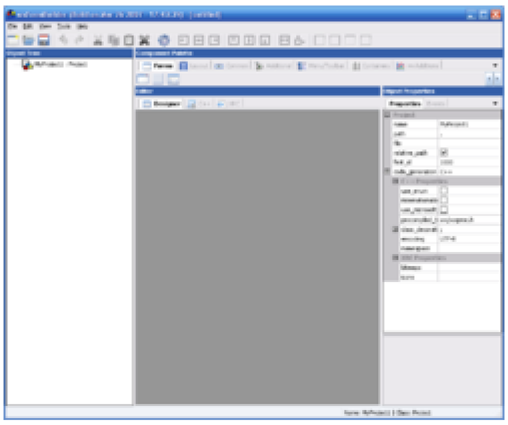
Tutorials:UsingWxFormBuilder

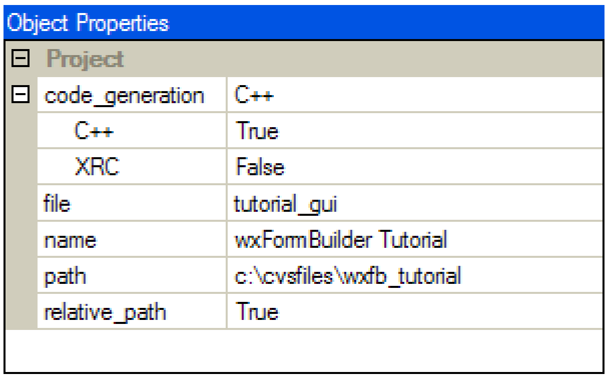
How to use wxFormBuilder v3.0

1 Open wxFormBuilder and start an empty project by pressing . You can also execute File | New.



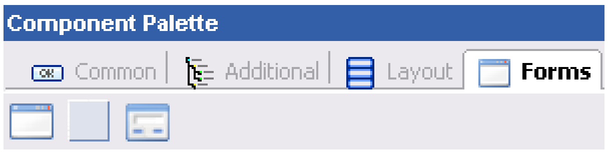
2 Configure the project settings from the Object Properties panel.

- Choose which code will be generated. Currently, you can generate C++ and/or XRC code.
- Set the file name of the generated file (only the name, without the extension).
- Type a name for your project.
- Set the path in which the code will be generated. The relative path "." will cause the code to be generated in the same directory as the project file.
- Check relative_path if you want all file referenced by your project (e.g. images) to be generated with a relative path. Save your Project.

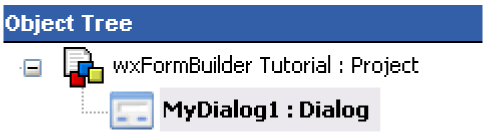

Object Properties

Project	
code_generation	C++
C++	True
XRC	False
file	tutorial_gui
name	wxFormBuilder Tutorial
path	c:\cvsfiles\wxfb_tutorial
relative_path	True

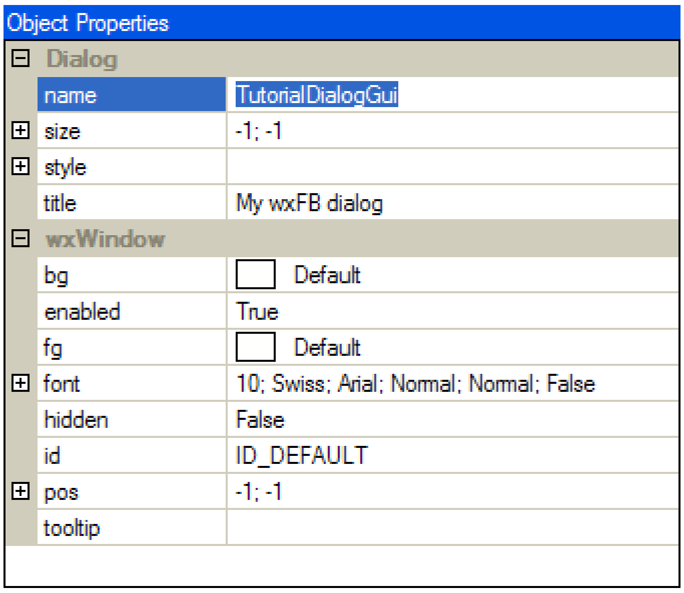
3 Go to the component palette and click on the Forms tab. Then, create a dialog by clicking on the third icon.



Now the dialog is shown in the Object Tree.


Object Tree
wxFormBuilder Tutorial : Project
MyDialog1 : Dialog

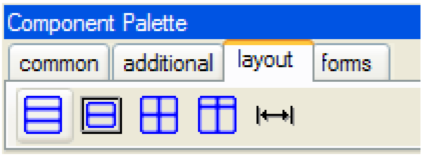
4 Now you can configure the dialog properties in the Object Properties panel. For example, change the name of the dialog to TutorialDialogGui. That name will be also the name of the generated C++ class.



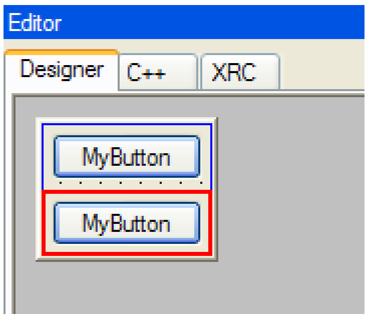
Object Properties

⊟ Dialog	
name	TutorialDialogGui
⊞ size	-1; -1
⊞ style	
title	My wxFB dialog
⊟ wxWindow	
bg	☐ Default
enabled	True
fg	☐ Default
⊞ font	10; Swiss; Arial; Normal; Normal; False
hidden	False
id	ID_DEFAULT
⊞ pos	-1; -1
tooltip	

5 After that, you are ready to populate your dialog with controls and sizers. wxFormBuilder only supports sizer based layouts, so you will have to add a sizer to be able to insert controls. Go to the Component Palette and click on the Layout tab. Then, create a box sizer by clicking on the first icon.

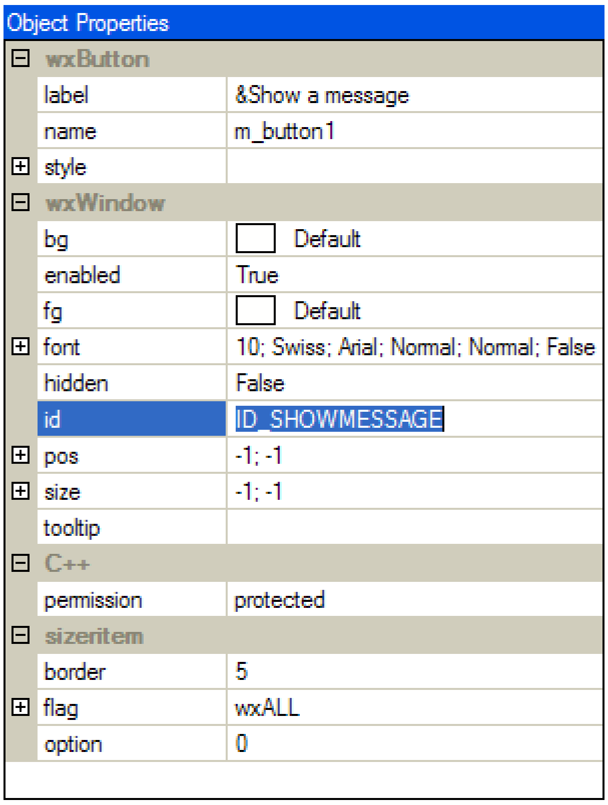


6 Add some controls to the sizer, for example, two buttons. So, go to the Common tab and press the first icon () twice.



7 Change the label of the icons by modifying the label property in the Object Properties panel. For example, write &Show a message in the label property of the first button and &Close in the second one.

Also, set the id property to ID_SHOWMESSAGE and wxID_OK in the first and second button, respectively.

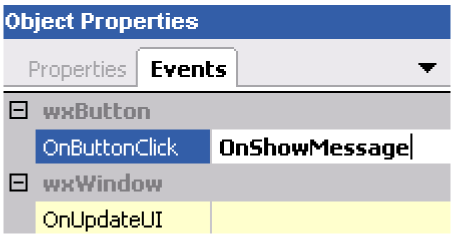


Object Properties

⊟ wxButton	
label	&Show a message
name	m_button1
⊞ style	
⊟ wxWindow	
bg	☐ Default
enabled	True
fg	☐ Default
⊞ font	10; Swiss; Arial; Normal; Normal; False
hidden	False
id	ID_SHOWMESSAGE
⊞ pos	-1; -1
⊞ size	-1; -1
tooltip	
⊟ C++	
permission	protected
⊟ sizeritem	
border	5
⊞ flag	wxALL
option	0

Properties of the first button.

8 Now, it is time to add event handlers to the buttons. Switch to the Events tab in the Object Properties panel. Set the value of OnButtonClick for the first button to OnShowMessage and set the value to On-Close for the second button.



Object Properties

Properties | Events

wxButton

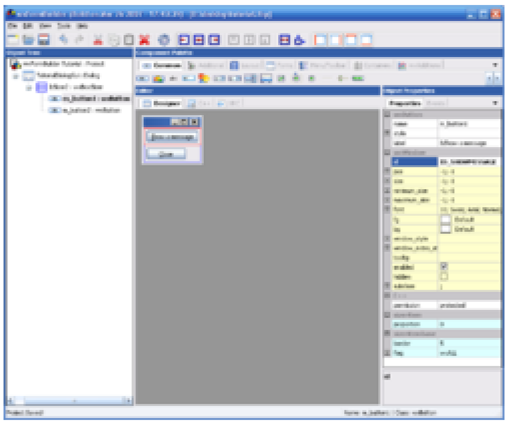
OnButtonClick	OnShowMessage

wxWindow

OnUpdateUI	

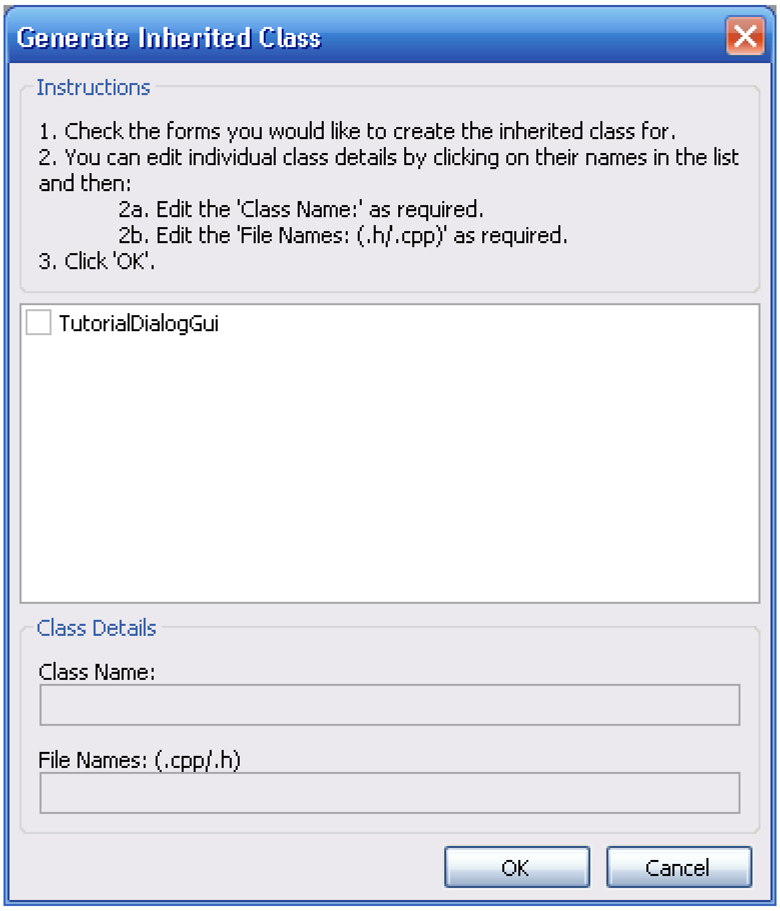
Events of the first button.

9 Now, you are ready to generate the code! Press F8 or . If everything is correct, you will see a message in the status bar which says the code has been generated. You can view the output code at any time in the C++ tab.



10 **Now it is time to integrate the generated code with your IDE (CodeLite, Code::Blocks, Dev-C++, Visual C++...). You will have to add the generated files (tutorial_gui.h/cpp) to the project.**

11 **Each time you generate code from wxFormBuilder, the file you generated the last time will be overwritten, so you cannot add your event handlers directly to that file. The approach chosen in wxFormBuilder is to subclass the generated classes. So, launch the Generate Inherited Class wizard from Tools | Generate Inherited Class.**



1. Check the forms you would like to create the inherited class for.
2. You can edit individual class details by clicking on their names in the list and then:
 2a. Edit the 'Class Name:' as required.
 2b. Edit the 'File Names: (.h/.cpp)' as required.
3. Click 'OK'.

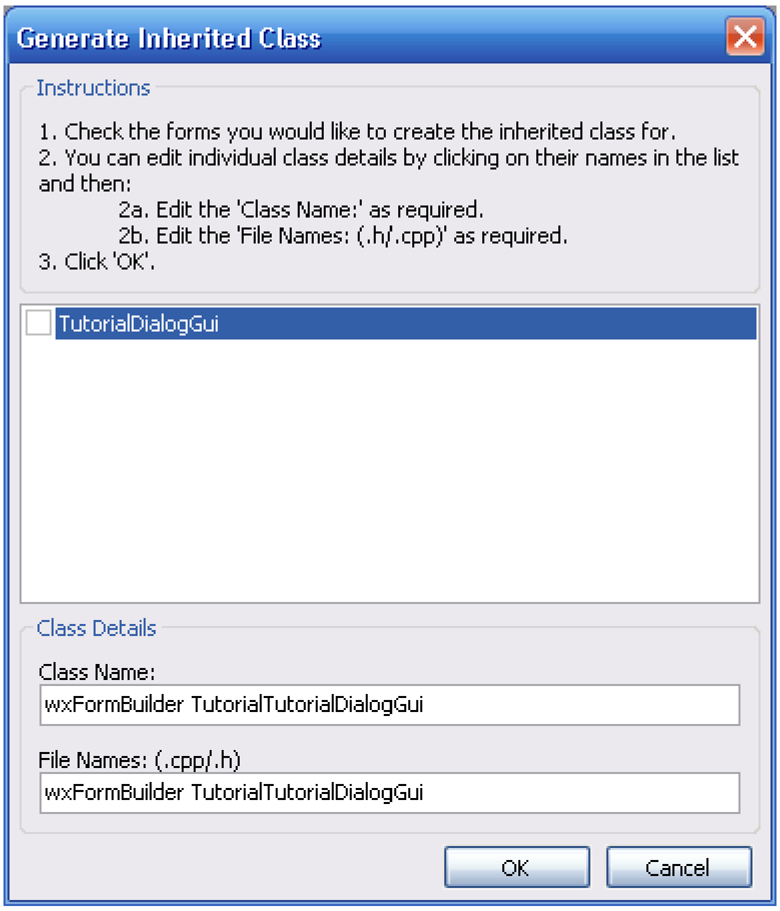
☐ TutorialDialogGui

Class Name:

File Names: (.cpp/.h)

Generate Inherited Class Dialog

12 Select the name of your form, TutorialDialogGui, so you can modify the name of the generated class.


Generate Inherited Class

Instructions

1. Check the forms you would like to create the inherited class for.
2. You can edit individual class details by clicking on their names in the list and then:
    2a. Edit the 'Class Name:' as required.
    2b. Edit the 'File Names: (.h/.cpp)' as required.
3. Click 'OK'.

TutorialDialogGui

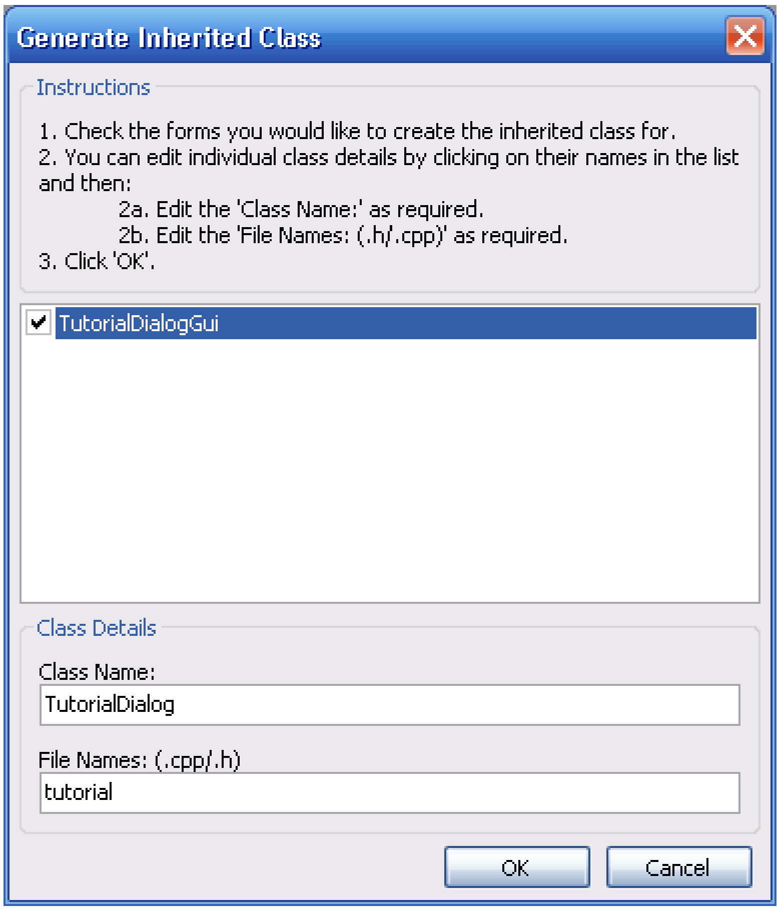
Class Details

Class Name: wxFormBuilder TutorialTutorialDialogGui

File Names: (.cpp/.h) wxFormBuilder TutorialTutorialDialogGui

Form Name Selected

13 Change the class name to TutorialDialog and the file name to tutorial. Make sure to check the box for your form before clicking "OK".


Generate Inherited Class

Instructions

1. Check the forms you would like to create the inherited class for.
2. You can edit individual class details by clicking on their names in the list and then:
   2a. Edit the 'Class Name:' as required.
   2b. Edit the 'File Names: (.h/.cpp)' as required.
3. Click 'OK'.

✓ TutorialDialogGui

Class Details

Class Name: TutorialDialog

File Names: (.cpp/.h) tutorial

Names Changed, Form Checked

14 Click "OK" in the dialog, if all goes well you will see a "Class Generated" message in the status bar. You should find tutorial.cpp and tutorial.h next to your other generated code, in the directory where you saved your project file.

The header file will look like this:

```
#ifndef __tutorial__ #define __tutorial__
/** @file Subclass of TutorialDialogGui, which is generated by wxFormBuilder.
@todo Add your event handlers directly to this file. */
#include "tutorial_gui.h"
/** Implementing TutorialDialogGui */
class TutorialDialog : public TutorialDialogGui
{ public: /** Constructor */ TutorialDialog( wxWindow* parent ); };
#endif // __tutorial__
```

This is the example source file:

```
#include "tutorial.h"
TutorialDialog::TutorialDialog( wxWindow* parent ) :
TutorialDialogGui( parent )
{
}
```

15 Now, you need to add the implementation of your event handlers to the generated TutorialDialog class. The event table is generated in the tutorial_gui files, so you just need to add the functions to the TutorialDialog class. Here are the same two files with event handlers added.

The header file will look like this:

```
#ifndef __tutorial__
#define __tutorial__
/** @file Subclass of TutorialDialogGui, which is generated by wxFormBuilder.
@todo Add your event handlers directly to this file. */
#include "tutorial_gui.h"
/** Implementing TutorialDialogGui */
class TutorialDialog : public TutorialDialogGui
{ public: /** Constructor */ TutorialDialog( wxWindow* parent );
protected: // Event handlers
void OnShowMessage( wxCommandEvent& event );
void OnClose( wxCommandEvent& event ); };
#endif // __tutorial__
```

This is the example source file:

```
#include "tutorial.h"
#include <wx/msgdlg.h>
#include <wx/app.h>
TutorialDialog::TutorialDialog( wxWindow* parent ) :
TutorialDialogGui( parent )
{
}
void TutorialDialog::OnShowMessage( wxCommandEvent& event )
{
wxMessageBox( wxT("wxFormBuilder Tutorial") );
}
void TutorialDialog::OnClose( wxCommandEvent& event )
{
wxTheApp->Exit();
}
```

16 All you are missing now are the source files for the main application. Add two new files to your project and in the OnInit() function of your wxApp derived class create an object of type TutorialDialog.

The header file might look like this:

```
#ifndef __WXWIDGETSAPP_H
#define __WXWIDGETSAPP_H
#include <wx/wx.h>
class wxWidgetsApp : public wxApp
{
public: wxWidgetsApp();
virtual ~wxWidgetsApp();
virtual bool OnInit();
};
DECLARE_APP(wxWidgetsApp)
#endif
```

The source file might look like this:

```
#include "wxWidgetsApp.h"
#include "tutorial.h"
IMPLEMENT_APP(wxWidgetsApp)
wxWidgetsApp::wxWidgetsApp() { }
wxWidgetsApp::~wxWidgetsApp() { }
bool wxWidgetsApp::OnInit()
{
TutorialDialog* dialog = new
TutorialDialog( (wxWindow*)NULL );
dialog ->Show();
SetTopWindow( dialog );
return true;
}
```

You can download all of the files from this tutorial here: tutorial.zip http://wxformbuilder.sourceforge.net/tutorials/tutorial.zip

Compile and run your project!