



wxGlue Documentation

Release 0.2 (Start)

Brian Toby

December 29, 2012

CONTENTS

1	<i>Introduction</i>	3
2	<i>What is Implemented</i>	5
3	<i>How to Use wxGlue</i>	7
4	<i>Overview of wxGlue Code</i>	9
4.1	Imports	9
4.2	Define Function	9
4.3	Define Callbacks	9
4.4	Control Initialization	10
4.5	Start the code	10
5	<i>Control-specific Method Use</i>	11
5.1	wx.TextCtrl methods	11
5.2	wx.Button methods	11
5.3	Calendar Methods	11
5.4	Option Selection Methods	11
6	<i>Classes and Functions</i>	13
	Python Module Index	19
	Index	21

wxGlue Info

Release 0.2 (Start)
Date December 29, 2012
Author Brian H. Toby
Target Novice Python programmers
Project Status in early development

Contents

- wxGlue: Classes to implement GUIs from wxGlade
 - *Introduction*
 - *What is Implemented*
 - *How to Use wxGlue*
 - *Overview of wxGlue Code*
 - * Imports
 - * Define Function
 - * Define Callbacks
 - * Control Initialization
 - * Start the code
 - *Control-specific Method Use*
 - * wx.TextCtrl methods
 - * wx.Button methods
 - * Calendar Methods
 - * Option Selection Methods
 - *Classes and Functions*

INTRODUCTION

This script contains a set of classes used to manage graphical user interface (GUI) controls in wxPython in a way that allows GUIs to be programmed without directly writing any wxPython calls. The GUI script is created using wxGlade (<http://wxglade.sourceforge.net/>) and then wxGlue takes over by providing a set of methods that interact with the GUI controls, such as loading settings into variables, validating inputs and responding to button presses.

To help in preparing the script that will use the wxGlue methods and functions, the wxGlue script also provides a function that will scan a script created with wxGlade and create a template script that has example code. The template can then be edited to better control the GUI and to introduce the functionality is desired.

WHAT IS IMPLEMENTED

This code has only been tested with wxGlade. In theory one should be able to use other GUI builders such as BoaConstructor, but this has not been tested. The top-level widget should be a wx.Frame instance and the following wxPython controls are currently supported:

TextCtrl, Button, BitmapButton, StaticBitmap, DatePickerCtrl, CalendarCtrl, ToggleButton, SpinCtrl, SpinButton, CheckBox, RadioButton, Gauge, Slider, StaticText, StaticBox, RadioBox, Choice, ComboBox, ListBox

This includes most of the capabilities in wxGlade, except for MenuBar, ToolBar, wx.ListCtrl, wx.TreeCtrl, wx.grid.Grid, Applications must have a wx.Frame base class, so at least for now, creation of frames using base class wx.MDICHildFrame or wx.Dialog is not supported. The following items have not been tested, but might work: Notebooks, Panels/ScrolledWindows, SplitterWindow.

HOW TO USE WXGLUE

The first step in using this package is to create a frame with a GUI using wxGlade.

1. Start wxGlade.
2. In the wxGlade Properties window, edit the application properties as needed. Be sure to set an output path: this should contain the entire name of the python file to be created. Note that wxGlue has been tested with settings: US-ASCII; no gettext support; Code generation: Single File; Language: Python (of course); wxWidgets: 2.8.
3. In the main wxGlade window click on the “Frame” button (upper left). A new window pops up, select the base class as wxFrame and name the class as you will. Click OK.
4. Note that a frame and a sizer has been added to the the wxGlade: Tree window. You will likely want to split this into more than one vertical section. Do this by right-clicking on the sizer entry in that window (Mac: control+click) and select the Add slot menu item as many times as needed.
5. To break a sizer section into multiple horizontally divided sections, click on the BoxSizer button in the main wxGlade window (near end). Then click in the Design window where this sizer will be placed. Select the Orientation: Horizontal, set the number of slots, as desired (usually 2 or more). Note that the Static Box option allows a label to be added to the border of the sizer.
6. Add controls to each section of the sizers, by clicking on the control icon in the main wxGlade window and then clicking in the design window where the widget will be placed. It is necessary to click on the on the control icon each time it a control will be added, even if it is a repeat of the previous.
7. Edit parameters for a control by changing information in the properties window. If needed, a control can be selected by clicking on it in the Design or wxGlade: Tree window. Note that the names assigned to controls are used in the glue code to reference them. Use care to select names that will help indicate which item is which later. You should not create any bindings for events here; they will be overridden later.
8. When the GUI design is complete, the code can be created using the File/Generate Code menu item. This creates the python file that was referenced in the second step. This file can be run in Python. It will display the frame and controls, but not do anything.

9. Create a template file by running the wxGlue script in Python:

```
python <path>/wxGlue/__init__.py <GUI_file.py>
```

If <GUI_file.py> is not placed on the command line, it is prompted for. The output file name is also prompted for as well.

10. Manually edit the template file to change the way that the controls are used.

OVERVIEW OF WXGLUE CODE

The template file created by wxGlue will have several sections.

4.1 Imports

The first section will define references for modules that need to be imported:

```
import sys
sys.path.insert(0, "/Users/toby/projects/wxglade")
import wxglue
sys.path.insert(0, "/Users/toby/myProject")
import myGUI
```

In the above case, myGUI.py (in directory /Users/toby/myProject) is the GUI created in wxGlade.

4.2 Define Function

The next section defines a function for each Frame class found in the the GUI file. It starts by creating a frame from the class and glue object for the frame:

```
def myGUI_MyFrame1():
    gluemod = wxglue.wxGlue(myGUI.MyFrame1)
```

4.3 Define Callbacks

Then functions are created for each callback that will be invoked by use of controls. Note that almost every control can be linked to a callback, but the template file will be created with examples only for every button, such as this:

```
def On_button_1():
    text_ctrl_1 = gluemod.GetValue('text_ctrl_1')
    datepicker_ctrl_1 = gluemod.GetValue('datepicker_ctrl_1')
    print('Called On_button_1 with values'
          + '\n\ttext_ctrl_1 = ' + str(text_ctrl_1)
          + '\n\tdatepicker_ctrl_1 = ' + str(datepicker_ctrl_1)
          )
```

This code shows how values are obtained for controls (in this case named *text_ctrl_1* and *datepicker_ctrl_1*). These values are then used in the print statement.

4.4 Control Initialization

The next section will have sample code for each control found in the frame, for example:

```
# gluemod.SetTextCtrlType("text_ctrl_1", float)
# gluemod.SetValue("text_ctrl_1", 1.0)
# gluemod.SetEmptyInvalid("text_ctrl_1", True)
# gluemod.SetMinValue("text_ctrl_1", -10.0)
# gluemod.SetMaxValue("text_ctrl_1", 10.0)

gluemod.SetAction("button_1", On_button_1)
# gluemod.SetLabel("button_1", 'Button Label')
# gluemod.Enable("button_1", True)

# gluemod.SetValue("datepicker_ctrl_1", (2013,1,31))
```

These examples will show the most commonly used methods for each type of control, but there is a complete list of methods documented for class `wxGlue`. Some methods, such as `wxGlue.Enable()`, are available for every type control. The `wxGlue.SetLabel()` or `wxGlue.SetValue()` methods (occasionally both) will be available for nearly every control, but the type for the value that will be passed varies depending on the control type. Note that if a method is not implemented for a control, it will be ignored. These statements are used in this section of the code to initialize controls, but they may also be placed in the function callbacks to change values, etc.

4.5 Start the code

The final section launches the application by creating the GUI and starting the event loop:

```
myGUI_MyFrame1()
wxglue.StartEventLoop()
```

CONTROL-SPECIFIC METHOD USE

5.1 wx.TextCtrl methods

One control with special treatment is `wx.TextCtrl`. This has a two unique methods that are used for validation of the control contents. The method `wxGlue.SetTextCtrlType()` sets a type for the control. At present, `str` (the default), `int` and `float` are all available. For `str` type controls, `wxGlue.SetEmptyInvalid()` indicates that the `TextCtrl` control may not be left blank. For the latter two numeric types, only digits, signs and for floats, a decimal point and exponent (e) are allowed to be entered into the control. Also for two numeric types, `wxGlue.SetMinValue()` and `wxGlue.SetMaxValue()` are optionally used to set a range for number that can be entered. If an invalid value is entered into a control, the color of the control is changed to highlight it and other controls may be disabled (see `wxGlue.SetValidationRequired()`).

5.2 wx.Button methods

There are no methods that are special to `wx.Button` controls, but several methods are used more commonly with buttons than with other controls. Method `wxGlue.SetAction()` is used to specify a function or method that will be called when the button is pressed. Note that optional arguments can specify a list of dictionary of arguments to be supplied in that call. Method `wxGlue.SetValidationRequired()` indicates that a button should be disabled if invalid values are present in any validated `TextCtrl`. `wxGlue.SetLabel()` can be used to change the label on a button. An OK button (with Id `wx.ID_OK`) is automatically set to be disabled if any input is invalid.

5.3 Calendar Methods

Calendar controls (`wx.calendar.CalendarCtrl` and `wx.DatePickerCtrl`) take special input with `wxGlue.SetValue()`. The value can be a list or tuple with three `int` elements, (year, month, day), such as (2013, 1, 13) or can be a Python `datetime.date` value. The `wxGlue.GetValue()` routine will always return a `datetime.date` value.

5.4 Option Selection Methods

Controls that present the user with a set of fixed choices (`RadioBox`, `Choice`, `ComboBox`, and `ListBox`) also work a bit differently. The `wxGlue.SetValue()` and `wxGlue.GetValue()` methods accept or return integer values corresponding to the option number (counting starts with zero). Likewise, `wxGlue.GetMinValue()` returns 0 and `wxGlue.GetMaxValue()` returns the number of options minus one. The `wxGlue.SetLabel()` function accepts a list or tuple of string values. For `Choice`, `ComboBox`, and `ListBox` controls, the number of options will be

set by the number of items in the list, while for `RadioBox` controls, this number cannot be changed and extra strings will be ignored. Note that `wxGlue.GetLabel()` can be used to determine the string value that has been selected in the control.

CLASSES AND FUNCTIONS

`wxGlue.GetInputFile` (*parent=None, multiple=False, startdir=None, title='Choose an input file'*)
Creates a dialog to get name(s) of existing file(s) and returns a list of files

Parameters

- **parent** (*frame*) – Name of parent to the dialog or None (default) if not needed
- **multiple** (*bool*) – Allow dialog to select only one file (False, default) or multiple files (True)
- **startdir** (*str*) – string with the name of a directory to initialize the dialog with. If None (default) the initial dialog will be the current working directory.
- **title** (*str*) – a title for the dialog (default is “Choose an input file”)

Returns a list of file(s) or an empty list if Cancel is pressed.

`wxGlue.GetNewFile` (*parent=None, startdir=None, defaultname='', title='Choose an output file'*)
Creates a dialog to get a name for a new file and returns that name

Parameters

- **parent** (*frame*) – Name of parent to the dialog or None (default) if not needed
- **startdir** (*str*) – string with the name of a directory to initialize the dialog with. If None (default) the initial dialog will be the current working directory.
- **defaultname** (*str*) – A default file name for the dialog (default is none)
- **title** (*str*) – a title for the dialog (default is “Choose an output file”)

Returns a list of file(s) or an empty list if Cancel is pressed.

`wxGlue.SetupTemplate` (*in_files=[]*)

This routine is used to process one or more GUI scripts created in wxGlade and create a template file to help in the creation of a script that uses the GUI script(s) and wxGlue. This will normally be used by invoking this script directly in a python interpreter (python wxglue.py).

`wxGlue.StartEventLoop` ()

Start the wx event loop after all code has been put in place

class `wxGlue.wxGlue` (*frameclass*)

An object that “glues” controls to a frame (window) object. The frame is created from a class that is typically coded with wxGlade. The name of the class that defines the frame is passed to wxGlue on initialization. Calls to methods are then used to define settings for controls or actions linked to them

Parameters *frameclass* (*class*) – defines a class that can be used to create a frame. This class is usually coded using wxGlade.

CloseFrame ()

Closes the frame (window) created by this object.

Disable (*lbl*)

Disables a control.

Parameters *lbl* (*str*) – the name of the control (widget)

Enable (*lbl*, *value=True*)

Enables or Disables a control. This can be used with all controls, but ones that do not have a used interaction may not change as a result of being disabled.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Determines if the control will be enabled (*value=True*, default) or disabled (*value=False*).

GetLabel (*lbl*)

Returns the label associated with a control or a list of labels in the case of controls that select options. Note that some controls do not display a label, but still can have a label set. Controls that contain multiple options (wx.RadioButton, wx.Choice, wx.ComboBox or wx.ListBox) return a list of labels (each an str).

Parameters *lbl* (*str*) – the name of the control (widget)

Returns the widget label, if defined (type str) or a list of (str) labels for the wx.RadioButton, wx.Choice, wx.ComboBox or wx.ListBox controls.

GetMaxValue (*lbl*)

Get a maximum value that is allowed for the setting of a control. This can be used for TextCtrl, SpinButton and SpinCtrl controls as well as controls that have options (RadioButton, Choice, ComboxBox, and ListBox) where it will supply the maximum allowed value.

Parameters *lbl* (*str*) – the name of the control (widget)

Returns the maximum allowed value for control *lbl* (int/float)

GetMinValue (*lbl*)

Get a minimum value that is allowed for the setting of a control. This can be used for TextCtrl, SpinButton and SpinCtrl controls as well as controls that have options (where it will supply the minimum valid value, 0).

Parameters *lbl* (*str*) – the name of the control (widget)

Returns the minimum allowed value for control *lbl* (int/float)

GetValue (*lbl*)

Returns the value associated with a control. The type for the value is dictated by the type of the control, as shown in the table below:

Parameters *lbl* (*str*) – the name of the control (widget)

control	type
wx.TextCtrl	str or [float/int, if set by <code>SetTextCtrlType()</code>]
wx.DatePickerCtrl	<i>datetime.date</i>
wx.calendar.CalendarCtrl	<i>datetime.date</i>
wx.ToggleButton	bool
wx.SpinCtrl	int
wx.CheckBox	bool
wx.RadioButton	bool
wx.RadioBox	int
wx.Gauge	float
wx.Slider	int
wx.Choice	int
wx.ComboBox	int
wx.ListBox	int
wx.Button	None
wx.StaticBitmap	None
wx.StaticText	None
wx.StaticBox	None

SetAction (*lbl*, *routine*, *args*=[], *kwargs*={})

Register a function that will be called when a control is activated. This will work with almost every type of input control, but will most commonly be used with button-type controls. For controls where input can be typed (TextCtrl and SpinBox, for example), these actions will be performed every time then mouse leaves the text box, so any action routine should be innocuous.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **routine** (*function*) – a function or method that will be called when the control is invoked, for example, by pressing a button
- **args** (*list*) – optional positional arguments to be supplied as parameters to the function *routine*
- **kwargs** (*dict*) – optional keyword arguments to be supplied as parameters to the function *routine*

SetEmptyInvalid (*lbl*, *value*=True)

Indicate that a TextCtrl control is invalid if it has a blank value. A control is considered to be blank if it has only whitespace characters [see `str.strip()`] or is null. The default is to allow controls to be considered valid if blank.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Value to set: True (defaults) indicates that a control is invalid when blank

SetLabel (*lbl*, *value*)

Sets the label(s) associated with a control. This can be used with all controls, but for controls do not display a label, for these items no change will be seen. Controls that contain multiple options (wx.RadioBox, wx.Choice, wx.ComboBox or wx.ListBox) require a list of labels (each an str). Note that for wx.RadioBox, the number of options cannot be changed, but for the rest, the number of labels in the list determines the new number of options in the control.

Parameters

- **lbl** (*str*) – the name of the control (widget)

- **value** (*str/list*) – the new label for control *lbl*, or a list of labels for `wx.RadioButton`, `wx.Choice`, `wx.ComboBox` or `wx.ListBox`.

SetMaxValue (*lbl, value*)

Set a maximum value that is allowed for the setting of a control. This can be used for `SpinButton` and `SpinCtrl` controls as well as `TextCtrls`.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*int/float*) – the maximum allowed value for control *lbl*

SetMinValue (*lbl, value*)

Set a minimum value that is allowed for the setting of a control. This can be used for `SpinButton` and `SpinCtrl` controls as well as `TextCtrls`.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*int/float*) – the minimum allowed value for control *lbl*

SetTextCtrlType (*lbl, typ*)

Set a variable type for a `TextCtrl` control. If the type is `int` or `float`, restrictions are placed on the characters that may be typed in the control. For `int`, valid characters are “+0123456789”, for `float`, the characters “.”, “e” and “E” are allowed in addition. Validation of the contents of the `TextCtrl` is performed when `Return` is pressed or another control is selected. This means checking if the can be converted to a valid `int` or `float`, if selected, or is non-blank, if `SetEmptyInvalid()` has been used to require non-blank values.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **typ** (*type*) – a type to be used for the `TextCtrl` value. Usually `int` or `float` (*str* is assumed if not called).

SetValidationRequired (*lbl, value=True*)

Indicate that a control (usually `Button`) will be disabled when validated `TextCtrl` controls are set with invalid values. Do not use `SetValidationRequired()` with `TextCtrls` because once the control has an invalid value, it will be disabled and then can't be changed.

Note that `OK` buttons (with `Id = wx.ID_OK`) are automatically set to be disabled when validated `TextCtrl` controls are set with invalid values.

`TextCtrl` controls are set to be validated using `SetTextCtrlType()`, `SetMinValue()` and `SetEmptyInvalid()`.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Value to set: `True` (default) indicates that a control should be disabled when validated controls are invalid.

SetValue (*lbl, value*)

Change the value associated with a control. Note that the type for the value must be appropriate for the control, as shown in the table below.

Parameters **lbl** (*str*) – the name of the control (widget) ;param many value: the value for control *lbl*, with type dictated by the table below. Note that for controls labeled (n/a), this method has no effect.

control	type
wx.TextCtrl	str or [float/int, if set by <code>SetTextCtrlType()</code>]
wx.DatePickerCtrl	<i>datetime.date</i> or list of 3 integers as (yyyy,mm,dd)
wx.calendar.CalendarCtrl	<i>datetime.date</i> or list of 3 integers as (yyyy,mm,dd)
wx.ToggleButton	bool
wx.SpinCtrl	int
wx.CheckBox	bool
wx.RadioButton	bool
wx.RadioBox	int
wx.Gauge	float
wx.Slider	int
wx.Choice	int
wx.ComboBox	int
wx.ListBox	int
wx.Button	(n/a)
wx.StaticBitmap	(n/a)
wx.StaticText	(n/a)
wx.StaticBox	(n/a)

ShowValues()

For development, print all values in the self.Values dict.

TestIfValid()

Check if all TextCtrl controls that have defined ranges or require non-empty values have been set with valid values. All controls that designated to require validation [with `self.SetValidationRequired()`] will be enabled or disabled accordingly. This will probably not be used externally.

Returns True if all validated TextCtrl controls have valid values

Values = None

A dict indexed by a control label. Contains the current value for that input control. Also see `GetValue()`, which returns this value.

PYTHON MODULE INDEX

W

wxGlue, 1

INDEX

C

CloseFrame() (wxGlue.wxGlue method), 13

D

Disable() (wxGlue.wxGlue method), 14

E

Enable() (wxGlue.wxGlue method), 14

G

GetInputFile() (in module wxGlue), 13

GetLabel() (wxGlue.wxGlue method), 14

GetMax Value() (wxGlue.wxGlue method), 14

GetMin Value() (wxGlue.wxGlue method), 14

GetNewFile() (in module wxGlue), 13

GetValue() (wxGlue.wxGlue method), 14

S

SetAction() (wxGlue.wxGlue method), 15

SetEmptyInvalid() (wxGlue.wxGlue method), 15

SetLabel() (wxGlue.wxGlue method), 15

SetMax Value() (wxGlue.wxGlue method), 16

SetMin Value() (wxGlue.wxGlue method), 16

SetTextCtrlType() (wxGlue.wxGlue method), 16

SetupTemplate() (in module wxGlue), 13

SetValidationRequired() (wxGlue.wxGlue method), 16

SetValue() (wxGlue.wxGlue method), 16

ShowValues() (wxGlue.wxGlue method), 17

StartEventLoop() (in module wxGlue), 13

T

TestIfValid() (wxGlue.wxGlue method), 17

V

Values (wxGlue.wxGlue attribute), 17

W

wxGlue (class in wxGlue), 13

wxGlue (module), 1