kivy using addtional class to store data from multiple screens

Asked 5 years, 3 months ago Modified 5 years, 3 months ago Viewed 1k times

2

I want to add an additional class to my kivy app that is just an info storage depot. I'm not sure where to do this or if this approach is even advised.

The app has multiple screens. Each screen allows the user to make a selection. The user's selection is what I want to store in the additional class. The additional class will store data then ultimately use MQTT to send the data out. Also, I wanted to keep the class definition in a separate file so I could organize the program into logically ordered chunks.

(keep in mind I'm sharing a small fraction of the code, but this should be representative enough to convey my question)

My kivy python code:

```
from kivy.app import App
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.screenmanager import ScreenManager, Screen
from storage import Storage  # <<========================== this is the addtional class

# Screen Manager
class MyScreenManager(ScreenManager):
    pass


# Background Color
class BG(FloatLayout):
    pass


class VendorScreen(Screen):
    pass


class InvoiceScreen(Screen):
    def __init__(self, **kwargs):
        super(InvoiceScreen, self).__init__(**kwargs)
        self.invoice = ''

    def keyEntry(self, number):  # ............ Digit Pressed
        invoice = self.ids.invoice  # ......... link to kivy Label
        invoice.text += number  # ............. append number to invoice

    def keyBack(self):  # ..................... Backspace
        invoice = self.ids.invoice  # ......... link to kivy Label
        invoice.text = invoice.text[:-1]  # ... remove last digit

    def set_invoice(self):
        invoice = self.ids.invoice
        self.invoice = invoice.text
```

```
class MainControlScreen(Screen):
    pass


# Main-Execution
class mySuperApp(App):
    def build(self):
        return MyScreenManager()


if __name__ == '__main__':
    mySuperApp().run()
```

my kv code:

```
#:kivy 1.0.9

<MyScreenManager>:
    VendorScreen:  # ........ Incoming Delivery, Screen 2b
        name: 'vendor_screen'
    InvoiceScreen: # ........ ................... Screen 3b
        name: 'invoice_screen'
    MainControlScreen:  # ... ................... Screen 4b
        name: 'main_control_screen'

<BG>
    AsyncImage:
        source: 'img/screen1_background4.png'
        size_hint: 1, 1

<VendorScreen>
    BG:
    FloatLayout:
        Button:
            text: 'name1'
            color: 1.0, 0.6, 0.0, 1
            font_size: 40
            size_hint_x: 0.45
            size_hint_y: 0.35
            pos_hint: {'x': 0.03, 'y': 0.50}
            on_release:
                root.manager.transition.direction = 'left'
                root.manager.current = 'invoice_screen'
        Button:
            text: 'name2'
            color: 1.0, 0.6, 0.0, 1
            font_size: 40
            size_hint_x: 0.45
            size_hint_y: 0.35
            pos_hint: {'x': 0.52, 'y': 0.50}
            on_release:
                root.manager.transition.direction = 'left'
                root.manager.current = 'invoice_screen'
        Button:
            text: 'name3'
            color: 1.0, 0.6, 0.0, 1
            font_size: 40
            size_hint_x: 0.45
            size_hint_y: 0.35
            pos_hint: {'x': 0.03, 'y': 0.10}
```

```
                on_release:
                    root.manager.transition.direction = 'left'
                    root.manager.current = 'invoice_screen'
            Button:
                text: 'name4'
                color: 1.0, 0.6, 0.0, 1
                font_size: 40
                size_hint_x: 0.45
                size_hint_y: 0.35
                pos_hint: {'x': 0.52, 'y': 0.10}
                on_release:
                    root.manager.transition.direction = 'left'
                    root.manager.current = 'invoice_screen'

<InvoiceScreen>
    BG:
        canvas:
            Color:
                rgba: 0.90, 0.90, 0.90, 0.5
            Rectangle:
                pos: (40, 295)
                size: (320, 35)
            Color:
                rgba: 0, 0, 0, 1
            Line:
                points: 40, 295, 360, 295, 360, 330, 40, 330, 40, 295
                width: 1
    BoxLayout:
        orientation: 'horizontal'  # break it up into left / right
        FloatLayout:
            Label:
                pos_hint: {'x':0, 'y':.25}
                font_size: 30
                text: 'Enter Invoice Number'
                color: 0.1, 0.1, 1, 1
            Label:
                id: invoice
                pos_hint: {'x':0, 'y':.15}
                font_size: 30
                text: ''  # initially blank
                color: 0, 0, 0, 1
        GridLayout:
            cols: 3  # number of columns
            rows: 4  # number of rows
            Button:
                text: '1'
                on_release: root.keyEntry('1')
            Button:
                text: '2'
                on_release: root.keyEntry('2')
            Button:
                text: '3'
                on_release: root.keyEntry('3')
            Button:
                text: '4'
                on_release: root.keyEntry('4')
            Button:
                text: '5'
                on_release: root.keyEntry('5')
            Button:
                text: '6'
                on_release: root.keyEntry('6')
```

```
                Button:
                    text: '7'
                    on_release: root.keyEntry('7')
                Button:
                    text: '8'
                    on_release: root.keyEntry('8')
                Button:
                    text: '9'
                    on_release: root.keyEntry('9')
                Button:
                    text: '< DEL'
                    on_release: root.keyBack()
                Button:
                    text: '0'
                    on_release: root.keyEntry('0')
                Button:
                    text: 'Done'
                    on_release:
                        root.set_invoice()
                        root.manager.transition.direction = 'left'
                        root.manager.current = 'main_control_screen'

<MainControlScreen>
    BG:
        canvas:
            Color:
                rgba: 0, 0, 1, 1
            Line:
                points: 500, 180, 770, 180, 770, 450, 500, 450, 500, 180
                width: 3
    FloatLayout:
        Label:  # foreground
            pos_hint: {'x': 0.30, 'y': 0.13}
            font_size: 80
            text: '5'
            color: 1.0, 0.6, 0.0, 1
        Button:
            size_hint_x: 0.2
            size_hint_y: 0.1
            pos_hint: {'x': 0.05, 'y': 0.05}
            text: 'Auto-Reject'
            on_release:
                root.manager.transition.direction = 'up'
                root.manager.current = 'vendor_screen'
        Button:
            size_hint_x: 0.2
            size_hint_y: 0.1
            pos_hint: {'x': 0.75, 'y': 0.05}
            text: 'Photo'
            on_release:
                root.manager.transition.direction = 'left'
                root.manager.current = 'invoice_screen'
```

And finally my additional class:

```
from datetime import datetime, date
import calendar


class Storage(object):
```

```
    def __init__(self):
        self.invoice = 0
        self.unit_name = ''
        self.unit_number = ''
        self.receiver = ''
        self.vendor = ''
        self.day = calendar.day_name[date.today().weekday()]
        self.delivery_time = datetime.now()
        self.package_condition = True
        self.email = ''
        self.location = ('32.0, -117.0',)
        self.duration = 0
        self.img = 'img.jpg'
```

So my question is, where and how do I use my addtional class 'Storage'? I want all the kivy classes to be able to access a single instance of it, but I cannot figure out how to do that.

I tried instantiating a class just before `mySuperApp().run()` but I cannot access it inside the other classes. I tried using `global` to access the instance, but that didn't work.

I thought about inheriting from the class, but I'm not sure how to do this... I started by inheriting from Storage in my different screens, however, that does not let me access a single instance where all my data is, rather I would have to pull instance variables from multiple classes to amass the full data set. Then I tried inheriting mySuperApp in Storage, then running the ScreenManager from Storage, but that didn't work. It ran, but I couldn't access the instance variables from the other classes. Maybe I need to keep inheriting from Storage in all the other classes?

I'm not sure how to approach this as the kivy part is only a fraction of the program. In the end I will need several additional classes, but I don't understand how or where to link these. Once I get the concept I will apply it to the other necessary classes, but I don't know where to start.

python class kivy

Share Follow

asked Aug 4, 2017 at 23:07

twegner
423 1 5 21

I just learned about Class variables. This seems to work. I'm currently inheriting Storage in all my screen classes. As I move between screens I am updating the Storage Class variables and this gives the behavior I wanted. – twegner Aug 5, 2017 at 0:50

1 Answer

Sorted by:

Highest score (default)

You could use the `ScreenManager` to share data.

2

The `ScreenManager` will allways be parent of all the screens. So here are three ways you can access the manager.

`root.manager`:

The `Screen`'s all have an attribute called `manager`, which is the `ScreenManager` which this `Screen` is in. So you can access it from any screen like this.

`root.parent`:

The `Widget`'s if they are in another widget will all have it's `parent`. In this case, the `Screen`'s parent is the `ScreenManager`. So you can access it from any screen like this.

`app.root`:

Every app has a `root`. In this case the root of the app, is the `ScreenManager`. So you can access it from anywhere like this.

Try this example, with three buttons demonstrating this:

```
from kivy.app import App
from kivy.uix.screenmanager import Screen, ScreenManager
from kivy.lang import Builder
from kivy.properties import StringProperty


class MyScreenManager(ScreenManager):
    shared_data = StringProperty("")


class Screen1(Screen):
    pass

class Screen2(Screen):
    pass


root = Builder.load_string('''

<Screen1>:
    name: "screen1"
    BoxLayout:
        orientation: "vertical"
        TextInput:
            on_text:
                root.manager.shared_data = self.text
        Label:
            text:
                root.manager.shared_data
        Button:
            text: "Go to screen2"
            on_release: root.manager.current = "screen2"

        Button:
            text: "root.manager.shared_data"
            on_release:
                print(root.manager.shared_data)
        Button:
            text: "root.parent.shared_data"
```

```
            on_release:
                print(root.parent.shared_data)
        Button:
            text: "app.root.shared_data"
            on_release:
                print(app.root.shared_data)



<Screen2>:
    name: "screen2"
    BoxLayout:
        orientation: "vertical"
        TextInput:
            on_text:
                root.manager.shared_data = self.text
        Label:
            text:
                root.manager.shared_data
        Button:
            text: "Go to screen1"
            on_release: root.manager.current = "screen1"


MyScreenManager:
    Screen1:
    Screen2:

''')



class MyApp(App):
    def build(self):
        return root
```

Share Follow

edited Aug 7, 2017 at 10:34

answered Aug 5, 2017 at 23:09

el3ien
5,239 1 16 32

In the kivy portion of your code; Screen1, under any of the widgets, when you write "root.manager.shared_data" does 'manager' refer to the Class Manager or is it a built-in call? I'm using your example, but slightly modified, and I'm having issues. – twegner Aug 7, 2017 at 3:41

Can you provide one more example? Add a button on screen1 that simple prints the string-text of 'shared_data' to the console. I've tried and it returns <StringProperty name=shared_data>. – twegner Aug 7, 2017 at 3:56

1 @twegner edited, and added three buttons, explaining three different ways to access the data. – el3ien Aug 7, 2017 at 10:05

1 this is a fantastic answer. I've seen pieces of this data spread over many different docs, but bringing it all together in a single extended explanation really clears this stuff up. A great many thanks! – twegner Aug 9, 2017 at 2:58

@twegner glad that it helped. Thank you. – el3ien Aug 9, 2017 at 5:13