
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

Manuál ke knihovně Functional Micro Widget

verze z 14. listopadu 2024

Knihovnu tvoří soubory `micro_widget.py` a `fmw.py`. Soubory se musí nalézat v témže adresáři jako program, který jej používá. Pro použití stačí importovat modul:

```
from fmw import *
```

Ovládací prvky

```
label(text="", x=0, y=0) => widget

  text: řetězec
  x: celé číslo
  y: celé číslo
  widget: ovládací prvek
```

Prvek *widget* je popiskem s textem *text* na souřadnicích *x*, *y*.

```
button(text="", action=None, x=0, y=0) => widget

  text: řetězec
  action: libovolná hodnota
  x: celé číslo
  y: celé číslo
  widget: ovládací prvek
```

Prvek *widget* je tlačítko s textem *text* na souřadnicích *x*, *y*. Při stisku v případě, že *action* není `None`, vyvolá událost *action*.

```
entry(text="", action=None, x=0, y=0) => widget

  text: řetězec
  action: libovolná hodnota
```

x: celé číslo
y: celé číslo
widget: ovládací prvek

Prvek *widget* je textovým polem s textem *text* na souřadnicích *x*, *y*. V případě změny uživatelem vyvolá akci [*action*, *new_text*], kde *new_text* je změněný text.

moved(*widget*, *dx*, *dy*) => *moved_widget*

widget: ovládací prvek
dx: celé číslo
dy: celé číslo
moved_widget: ovládací prvek

Prvek *moved_widget* je prvkem *widget* posunutým o přírůstky *dx* a *dy*.

action_changed(*widget*, *change*) => *changed_widget*

widget: ovládací prvek
change: funkce jednoho parametru nebo hodnota, která není funkcí
changed_widget: vládací prvek

Prvek *changed_widget* je shodný s prvkem *widget* až na to, že každá akce *action* vyvolaná prvkem *widget* se změní na *change*(*action*), pokud je *change* funkce, jinak na [*change*, *action*].

group(*widget1*, *widget2*) => *group*

group: ovládací prvek
widget1: ovládací prvek
widget2: ovládací prvek

Ovládací prvek *group* se skládá z prvků *widget1* a *widget2*.

empty_widget

Prázdný ovládací prvek.

Zobrazení okna

```
display_window(content,
               init_state=None,
               next_state=None,
               process_effect=None,
               trace=False) => emit_action
```

content: funkce jednoho parametru nebo ovládací prvek
init_state: libovolná hodnota
next_state: hodnota `None` nebo funkce dvou parametrů
process_effect: hodnota `None` nebo funkce jednoho parametru
emit_action: funkce jednoho parametru
trace: logická hodnota

Pokud *content* není funkce, pak se místo *content* používá funkce jednoho parametru, která vždy vrací *content*. Neboli funkce:

```
lambda state: content
```

Pokud je *next_state* hodnota `None`, pak se místo *next_state* používá funkce dvou parametrů, která vždy vrací druhý argument. Což je funkce:

```
lambda state, action: action
```

Pokud je *process_effect* hodnota `None`, pak se místo *process_effect* používá funkce jednoho parametru, která pouze vrací hodnotu `None`; tedy funkce:

```
lambda effect: None
```

Aktuální **stav okna** si označíme *state*. Na začátku je *state* rovno *init_state*. Obsah okna se získá zavoláním funkce *content* na *state*. Okno tedy zobrazuje ovládací prvek:

```
content(state)
```

Po každé změně stavu okna se znovu získá jeho obsah.

Při vyvolání akce *action* ovládacím prvkem v okně, se změní stav okna *state* na výsledek volání funkce *next_state* na původní stav okna *state* a akci *action*. Tedy:

```
state = next_state(state, action)
```

Funkce *next_state* může vrátit i hodnotu

```
with_effect(state, effect)
```

V takovém případě je *state* nový stav okna a dojde k aplikaci funkce *process_effect* na *effect*. Tedy:

```
process_effect(effect)
```

Funkci *emit_action* je možné aplikovat na libovolnou hodnotu:

```
emit_action(action)
```

čímž dojde k vyvolání akce *action*.

Pokud *trace* je pravda, pak se při získávání obsahu okna tiskne jeho stav a získaný obsah okna a při vyvolání akce se tiskne starý stav a vyvolaná akce.