DESIGN PATTERNS - CHAPITRE 13

Proxy en Python

Contrôler l'accès à un service tout en exposant la même interface.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Contrôler l'accès à un service tout en exposant la même interface.
- Expliquer la décision de déléguer et ses limites dans une simulation.

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

Proxy : intention et expérience

Proxy : contrôler l'accès

Un **Proxy** conserve l'interface du service et décide quand déléguer. Ce peut être un contrôle d'accès, une initialisation différée ou une représentation distante. Ici, les messages illustrent la décision ; ils ne réalisent pas une authentification.

Python

```python
# example.py
# Subject interface
class Service:
    def operation(self):
        pass

# Real Subject
class RealService(Service):
    def operation(self):
        return "RealService: Handling request."

# Proxy
class Proxy(Service):
    def __init__(self, real_service):
        self.real_service = real_service

    def operation(self):
        # Add some logic here, like access control, logging, etc.
        print("Proxy: Checking access prior to firing a real request.")
        if self.check_access():
            result = self.real_service.operation()
            self.log_access()
            return result
        else:
            return "Proxy: Access denied."

    def check_access(self):
        # Simulate access check
        return True

    def log_access(self):
        print("Proxy: Logging the time of request.")

# Client code
if __name__ == "__main__":
    real_service = RealService()
```

```
    proxy = Proxy(real_service)
    print(proxy.operation())
```

Vérifier et interpréter

Remplacez le service par un faux qui compte les délégations. Vérifiez qu'une requête autorisée atteint le service exactement une fois. Si vous ajoutez un refus, ce compteur doit rester inchangé. Le proxy doit préserver les résultats et les erreurs contractuels ; son contrôle ajoute une étape d'exécution et de diagnostic.

Version web : https://www.clearcode.fr/courses/programming/design-patterns/structure/proxy