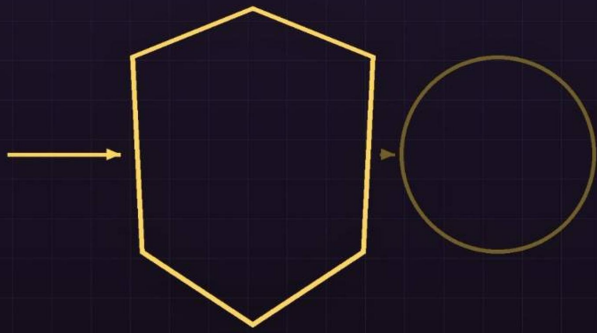
GoF · 12/23 СТРУКТУРНЫЙ

Python C++23 Rust Java



Заместитель

Proxy

Алексей Примак

Алексей Примак

Заместитель. Паттерн GoF на Python, C++23, Rust и Java

https://litres.ru/74810698
SelfPub; 2026

Аннотация

Когда объект тяжёлый, удалённый или требует контроля доступа, между клиентом и реальным объектом можно поставить заместителя. «Заместитель» разбирает ленивую загрузку, кэширование и контроль доступа на практических примерах. Паттерн GoF реализован на Python, C++23, Rust и Java. Книга помогает понять, как сохранить общий интерфейс и при этом управлять моментом и условиями обращения к реальному объекту.

Содержание

Введение 4
Как читать 5
Глава 12. Реализация на Python 6
12.1. Зачем нужен заместитель 6
12.2. Участники и схема 8
12.3. Виртуальный заместитель 11
Конец ознакомительного фрагмента. 14

Заместитель. Паттерн GoF на Python, C ++23, Rust и Java

Введение

Эта книга посвящена одному паттерну проектирования — **Заместитель**(Proxy). Он относится к каталогу «Банды четырёх» (Gang of Four, GoF). Здесь паттерн разобран сразу в четырёх реализациях: на Python, C++23, Rust и Java. Каждая реализация — отдельная часть книги: сначала задача и общая схема, затем код, модификации, гибридные варианты и самостоятельные работы.

Как читать

Основы каждого языка здесь не повторяются: они собраны в книгах серий по соответствующему языку. Разделы каждой части нумеруются заново, поэтому ссылка «раздел 1.2» указывает на раздел той же части, где она сделана.

Глава 12. Реализация на Python

12.1. Зачем нужен заместитель

Объект изображения весит сотни мегабайт, и загружать его стоит только тогда, когда картинку действительно покажут. Удалённый сервис отвечает медленно, и повторные запросы стоит кешировать. Административные операции должны быть доступны только администраторам. Во всех трёх случаях хочется поставить между клиентом и объектом посредника с тем же интерфейсом.

Заместитель — объект, который подменяет настоящий и контролирует доступ к нему. Клиент не замечает подмены: интерфейс тот же. Различают несколько видов заместителей:

виртуальный— откладывает создание дорогого объекта до первого обращения;

защищающий— проверяет права доступа;

удалённый— представляет объект в другом процессе или на другой машине;

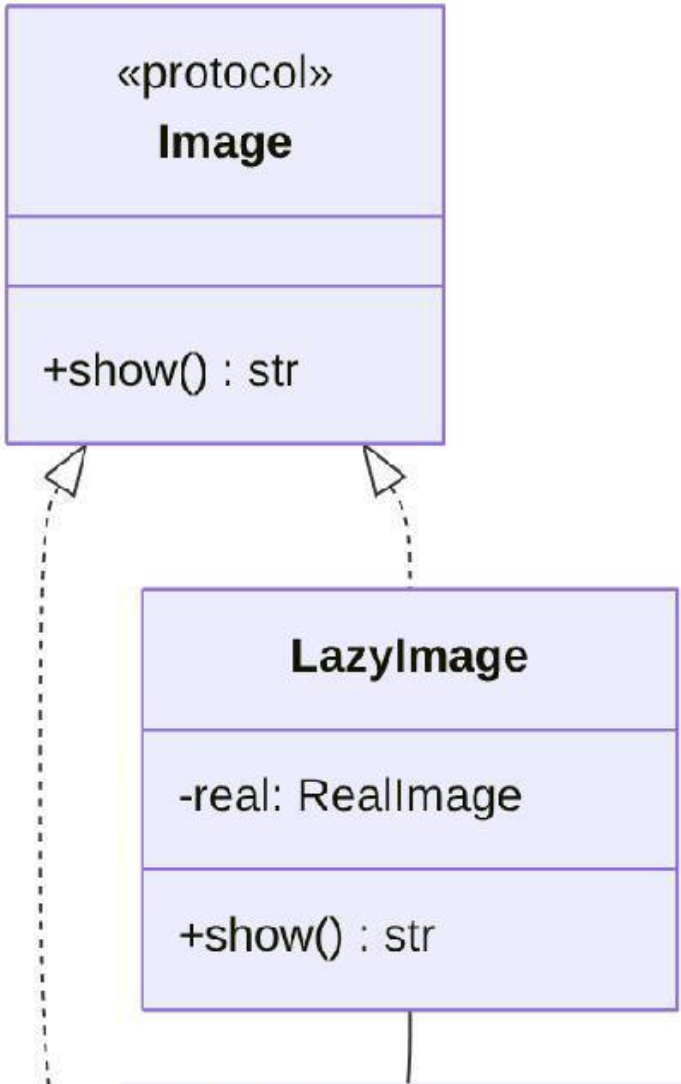
кеширующий— запоминает результаты;

журналирующий— записывает обращения.

Устройством заместитель похож на декоратор, но цель другая: декоратор добавляет поведение, а заместитель управляет доступом к объекту и часто сам управляет его

жизненным циклом.

12.2. Участники и схема


«protocol»
Image
+show() : str
LazyImage
-real: RealImage
+show() : str

Subject— общий интерфейс.

Real Subject— настоящий объект.

Proxy— хранит ссылку на настоящий объект (или способ его получить) и контролирует обращения.

12.3. Виртуальный заместитель

Файл 01_virtual_proxy.py:

01_virtual_proxy.py Python

```python
from typing import Protocol


class Image(Protocol):
    def show(self) -> str: ...


class RealImage:
    loaded = 0

    def __init__(self, path: str) -> None:
        RealImage.loaded += 1
        self.path = path
        self.pixels = bytes(1_000_000)  # тяжёлая загрузка
        print(f"  [загружен {path}]")

    def show(self) -> str:
        return f"{self.path}: {len(self.pixels) // 1000} КБ"


class LazyImage:
    def __init__(self, path: str) -> None:
        self.path = path
        self._real: RealImage | None = None

    def show(self) -> str:
        if self._real is None:
            self._real = RealImage(self.path)
        return self._real.show()


if __name__ == "__main__":
    gallery: list[Image] = [LazyImage(f"photo_{i}.jpg") for i in
        range(100)]
    print("Галерея создана, загружено:", RealImage.loaded)
    for img in gallery[:3]:  # пользователь посмотрел три фото
        print(img.show())
    print(gallery[0].show())  # повторный показ без загрузки
    print("Загружено:", RealImage.loaded)
```

```
$ python 01_virtual_proxy.py
```

Листинг 12.1 *— Виртуальный заместитель*

Разбор. Сто заместителей почти ничего не весят, а загружены только три изображения, которые реально показали. Повторный показ использует уже загруженный объект. Для простых случаев в Python есть functools.cached_property: атрибут вычисляется при первом обращении и запоминается.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.