# psforms

***Release 0.3.1***

September 16, 2015

Contents

Hassle free PySide forms.

```
from psforms import (Form, IntField, StringField,
                     StringOptionField, BoolField)


class MyForm(Form):
    '''My amazing form, useful in many scenarios.'''

    title = 'My Form'
    int_field = IntField('Integer Value')
    str_field = StringOptionField('String Value', options=['A', 'B', 'C'])
    bool_field = BoolField('Boolean Value')
    strb_field = StringField('String Value B')


myform_dialog = MyForm.as_dialog()
if myform_dialog.exec_():
    print dialog.get_value()
```

CHAPTER 1

# Features

- Easy Form creation
- Parent forms to your own window or use them as their own stand alone dialog
- Unified api for all standard PySide input widgets

CHAPTER 2

# Get psforms

You can install psforms using pip:

```
pip install psforms
```

or you can use setuptools:

```
git clone git@github.com/danbradham/psforms.git
cd psforms
python setup.py install
```

CHAPTER 3

# Table of Contents

## 3.1 Guide

This guide will walk you through using psforms. Let's start by expanding on the example from the ReadMe.

```
from psforms import (Form, IntField, StringField,
                     StringOptionField, BoolField)


class MyForm(Form):
    '''My amazing form, useful in many scenarios.'''

    title = 'My Form'
    int_field = IntField('Integer Value')
    str_field = StringOptionField('String Value', options=['A', 'B', 'C'])
    bool_field = BoolField('Boolean Value')
    strb_field = StringField('String Value B')
```

The `psforms.Form` is a factory for creating various types of forms. `psforms.Field` attributes are used to describe the input fields. `psforms.Form` subclasses are only skeletons of a widget, waiting to be created. To create an actual form widget, use one of following methods, all of which start with the prefix **as_**.

### 3.1.1 Forms as Dialogs

```
myform_dialog = MyForm.as_dialog()

if myform_dialog.exec_():
    print dialog.get_value()
```

The `as_dialog()` returns a `psforms.Dialog` instance with the fields specified in `MyForm`. `psforms.Dialog` accepts two keyword arguments; columns and parent. `get_value()` returns a `FormData` including the names and values for all the fields in MyForm. `FormData` supports both dictionary access and attribute access.

In this next example `ParentWidget` refers to a parent applications `QtGui.QWidget` or `QtGui.QMainWindow`.

```
myform_dialog = MyForm.as_dialog(columns=2, parent=ParentWidget)
```

The previous examples create a modal dialog, which blocks all other PySide widgets from receiving input until after the dialog is accepted or rejected.

### 3.1.2 Forms as Widgets and Groups

You can also get a Form as a mulit-column `psforms.Widget` or `psforms.Group`.

```
myform_widget = MyForm.as_widget(columns=2)
myform_group = MyForm.as_group(columns=1, collapsable=True)
```

The above `psforms.Widget` and `psforms.Group` are derived from a standard `QtGui.QWidget` and a standard `QtGui.QGroupBox`; therefore, they can be added to any PySide layout. The collapsable parameter refers to whether or not the entire `psforms.Group` can be collapsed. Both of these also have a `get_value()` like the dialog above.

### 3.1.3 Getting the value of a control

All psform Field controls share the same api. You can use `set_value()` to set them and `get_value()` to retrieve them.

```
myform_dialog.int_field.set_value(40)
assert myform_dialog.int_field.get_value() == 40
```

All controls also have changed signal that are emitted whenever their values are modified by user interaction.

## 3.2 API Documentation

### 3.2.1 Form

**class** `psforms.form.`**`Form`**

**classmethod `as_dialog`** (*frameless=False*, *dim=False*, *parent=None*)
Get this form as a dialog

**classmethod `as_widget`** (*parent=None*)
Get this form as a widget

**`columns` = 1**

**classmethod `create`** (*parent=None*)
Create a widget for this form using all Field attributes

**`description` = None**

**`header` = False**

**`icon` = None**

**`labeled` = True**

**`labels_on_top` = True**

**`title` = None**

### 3.2.2 Fields

**class** `psforms.fields.`**`BoolField`**(*name*, *label_on_top=False*, ***kwargs*)
Represented by a `CheckBox` control.

**Parameters**

- **name** – Nice name of the field (str)
- **default** – Default value (str)

**control_cls = <Mock spec='str' id='140189845288592'>**

**class** psforms.fields.**Field**(*name*, *labeled=None*, *label_on_top=None*, *default=None*)
Form calls the create() to retrieve an appropriate control.

**Parameters**

- **name** – Nice name of the field (str)
- **labeled** – Field Control has label (bool) Overrides the parent Forms labeled attribute for this field only
- **label_on_top** – Label appears on top of the field control (bool) Overrides the parent Forms label_on_top attribute for this field only
- **default** – Default value (str)

**class** psforms.fields.**Float2Field**(*name*, *range1=None*, *range2=None*, ***kwargs*)
Represented by a TwinDoubleSpinBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **range1** – Tuple of minimum and maximum values
- **range2** – Tuple of minimum and maximum values
- **default** – Default value (float)

**control_cls = <Mock spec='str' id='140189845229520'>**

**class** psforms.fields.**FloatField**(*name*, *range=None*, ***kwargs*)
Represented by a DoubleSpinBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **range** – Tuple of minimum and maximum values
- **default** – Default value (float)

**control_cls = <Mock spec='str' id='140189845229264'>**

**class** psforms.fields.**Int2Field**(*name*, *range1=None*, *range2=None*, ***kwargs*)
Represented by a TwinSpinBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **range1** – Tuple of minimum and maximum values
- **range2** – Tuple of minimum and maximum values
- **default** – Default value (float, float)

**control_cls = <Mock spec='str' id='140189845287568'>**

**class** psforms.fields.**IntField**(*name*, *range=None*, ***kwargs*)
Represented by a SpinBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **range** – Tuple of minimum and maximum values
- **default** – Default value (int)

**control_cls = <Mock spec='str' id='140189845287312'>**

**class** psforms.fields.**IntOptionField**(*name*, *options*, ***kwargs*)
Represented by an IntComboBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **options** – List of options
- **default** – Default value (int)

**control_cls = <Mock spec='str' id='140189845288208'>**

**class** psforms.fields.**ListField**(*name*, *labeled=None*, *label_on_top=None*, *default=None*)
Represented by a List control

**Parameters**

- **name** – Nice name of field (str)
- **default** – Default value (list of strings)

**control_cls = <Mock spec='str' id='140189845289360'>**

**class** psforms.fields.**StringField**(*name*, *labeled=None*, *label_on_top=None*, *default=None*)
Represented by a LineEdit control.

**Parameters**

- **name** – Nice name of the field (str)
- **default** – Default value (str)

**control_cls = <Mock spec='str' id='140189845288976'>**

**class** psforms.fields.**StringOptionField**(*name*, *options*, ***kwargs*)
Represented by an ComboBox control.

**Parameters**

- **name** – Nice name of the field (str)
- **options** – List of options
- **default** – Default value (str)

**control_cls = <Mock spec='str' id='140189845287952'>**

### 3.2.3 Controls

#### psforms.controls

Wraps standard PySide input widgets providing a unified api for getting and setting their values. Each control implements get_value() and set_value(). A required position argument value or values is used to set the default value of the control or in the case of ComboBox and IntComboBox a sequence of items to add to the

wrapped QComboBox. In addition each control emits a Signal named *changed* whenever the value is changed by user interaction.

### 3.2.4 Exceptions

**exception** psforms.exc.**FieldNotFound**

**exception** psforms.exc.**FieldNotInstantiated**

**exception** psforms.exc.**FormNotInstantiated**

## p

## A



## B



## C



## D



## F



## H



## I



## L



## P



## S



## T