

# ライブラリスクリプトを利用した vimエディタプラグインの構築

名無しのvim使い <http://nanasi.jp/>  
小見 拓 [mail@nanasi.jp](mailto:mail@nanasi.jp)

ライブラリスクリプトとは

ライブラリスクリプト

=

vimエディタのスクリプト

読み込みの仕組み

+

名前空間もどき

# ライブラリスクリプトの使い方①

- runtimepathディレクトリのautoload以下のディレクトリにvimスクリプトを置く。

" Windowsの場合

\$HOME/.vim/autoload/

" Mac OSX、Linux系OS環境の場合

\$HOME/vimfiles/autoload/

" 例

\$HOME/.vim/autoload/sample/xxx/testa.vim

## ライブラリスクリプトの使い方②

- autoloadに置いたスクリプトで、一定の命名規則に従ってファンクション、変数を定義。

```
" vimスクリプトファイル  
$HOME/.vim/autoload/sample/xxx/testa.vim
```

```
" ファンクションの例  
function sample#xxx#testa#testFunc()  
    " 適当なコード  
endfunction
```

```
" 変数の例  
let sample#xxx#testa#sampleVar = "変数の値(文字列型)"
```

# ライブラリスクリプトの使い方③

- ライブラリスクリプトのコードを呼び出すには。

" 関数呼び出し

```
call sample#xxx#testa#testFunc()
```

" 変数読み込み

```
let value = sample#xxx#testa#sampleVar
```

# ライブラリスクリプトの使い方④

- 詳しくは次のヘルプを読むべし。

`:help write-library-script`

`:help autoload`

# おまけ

ライブラリスクリプトを使って、  
実際にvimエディタのプラグインを  
作ります。

# ライブラリっぽく使われてるスクリプト

- cecutil : Some utilities used by several of my scripts (window positioning, mark handling)  
[http://www.vim.org/scripts/script.php?script\\_id=1066](http://www.vim.org/scripts/script.php?script_id=1066)
- genutils : General utility functions  
[http://www.vim.org/scripts/script.php?script\\_id=197](http://www.vim.org/scripts/script.php?script_id=197)
- multvals.vim : Array library that uses patterns as separators  
[http://www.vim.org/scripts/script.php?script\\_id=171](http://www.vim.org/scripts/script.php?script_id=171)
- Align : Provides commands and maps to help produce aligned text, eqns, declarations, etc  
[http://www.vim.org/scripts/script.php?script\\_id=294](http://www.vim.org/scripts/script.php?script_id=294)
- alice.vim : Chalice  
<http://www.kaoriya.net/>

# 今回使用するライブラリ

## tlib : Some utility functions

Thomas Link <micathom at gmail com>

[http://www.vim.org/scripts/script.php?script\\_id=1863](http://www.vim.org/scripts/script.php?script_id=1863)

- ウィンドウを分割して動作するシステムのプラグインを簡単に作れるライブラリ。  
(本当は他にも機能はある)

tlibを使うと、こういうのが簡単に作れます。

ウィンドウを開いて、何か  
データを表示する。

```
demo_tlib.vim (~//l_vimm_200807) - VIM
37 \ {'key':16, 'key_name': '<c-p>', 'help': 'push Control-P.', 'agent': 'g:CallCtrlP' },
38 \ {'key':21, 'key_name': '<c-u>', 'help': 'push Control-U.', 'agent': 'g:CallCtrlU' },
39 \ ]
40
41
42 " 開いたスクラッチウィンドウで表示する一覧
43 " 仮実装のコメントアウト
44 " let l:world.base = [ "A", "B", "C" ]
45
46 " 外部コマンドを実行
47 let l:ls_result = system('ls /usr/bin')
48
49 " 結果を編集してリストに変換
50 let l:ls_list = split(l:ls_result, '\n')
51
52 " baseキーにセットする
53 let l:world.base = l:ls_list
54
55
56 " 渡されたWorldオブジェクトのパラメータを元に、
57 " スクラッチウィンドウを開きます。
58 call tlib#input#ListW(l:world)
59
60 endfunction
61
62
63 " Control-Aを押すと呼び出される
demo_tlib.vim [cp932][unix]
```

ユーザがリストを選択したら、  
何か処理を実行する。

```
CH_NAME (~//l_vimm_200807) - VIM
37 \ {'key':16, 'key_name': '<c-p>', 'help': 'push Control-P.', 'agent': 'g:CallCtrlP' },
38 \ {'key':21, 'key_name': '<c-u>', 'help': 'push Control-U.', 'agent': 'g:CallCtrlU' },
39 \ ]
40
41
42 " 開いたスクラッチウィンドウで表示する一覧
43 " 仮実装のコメントアウト
44 " let l:world.base = [ "A", "B", "C" ]
45
46 " 外部コマンドを実行
47 let l:ls_result = system('ls /usr/bin')
48
49 " 結果を編集してリストに変換
50 let l:ls_list = split(l:ls_result, '\n')
51
52 " baseキーにセットする
53 let l:world.base = l:ls_list
54
55
56 " 渡されたWorldオブジェクトのパラメータを元に、
57 " スクラッチウィンドウを開きます。
58 call tlib#input#ListW(l:world)
59
60 endfunction
61
62
63 " Control-Aを押すと呼び出される
(filter: (); press "?" for help) (filter: (); press "?" for help)
OutCommandViewer
```

# tlibの簡単な使い方

- Worldオブジェクトを生成。(正体はディクショナリ)  
let world = tlib#World#New()
- ListWファンクションに渡す。  
call tlib#input#ListW(world)
- 別ウィンドウが開かれる。

# どのようなプラグインを作るか

- コマンドを実行すると、外部コマンドを実行する。
- 別ウィンドウを開いて、そこに外部コマンドの結果を表示する。
- ユーザに外部コマンドの結果の中から何かを選択させる。
- 何らかの処理を実行する。

# tlibの使い方がわからない場合

- tlibのドキュメントを読もう。
- コードを読もう。
- tlibを使っているプラグインがいくつかあるから、そいつらを参考にしよう。

ご静聴ありがとうございました。

気が向いたら、何かプラグインを作ってみて  
ください

終わり