RubyKaigi2009-RejectKaigi（2009/07/19）

Ruby/Tk のデフォルトウィジェットセット制御におけるトップレベル定数定義の操作

永井 秀利 (*nagai@ai.kyutech.ac.jp*)

九州工業大学 / Rubyist 九州

テーマ付きウィジェットの標準化

- Tcl/Tk において tile または Ttk 拡張と呼ばれていたもの
- 標準ウィジェットと個々に対応し，１対１の置き換えとなりうる

Ttk を活用するサンプル： ttk_wrapper.rb

- 旧来の Ruby/Tk スクリプトを（100% ではないが）テーマエンジンに対応させる wrapper スクリプト

例： tkhello.rb

```
require 'tk'
TkButton.new(:text=>'hello',
             :command=>proc{puts 'hello'}).pack(:fill=>:x)
TkButton.new(:text=>'quit',
             :command=>proc{exit}).pack(:fill=>:x)
Tk.mainloop
```

ruby tkhello.rb

⇒

ruby ttk_wrapper -t *theme* tkhello.rb

theme:alt

theme:clam

theme:classic

theme:blue

theme:kroc

theme:plastik

Tk.default_widget_set = *widget_set*

- 旧来の TkButton などが指し示すウィジェット集合の切替え
  - 任意の時点で切替え可能
  - デフォルトでは :Tk と :Ttk とを用意
  - 自分で組合せを作ることも可能
- Tk.default_widget_set = :Tk　（デフォルトの状態）
  TkButton → Tk::Button
  TkLabel　→ Tk::Label　などと標準ウィジェットに設定
- Tk.default_widget_set = :Ttk
  TkButton → Ttk::Button
  TkLabel　→ Ttk::Label　などと Ttk ウィジェットに設定

… と，一見，簡単な実装のようにみえるんですが …

実は条件の組合せがすごくややこしい！

- TkButton 等は切替時に autoload 設定のままかもしれないし，すでに autoload 済みかもしれない．
  “class TkButton” などとした場合は autoload が働かなければならない．
- TkCombobox のように :Ttk ウィジェットセットに登録されているものの，:Tk ウィジェットセットには登録されていないものの場合，
  - デフォルトウィジェットセットが :Tk の時，ユーザが autoload 設定と無関係にクラス定義しようとするかもしれない．
    いきなり TkCombobox.new とした際は autoload が働かなくてはならないが，“class TkCombobox < TkWindow” の場合は autoload が働いてはならない
  - でも :Ttk がデフォルトの時は autoload が優先されなければならない
- 切替時に定義されているクラスは，autoload されたものかもしれないし，ユーザが独自に定義したものかもしれない．
- ユーザ独自作成のクラスを交えたウィジェットセットが利用されるかもしれない

…などなど

結果として

Tk.default_widget_set = *widget_set*

というシンプルな命令の実装は …

こんな条件判断と動作に…

```
>>> widget_set 管理の改善
関係するもの
 TABLE[set][sym] : obj, string, nil
 ALIASES[sym] : obj, string, undefined
 Object[sym] : obj, string, undefined
 OWNER[sym] : false, set, other, nil

OWNER[sym] は Object[sym] の支配権を表す.
OWNER[sym] が false の時は Object[sym] は設定管理外.
OWNER[sym] == current_widget_set の場合には Object[sym] を設定する.
OWNER[sym] != current_widget_set の場合,
  autoload 設定の場合は Object[sym] には設定しない.
  具体的なオブジェクトを設定する場合は, Object[sym] が未定義なら設定する.

autoload 対象のファイルにおいて,
ファイルとクラスとの対応をテーブルに設定する処理を呼ぶようにする.
(1) __regist_toplevel_aliases__(set, obj, sym) を呼んだとき
    sym を管理下に置くための (初期) 登録処理を行う.
    obj は実 object または autoload file である.
    TABLE[set][sym] = obj
    ALIASES[sym] が未定義, または set == current_widget_set であるとき,
    |---->
        管理権を取り ALIASES を設定.
        OWNER[sym] <- set
        ALIASES[sym] <- obj
        obj は file ではないとき
        |---->
            obj を Object[sym] に登録する.
            Object[sym] <- obj

(2) __set_toplevel_aliases__(set, obj, sym) を呼んだとき
    obj は autoload file ではない.
    set == cur か?
    |YES ->
        OWNER[sym] は?
        |false ->
            Object[sym] は操作対象外.
            set == cur なので ALIAS[sym] は支配下として設定.
            Object[sym] はそのまま.
            OWNER[sym] <- false (==変更なし)
            ALIASES[sym] <- TABLE[set][sym] <- obj
        |set ->
            Object[sym] が定義済みか?
            |YES ->
                Object[sym] の定義あり.
                ALIASES[sym] == TABLE[set][sym] のはず.
                ( もし違っていれば不正規に設定がなされたことを意味するので,
                  動作を保証する必要なし. )
                Object[sym] == obj か?
                |YES ->
                    定義済みの Object[sym] == obj を
                    管理下に移行しようとしているものと推定.
                    Object[sym] はそのまま.
                    OWNER[sym] <- set (==変更なし)
                    Object[sym] <- obj (==変更なし)
                    ALIASES[sym] <- TABLE[set][sym] <- obj
                    ( ALIASES[sym] と TABLE[set][sym] には
                      変更される場合と変更なしの場合があるが,
                      どちらであっても問題なし. )
                |NO ->
                    Object[sym] == ALIASES[sym] ( == TABLE[set][sym] ) か?
                    |YES ->
                        管理下で適切に設定されたものと推定されるため,
                        やりたいことは, 多分, set の管理対象の設定変更.
                        OWNER[sym] <- set (==変更なし)
                        Object[sym] <- obj
                        ALIASES[sym] <- TABLE[set][sym] <- obj
                    |NO  ->
                        Object[sym] は, 多分, ユーザが別途定義したものなので,
                        Object[sym] の変更はしない.
                        OWNER[sym] <- false
                        Object[sym] はそのまま.
                        ALIASES[sym] <- TABLE[set][sym] <- obj
            |NO ->
                Object[sym] の定義なし.
                set についての sym の単純な設定変更.
                set == cur についての設定なので, Object[sym] も新規設定.
                OWNER[sym] <- set (==変更なし)
                Object[sym] <- obj
                ALIASES[sym] <- TABLE[set][sym] <- obj
    |other ->
        現 set での登録なので, 支配下に置くことを意図していると推測.
        OWNER[sym] <- set
        Object[sym] <- obj
        ALIASES[sym] <- TABLE[set][sym] <- obj
    |nil ->
        新規設定.
        支配者なしなので, 現 set の支配下に置くことを意図と推測.
        OWNER[sym] <- set
        Object[sym] <- obj
        ALIASES[sym] <- TABLE[set][sym] <- obj
    |NO ->
        set != cur
        OWNER[sym] は?
        |false ->
            Object[sym] は操作対象外.
            ALIASES[sym] は定義済みか?
            |YES ->
                ALIASES[sym] は他に設定したもの (cur?) がいるので,
                TABLE[set][sym] についての設定のみを更新.
                Object[sym] はそのまま (定義の有無も維持)
                OWNER[sym] <- false (==変更なし)
                ALIASES[sym] はそのまま.
                TABLE[set][sym] <- obj
            |NO ->
                ALIASES[sym] を誰も管理していない状況,
                つまり, 少なくとも cur が管理する sym ではないので,
                とりあえず ALIASES[sym] は obj に設定しておく.
                Object[sym] はそのまま (定義の有無も維持)
                OWNER[sym] <- false (==変更なし)
                ALIASES[sym] <- TABLE[set][sym] <- obj
        |set ->
            Object[sym] は支配下なので, 対応オブジェクトの変更と推測.
            set != cur ではあるが, cur が支配対象としていない状況での
            明示的な指定であるため, Object[sym] への設定を行う.
            OWNER[sym] <- set (==変更なし)
            Object[sym] <- obj
            ALIASES[sym] <- TABLE[set][sym] <- obj
        |other ->
            他者が支配しているため, TABLE[set][sym] の設定のみを行う.
            Object[sym] はそのまま (定義の有無も維持)
            OWNER[sym] はそのまま
            ALIASES[sym] はそのまま
            TABLE[set][sym] <- obj
        |nil ->
            新規設定.
            set != cur ではあるが, cur が支配対象としていない状況での
            明示的な指定であるため, Object[sym] への設定を行う.
            cur は ALIAS[sym] と無関係なので, ALIAS[sym] は支配下に置く.
            OWNER[sym] <- set
            Object[sym] <- obj
            ALIASES[sym] <- TABLE[set][sym] <- obj
```

… まだ続く

これだけ手間をかけたおかげで…

```
Tk.default_widget_set = widget_set
```

（もし詳しい動作を知りたいという物好きな人がいたら，尋ねてね）

開発者が苦労して，利用者は快適に楽しく使う…

開発者が苦労して，利用者は快適に楽しく使う…

まさに Ruby 的！

開発者が苦労して，利用者は快適に楽しく使う…

まさに Ruby 的！

快適な機能や便利メソッドを使うときには
優雅な水鳥の水掻きに思いをはせ，
開発者への感謝の気持ちを忘れないようにしよう！

（Ruby/Tk に関しては決して求めたりしてはいないので念のため）