KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

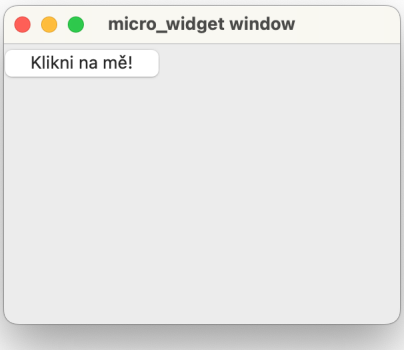
Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Úvod do programovacích stylů ◇ poznámky k přednášce

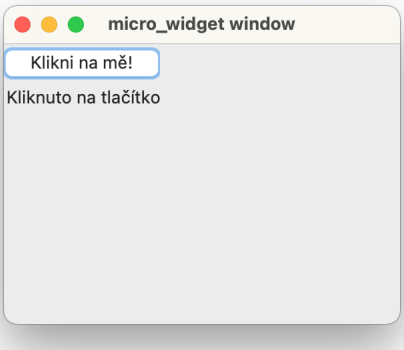
5. Události

verze z 17. října 2023

Začneme příkladem. Chceme vytvořit tlačítko:



a zobrazit text poté, co se na něj kliklo:



Začneme definicí třídy:

```
class LabeledButton(Group):
    def __init__(self):
        super().__init__()
        button = Button().set_text("Klikni na mě!")
        label = Label().move(0, 30)
```

```
        self.set_items([button, label])

    def get_label(self):
        return self.get_items()[1]

    def display_text(self):
        self.get_label().set_text("Kliknuto na tlačítko")
        return self
```

Zobrazení textu má na starosti metoda `display_text`.

Instanci třídy vložíme do okna:

```
window = Window()
labeled_button = LabeledButton()
window.set_widget(labeled_button)
```

Chceme zajistit, aby se objektu `labeled_button` zaslala zpráva `display_text` poté, co uživatel klikne na tlačítko. Tlačítko ale neví, co se má stát, když se na něj klikne. Rozhodnutí musí nechat na objektu, který je za něj zodpovědný. Takový objekt nazýváme *delegátem objektu*. Obecně platí, že skupina (instance třídy `Group`) je delegátem všech svých prvků a okno je delegátem zobrazeného ovládacího prvku. Tedy `labeled_button` je delegátem tlačítka, které je jeho prvkem.

Pokud objekt potřebuje informovat svého delegáta o nějaké skutečnosti *zašle událost*. Konkrétněji objekt ***sender*** zašle událost ***event*** s argumenty ***arg1***, ***arg2***, ... Nejprve se zjistí, zda má objekt ***sender*** delegáta a zda delegát rozumí zprávě ***event***. V kladném případě se delegátovi zašle zpráva ***event*** s argumenty ***sender***, ***arg1***, ***arg2***, ... Odesílatel události se stane prvním argumentem zaslané zprávy. Například tlačítko zašle událost `ev_button_click` v případě, že na něj uživatel klikne. Jména událostí vždy začínají písmeny `ev` (což je zkráveně za *event* – událost).

K zajištění zobrazení textu v našem příkladě stačí reagovat na událost `ev_button_click`. Do třídy `LabeledButton` doplníme její obsluhu:

```
    def ev_button_clicked(self, sender, button):
        super().ev_button_clicked(sender, button)
        self.display_text()
```

Všimněte si, že nejprve voláme přepsanou metodu.

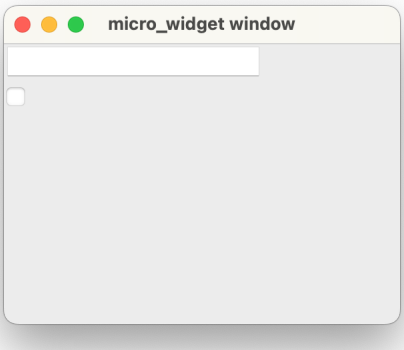
Okna i ovládací prvky souhrně nazýváme objekty knihovny `omw`. Objekt knihovny zašle událost `ev_change` poté, co se změní. Můžeme tak snadno vytvořit třídu, která zaznamenává změny svých prvků:

```
class LoggingGroup(Group):
    def ev_change(self, sender):
        super().ev_change(sender)
        print("Změna prvku:", sender)
```

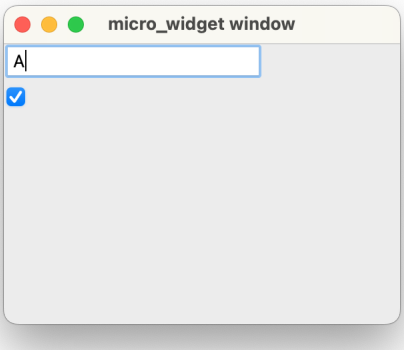
Vytvoříme instanci:

```
window = Window()
entry = Entry()
checkbox = Checkbox().move(0, 30)
group = LoggingGroup().set_items([entry, checkbox])
window.set_widget(group)
```

Uvidíme:



Uživatelský vstup:



vyvolává změny:

```
Změna prvku: <omw.Checkbox object at 0x103d55310>
Změna prvku: <omw.Entry object at 0x103f14b50>
```

Změnu můžeme vyvolat i programově:

```
>>> checkbox.move(10, 0)
Změna prvku: <omw.Checkbox object at 0x103d55310>
```

Každý objekt knihovny rozumí zprávě *object*`.send_event(`*event*`, `*arg1*`, `*arg2*`, ...)`, která zašle událost *event* s argumenty *arg1*, *arg2*, ...

Můžeme například vytvořit třídu pro textové pole, které bude zasílat událost `ev_text_change` při každé změně textu:

```
class TextChangeEntry(Entry):
    def set_text(self, text):
        super().set_text(text)
        self.send_event("ev_text_change")
        return self
```

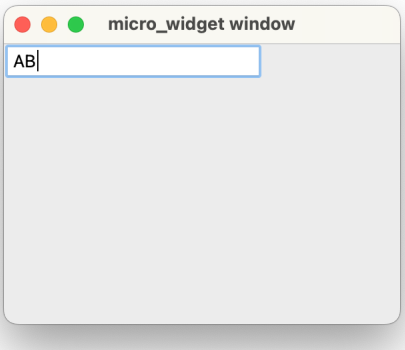
Vytvoříme třídu pro skupinu s obsluhou události `ev_text_change`:

```
class LoggingGroup(Group):
    def ev_text_change(self, sender):
        print("Change of text of entry", sender)
```

Po vytvoření instancí:

```
window = Window()
entry = TextChangeEntry()
group = LoggingGroup().set_items([entry])
window.set_widget(group)
```

každá změna vlastnosti `text` objektu `entry`:



bude hlášena:

```
Change of text of entry: <__main__.TextChangeEntry object at 0x108f68910>
Change of text of entry: <__main__.TextChangeEntry object at 0x108f68910>
```

Nahlásí se i změna provedená programem:

```
>>> entry.set_text("ABC")
Change of text of entry: <__main__.TextChangeEntry object at 0x108f68910>
<__main__.TextChangeEntry object at 0x108f68910>
```

Posun hlášen nebude:

```
>>> entry.move(10, 0)
<__main__.TextChangeEntry object at 0x108f68910>
```