Design Patterns et Python

TP Design Pattern en Python

Eric Ramat
ramat@lil.univ-littoral.fr

17 janvier 2008

Durée : 6 heures

---

1 Objectif

On désire développer une petite application en Python qui propose une fenêtre graphique dans laquelle on puisse instancier des plugins interagissant sur une donnée simple. Cette donnée simple peut être un simple compteur entier initialisé à zéro. Deux types de plugins sont à developper :

- les boutons :
  - un bouton '+' sur lequel on appuie pour que le compteur s'incrémente ;
  - un bouton '-' sur lequel on appuie pour que le compteur se décremente ;
- les vues :
  - une représentation graphique du compteur à l'aide d'une barre graphique ;
  - une réprésentation du compteur sous forme d'un texte.

2 Contraintes

- le compteur est naturellement unique pour l'application. S'il n'existe pas au lancement des plugins, il faut qu'il soit créé.
- dès qu'un bouton modifie le compteur alors les vues sont mises à jour ;
- les boutons et les vues sont chargés dynamiquement c'est à dire que la fonction principale __main__ doit possèder des appels du type :

```
self.add_button("Plus")
self.add_button("Moins")
self.add_button("RAZ")

self.add_view("TextView")
self.add_view("ProgressBarView")
```

- à l'aide d'une chaîne de caractères et du type d'objets, l'application doit être capable de construire une instance de contrôleur et de vues et de la placer dans l'interface ;
- **on utilisera au maximum les design patterns.**

3 Exemple

4 Outils

On utilisera pour cela les packages IMP (fonction d'importation dynamique de modules), INSPECT (fonction d'introspection des classes et objets) et PyGTK (portage Python de GTK - http://www.pygtk.org) de Python. Les documentations sont disponibles aux urls suivantes :

- file:///usr/share/doc/python2.5-doc/html/index.html
- file:///usr/share/doc/python-gtk2-doc/html/index.html

5 Quelques pistes

Le début du code ressemble à ça :

```
class Application:
  """ """
  def destroy(self, widget, data=None):
      gtk.main_quit()

  def __init__(self):
      self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
      self.window.connect("destroy", self.destroy)

      self.window.show()

  def main(self):
      gtk.main()

if __name__ == "__main__":
  counter = Counter(3)
  application = Application()
  application.main()
```

- la fenêtre principale est une Window découpée avec des VBox et HBox ;
- une vue est une Frame attachée à une VBox ou HBox ;
- les contrôleurs et les vues sont des plugins chargés dynamiquement avec une interface fonctionnelle identique.

6 Références

- Python : http://www.python.org
- IMP : http://docs.python.org/lib/module-imp.html
- INSPECT : http://docs.python.org/lib/module-inspect.html
- PyGTK : http://www.pygtk.org