
Pyforms GUI Documentation

Release 4.0

Ricardo Jorge Vieira Ribeiro

Feb 08, 2019

1	Overview	3
1.1	Pyforms GUI	3
1.2	Pyforms	3
1.3	Developer	4
2	Install & configure	5
3	First application	7
3.1	Create the first app	7
4	The basic	11
4.1	Prepare the application class	11
4.1.1	Import the library	11
4.1.2	Create your application class.	11
4.1.2.1	Add an action to the button	12
4.2	Create the action	12
4.3	Set the button action	12
4.3.1	Move to the next chapter.	16
4.3.2	Find out what you can do with other Controls here.	16
5	Multiple windows	19
5.1	Create the Model	19
5.1.1	Data model	19
5.1.2	Let's go for the GUI	20
5.1.3	Implement the GUI to manage the People model	21
5.2	EmptyWidget Control	22
5.3	DockWidget Control	23
6	Mdi Applications	27
7	Style and layout with CSS	29
8	Python	31
8.1	BaseWidget	31
8.1.1	Overview	31
8.1.2	API	31
8.2	Controls	32

8.2.1	ControlBase	32
8.2.2	ControlBoundingSlider	34
8.2.3	ControlButton	35
8.2.4	ControlCheckBox	35
8.2.5	ControlCheckBoxList	36
8.2.6	ControlCodeEditor	37
8.2.7	ControlCombo	38
8.2.8	ControlDir	39
8.2.9	ControlDockWidget	39
8.2.10	ControlEmptyWidget	40
8.2.11	ControlFile	40
8.2.12	ControlFilesTree	41
8.2.13	ControlImage	41
8.2.14	ControlLabel	42
8.2.15	ControlList	43
8.2.16	ControlPlayer	44
8.2.17	ControlMatplotlib	46
8.2.18	ControlMdiArea	46
8.2.19	ControlNumber	46
8.2.20	ControlPassword	47
8.2.21	ControlOpenGL	47
8.2.22	ControlProgress	48
8.2.23	ControlSlider	48
8.2.24	ControlText	49
8.2.25	ControlTextArea	49
8.2.26	ControlToolBox	49
8.2.27	ControlToolButton	50
8.2.28	ControlTree	51
8.2.29	ControlTreeView	52
8.2.30	ControlVisVis	52
8.2.31	ControlVisVisVolume	53
8.2.32	ControlWeb	53
8.2.33	ControlEventTimeline	54
8.2.34	ControlEventsGraph	55
8.3	Settings	56
8.3.1	General configurations	56
8.3.2	GUI layout	56
8.3.3	Controls	57

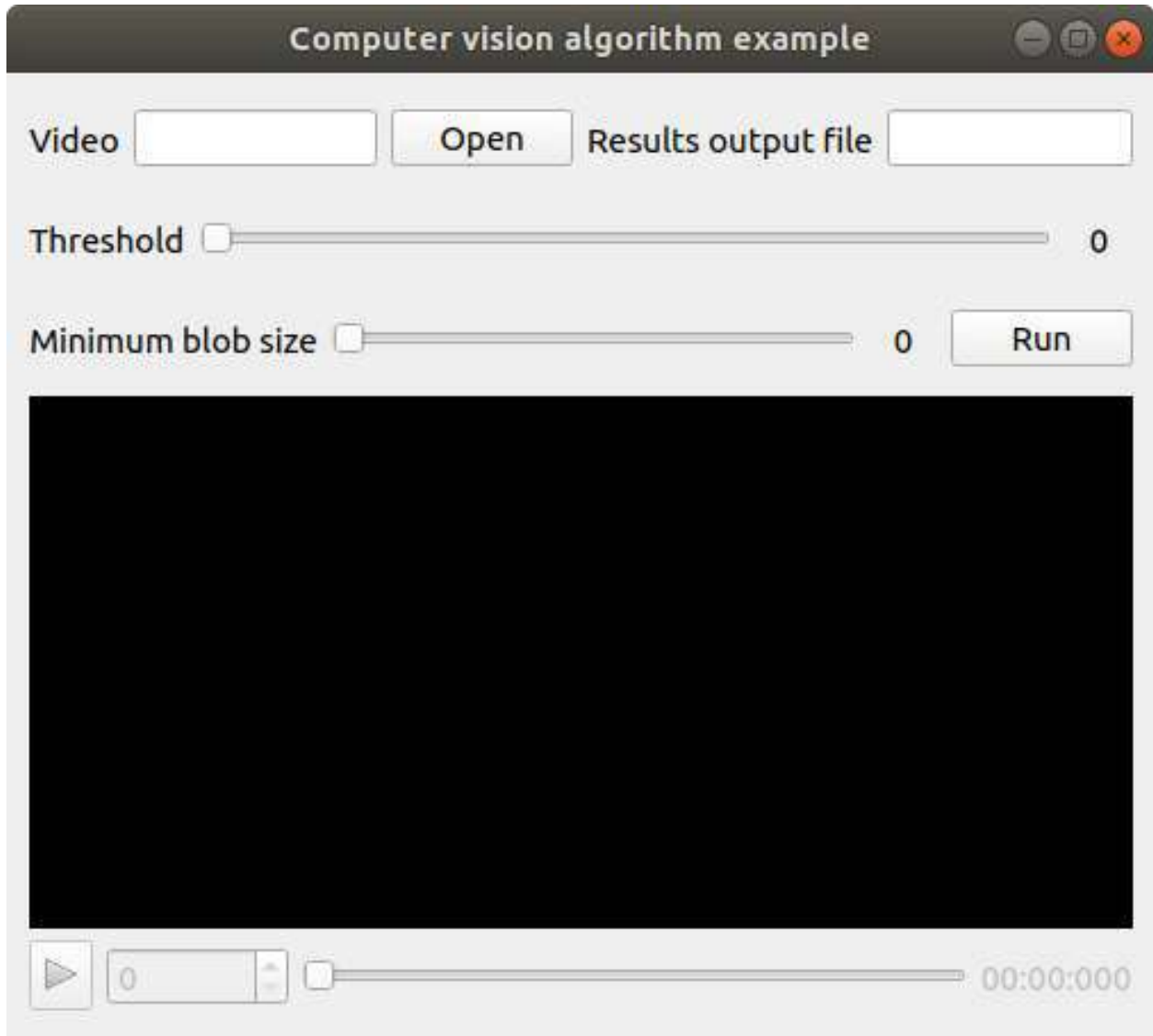
9 Indices and tables **59**

Python Module Index **61**

Pyforms GUI is Python 3 framework to allow pyforms applications to execute in Windows GUI mode.

The framework aims to boost the development productivity by providing an API in Python to allow the execution of applications developed for GUI and Web mode in terminal mode.

Source code <https://github.com/UmSenhorQualquer/pyforms-gui>



Note: This framework is a software layer part of the Pyforms framework.

Pyforms <https://pyforms.readthedocs.io>

1.1 Pyforms GUI



Pyforms GUI is part the Pyforms framework. It implements a software layer that handles the execution of pyforms applications in Windows GUI mode.

1.2 Pyforms



[Pyforms](#) is a Python 3 framework to develop applications capable of executing in 3 diferent environments, Desktop GUI, Terminal and Web.

1.3 Developer

Ricardo Ribeiro	Champalimaud Scientific Software Platform ricardo.ribeiro@research.fchampalimaud.org ricardojvr@gmail.com
--------------------	---

Note: Please **star** the project at the [Github repository](#) to support the project.

CHAPTER 2

Install & configure

- Install Pyforms using **pip**.

```
pip install pyforms-gui
```

Note: More documentation to read about this example at:

- `pyforms_gui.basewidget.BaseWidget`
 - `pyforms_gui.controls.control_base.ControlBase`
-

Here it is shown how to create the first pyforms app.

3.1 Create the first app

Create the file `example.py` and add the next code to it.

```
from pyforms.basewidget import BaseWidget
from pyforms.controls import ControlFile
from pyforms.controls import ControlText
from pyforms.controls import ControlSlider
from pyforms.controls import ControlPlayer
from pyforms.controls import ControlButton

class ComputerVisionAlgorithm(BaseWidget):

    def __init__(self, *args, **kwargs):
        super().__init__('Computer vision algorithm example')

        #Definition of the forms fields
        self._videofile = ControlFile('Video')
        self._outputfile = ControlText('Results output file')
        self._threshold = ControlSlider('Threshold', default=114, minimum=0, ↵
↵maximum=255)
        self._blobsize = ControlSlider('Minimum blob size', default=110, ↵
↵minimum=100, maximum=2000)
```

(continues on next page)

(continued from previous page)

```
self._player      = ControlPlayer('Player')
self._runbutton   = ControlButton('Run')

#Define the function that will be called when a file is selected
self._videofile.changed_event = self.__videoFileSelectionEvent
#Define the event that will be called when the run button is processed
self._runbutton.value = self.__runEvent
#Define the event called before showing the image in the player
self._player.process_frame_event = self.__process_frame

#Define the organization of the Form Controls
self._formset = [
    ('_videofile', '_outputfile'),
    '_threshold',
    ('_blobsize', '_runbutton'),
    '_player'
]

def __videoFileSelectionEvent(self):
    """
    When the videofile is selected instanciate the video in the player
    """
    self._player.value = self._videofile.value

def __process_frame(self, frame):
    """
    Do some processing to the frame and return the result frame
    """
    return frame

def __runEvent(self):
    """
    After setting the best parameters run the full algorithm
    """
    pass

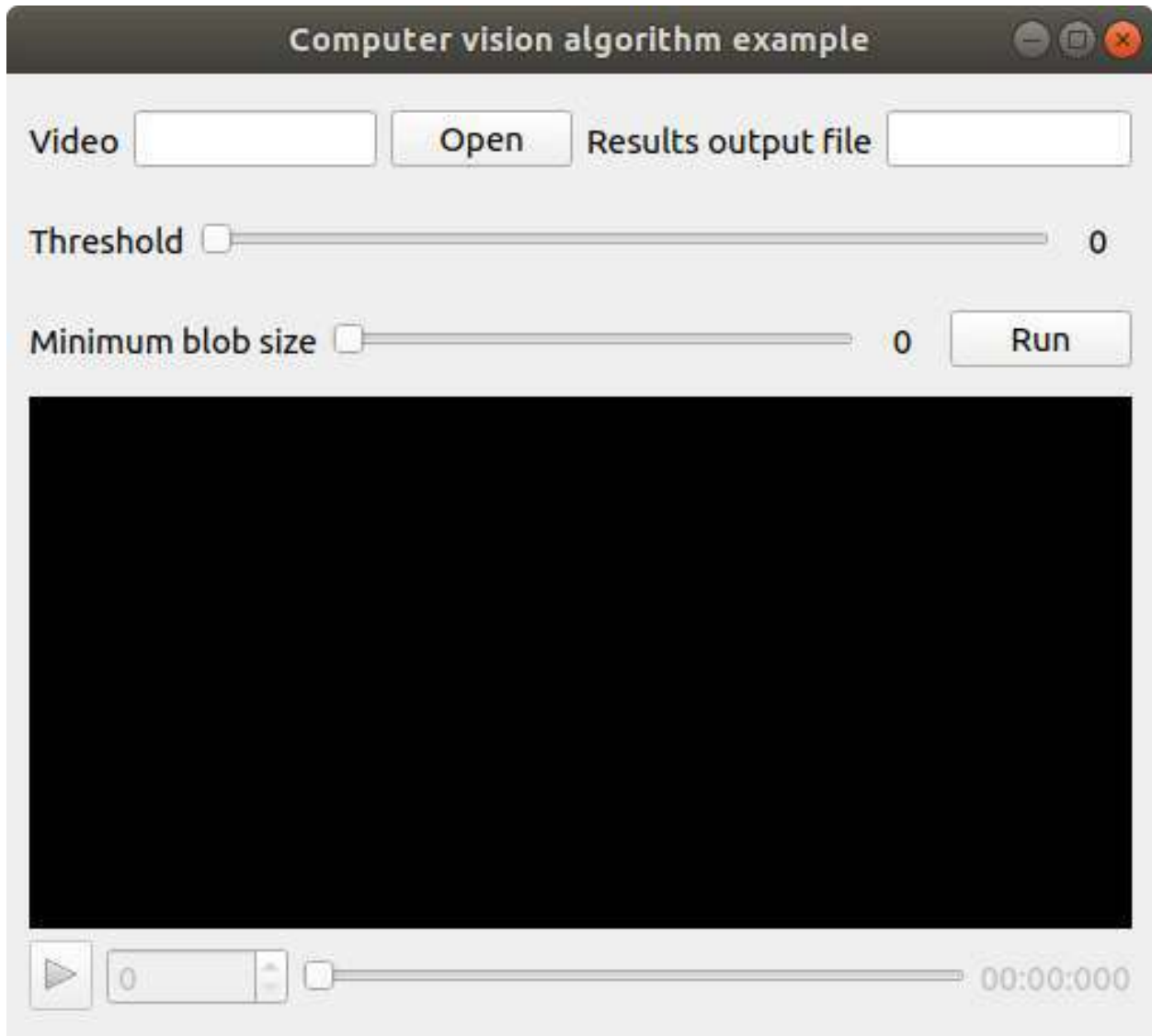
if __name__ == '__main__':

    from pyforms import start_app
    start_app(ComputerVisionAlgorithm)
```

Now execute in the terminal the next command:

```
$ python example.py
```

You will visualize the next result:



This page was based on the examples available at the github folder: [Tutorial - SimpleExamples](#)

4.1 Prepare the application class

Create the Python file that will store your applications.

Example: **SimpleExample.py**

4.1.1 Import the library

Import the pyforms library, the BaseWidget and the Controls classes that you will need:

```
import pyforms
from pyforms.basewidget import BaseWidget
from pyforms.controls import ControlText
from pyforms.controls import ControlButton
```

4.1.2 Create your application class.

This class should inherit from the class BaseWidget.

```
class SimpleExample1(BaseWidget):

    def __init__(self):
        super(SimpleExample1, self).__init__('Simple example 1')

        #Definition of the forms fields
        self._firstname = ControlText('First name', 'Default value')
        self._middlename = ControlText('Middle name')
```

(continues on next page)

(continued from previous page)

```
self._lastname = ControlText('Lastname name')
self._fullname = ControlText('Full name')
self._button = ControlButton('Press this button')

#Execute the application
if __name__ == "__main__": pyforms.start_app( SimpleExample1 )
```

If you run this file, it will produce the next window.

SimpleExample1



4.1.2.1 Add an action to the button

4.2 Create the action

Create the class function that will work as the button action.

```
def __buttonAction(self):
    """Button action event"""
    self._fullname.value = self._firstname.value + " " + self._middlename.value + "
↪"+self._lastname.value
```

4.3 Set the button action

Configure the button to execute your function when pressed. Inside the class constructor add the code:

```
#Define the button action
self._button.value = self.__buttonAction
```


The final code should look like:

```
import pyforms
from pyforms.basewidget import BaseWidget
from pyforms.controls import ControlText
from pyforms.controls import ControlButton

class SimpleExample1(BaseWidget):

    def __init__(self):
        super(SimpleExample1,self).__init__('Simple example 1')

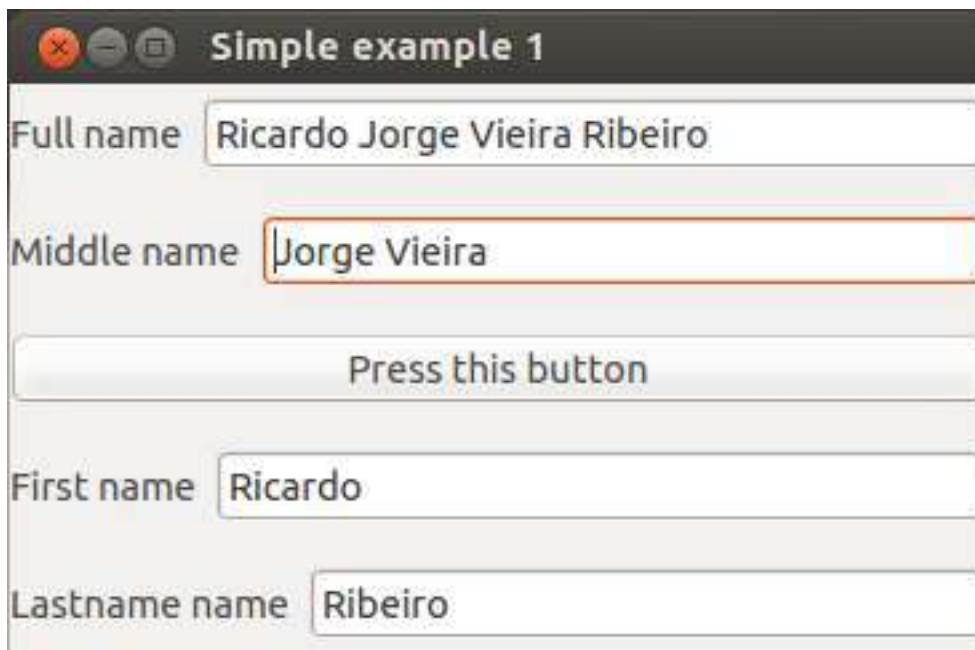
        #Definition of the forms fields
        self._firstname = ControlText('First name', 'Default value')
        self._middlename = ControlText('Middle name')
        self._lastname = ControlText('Lastname name')
        self._fullname = ControlText('Full name')
        self._button = ControlButton('Press this button')

        #Define the button action
        self._button.value = self.__buttonAction

    def __buttonAction(self):
        """Button action event"""
        self._fullname.value = self._firstname.value + " " + self._middlename.value + \
            " " + self._lastname.value

#Execute the application
if __name__ == "__main__": pyforms.start_app( SimpleExample1 )
```

The previous code produces the next window, after you had pressed the button:



Use the `BaseWidget.formset` variable to organize the Controls inside the Window. [Find here more details about the formset variable](#)

```

...

class SimpleExample1(BaseWidget):

    def __init__(self):
        ...

        #Define the organization of the forms
        self.formset = [ ('_firstname','_middlename','_lastname'), '_button', '_
↪fullname', ' ' ]
        #The ' ' is used to indicate that a empty space should be placed at the_
↪bottom of the window
        #If you remove the ' ' the forms will occupy the entire window

        ...

```

Result:



Try now:

```

self.formset = [ {
    'Tab1':['_firstname','|||','_middlename','|||','_lastname'],
    'Tab2':['_fullname']
},
'=',
(' ','_button', ' ')
]
#Use dictionaries for tabs
#Use the sign '=' for a vertical splitter
#Use the signs '||' for a horizontal splitter

```

Note: In the name of each tab use the format **a:Tab1** and **b:Tab2** to define the order of the tabs. Example:

```

self.formset = [ {
    'a:Tab1':['_firstname','|||','_middlename','|||','_lastname'],
    'b:Tab2':['_fullname']
}
]

```

To add a main menu to your application, first you need to define the functions that will work as the options actions.

```

...

class SimpleExample1(BaseWidget):
    ...

```

(continues on next page)

(continued from previous page)

```

def __openEvent(self):
    ...

def __saveEvent(self):
    ...

def __editEvent(self):
    ...

def __pastEvent(self):
    ...

```

After you just need to set the `BaseWidget.mainmenu` property inside your application class constructor as the example below.

```

...

class SimpleExample1(BaseWidget):

    def __init__(self):
        ...
        self.mainmenu = [
            { 'File': [
                {'Open': self.__openEvent},
                '-',
                {'Save': self.__saveEvent},
                {'Save as': self.__saveAsEvent}
            ]
            },
            { 'Edit': [
                {'Copy': self.__editEvent},
                {'Past': self.__pastEvent}
            ]
            }
        ]
        ...

```

Create the functions that will work as the popup menu options actions, as you have than in the main menu chapter. After use the functions `add_popup_menu_option` or `add_popup_sub_menu_option` to add a popup menu or a popup submenu to your Control.

[Find here more details about the functions `add_popup_menu_option` and `add_popup_sub_menu_option`.](<http://pyforms.readthedocs.org/en/latest/api-documentation/controls/#controlbase>)

```

...

class SimpleExample1(BaseWidget):

    def __init__(self):
        ...

        self._fullname.addPopupSubMenuOption('Path',
            {
                'Delete': self.__dummyEvent,
                'Edit': self.__dummyEvent,

```

(continues on next page)

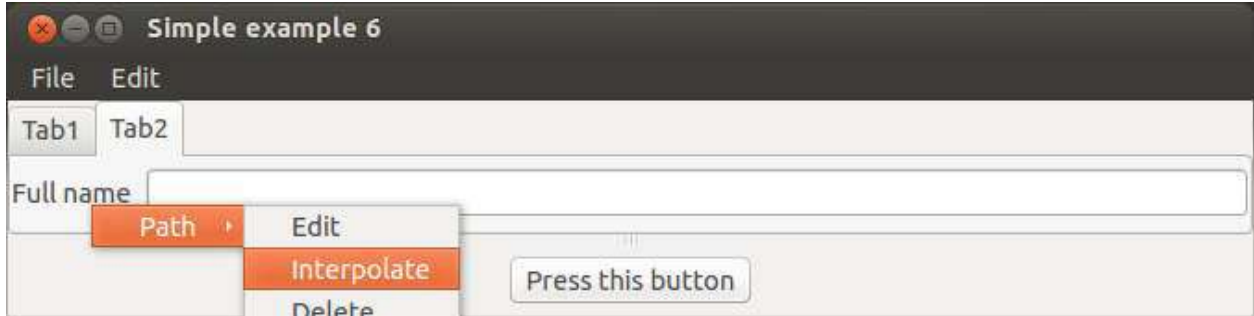
(continued from previous page)

```

        'Interpolate': self.__dummyEvent
    })
    ...

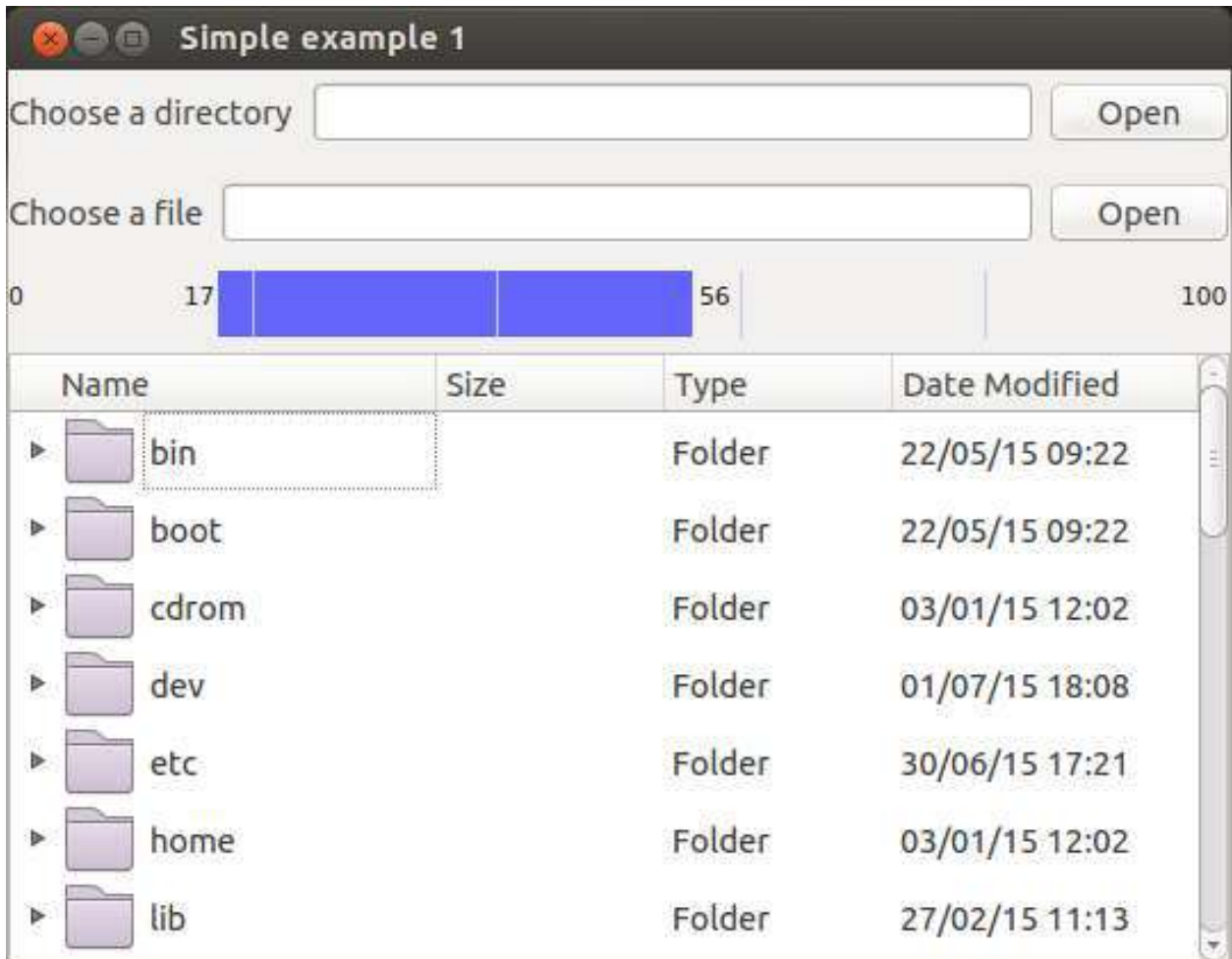
```

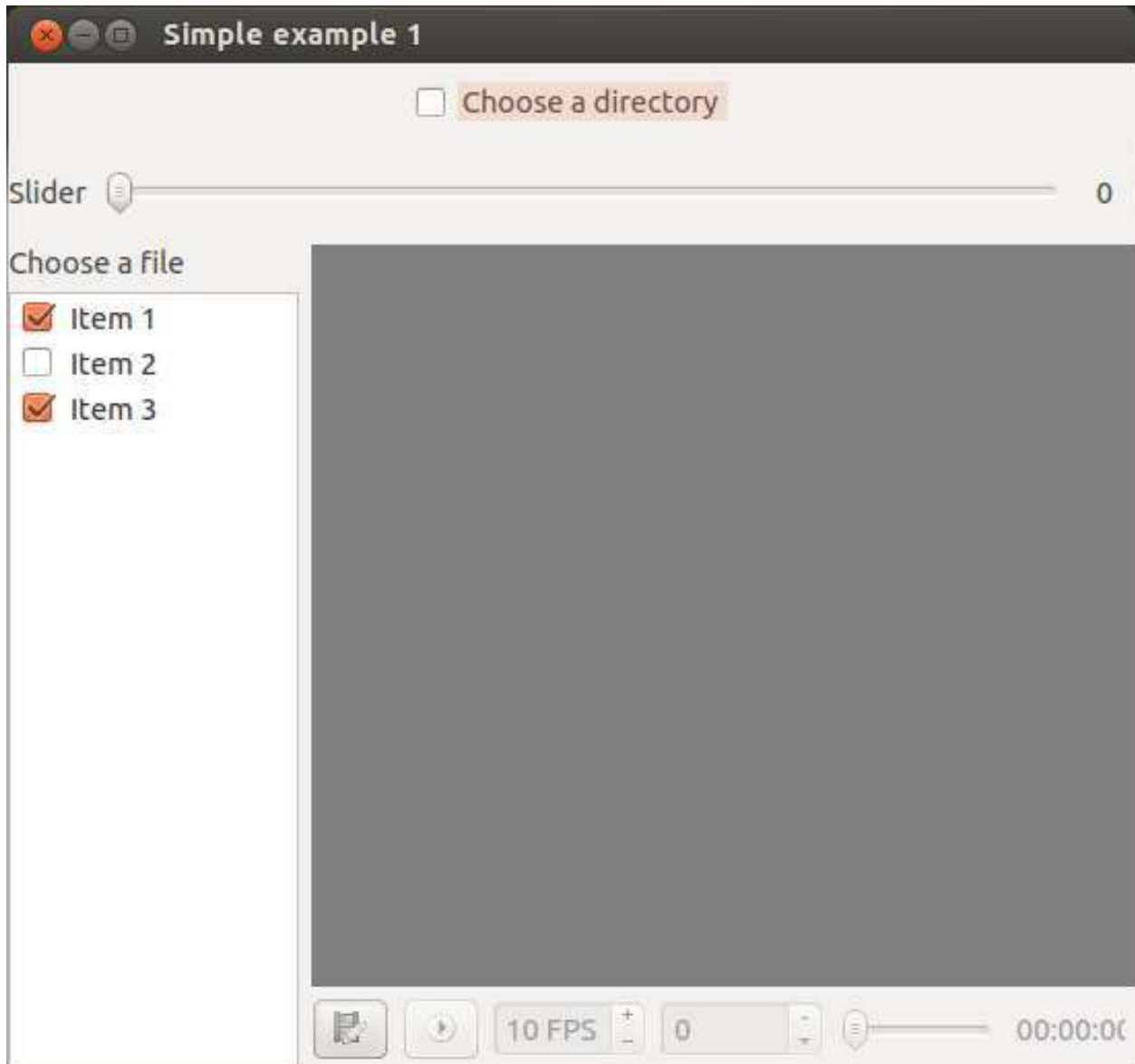
Result:

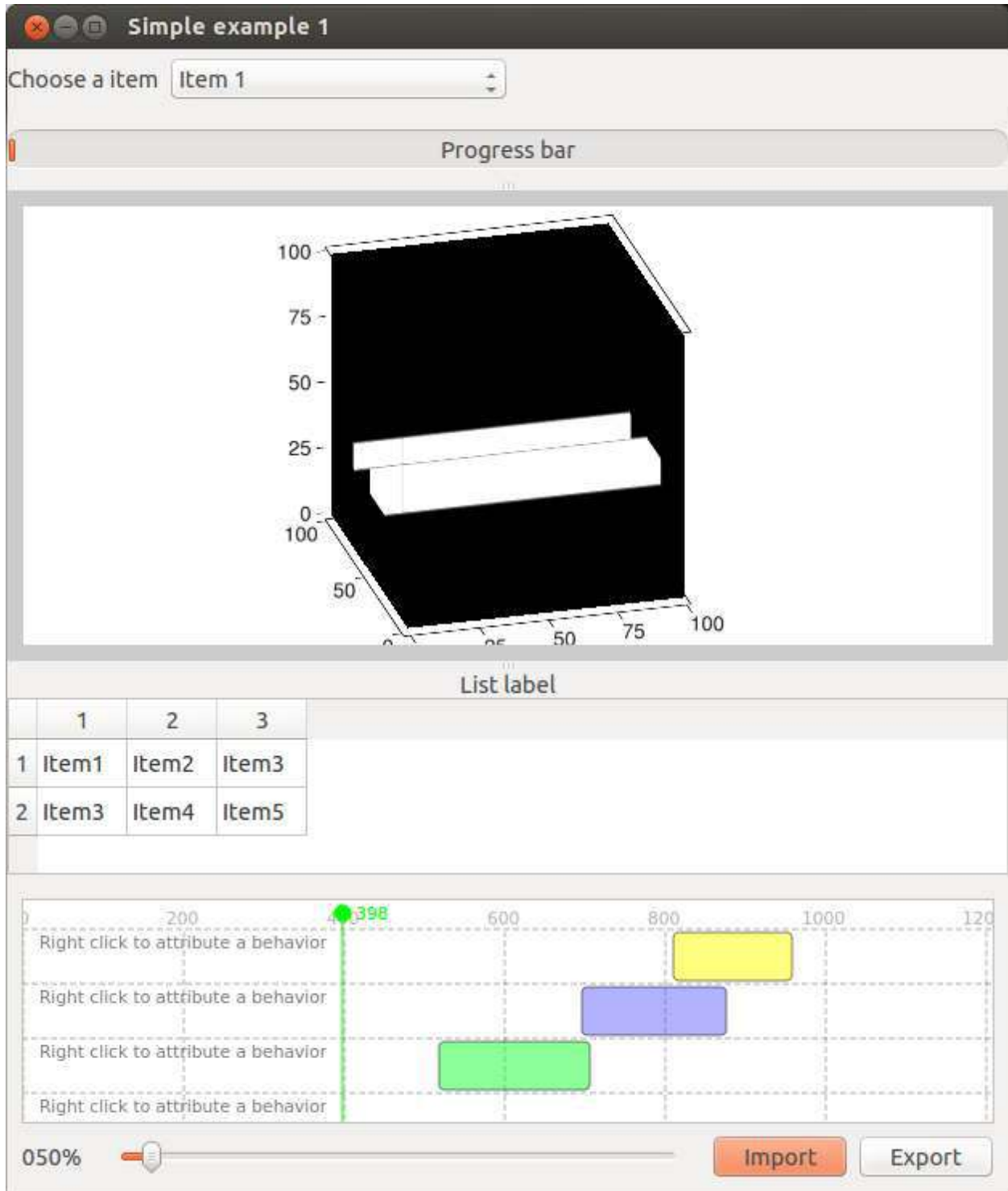


4.3.1 Move to the next chapter.

4.3.2 Find out what you can do with other Controls here.







This page was based on the examples available at the github folder: [Tutorial - Code Organization](#)

The application described on this page will allow us to add People details to a list.

5.1 Create the Model

Instead of starting by showing you how to develop the GUI I will suggest first how to modularize the code in a Model View Control (MVC) style.

First we will create our data model which may be used outside the GUI.

5.1.1 Data model

Start by creating the file Person.py where we will implement the model responsible for storing the a person information.

```
class Person(object):

    def __init__(self, firstName, middleName, lastName):
        self._firstName = firstName
        self._middleName = middleName
        self._lastName = lastName

    @property
    def fullName(self):
        return "{0} {1} {2}".format(self._firstName, self._middleName, self._lastName)
```

After, create the file People.py and implement the People class which will keep and manage the list of people.

```
import pickle

class People(object):
```

(continues on next page)

(continued from previous page)

```

def __init__(self):
    self._people = []

def addPerson(self, person):
    self._people.append(person)

def removePerson(self, index):
    return self._people.pop(index)

def save(self, filename):
    output = open(filename, 'wb')
    pickle.dump(self._people, output)

def load(self, filename):
    pkl_file = open(filename, 'rb')
    self._people = pickle.load(pkl_file)

```

5.1.2 Let's go for the GUI

To make our code modular and easy to navigate we will split the edition of the 2 Models in 2 different windows.

Implement the GUI to manage the Person Model.

Create the file PersonWindow.py and implement the window that will allow us the edit the Person Model. This window should inherit from the BaseWidget and Person classes.

```

import pyforms
from pyforms.basewidget import BaseWidget
from pyforms.controls import ControlText
from pyforms.controls import ControlButton
from Person import Person

class PersonWindow(Person, BaseWidget):

    def __init__(self):
        Person.__init__(self, '', '', '')
        BaseWidget.__init__(self, 'Person window')

        #Definition of the forms fields
        self._firstnameField = ControlText('First name')
        self._middlenameField = ControlText('Middle name')
        self._lastnameField = ControlText('Lastname name')
        self._fullnameField = ControlText('Full name')
        self._buttonField = ControlButton('Press this button')

        #Define the button action
        self._buttonField.value = self.__buttonAction

    def __buttonAction(self):
        self._firstName = self._firstnameField.value
        self._middleName = self._middlenameField.value
        self._lastName = self._lastnameField.value
        self._fullnameField.value = self.fullName

```

(continues on next page)

(continued from previous page)

```

    #In case the window has a parent
    if self.parent!=None: self.parent.addPerson(self)

#Execute the application
    if __name__ == "__main__": pyforms.start_app( PersonWindow )

```

Note: Test the window by executing the file.

5.1.3 Implement the GUI to manage the People model

Create the file PeopleWindow.py and implement the window that will allow us the manager the People Model. This window should inherit from the BaseWidget and People classes.

```

import pyforms
from pyforms.basewidget      import BaseWidget
from pyforms.controls        import ControlList
from People                  import People
from PersonWindow            import PersonWindow
from AddMenuFuntionality    import AddMenuFuntionality

class PeopleWindow(AddMenuFuntionality, People, BaseWidget):
    """
    This applications is a GUI implementation of the People class
    """

    def __init__(self):
        People.__init__(self)
        BaseWidget.__init__(self,'People window')

        #Definition of the forms fields
        self._peopleList      = ControlList('People',
            plusFunction        = self.__addPersonBtnAction,
            minusFunction       = self.__rmPersonBtnAction)

        self._peopleList.horizontalHeaders = ['First name', 'Middle name', 'Last name
↪']

    def addPerson(self, person):
        """
        Reimplement the addPerson function from People class to update the GUI
        everytime a new person is added.
        """
        super(PeopleWindow, self).addPerson(person)
        self._peopleList += [person._firstName, person._middleName, person._lastName]
        person.close() #After adding the person close the window

    def removePerson(self, index):
        """
        Reimplement the removePerson function from People class to update the GUI
        everytime a person is removed.
        """
        super(PeopleWindow, self).removePerson(index)
        self._peopleList -= index

    def __addPersonBtnAction(self):

```

(continues on next page)

(continued from previous page)

```

"""
Add person button event.
"""
# A new instance of the PersonWindow is opened and shown to the user.
win = PersonWindow()
win.parent = self
win.show()

def __rmPersonBtnAction(self):
    """
    Remove person button event
    """
    self.removePerson( self._peopleList.selected_row_index )

#Execute the application
if __name__ == "__main__":    pyforms.start_app( PeopleWindow )

```

The application will look like:



5.2 EmptyWidget Control

Instead of opening a new window everytime we want to add a new Person, we will change the Application to open the PersonWindow inside the PeopleWindow. For this we will use the ControlEmptyWidget.

```

from pyforms.controls      import ControlEmptyWidget
...

def __init__(self):
    ...
    self._panel = ControlEmptyWidget()

def __addPersonBtnAction(self):
    """
    Add person button event.

```

(continues on next page)

(continued from previous page)

```
"""
# A new instance of the PersonWindow is opened and shown to the user.
win = PersonWindow()
win.parent = self
self._panel.value = win
...

```

5.3 DockWidget Control

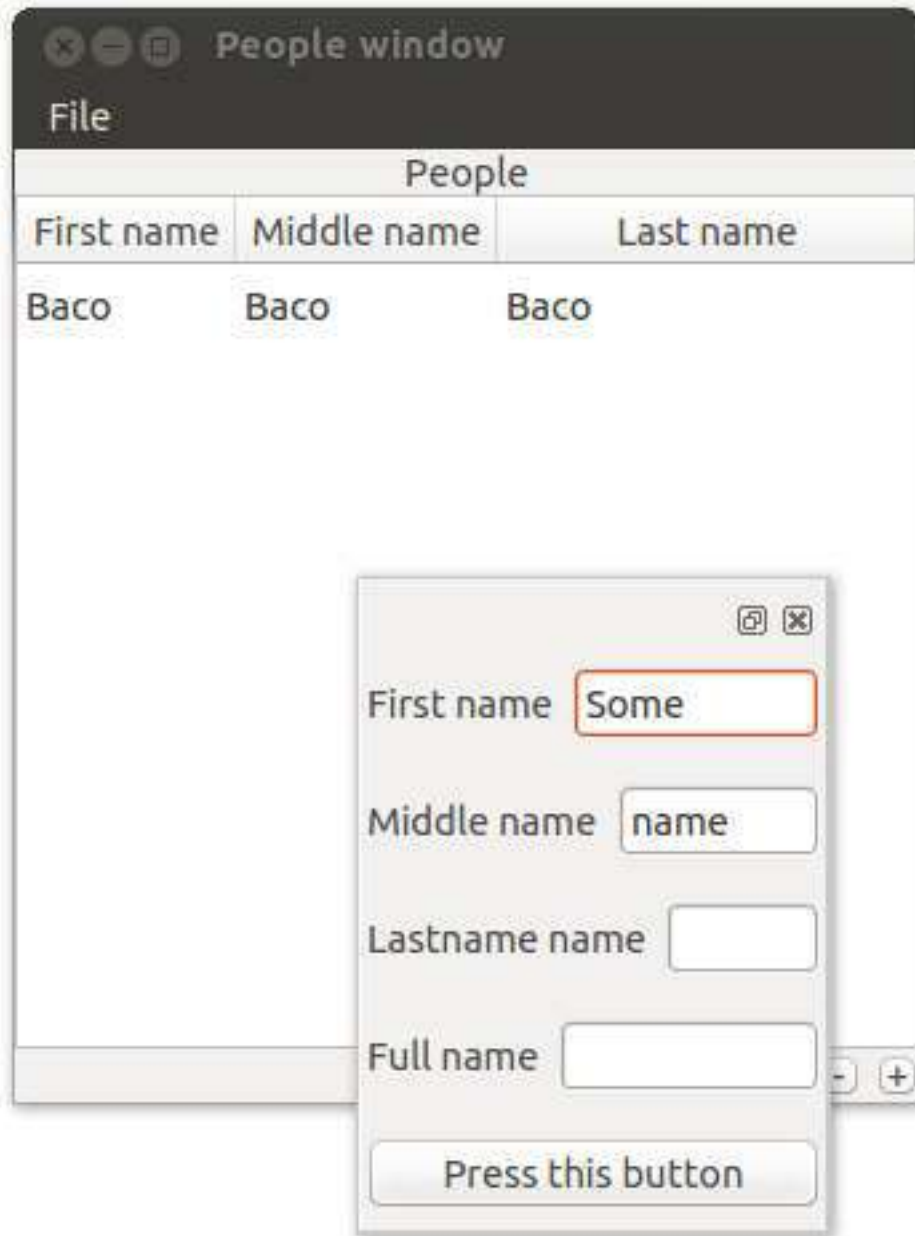
A DockWidget works like the EmptyWidget but can be detached or moved around the sides of the main Window.

```
from pyforms.controls import ControlDockWidget
...

def __init__(self):
    ...
    self._panel = ControlDockWidget()
...

```





CHAPTER 6

Mdi Applications

This page was based in the examples available on the github folder: [Tutorial - Mdi Application](#)

Style and layout with CSS

This page was based on the examples available at the github folder: [Tutorial - Code Organization](#)

PyForms takes advantage of the Qt framework to split the layout from the implementation of the functionalities. It is possible to configure the settings to import a stylesheet file which will change the application layout.

To do it, you need to add to your settings file the variable `PYFORMS_STYLESHEET` with the path to the css file you want to use:

```
PYFORMS_STYLESHEET = 'style.css'
```

You may would like also to adapt the layout for a specific operating system.

The next variables will allow to do this. You can complement the style configured in `PYFORMS_STYLESHEET` with a stylesheet for a specific operating system.

```
PYFORMS_STYLESHEET_DARWIN = 'style_darwin.css'  
PYFORMS_STYLESHEET_LINUX = 'style_linux.css'  
PYFORMS_STYLESHEET_WINDOWS = 'style_window.css'
```

Check the example: `style.css`

```
QMainWindow{  
    background-color: white;  
}  
  
QLabel{  
    min-width: 110px;  
}  
  
QLineEdit{  
    min-width: 200px;  
    border: 1px solid #CCC;  
    height: 30px;  
    padding-left: 10px;  
}
```

(continues on next page)

(continued from previous page)

```
QPushButton{
    background: #3498db;
    color: #ffffff;
    padding: 10px 20px 10px 20px;

    border-radius: 6px;
}

QPushButton:hover {
    background: #3cb0fd;
}

/*Use the # and the name of the variable to access to a specific the Control*/
#_firstnameField QLineEdit{
    color:red;
}
```



8.1 BaseWidget

8.1.1 Overview

The BaseWidget class is the base class of all pyforms applications.

8.1.2 API

class `pyforms_gui.basewidget.BaseWidget` (*args, **kwargs)

Bases: `PyQt5.QtWidgets.QFrame`

The class implements the most basic widget or window.

init_form()

Generate the module Form

generate_panel (formset)

Generate a panel for the module form with all the controls formset format example: [('_video', '_arenas', '_run'), {"Player":['_threshold', "_player", "=", "_results", "_query"], "Background image":[('_selectBackground', '_paintBackground'), '_image']}, "_progress"] tuple: will display the controls in the same horizontal line list: will display the controls in the same vertical line dict: will display the controls in a tab widget 'll': will split the controls in a horizontal line '=': will split the controls in a vertical line @param formset: Form configuration @type formset: list

show (self)

close (self) → bool

input_text (msg, title=", default=None)

input_double (msg, title=", default=0, min=-2147483647, max=2147483647, decimals=1)

input_int (msg, title=", default=0, min=-2147483647, max=2147483647)

question (msg, title=None, buttons=['no', 'yes'])

```
message (msg, title=None, msg_type=None)
success (msg, title=None)
info (msg, title=None)
warning (msg, title=None)
alert (msg, title=None)
critical (msg, title=None)
about (msg, title=None)
aboutQt (msg, title=None)
message_popup (msg, title="", buttons=None, handler=None, msg_type='success')
success_popup (msg, title="", buttons=None, handler=None)
info_popup (msg, title="", buttons=None, handler=None)
warning_popup (msg, title="", buttons=None, handler=None)
alert_popup (msg, title="", buttons=None, handler=None)
set_margin (margin)
controls
    Return all the form controls from the the module
form_has_loaded
parent_widget
form
title
formset
uid
closeEvent (self, QCloseEvent)
```

8.2 Controls

A form Control is a UI interface for the user to interact with the application.

Bellow we can find the description of all the Controls implemented in the PyForms library.

8.2.1 ControlBase

```
class pyforms_gui.controls.control_base.ControlBase (*args, **kwargs)
    Bases: object
```

All the Controls inherit from this Control, therefore you can find its functions and properties in all the other controls listed below.

Parameters

- **label** (*str*) – Control label. Default = ‘’.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.

- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

init_form()

Load the control UI and initiate all the events.

load_form (*data*, *path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

save_form (*data*, *path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

show()

Show the control

hide()

Hide the control

add_popup_submenu (*label*, *submenu=None*)

It returns a new sub popup menu. If submenu is open the menu is added to the main popup menu.

Parameters

- **label** (*str*) – Label of the option
- **submenu** (*QMenu*) – Parent submenu to which the option should be added. If no value is set, then the option will be added to the main popup menu.

add_popup_menu_option (*label*, *function_action=None*, *key=None*, *icon=None*, *menu=None*)

Add an option to the Control popup menu.

Parameters

- **label** (*str*) – Label of the option
- **function_action** (*function*) – The function that should be executed when the menu is selected.
- **key** (*str*) – Short key.
- **or str icon** (*QIcon*) – Icon.
- **submenu** (*QMenu*) – Parent submenu to which the option should be added. If no value is set, then the option will be added to the main popup menu.

```
control.add_popup_menu_option('option 0', function_action=self._do_something)
submenu1 = control.add_popup_submenu('menu 1')
submenu2 = control.add_popup_submenu('menu 2', submenu=submenu1)
control.add_popup_menu_option('option 1', function_action=self._do_something,
↵key='Control+Q', submenu=submenu2)
```

changed_event ()

Function called when ever the Control value is changed. The event function should return True if the data was saved with success.

about_to_show_contextmenu_event ()

Function called before the Control popup menu is opened.

enabled

Returns or set if the control is enable or disable.

value

This property returns or set what the control should manage or store.

name

This property returns or set the name of the control.

label

Returns or sets the label of the control.

parent

Returns or set the parent basewidget where the Control is.

visible

Return the control visibility.

help

Returns or set the tip box of the control.

readonly

Set and return the control readonly state.

form

Returns the QWidget of the control.

8.2.2 ControlBoundingSlider

```
class pyforms_gui.controls.control_boundingslider.ControlBoundingSlider(*args,
**kwargs)
```

Bases: `pyforms_gui.controls.control_base.ControlBase`

**Parameters**

- **default** (*tuple*) – The default value is a list containing in the first element the lower value and in the second element the upper value. Default = [20,40].
- **horizontal** (*bool*) – Flag indicating if the Bounding slider should be draw horizontally or vertically. Default = True.

- **show_spinboxes** (*bool*) – Show or hide the spinboxes. Default = True
- **minimum** (*float*) – Defines the minimum value that can be selected.
- **maximum** (*float*) – Defines the maximum value that can be selected.
- **convert_2_int** (*bool*) – Flag to define if the control should return floats or integers.

value

Sets and gets the control value. It should be a list or tuple of 2 values.

min

Sets and gets the minimum value possible.

max

Sets and gets the maximum value possible.

scale

Sets and gets the scale value.

convert_2_int

Flag to define if the control should return floats or integers.

8.2.3 ControlButton

```
class pyforms_gui.controls.control_button.ControlButton (*args, **kwargs)
```

```
    Bases: pyforms_gui.controls.control_base.ControlBase
```

```
    ...
```

Parameters

- **icon** (*str*) – Button icon
- **checkable** (*bool*) – Flag to set the button checkable.

click()

Trigger a click event

icon

Sets and gets the icon of the button.

value

Sets and gets the value of the Button. The value should be a function

checked

Sets and gets the button checked state

8.2.4 ControlCheckBox

```
class pyforms_gui.controls.control_checkbox.ControlCheckBox (*args, **kwargs)
```

```
    Bases: pyforms_gui.controls.control_base.ControlBase
```

Parameters

- **label** (*str*) – Control label. Default = “.
 - **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
-

- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

load_form (*data*, *path=None*)
Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

save_form (*data*, *path=None*)
Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

value
This property returns or set what the control should manage or store.

8.2.5 ControlCheckBoxList

class `pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList` (**args*,
***kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

save_form (*data={}*, *path=None*)
Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data*, *path=None*)
Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

item_changed (*item*)

clear ()

refresh ()

selection_changed_event ()

count

checked_indexes

value

This property returns or set what the control should manage or store.

selected_row_index

items

8.2.6 ControlCodeEditor

class `pyforms_gui.controls.control_codeeditor.ControlCodeEditor` (**args*,
***kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

Control that offers a code editor with pretty-print and line numbers and a save button

Parameters

- **label** –
- **default** –
- **helptext** –

ARROW_MARKER_NUM = 8

on_margin_clicked (*nmargin*, *nline*, *modifiers*)

On margin clicked, toggle marker for the line the margin was clicked on :param nmargin: :type nmargin:
:param nline: :type nline: :param modifiers: :type modifiers:

on_modification_changed ()

On modification change, re-enable save button

on_save_changes ()

On button save clicked, save changes made on the code editor to file

on_discart_changes ()

discart_event ()

key_pressed_event (*event*)

Override KeyPressed event as you like :param event: key event

is_modified

lexer

value

This property returns or set what the control should manage or store.

changed_event

Function called when ever the Control value is changed. The event function should return True if the data was saved with success.

8.2.7 ControlCombo

class `pyforms_gui.controls.control_combo.ControlCombo` (*args, **kwargs)

Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QWidget`

This class represents a wrapper to the combo box

clear ()

add_item (label, value=<class 'pyforms_gui.controls.control_combo.ValueNotSet'>)

get_item_index_by_name (item_name)

Returns the index of the item containing the given name :param item_name: item name in combo box
:type item_name: string

count ()

show ()

Show the control

hide ()

Hide the control

current_index_changed_event (index)

Called when the user chooses an item in the combobox and the selected choice is different from the last one selected. @index: item's index

activated_event (index)

Called when the user chooses an item in the combobox. Note that this signal happens even when the choice is not changed @index: item's index

highlighted_event (index)

edittext_changed_event (text)

form

Returns the QWidget of the control.

current_index

values

items

value

This property returns or set what the control should manage or store.

text

label

Returns or sets the label of the control.

8.2.8 ControlDir

class `pyforms_gui.controls.control_dir.ControlDir` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = ‘’.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

click ()

finishEditing ()

Function called when the lineEdit widget is edited

value

This property returns or set what the control should manage or store.

label

Returns or sets the label of the control.

8.2.9 ControlDockWidget

class `pyforms_gui.controls.control_dockwidget.ControlDockWidget` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_emptywidget.ControlEmptyWidget`

SIDE_LEFT = 'left'

SIDE_RIGHT = 'right'

SIDE_TOP = 'top'

SIDE_BOTTOM = 'bottom'

SIDE_DETACHED = 'detached'

label

Returns or sets the label of the control.

save_form (*data, path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

show()
Show the control

hide()
Hide the control

8.2.10 ControlEmptyWidget

class `pyforms_gui.controls.control_emptywidget.ControlEmptyWidget` (**args*,
***kwargs*)
Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QWidget`

value
This property returns or set what the control should manage or store.

form
Returns the QWidget of the control.

save_form (*data*, *path=None*)
Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data*, *path=None*)
Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

show()
Show the control

hide()
Hide the control

8.2.11 ControlFile

class `pyforms_gui.controls.control_file.ControlFile` (**args*, ***kwargs*)
Bases: `pyforms_gui.controls.control_base.ControlBase`

finishEditing()
Function called when the lineEdit widget is edited

click()

value

This property returns or set what the control should manage or store.

label

Returns or sets the label of the control.

8.2.12 ControlFilesTree

class `pyforms_gui.controls.control_filestree.ControlFilesTree` (*args, **kwargs)
 Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

value

This property returns or set what the control should manage or store.

8.2.13 ControllImage

class `pyforms_gui.controls.control_image.ControlImage` (*args, **kwargs)
 Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

save_form (*data*, *path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.

- **path** (*str*) – Optional parameter that can be used to load the data.

value

This property returns or set what the control should manage or store.

double_click_event

click_event

drag_event

end_drag_event

key_release_event

8.2.14 ControlLabel

class `pyforms_gui.controls.control_label.ControlLabel` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = ‘’.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

load_form (*data, path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

save_form (*data, path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

selectable

form

Returns the QWidget of the control.

value

This property returns or set what the control should manage or store.

8.2.15 Controllist

class `pyforms_gui.controls.control_list.ControlList` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QWidget`

This class represents a wrapper to the table widget It allows to implement a list view

CELL_VALUE_BEFORE_CHANGE = `None`

clear (*headers=False*)

save_form (*data, path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data, path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

set_value (*column, row, value*)

get_value (*column, row*)

resize_rows_contents ()

get_currentrow_value ()

get_cell (*column, row*)

set_sorting_enabled (*value*)

Enable or disable columns sorting

Parameters value (*bool*) – True to enable sorting, False otherwise

data_changed_event (*row, col, item*)

item_selection_changed_event ()

current_cell_changed_event (*next_row, next_col, previous_row, previous_col*)

current_item_changed_event (*current, previous*)

cell_double_clicked_event (*row, column*)

horizontal_headers

word_wrap

readonly

Set and return the control readonly state.

select_entire_row

rows_count

columns_count

value

This property returns or set what the control should manage or store.

selected_rows_indexes

selected_row_index

label

Returns or sets the label of the control.

form

Returns the QWidget of the control.

icon_size

autoscroll

resizecolumns

tableWidgetCellChanged (*nextRow, nextCol, previousRow, previousCol*)

tableWidgetItemChanged (*current, previous*)

tableWidgetItemSelectionChanged ()

tableWidgetCellDoubleClicked (*row, column*)

(From PyQt) This signal is emitted whenever a cell in the table is double clicked. The row and column specified is the cell that was double clicked.

Besides firing this signal, we save the current value, in case the user needs to know the old value. :param row: :param column: :return:

empty_signal (**args, **kwargs*)

Use this function if you want to disconnect a signal temporarily

8.2.16 ControlPlayer

class `pyforms_gui.controls.control_player.control_player.ControlPlayer` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QFrame`

play ()

stop ()

hide ()

Hide the control

show ()

Show the control

refresh ()

save_form (*data, path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data, path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

process_frame_event (*frame*)

double_click_event

click_event

drag_event

end_drag_event

key_press_event

key_release_event

next_frame_step

view_in_3D

video_index

max

frame

fps

Return the video frames per second

help_text

form

Returns the QWidget of the control.

frame_width

frame_height

is_playing

value

This property returns or set what the control should manage or store.

call_next_frame (*update_slider=True, update_number=True, increment_frame=True*)

videoPlay_clicked()

Slot for Play/Pause functionality.

convertFrameToTime (*totalMilliseconds*)

videoProgress_valueChanged()

videoProgress_sliderReleased()

video_frames_value_changed (*pos*)

jump_forward()

jump_backward()

8.2.17 ControlMatplotlib

```
class pyforms_gui.controls.control_matplotlib.ControlMatplotlib (*args,  
                                                                **kwargs)  
    Bases: pyforms_gui.controls.control_base.ControlBase, PyQt5.QtWidgets.QWidget  
value  
    This property returns or set what the control should manage or store.  
draw ()  
on_draw (figure)  
    Redraws the figure  
fig  
form  
    Returns the QWidget of the control.
```

8.2.18 ControlMdiArea

```
class pyforms_gui.controls.control_mdiarea.ControlMdiArea (*args, **kwargs)  
    Bases: pyforms_gui.controls.control_base.ControlBase, PyQt5.QtWidgets.QMdiArea  
  
    The ControlMdiArea wraps a QMdiArea widget which provides an area in which MDI windows are displayed.  
  
    show_subwin_close_button  
  
    label  
    Returns or sets the label of the control.  
  
    form  
    Returns the QWidget of the control.
```

8.2.19 ControlNumber

```
class pyforms_gui.controls.control_number.ControlNumber (*args, **kwargs)  
    Bases: pyforms_gui.controls.control_base.ControlBase  
  
    Parameters  
    • minimum (int) – Minimum value.  
    • maximum (int) – Maximum value.  
    • default (float) – Set the value. Default = 0.  
    • decimals (int) – Decimals precision.  
    • step (float) – Step jump value.  
  
    update_event (value)  
  
    label  
    Returns or sets the label of the control.
```

value

This property returns or set what the control should manage or store.

min**max****decimals****step**

8.2.20 ControlPassword

class `pyforms_gui.controls.control_password.ControlPassword(*args, **kwargs)`

Bases: `pyforms_gui.controls.control_text.ControlText`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

8.2.21 ControlOpenGL

class `pyforms_gui.controls.control_opengl.ControlOpenGL(*args, **kwargs)`

Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

repaint ()

reset_zoom_and_rotation ()

value

This property returns or set what the control should manage or store.

`clear_color`
`width`
`height`

8.2.22 ControlProgress

class `pyforms_gui.controls.control_progress.ControlProgress` (*args, **kwargs)

Bases: `pyforms_gui.controls.control_base.ControlBase`

label

Returns or sets the label of the control.

value

This property returns or set what the control should manage or store.

min

max

8.2.23 ControlSlider

class `pyforms_gui.controls.control_slider.ControlSlider` (*args, **kwargs)

Bases: `pyforms_gui.controls.control_base.ControlBase`

valueChanged (*value*)

load_form (*data*, *path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

save_form (*data*, *path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

value

This property returns or set what the control should manage or store.

min

max

8.2.24 ControlText

class `pyforms_gui.controls.control_text.ControlText` (*args, **kwargs)
 Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

finishEditing ()

Function called when the lineEdit widget is edited

key_pressed_event (*evt*)

value

This property returns or set what the control should manage or store.

label

Returns or sets the label of the control.

readonly

Set and return the control readonly state.

8.2.25 ControlTextArea

class `pyforms_gui.controls.control_textarea.ControlTextArea` (*args, **kwargs)
 Bases: `pyforms_gui.controls.control_base.ControlBase`

finishEditing ()

Function called when the lineEdit widget is edited

value

This property returns or set what the control should manage or store.

readonly

Set and return the control readonly state.

autoscroll

8.2.26 ControlToolBox

class `pyforms_gui.controls.control_toolbox.ControlToolBox` (*args, **kwargs)
 Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “”.

- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

value

This property returns or set what the control should manage or store.

set_item_enabled (*index, enabled*)

Enable or disable an item

is_item_enabled (*index*)

Check if an item is enabled or disabled

8.2.27 ControlToolButton

class `pyforms_gui.controls.control_toolbutton.ControlToolButton` (**args, **kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

click ()

load_form (*data, path=None*)

Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

save_form (*data, path=None*)

Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

label

Returns or sets the label of the control.

icon

value

This property returns or set what the control should manage or store.

checked

8.2.28 ControlTree

class `pyforms_gui.controls.control_tree.ControlTree` (**args, **kwargs*)
 Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QTreeWidgetItem`

This class represents a wrapper to the `QTreeWidgetItem`

save_form (*data, path=None*)
 Save a value of the control to a dictionary.

Parameters

- **data** (*dict*) – Dictionary where the control value should be saved.
- **path** (*str*) – Optional parameter that can be used to load the data.

load_form (*data, path=None*)
 Loads the value of the control.

Parameters

- **data** (*dict*) – It is a dictionary with the required information to load the control.
- **path** (*str*) – Optional parameter that can be used to save the data.

add_popup_menu_option (*label=*”, *function_action=None, key=None, item=None, icon=None, submenu=None*)
 Add an option to the Control popup menu @param label: label of the option. @param function_action: function called when the option is selected. @param key: shortcut key @param key: shortcut key

clear (*self*)

expand_item (*item, expand=True, parents=True*)

create_child (*name, parent=None, icon=None*)

Create a new child for to the parent item. If the parent is None it add to the root.

item_changed_event (*item*)

item_selection_changed_event ()

item_double_clicked_event (*item*)

key_press_event (*event*)

rows_inserted_event (*parent, start, end*)

This event is called every time a new row is added to the tree

show_header

selected_rows_indexes

selected_row_index

selected_item

form

Returns the `QWidget` of the control.

value

This property returns or set what the control should manage or store.

icon_size

rowsInserted (*self, QModelIndex, int, int*)

selectionChanged (*self, QItemSelection, QItemSelection*)

keyPressEvent (*self*, *QKeyEvent*)

about_to_show_contextmenu_event ()
Function called before open the Control popup menu

clone_item (*parent*, *item*, *copy_function=None*)

clone_tree (*tree*, *copy_function=None*)

8.2.29 ControlTreeView

class `pyforms_gui.controls.control_treeview.ControlTreeView` (**args*, ***kwargs*)
Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QTreeView`

default_width = None

item_selection_changed_event (*selected*, *deselected*)

item_double_clicked_event (*evt*)

selected_row_index

selected_item

value
This property returns or set what the control should manage or store.

form
Returns the QWidget of the control.

8.2.30 ControlVisVis

class `pyforms_gui.controls.control_visvis.ControlVisVis` (**args*, ***kwargs*)
Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = “.”
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

refresh ()

paint (*visvis*)

legend

show_grid

title
xlabel
ylabel
zlabel
value

This property returns or set what the control should manage or store.

8.2.31 ControlVisVisVolume

class `pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume` (**args*,
***kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`

Parameters

- **label** (*str*) – Control label. Default = ‘’.
- **helptext** (*str*) – Text shown when the mouse is over the control. Default = None.
- **default** (*str*) – Initial value of the control. Default = None.
- **visible** (*bool*) – Flag to set the control visible or hidden. Default = True.
- **enabled** (*bool*) – Flag to set the control enabled or Disabled. Default = True.
- **readonly** (*bool*) – Flag to set the control readonly. Default = False.
- **changed_event** (*function*) – Function to call whenever the control value is updated. Default = None.

color_map

refresh ()

value

This property returns or set what the control should manage or store.

colors_limits

visvis

8.2.32 ControlWeb

class `pyforms_gui.controls.control_web.ControlWeb` (**args*, ***kwargs*)

Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWebEngineWidgets.QWebEngineView`

load_finished_event (*ok*)

value

This property returns or set what the control should manage or store.

html

form

Returns the QWidget of the control.

8.2.33 ControlEventTimeline

```
class pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline
```

Bases: *pyforms_gui.controls.control_base.ControlBase*, *PyQt5.QtWidgets.QWidget*

Timeline events editor

```
rename_graph (graph_index, newname)
```

```
add_period (value, row=0, color=None)
```

Parameters

- **value** –
- **row** –
- **color** –

Returns

```
add_graph (name, data)
```

Parameters

- **name** –
- **data** –

Returns

```
import_graph (filename, frame_col=0, val_col=1)
```

Parameters

- **filename** –
- **frame_col** –
- **val_col** –

Returns

```
import_graph_file (filename, separator=';', ignore_rows=0)
```

Parameters

- **filename** –
- **separator** –
- **ignore_rows** –

Returns

```
show_graphs_properties ()
```

```
import_csv (csvfile)
```

Parameters csvfile –

```
export_csv_file (filename)
```

```

import_csv_file (filename)
mouse_moveover_timeline_event (event)
pointer_changed_event
value
    This property returns or set what the control should manage or store.
max
form
    Returns the QWidget of the control.
rows
graphs
key_release_event
about_to_show_contextmenu_event ()
    Function called before the Control popup menu is opened.
clean ()

```

8.2.34 ControlEventsGraph

```

class pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph (label,
de-
fault:
min=
max=
**kw

```

Bases: `pyforms_gui.controls.control_base.ControlBase`, `PyQt5.QtWidgets.QWidget`

Timeline events editor

Parameters

- **label** –
- **default** –
- **min** –
- **max** –
- **kwargs** –

add_track (*title=None*)

Parameters **title** –

add_event (*begin, end, title=""*, *track=0*, *color='#FFFF00'*)

Parameters

- **begin** –
- **end** –
- **title** –
- **track** –

- `color` –

Returns

`get_export_filename()`

`export_csv(filename)`

Export annotations to a file. :param str filename: filename to open

`repaint()`

`changed_event`

Function called when ever the Control value is changed. The event function should return True if the data was saved with success.

`value`

This property returns or set what the control should manage or store.

`form`

Returns the QWidget of the control.

`tracks`

`tracks_height`

`scale`

8.3 Settings

Pyforms is using the confapp library to manage it settings. Here it is described some of the settings of the library.

8.3.1 General configurations

`PYFORMS_MODE = os.environ.get('PYFORMS_MODE', 'GUI')`

It defines the mode that the pyforms should run. Currently pyforms can run as **GUI** or **TERMINAL** mode.

`PYFORMS_LOG_HANDLER_FILE_LEVEL = logging.DEBUG`

Logging level.

`PYFORMS_LOG_HANDLER_CONSOLE_LEVEL = logging.INFO`

Logging level.

8.3.2 GUI layout

`PYFORMS_STYLESHEET = None`

Path to the stylesheet file of the application.

`PYFORMS_STYLESHEET_DARWIN = None`

`PYFORMS_STYLESHEET_LINUX = None`

`PYFORMS_STYLESHEET_WINDOWS = None`

Frequently it is necessary to adapt the layout of an application for each operating system. These variables allow you to do just that. For each operating system you can define a stylesheet that will complement the default stylesheet for a specific OS.

8.3.3 Controls

PYFORMS_CONTROL_CODE_EDITOR_DEFAULT_FONT_SIZE = '12'

PYFORMS_CONTROL_EVENTS_GRAPH_DEFAULT_SCALE = 1

PYFORMS_CONTROLPLAYER_FONT = 9

CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

p

`pybpod_web.basewidget.BaseWidget`, 31
`pyforms_gui.basewidget`, 31
`pyforms_gui.controls`, 32

A

- about () (*pyforms_gui.basewidget.BaseWidget* method), 32
- about_to_show_contextmenu_event () (*pyforms_gui.controls.control_base.ControlBase* method), 34
- about_to_show_contextmenu_event () (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline* method), 55
- about_to_show_contextmenu_event () (*pyforms_gui.controls.control_tree.ControlTree* method), 52
- aboutQt () (*pyforms_gui.basewidget.BaseWidget* method), 32
- activated_event () (*pyforms_gui.controls.control_combo.ControlCombo* method), 38
- add_event () (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph* method), 55
- add_graph () (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline* method), 54
- add_item () (*pyforms_gui.controls.control_combo.ControlCombo* method), 38
- add_period () (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline* method), 54
- add_popup_menu_option () (*pyforms_gui.controls.control_base.ControlBase* method), 33
- add_popup_menu_option () (*pyforms_gui.controls.control_tree.ControlTree* method), 51
- add_popup_submenu () (*pyforms_gui.controls.control_base.ControlBase* method), 33
- add_track () (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph* method), 55
- alert () (*pyforms_gui.basewidget.BaseWidget* method), 32
- alert_popup () (*pyforms_gui.basewidget.BaseWidget* method), 32
- ARROW_MARKER_NUM (*pyforms_gui.controls.control_codeeditor.ControlCodeEditor* attribute), 37
- autoscroll (*pyforms_gui.controls.control_list.ControlList* attribute), 44
- autoscroll (*pyforms_gui.controls.control_textarea.ControlTextArea* attribute), 49

B

BaseWidget (class in *pyforms_gui.basewidget*), 31

C

- call_next_frame () (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45
- changed_event (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph* attribute), 56
- changed_event () (*pyforms_gui.controls.control_base.ControlBase* method), 34
- checked (*pyforms_gui.controls.control_button.ControlButton* attribute), 35
- checked (*pyforms_gui.controls.control_toolbutton.ControlToolButton* attribute), 50
- checked_indexes (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* attribute), 37
- CELL_VALUE_BEFORE_CHANGE (*pyforms_gui.controls.control_list.ControlList* attribute), 43

`clean()` (`pyforms_gui.controls.control_event_timeline.ControlEventTimeline` method), 55
`clear()` (`pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList` method), 37
`clear()` (`pyforms_gui.controls.control_combo.ControlCombo` method), 38
`clear()` (`pyforms_gui.controls.control_list.ControlList` method), 43
`clear()` (`pyforms_gui.controls.control_tree.ControlTree` method), 51
`clear_color` (`pyforms_gui.controls.control_opengl.ControlOpenGL` attribute), 48
`click()` (`pyforms_gui.controls.control_button.ControlButton` method), 35
`click()` (`pyforms_gui.controls.control_dir.ControlDir` method), 39
`click()` (`pyforms_gui.controls.control_file.ControlFile` method), 40
`click()` (`pyforms_gui.controls.control_toolbutton.ControlToolButton` method), 50
`click_event` (`pyforms_gui.controls.control_image.ControlImage` attribute), 42
`click_event` (`pyforms_gui.controls.control_player.control_player.ControlPlayer` attribute), 45
`clone_item()` (`pyforms_gui.controls.control_tree.ControlTree` method), 52
`clone_tree()` (`pyforms_gui.controls.control_tree.ControlTree` method), 52
`close()` (`pyforms_gui.basewidget.BaseWidget` method), 31
`closeEvent()` (`pyforms_gui.basewidget.BaseWidget` method), 32
`color_map` (`pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume` attribute), 53
`colors_limits` (`pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume` attribute), 53
`columns_count` (`pyforms_gui.controls.control_list.ControlList` attribute), 43
`ControlBase` (class in `pyforms_gui.controls.control_base`), 32
`ControlBoundingSlider` (class in `pyforms_gui.controls.control_boundingslider`), 34
`ControlButton` (class in `pyforms_gui.controls.control_button`), 35
`ControlCheckBox` (class in `pyforms_gui.controls.control_checkbox`), 35
`ControlCheckBoxList` (class in `pyforms_gui.controls.control_checkboxlist`), 36
`ControlCodeEditor` (class in `pyforms_gui.controls.control_codeeditor`),
`ControlCombo` (class in `pyforms_gui.controls.control_combo`), 38
`ControlDir` (class in `pyforms_gui.controls.control_dir`), 39
`ControlDockWidget` (class in `pyforms_gui.controls.control_dockwidget`), 39
`ControlEmptyWidget` (class in `pyforms_gui.controls.control_emptywidget`), 39
`ControlOpenGL` (class in `pyforms_gui.controls.control_opengl`), 47
`ControlEventsGraph` (class in `pyforms_gui.controls.control_events_graph.control_eventsgraph`), 55
`ControlEventTimeline` (class in `pyforms_gui.controls.control_event_timeline.control_eventtimeline`), 54
`ControlFile` (class in `pyforms_gui.controls.control_file`), 40
`ControlFilesTree` (class in `pyforms_gui.controls.control_filetree`), 41
`ControlImage` (class in `pyforms_gui.controls.control_image`), 41
`ControlLabel` (class in `pyforms_gui.controls.control_label`), 42
`ControlList` (class in `pyforms_gui.controls.control_list`), 43
`ControlMatplotlib` (class in `pyforms_gui.controls.control_matplotlib`), 46
`ControlMdiArea` (class in `pyforms_gui.controls.control_mdiarea`), 46
`ControlNumber` (class in `pyforms_gui.controls.control_number`), 46
`ControlOpenGL` (class in `pyforms_gui.controls.control_opengl`), 47
`ControlPassword` (class in `pyforms_gui.controls.control_password`), 47
`ControlPlayer` (class in `pyforms_gui.controls.control_player.control_player`), 44
`ControlProgress` (class in `pyforms_gui.controls.control_progress`), 48
`controls` (`pyforms_gui.basewidget.BaseWidget` attribute), 32
`ControlSlider` (class in `pyforms_gui.controls.control_slider`), 48
`ControlText` (class in `pyforms_gui.controls.control_text`), 49
`ControlTextArea` (class in `pyforms_gui.controls.control_textarea`), 49
`ControlToolBox` (class in `pyforms_gui.controls.control_toolbox`), 49
`ControlToolButton` (class in `pyforms_gui.controls.control_toolbutton`), 50

ControlTree (class in *pyforms_gui.controls.control_tree*), 51

ControlTreeView (class in *pyforms_gui.controls.control_treeview*), 52

ControlVisVis (class in *pyforms_gui.controls.control_visvis*), 52

ControlVisVisVolume (class in *pyforms_gui.controls.control_visvisvolume*), 53

ControlWeb (class in *pyforms_gui.controls.control_web*), 53

convert_2_int (*pyforms_gui.controls.control_boundingslider.ControlBoundingSlider* attribute), 35

convertFrameToTime () (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45

count (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* attribute), 37

count () (*pyforms_gui.controls.control_combo.ControlCombo* method), 38

create_child () (*pyforms_gui.controls.control_tree.ControlTree* method), 51

critical () (*pyforms_gui.basewidget.BaseWidget* method), 32

current_cell_changed_event () (*pyforms_gui.controls.control_list.ControlList* method), 43

current_index (*pyforms_gui.controls.control_combo.ControlCombo* attribute), 38

current_index_changed_event () (*pyforms_gui.controls.control_combo.ControlCombo* method), 38

current_item_changed_event () (*pyforms_gui.controls.control_list.ControlList* method), 43

D

data_changed_event () (*pyforms_gui.controls.control_list.ControlList* method), 43

decimals (*pyforms_gui.controls.control_number.ControlNumber* attribute), 47

default_width (*pyforms_gui.controls.control_treeview.ControlTreeView* attribute), 52

discart_event () (*pyforms_gui.controls.control_codeeditor.ControlCodeEditor* method), 37

double_click_event (*pyforms_gui.controls.control_image.ControlImage* attribute), 42

double_click_event (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

drag_event (*pyforms_gui.controls.control_image.ControlImage* attribute), 42

drag_event (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

draw () (*pyforms_gui.controls.control_matplotlib.ControlMatplotlib* method), 46

E

edittext_changed_event () (*pyforms_gui.controls.control_combo.ControlCombo* method), 38

empty_signal () (*pyforms_gui.controls.control_list.ControlList* method), 44

enabled (*pyforms_gui.controls.control_base.ControlBase* attribute), 34

end_drag_event (*pyforms_gui.controls.control_image.ControlImage* attribute), 42

end_drag_event (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

expand_item () (*pyforms_gui.controls.control_tree.ControlTree* method), 51

export_csv () (*pyforms_gui.controls.control_events_graph.control_events_graph* method), 56

export_csv_file () (*pyforms_gui.controls.control_event_timeline.control_eventtimeline* method), 54

F

fig (*pyforms_gui.controls.control_matplotlib.ControlMatplotlib* attribute), 46

finishEditing () (*pyforms_gui.controls.control_dir.ControlDir* method), 39

finishEditing () (*pyforms_gui.controls.control_file.ControlFile* method), 40

finishEditing () (*pyforms_gui.controls.control_text.ControlText* method), 49

finishEditing () (*pyforms_gui.controls.control_textarea.ControlTextArea* method), 49

form (*pyforms_gui.basewidget.BaseWidget* attribute), 32

form (*pyforms_gui.controls.control_base.ControlBase* attribute), 34

form (*pyforms_gui.controls.control_combo.ControlCombo* *get_value()* (*pyforms_gui.controls.control_list.ControlList* *attribute*), 38 *method*), 43

form (*pyforms_gui.controls.control_emptywidget.ControlEmptyWidget* *pyforms_gui.controls.control_event_timeline.control_eventtimeline* *attribute*), 40 *attribute*), 55

form (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline* *attribute*), 55

H

form (*pyforms_gui.controls.control_events_graph.control_events_graph.pyforms_gui.controls.control_opengl.ControlOpenGL* *attribute*), 56 *attribute*), 48

form (*pyforms_gui.controls.control_label.ControlLabel* *help* (*pyforms_gui.controls.control_base.ControlBase* *attribute*), 42 *attribute*), 34

form (*pyforms_gui.controls.control_list.ControlList* *help_text* (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 44 *attribute*), 45

form (*pyforms_gui.controls.control_matplotlib.ControlMatplotlib* *plot()* (*pyforms_gui.controls.control_base.ControlBase* *attribute*), 46 *method*), 33

form (*pyforms_gui.controls.control_mdiaarea.ControlMdiArea* *hide()* (*pyforms_gui.controls.control_combo.ControlCombo* *attribute*), 46 *method*), 38

form (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *pyforms_gui.controls.control_dockwidget.ControlDockWidget* *attribute*), 45 *method*), 40

form (*pyforms_gui.controls.control_tree.ControlTree* *hide()* (*pyforms_gui.controls.control_emptywidget.ControlEmptyWidget* *attribute*), 51 *method*), 40

form (*pyforms_gui.controls.control_treeview.ControlTreeView* *hide()* (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 52 *method*), 44

form (*pyforms_gui.controls.control_web.ControlWeb* *highlighted_event()* (*pyforms_gui.controls.control_combo.ControlCombo* *attribute*), 53 *method*), 38

form_has_loaded (*pyforms_gui.basewidget.BaseWidget* *attribute*), 32 *horizontal_headers* (*pyforms_gui.controls.control_list.ControlList* *attribute*), 43

formset (*pyforms_gui.basewidget.BaseWidget* *attribute*), 32 *html* (*pyforms_gui.controls.control_web.ControlWeb* *attribute*), 53

fps (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 45

frame (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 45 *icon* (*pyforms_gui.controls.control_button.ControlButton* *attribute*), 45

frame_height (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 45 *icon* (*pyforms_gui.controls.control_toolbutton.ControlToolButton* *attribute*), 36

frame_width (*pyforms_gui.controls.control_player.control_player.ControlPlayer* *attribute*), 45 *icon_size* (*pyforms_gui.controls.control_list.ControlList* *attribute*), 44

G

generate_panel (*pyforms_gui.basewidget.BaseWidget* *method*), 31 *import_csv()* (*pyforms_gui.controls.control_event_timeline.control_eventtimeline* *method*), 54

get_cell (*pyforms_gui.controls.control_list.ControlList* *import_csv_file()* (*pyforms_gui.controls.control_event_timeline.control_eventtimeline* *method*), 43 *method*), 54

get_currentrow_value (*pyforms_gui.controls.control_list.ControlList* *import_graph()* (*pyforms_gui.controls.control_event_timeline.control_eventtimeline* *method*), 43 *method*), 54

get_export_filename (*pyforms_gui.controls.control_events_graph.control_events_graph.pyforms_gui.controls.control_events_graph* *method*), 56 *method*), 54

get_item_index_by_name (*pyforms_gui.controls.control_combo.ControlCombo* *info()* (*pyforms_gui.basewidget.BaseWidget* *method*), 38 *method*), 32

- info_popup() (*pyforms_gui.basewidget.BaseWidget* method), 32
 init_form() (*pyforms_gui.basewidget.BaseWidget* method), 31
 init_form() (*pyforms_gui.controls.control_base.ControlBase* method), 33
 input_double() (*pyforms_gui.basewidget.BaseWidget* method), 31
 input_int() (*pyforms_gui.basewidget.BaseWidget* method), 31
 input_text() (*pyforms_gui.basewidget.BaseWidget* method), 31
 is_item_enabled() (*pyforms_gui.controls.control_toolbox.ControlToolBox* method), 50
 is_modified (*pyforms_gui.controls.control_codeeditor.ControlCodeEditor* attribute), 37
 is_playing (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45
 item_changed() (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* method), 37
 item_changed_event() (*pyforms_gui.controls.control_tree.ControlTree* method), 51
 item_double_clicked_event() (*pyforms_gui.controls.control_tree.ControlTree* method), 51
 item_double_clicked_event() (*pyforms_gui.controls.control_treeview.ControlTreeView* method), 52
 item_selection_changed_event() (*pyforms_gui.controls.control_list.ControlList* method), 43
 item_selection_changed_event() (*pyforms_gui.controls.control_tree.ControlTree* method), 51
 item_selection_changed_event() (*pyforms_gui.controls.control_treeview.ControlTreeView* method), 52
 items (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* attribute), 37
 items (*pyforms_gui.controls.control_combo.ControlCombo* attribute), 38
- J**
- jump_backward() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45
 jump_forward() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45
- K**
- key_press_event (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45
 key_press_event() (*pyforms_gui.controls.control_tree.ControlTree* method), 51
 key_pressed_event() (*pyforms_gui.controls.control_codeeditor.ControlCodeEditor* method), 37
 key_pressed_event() (*pyforms_gui.controls.control_text.ControlText* method), 49
 key_release_event (*pyforms_gui.controls.control_event_timeline.control_eventtimeline* attribute), 55
 key_release_event (*pyforms_gui.controls.control_image.ControlImage* attribute), 42
 key_release_event (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45
 keyPressEvent() (*pyforms_gui.controls.control_tree.ControlTree* method), 52
- L**
- label (*pyforms_gui.controls.control_base.ControlBase* attribute), 34
 label (*pyforms_gui.controls.control_combo.ControlCombo* attribute), 38
 label (*pyforms_gui.controls.control_dir.ControlDir* attribute), 39
 label (*pyforms_gui.controls.control_dockwidget.ControlDockWidget* attribute), 39
 label (*pyforms_gui.controls.control_file.ControlFile* attribute), 41
 label (*pyforms_gui.controls.control_list.ControlList* attribute), 44
 label (*pyforms_gui.controls.control_mdiaarea.ControlMdiArea* attribute), 46
 label (*pyforms_gui.controls.control_number.ControlNumber* attribute), 46
 label (*pyforms_gui.controls.control_progress.ControlProgress* attribute), 48
 label (*pyforms_gui.controls.control_text.ControlText* attribute), 49
 label (*pyforms_gui.controls.control_toolbutton.ControlToolButton* attribute), 50
 legend (*pyforms_gui.controls.control_visvis.ControlVisVis* attribute), 52
 lexer (*pyforms_gui.controls.control_codeeditor.ControlCodeEditor* attribute), 37

load_finished_event() (pyforms_gui.controls.control_web.ControlWeb method), 53

load_form() (pyforms_gui.controls.control_base.ControlBase method), 33

load_form() (pyforms_gui.controls.control_checkbox.ControlCheckBox method), 36

load_form() (pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList method), 36

load_form() (pyforms_gui.controls.control_dockwidget.ControlDockWidget method), 39

load_form() (pyforms_gui.controls.control_emptywidget.ControlEmptyWidget method), 40

load_form() (pyforms_gui.controls.control_label.ControlLabel method), 42

load_form() (pyforms_gui.controls.control_list.ControlList method), 43

load_form() (pyforms_gui.controls.control_player.control_player.ControlPlayer method), 44

load_form() (pyforms_gui.controls.control_slider.ControlSlider method), 48

load_form() (pyforms_gui.controls.control_toolbutton.ControlToolButton method), 50

load_form() (pyforms_gui.controls.control_tree.ControlTree method), 51

N

O

M

max (pyforms_gui.controls.control_boundingslider.ControlBoundingSlider attribute), 35

max (pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline attribute), 55

max (pyforms_gui.controls.control_number.ControlNumber attribute), 47

max (pyforms_gui.controls.control_player.control_player.ControlPlayer attribute), 45

max (pyforms_gui.controls.control_progress.ControlProgress attribute), 48

max (pyforms_gui.controls.control_slider.ControlSlider attribute), 48

message() (pyforms_gui.basewidget.BaseWidget method), 32

message_popup() (pyforms_gui.basewidget.BaseWidget method), 32

min (pyforms_gui.controls.control_boundingslider.ControlBoundingSlider attribute), 35

min (pyforms_gui.controls.control_number.ControlNumber attribute), 47

min (pyforms_gui.controls.control_progress.ControlProgress attribute), 48

min (pyforms_gui.controls.control_slider.ControlSlider attribute), 48

mouse_moveover_timeline_event() (pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline attribute), 34

on_discart_changes() (pyforms_gui.controls.control_codeeditor.ControlCodeEditor method), 37

on_draw() (pyforms_gui.controls.control_matplotlib.ControlMatplotlib method), 46

on_margin_clicked() (pyforms_gui.controls.control_codeeditor.ControlCodeEditor method), 37

on_modification_changed() (pyforms_gui.controls.control_codeeditor.ControlCodeEditor method), 37

on_save_changes() (pyforms_gui.controls.control_codeeditor.ControlCodeEditor method), 37

P

Q

R

readonly (*pyforms_gui.controls.control_list.ControlList* save_form() (*pyforms_gui.controls.control_label.ControlLabel* attribute), 43 method), 42

readonly (*pyforms_gui.controls.control_text.ControlText* save_form() (*pyforms_gui.controls.control_list.ControlList* attribute), 49 method), 43

readonly (*pyforms_gui.controls.control_textarea.ControlTextArea* save_form() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 49 method), 44

refresh() (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList*) (*pyforms_gui.controls.control_slider.ControlSlider* method), 37 method), 48

refresh() (*pyforms_gui.controls.control_player.control_player.ControlPlayer*) (*pyforms_gui.controls.control_toolbutton.ControlToolButton* method), 44 method), 50

refresh() (*pyforms_gui.controls.control_visvis.ControlVisVis*) save_form() (*pyforms_gui.controls.control_tree.ControlTree* method), 52 method), 51

refresh() (*pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume*) (*pyforms_gui.controls.control_slider.ControlSlider* method), 53 attribute), 35

rename_graph() (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph*) (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline* method), 54 method), 56

repaint() (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph*) (*pyforms_gui.controls.control_events_graph.control_eventsgraph.ControlEventsGraph* attribute), 43

repaint() (*pyforms_gui.controls.control_opengl.ControlOpenGL*) (*pyforms_gui.controls.control_label.ControlLabel* method), 47 attribute), 42

reset_zoom_and_rotation() (*pyforms_gui.controls.control_opengl.ControlOpenGL*) (*pyforms_gui.controls.control_tree.ControlTree* selected_item method), 47 attribute), 51

resize_rows_contents() (*pyforms_gui.controls.control_list.ControlList*) (*pyforms_gui.controls.control_treeview.ControlTreeView* selected_item method), 43 attribute), 52

resizecolumns (*pyforms_gui.controls.control_list.ControlList*) (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* selected_row_index attribute), 44 attribute), 37

rows (*pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline*) (*pyforms_gui.controls.control_list.ControlList* attribute), 55 attribute), 44

rows_count (*pyforms_gui.controls.control_list.ControlList*) (*pyforms_gui.controls.control_tree.ControlTree* selected_row_index attribute), 43 attribute), 51

rows_inserted_event() (*pyforms_gui.controls.control_tree.ControlTree*) (*pyforms_gui.controls.control_treeview.ControlTreeView* selected_row_index method), 51 attribute), 52

rowsInserted() (*pyforms_gui.controls.control_tree.ControlTree*) (*pyforms_gui.controls.control_list.ControlList* selected_row_index method), 51 attribute), 43

rowsInserted() (*pyforms_gui.controls.control_treeview.ControlTreeView*) (*pyforms_gui.controls.control_list.ControlList* selected_rows_indexes method), 51 attribute), 43

save_form() (*pyforms_gui.controls.control_base.ControlBase*) (*pyforms_gui.controls.control_treeview.ControlTreeView* selected_rows_indexes method), 33 attribute), 51

save_form() (*pyforms_gui.controls.control_checkbox.ControlCheckBox*) (*pyforms_gui.controls.control_treeview.ControlTreeView* selection_changed_event() method), 36 method), 36

save_form() (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList*) (*pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList* method), 36 method), 37

save_form() (*pyforms_gui.controls.control_dockwidget.ControlDockWidget*) (*pyforms_gui.controls.control_treeview.ControlTreeView* selection_changed() method), 39 method), 51

save_form() (*pyforms_gui.controls.control_emptywidget.ControlEmptyWidget*) (*pyforms_gui.controls.control_treeview.ControlTreeView* set_item_enabled() method), 40 method), 51

save_form() (*pyforms_gui.controls.control_image.ControlImage*) (*pyforms_gui.controls.control_toolbox.ControlToolBox* method), 41 method), 50

set_margin() (pyforms_gui.basewidget.BaseWidget method), 32	tableWidgetCellDoubleClicked() (pyforms_gui.controls.control_list.ControlList method), 44
set_sorting_enabled() (pyforms_gui.controls.control_list.ControlList method), 43	tableWidgetItemChanged() (pyforms_gui.controls.control_list.ControlList method), 44
set_value() (pyforms_gui.controls.control_list.ControlList method), 43	tableWidgetItemSelectionChanged() (pyforms_gui.controls.control_list.ControlList method), 44
show() (pyforms_gui.basewidget.BaseWidget method), 31	text (pyforms_gui.controls.control_combo.ControlCombo attribute), 38
show() (pyforms_gui.controls.control_base.ControlBase method), 33	title (pyforms_gui.basewidget.BaseWidget attribute), 32
show() (pyforms_gui.controls.control_combo.ControlCombo method), 38	toggleWidget (pyforms_gui.controls.control_visvis.ControlVisVis attribute), 52
show() (pyforms_gui.controls.control_dockwidget.ControlDockWidget method), 40	toggleWidget (pyforms_gui.controls.control_events_graph.control_eventsgraph attribute), 56
show() (pyforms_gui.controls.control_emptywidget.ControlEmptyWidget method), 40	toggleWidget (pyforms_gui.controls.control_events_graph.control_eventsgraph attribute), 56
show() (pyforms_gui.controls.control_player.control_player.ControlPlayer method), 44	toggleWidget (pyforms_gui.controls.control_events_graph.control_eventsgraph attribute), 56
show_graphs_properties() (pyforms_gui.controls.control_event_timeline.control_eventtimeline.ControlEventTimeline method), 54	
show_grid (pyforms_gui.controls.control_visvis.ControlVisVis attribute), 52	update_event() (pyforms_gui.controls.control_number.ControlNumber method), 46
show_header (pyforms_gui.controls.control_tree.ControlTree attribute), 51	
show_subwin_close_button (pyforms_gui.controls.control_mdiarea.ControlMdiArea attribute), 46	
SIDE_BOTTOM (pyforms_gui.controls.control_dockwidget.ControlDockWidget attribute), 39	value (pyforms_gui.controls.control_base.ControlBase attribute), 34
SIDE_DETACHED (pyforms_gui.controls.control_dockwidget.ControlDockWidget attribute), 39	value (pyforms_gui.controls.control_boundingslider.ControlBoundingSlider attribute), 35
SIDE_LEFT (pyforms_gui.controls.control_dockwidget.ControlDockWidget attribute), 39	value (pyforms_gui.controls.control_button.ControlButton attribute), 35
SIDE_RIGHT (pyforms_gui.controls.control_dockwidget.ControlDockWidget attribute), 39	value (pyforms_gui.controls.control_checkbox.ControlCheckBox attribute), 36
SIDE_TOP (pyforms_gui.controls.control_dockwidget.ControlDockWidget attribute), 39	value (pyforms_gui.controls.control_checkboxlist.ControlCheckBoxList attribute), 37
step (pyforms_gui.controls.control_number.ControlNumber attribute), 47	value (pyforms_gui.controls.control_codeeditor.ControlCodeEditor attribute), 38
stop() (pyforms_gui.controls.control_player.control_player.ControlPlayer method), 44	value (pyforms_gui.controls.control_combo.ControlCombo attribute), 38
success() (pyforms_gui.basewidget.BaseWidget method), 32	value (pyforms_gui.controls.control_dir.ControlDir attribute), 39
success_popup() (pyforms_gui.basewidget.BaseWidget method), 32	value (pyforms_gui.controls.control_emptywidget.ControlEmptyWidget attribute), 40
	value (pyforms_gui.controls.control_event_timeline.control_eventtimeline attribute), 55
	value (pyforms_gui.controls.control_events_graph.control_eventsgraph attribute), 56
	value (pyforms_gui.controls.control_file.ControlFile attribute), 40
	value (pyforms_gui.controls.control_filetree.ControlFilesTree attribute), 41

T

value (*pyforms_gui.controls.control_image.ControlImage* attribute), 42

value (*pyforms_gui.controls.control_label.ControlLabel* attribute), 42

value (*pyforms_gui.controls.control_list.ControlList* attribute), 43

value (*pyforms_gui.controls.control_matplotlib.ControlMatplotlib* attribute), 46

value (*pyforms_gui.controls.control_number.ControlNumber* attribute), 46

value (*pyforms_gui.controls.control_opengl.ControlOpenGL* attribute), 47

value (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

value (*pyforms_gui.controls.control_progress.ControlProgress* attribute), 48

value (*pyforms_gui.controls.control_slider.ControlSlider* attribute), 48

value (*pyforms_gui.controls.control_text.ControlText* attribute), 49

value (*pyforms_gui.controls.control_textarea.ControlTextArea* attribute), 49

value (*pyforms_gui.controls.control_toolbox.ControlToolBox* attribute), 50

value (*pyforms_gui.controls.control_toolbutton.ControlToolButton* attribute), 50

value (*pyforms_gui.controls.control_tree.ControlTree* attribute), 51

value (*pyforms_gui.controls.control_treeview.ControlTreeView* attribute), 52

value (*pyforms_gui.controls.control_visvis.ControlVisVis* attribute), 53

value (*pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume* attribute), 53

value (*pyforms_gui.controls.control_web.ControlWeb* attribute), 53

valueChanged() (*pyforms_gui.controls.control_slider.ControlSlider* method), 48

values (*pyforms_gui.controls.control_combo.ControlCombo* attribute), 38

video_frames_value_changed() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45

video_index (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

videoPlay_clicked() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45

videoProgress_sliderReleased() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45

videoProgress_valueChanged() (*pyforms_gui.controls.control_player.control_player.ControlPlayer* method), 45

view_in_3D (*pyforms_gui.controls.control_player.control_player.ControlPlayer* attribute), 45

visible (*pyforms_gui.controls.control_base.ControlBase* attribute), 34

visvis (*pyforms_gui.controls.control_visvisvolume.ControlVisVisVolume* attribute), 53

W

warning() (*pyforms_gui.basewidget.BaseWidget* method), 32

warning_popup() (*pyforms_gui.basewidget.BaseWidget* method), 32

width (*pyforms_gui.controls.control_opengl.ControlOpenGL* attribute), 48

word_wrap (*pyforms_gui.controls.control_list.ControlList* attribute), 43

X

xlabel (*pyforms_gui.controls.control_visvis.ControlVisVis* attribute), 53

Y

ylabel (*pyforms_gui.controls.control_visvis.ControlVisVis* attribute), 53

Z

zlabel (*pyforms_gui.controls.control_visvis.ControlVisVis* attribute), 53