Gura モジュールリファレンス – tcl, tk

Updated: August 25, 2012

目次

1. このリファレンスについて 3
2. オーバービュー 4
2.1. モジュール構成 4
2.2. 命名規則 4
2.3. イベントの扱い 4
3. tcl モジュール 6
3.1. モジュール関数 6
3.2. tcl.interp クラス 6
3.2.1. インスタンスの生成 6
3.2.2. インスタンスメソッド 6
3.3. tcl.variable クラス 6
3.3.1. インスタンスプロパティ 6
3.4. tcl.timer クラス 6
3.4.1. インスタンスメソッド 6
4. tk モジュール 7
4.1. モジュール関数 7
4.2. image クラスの拡張 7
4.2.1. インスタンスメソッド 7
4.3. tk.Widget クラス 8
4.3.1. インスタンスの生成 8
4.3.2. インスタンスメソッド 9
4.4. tk.Menu クラス 12
4.4.1. インスタンスメソッド 12
4.5. tk.Canvas クラス 13
4.6. tk.CanvasItem クラス 13
4.7. tk.CanvasTag クラス 13
4.8. tk.Text クラス 13
4.9. tk.TextTag クラス 13
4.10. tk.Notebook クラス 13
4.11. tk.Treeview クラス 13
4.12. tk.TreeviewItem クラス 13
4.13. tk.TreeviewTag クラス 13
4.14. tk.FontT クラス 13
4.15. tk.Image クラス 13
4.16. 注意事項 13

1. このリファレンスについて

本リファレンスは Gura の tk モジュールで定義されている関数やクラスの仕様について説明します。
Tcl/Tk のオフィシャルサイトは http://www.tcl.tk/ です。

2. オーバービュー

2.1. モジュール構成

モジュールは、Tcl インタープリタとのインターフェースを提供するモジュール `tcl`、そのモジュールを使って Tk の機能を実現するモジュール `tk` からなります。

Tcl/Tk には多くの関数がありますが、`tcl` や `tk` の中でそれらの関数を個別に実装しているわけではありません。ユーザが呼び出したメソッド名と引数をもとにして tcl のコマンドを生成し、tcl インタープリタに渡して実行しています。

2.2. 命名規則

ウィジェットを生成する手続きは、親ウィジェットを表す変数のメソッドとして実装されます。

ウィジェットを生成する手続きは、最初の文字を大文字にした名前のメソッドになります。例えば、canvas は Canvas になります。

名前空間 ttk に属する手続きは、`"ttk$"` につづけて、最初の文字を大文字にした名前をつけたメソッドになります。例えば、`ttk::button` は `ttk$Button` になります。

2.3. イベントの扱い

Tcl/Tk は、イベントが発生すると関連付けられたプロシージャを評価します。この際、プロシージャ中にパーセント記号 `"%"` と一文字のアルファベットまたは記号が記述されていると、評価の前にその部分をイベントに付随するパラメータ値に置換します。例えば、マウスをクリックしたときに発生する ButtonPress イベントが発生すると、関連付けたプロシージャ中の `%x` や `%y` という部分をそれぞれマウスの x, y 座標数値に置換してから評価が行われます。置換文字列の一覧は http://www.tcl.tk/man/tcl8.5/TkCmd/bind.htm にまとめられています。

Gura はブロックでイベントハンドラを指定できますが、イベントのパラメータ値はブロック引数として受け取ります。このとき、ブロック引数の名前が、Tcl/Tk の置換文字列の名前に対応します。上の例を使えば、ブロック引数に x, y という名前の引数を指定すると、これらに x, y 座標値が入ります。一般の関数呼び出しの慣例では引数の位置によって受け渡しする値を指定しますが、Tcl/Tk のイベントハンドラでは名前で受け取る値が決まることに注意してください。

また、Tcl/Tk が置換する結果は文字列になるので、数値など他の方で受け取りたい場合はブロック引数に明示的に型指定をしてください。

ブロック引数では、Tcl/Tk が定義する一文字の識別子のほかに、可読性を高めるため以下の別名も受け付けられるようにしています。

引数名	Tcl/Tk
serial	#
above	a
numbuttons	b
count	c
detail	d

引数名	Tcl/Tk
place	p
state	s
time	t
width	w
x	x

引数名	Tcl/Tk
root	R
subwidget	S
type	T
widget	W
x_root	X

user_data	d
focus	f
height	h
widget_num	i
keycode	k
mode	m
override_redirect	o

y	y
char	A
border_width	B
delta	D
send_event	E
keysym	K
keysym_num	N

y_root	Y
action	d
index	i
wouldbe	P
current	s
edited	S
validate	V

この対応づけは、tk.bindSubstDict 変数に辞書として登録されており、必要に応じて追加することができます。例えば、%h で置換されるイベントパラメータを hoge という引数名で受け取りたい場合は、以下のようなコードを追加します。

```
tk.bindSubstDict[`hoge] = "h"
```

3. tcl モジュール

3.1. モジュール関数

```
tcl.Tk_MainLoop()
```

3.2. tcl.interp クラス

3.2.1. インスタンスの生成

```
tcl.interp()
```

3.2.2. インスタンスメソッド

```
tcl.interp#command(func:function)
```

```
tcl.interp#eval(objs+)
```

```
tcl.interp#evalscript(script:string)
```

```
tcl.interp#timer()
```
tcl.timer インスタンスを生成します。

```
tcl.interp#variable(value?, varName?:string)
```
tcl.variable インスタンスを生成します。

3.3. tcl.variable クラス

3.3.1. インスタンスプロパティ

3.4. tcl.timer クラス

3.4.1. インスタンスメソッド

```
tcl.timer#cancel()
```

```
tcl.timer#start(msec:number, msecCont?:number, count?:number):reduce {block}
```

4. tk モジュール

4.1. モジュール関数

```
tk.bell(args*, opts%):map

tk.bind(args*, opts%):map

tk.bindtags(args*, opts%):map

tk.tkerror(args*, opts%):map

tk.update()

tk.winfo$atom(args*, opts%):map

tk.winfo$atomname(args*, opts%):map

tk.winfo$containing(args*, opts%):map

tk.winfo$interps(args*, opts%):map

tk.winfo$pathname(args*, opts%):map

tk.tk$FocusFollowsMouse(args*, opts%):map

tk.tk$SetPalette(args*, opts%):map

tk.tk$bisque(args*, opts%):map

tk.tkwait$variable(args*, opts%):map

tk.tkwait$visibility(args*, opts%):map

tk.tkwait$window(args*, opts%):map

tk.mainloop()

tk.tclDump(flag:boolean => true)
```

4.2. image クラスの拡張

4.2.1. インスタンスメソッド

tk モジュールをインポートすると、image クラスに以下のメソッドが追加されます。

```
image#to_tk()
```

4.3. tk.Widget クラス

4.3.1. インスタンスの生成

例えば、button ウィジェットを生成するメソッドの一般式は以下のようになります。

```
tk.Widget#Button(options%)
```

このメソッドは、Tcl コマンドとして"button pathName %options%" を実行します。pathName は新しく生成する button ウィジェットのパス名、options は tk.Widget#Button メソッドに辞書形式で渡した引数の内容が渡されます。

ウィジェット生成メソッドの戻り値は、新しく生成したウィジェットのパス名を持った tk.Widget インスタンスです。menu ウィジェットや text ウィジェットのように、ウィジェット特有のメソッドを持つものについては、tk.Widget クラスから派生したクラスのインスタンスを返します。

ウィジェット生成メソッドと、生成する Tk ウィジェットの対応表を以下にまとめます。

メソッド	生成する Tk ウィジェット	メソッドの戻り値
tk.Widget#Button	button	tk.Widget
tk.Widget#Canvas	canvas	tk.Widget
tk.Widget#Checkbutton	checkbutton	tk.Widget
tk.Widget#Entry	entry	tk.Widget
tk.Widget#Frame	frame	tk.Widget
tk.Widget#Label	label	tk.Widget
tk.Widget#Labelframe	labelframe	tk.Widget
tk.Widget#Listbox	listbox	tk.Widget
tk.Widget#Menu	menu	tk.Menu
tk.Widget#Menubutton	menubutton	tk.Widget
tk.Widget#Message	message	tk.Widget
tk.Widget#Panedwindow	panedwindow	tk.Widget
tk.Widget#Radiobutton	radiobutton	tk.Widget
tk.Widget#Scrollbar	scrollbar	tk.Widget
tk.Widget#Spinbox	spinbox	tk.Widget
tk.Widget#Text	text	tk.Text
tk.Widget#Toplevel	toplevel	tk.Widget
tk.Widget#ttk$Button	ttk::button	tk.Widget
tk.Widget#ttk$Checkbutton	ttk::checkbutton	tk.Widget
tk.Widget#ttk$Combobox	ttk::combobox	tk.Widget
tk.Widget#ttk$Entry	ttk::entry	tk.Widget
tk.Widget#ttk$Frame	ttk::frame	tk.Widget
tk.Widget#ttk$Intro	ttk::intro	tk.Widget
tk.Widget#ttk$Label	ttk::label	tk.Widget

tk.Widget#ttk$Labelframe	ttk::labelframe	tk.Widget
tk.Widget#ttk$Menubutton	ttk::menubutton	tk.Widget
tk.Widget#ttk$Notebook	ttk::notebook	tk.Notebook
tk.Widget#ttk$Panedwindow	ttk::panedwindow	tk.Widget
tk.Widget#ttk$Progressbar	ttk::progressbar	tk.Widget
tk.Widget#ttk$Radiobutton	ttk::radiobutton	tk.Widget
tk.Widget#ttk$Scale	ttk::scale	tk.Widget
tk.Widget#ttk$Scrollbar	ttk::scrollbar	tk.Widget
tk.Widget#ttk$Separator	ttk::separator	tk.Widget
tk.Widget#ttk$Sizegrip	ttk::sizegrip	tk.Widget
tk.Widget#ttk$Spinbox	ttk::spinbox	tk.Widget
tk.Widget#ttk$Style	ttk::style	tk.Widget
tk.Widget#ttk$Treeview	ttk::treeview	tk.Treeview
tk.Widget#ttk$Widget	ttk::widget	tk.Widget
tk.Widget#ttk$Image	ttk::image	tk.Widget
tk.Widget#ttk$Vsapi	ttk::vsapi	tk.Widget

4.3.2. インスタンスメソッド

`tk.Widget#wm$aspect(minNumber?, minDenom?, maxNumber?, maxDenom?):map`

Tcl コマンド: `wm aspect window ?minNumber mindenom maxNumber maxDenom`

`tk.Widget#wm$attributes(opts%):map`

Tcl コマンド: `wm attributes window`

`tk.Widget#wm$client(name?):map`

Tcl コマンド: `wm client window ?name?`

`tk.Widget#wm$colormapwindows(windowList?):map`

Tcl コマンド:

`tk.Widget#wm$command(value?):map`

Tcl コマンド:

`tk.Widget#wm$deiconify():map`

Tcl コマンド:

`tk.Widget#wm$focusmodel(args*, opts%):map`

Tcl コマンド:

`tk.Widget#wm$forget():map`

Tcl コマンド:

tk.Widget#wm$frame():map

Tcl コマンド:

tk.Widget#wm$geometry(newGeometry):map

Tcl コマンド:

tk.Widget#wm$grid(baseWidth?, baseHeight?, widthInc?, heightInc?):map

Tcl コマンド:

tk.Widget#wm$group(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconbitmap(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconify(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconmask(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconname(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconphoto(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconposition(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$iconwindow(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$manage(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$maxsize(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$minsize(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$overrideredirect(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$positionfrom(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$protocol(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$resizable(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$sizefrom(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$stackorder(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$state(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$title(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$transient(args*, opts%):map

Tcl コマンド:

tk.Widget#wm$withdraw():map

Tcl コマンド:

tk.Widget#winfo$(args*, opts%):map

Tcl コマンド:

tk.Widget#grid$(args*, opts%):map

Tcl コマンド:

tk.Widget#place$(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$bisque(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$chooseColor(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$chooseDirectory(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$dialog(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$focusFollowsMouse(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$focusNext(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$focusPrev(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$getOpenFile(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$getSaveFile(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$menuSetFocus(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$messageBox(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$optionMenu(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$popup(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$setPalette(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$textCopy(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$textCut(args*, opts%):map

Tcl コマンド:

tk.Widget#tk$textPaste(args*, opts%):map

Tcl コマンド:

4.4. tk.Menu クラス

4.4.1. インスタンスメソッド

tk.Menu#Separator(opts%)

tk.Menu#Cascade(opts%) {block?}

tk.Menu#Command(opts%) {block?}

tk.Menu#Checkbutton(opts%) {block?}

```
tk.Menu#Radiobutton(opts%) {block?}
```

4.5. tk.Canvas クラス

```
tk.Canvas#Tag() {block?}
```

4.6. tk.CanvasItem クラス

4.7. tk.CanvasTag クラス

4.8. tk.Text クラス

4.9. tk.TextTag クラス

4.10. tk.Notebook クラス

4.11. tk.Treeview クラス

4.12. tk.TreeviewItem クラス

4.13. tk.TreeviewTag クラス

4.14. tk.FontT クラス

4.15. tk.Image クラス

4.16. 注意事項

Tk イメージオブジェクトは、明示的にメソッド `destroy()` を実行しないとメモリから削除されません。