LE DESIGN PATTERN COMMAND

1. Introduction

1. INTRODUCTION

Le pattern Command est un pattern comportemental qui transforme une requête en un objet indépendant contenant toutes les informations sur cette requête. Cette transformation permet de paramétrer des méthodes avec différentes requêtes, de mettre en file d'attente ou de journaliser les requêtes, et de supporter les opérations d'annulation.

2. INTENTION

'intention principale du pattern Command est de :

- Encapsuler une requête sous forme d'objet
- Découpler l'objet qui invoque l'opération de celui qui la réalise
- Permettre la paramétrisation des clients avec différentes requêtes
- Permettre la paramétrisation des clients avec différentes requêtes
- Supporter l'annulation (undo) et la répétition (redo) d'opérations
- Journaliser les requêtes pour audit ou persistance

3. MOTIVATION

Dans de nombreuses applications, nous devons :

- Exécuter des actions à différents moments
- Annuler des opérations précédentes
- Journaliser les actions pour audit
- Créer des macros ou des scripts d'automatisation
- Implémenter des systèmes de file d'attente

Le pattern Command résout ces problèmes en encapsulant les requêtes dans des objets, permettant ainsi leur manipulation comme n'importe quel autre objet.

4. STRUCTURE DE CLASSE

Composants principaux :
Command (Interface) : Déclare une interface pour exécuter une opération
ConcreteCommand : Définit une liaison entre un objet Receiver et une action. Implémente execute() en invoquant les opérations correspondantes sur Receiver
Client : Crée un objet ConcreteCommand et définit son receiver
Invoker : Demande à la commande d'exécuter la requête
Receiver : Sait comment effectuer les opérations associées à une requête

5. EXEMPLE CONCRET : SYSTÈME DE TÉLÉCOMMANDE UNIVERSELLE

- Prenons l'exemple d'une télécommande universelle pour maison intelligente qui peut contrôler différents appareils (lumières, télévision, climatisation, etc.).
- Avantages de ce choix :
- Flexibilité : Possibilité d'ajouter de nouveaux appareils sans modifier la télécommande
- Undo/Redo : Possibilité d'annuler la dernière action
- Macros : Création de séquences d'actions
- Journalisation : Enregistrement de toutes les actions effectuées
-

7. AVANTAGES ET INCONVÉNIENTS

Avantages :

- Découplage : Sépare l'objet qui invoque l'opération de celui qui l'exécute
- Flexibilité : Permet de composer des opérations complexes
- Extensibilité : Facile d'ajouter de nouvelles commandes
- Undo/Redo : Support natif des opérations d'annulation
- Journalisation : Possibilité de logger toutes les opérations
- Transactions : Possibilité de regrouper plusieurs opérations

INCONVÉNIENTS :

- Complexité : Augmente le nombre de classes
- Mémoire : Peut consommer plus de mémoire si beaucoup de commandes sont stockées
- Performance : Léger surcoût dû à l'indirection

9. CAS D'USAGE TYPIQUES

- Interfaces utilisateur : Boutons, menus, raccourcis clavier
- Systèmes de file d'attente : Traitement asynchrone de tâches
- Systèmes transactionnels : Base de données, systèmes financiers
- Éditeurs de texte : Undo/Redo, macros
- Systèmes de workflow : Orchestration de processus métier
- APIs REST : Encapsulation des requêtes HTTP

CONCLUSION

Le pattern Command est un outil puissant pour découpler les demandes de leur exécution. Il offre une grande flexibilité dans la gestion des opérations et permet d'implémenter facilement des fonctionnalités avancées comme l'annulation, la journalisation et les macros. Bien qu'il introduise une certaine complexité, ses avantages en font un pattern incontournable dans de nombreux contextes applicatifs.