DESIGN PATTERNS - CHAPITRE 6

Prototype en Python

Créer une variante à partir d'un exemplaire existant en choisissant ce qui reste partagé.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Créer une variante à partir d'un exemplaire existant en choisissant ce qui reste partagé.
- Comparer copie superficielle et copie profonde d'un état imbriqué.

Prérequis. Savoir utiliser classes, héritage, composition, listes, dictionnaires et exceptions Python. Les mécanismes spécifiques sont expliqués avant leur emploi.

Prototype : intention et expérience

Copier ou reconstruire ?

Vous disposez d'un modèle déjà configuré et voulez produire une variante. Le patron **Prototype** crée un objet à partir d'un exemplaire existant au lieu de reconstruire sa configuration. Définissez d'abord quelles parties doivent rester partagées et lesquelles doivent être indépendantes.

Le module standard `copy` propose une copie superficielle (`copy`) qui conserve les références vers les objets imbriqués, et une copie profonde (`deepcopy`) qui copie récursivement le graphe selon les protocoles des objets. Ce n'est pas une garantie de duplication de toutes les ressources : fichiers ouverts et connexions demandent une politique explicite.

Dans l'expérience, ajoutez une section au clone profond et vérifiez que le modèle reste inchangé. Faites ensuite le même ajout sur la copie superficielle : sa liste est celle du modèle. Copier coûte du temps et de la mémoire ; un constructeur explicite reste préférable si l'état est petit ou doit être revalidé.

Python

```python
# example.py
from copy import copy, deepcopy

class ReportPrototype:
    def __init__(self, title, sections):
        self.title = title
        self.sections = sections

original = ReportPrototype("Model", ["Introduction"])
shallow = copy(original)
independent = deepcopy(original)
independent.title = "New report"
independent.sections.append("Results")

print(original.sections)     # ['Introduction']
print(independent.sections)  # ['Introduction', 'Results']
assert shallow.sections is original.sections
assert independent.sections is not original.sections
```

Pour approfondir, consultez **la documentation des copies superficielles et profondes**.

Vérifier et interpréter

Modifiez la liste imbriquée d'une copie superficielle et observez le modèle, puis recommencez avec une copie profonde. Les deux politiques ne promettent pas la même indépendance. Définissez ce qui doit être partagé avant de copier : dupliquer une configuration ne revient pas à cloner ses ressources externes.

Version web : https://www.clearcode.fr/courses/programming/design-patterns/creation/prototype