DESIGN PATTERNS - CHAPITRE 21

Memento en Python

Capturer un état puis le restaurer sans demander à l'historique de le reconstruire.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Capturer un état puis le restaurer sans demander à l'historique de le reconstruire.
- Vérifier qu'un cliché conserve le texte et la position du curseur.

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

Memento : intention et expérience

Restaurer sans rejouer les actions

Comment annuler une modification sans demander à l'historique de comprendre tous les détails de l'éditeur ? Le patron **Memento** sauvegarde un état puis permet à son propriétaire de le restaurer. L'éditeur est l'origine de l'état ; le cliché est le memento ; la liste d'historique conserve les clichés.

`dataclass`, du module standard `dataclasses`, génère notamment un constructeur à partir des champs déclarés. `frozen=True` interdit ici de réaffecter ces champs. Leurs valeurs sont une chaîne et un entier, donc immuables ; cela évite qu'un changement de l'éditeur modifie le cliché.

Le contrat `save()` capture texte et position du curseur ; `restore()` retrouve les deux. Testez aussi deux clichés successifs. Cette version ne masque pas techniquement les champs : l'historique doit les traiter comme opaques. Des états volumineux ou très fréquents rendent une sauvegarde complète coûteuse.

Python

```python
# example.py
from dataclasses import dataclass

@dataclass(frozen=True)
class Snapshot:
    text: str
    cursor: int

class Editor:
    def __init__(self):
        self.text = ""
        self.cursor = 0

    def insert(self, fragment):
        self.text = self.text[:self.cursor] + fragment +
self.text[self.cursor:]
        self.cursor += len(fragment)

    def save(self):
        return Snapshot(self.text, self.cursor)

    def restore(self, snapshot):
        self.text = snapshot.text
        self.cursor = snapshot.cursor

editor = Editor()
editor.insert("Hello")
history = [editor.save()]
editor.insert(" world")
editor.restore(history.pop())
print(editor.text, editor.cursor)  # Hello 5
```

Pour approfondir, consultez **la documentation de dataclasses**.

Vérifier et interpréter

Sauvegardez deux états, puis restaurez-les dans l'ordre inverse. Texte et curseur doivent revenir ensemble à leurs valeurs antérieures. Un cliché complet simplifie cette restauration, mais l'historique peut coûter cher en mémoire ; ne restaurez pas un état devenu incompatible avec des ressources externes modifiées depuis.

Version web : https://www.clearcode.fr/courses/programming/design-patterns/comportement/memento