Wzorzec Proxy (Pełnomocnik)
Autor: Tymofii Kyrychenko

Wprowadzenie

Wzorzec Proxy (z ang. pełnomocnik) należy do wzorców strukturalnych. Jego głównym celem jest umożliwienie pośredniego dostępu do innego obiektu — czyli takiego, który faktycznie wykonuje dane operacje. Pełnomocnik przejmuje część odpowiedzialności za tworzenie, inicjalizację lub kontrolę dostępu do tego obiektu. Dzięki temu klient nie musi znać szczegółów implementacji rzeczywistego obiektu.

Przeznaczenie

Wzorzec Proxy (Pełnomocnik) zapewnia kontrolowany dostęp do innego obiektu. Zamiast bezpośrednio odwoływać się do obiektu docelowego, klient korzysta z pośrednika — tzw. pełnomocnika, który może dodawać dodatkowe zachowania, takie jak buforowanie, zdalne wywołania czy kontrola dostępu.

Rodzaje pełnomocników

Typ Proxy	Opis
Remote Proxy	Reprezentuje obiekt znajdujący się w innym adresie (np. na innym serwerze). Stosowany w systemach rozproszonych.
Virtual Proxy	Opóźnia tworzenie kosztownych obiektów, dopóki nie są potrzebne (leniwe ładowanie).
Protection Proxy	Odpowiada za kontrolę uprawnień dostępu do obiektu (np. administrator, użytkownik).
Smart Proxy	Dodaje dodatkowe działania podczas dostępu (np. logowanie, zliczanie referencji, synchronizacja).
Caching Proxy	Buforuje wyniki kosztownych operacji, aby uniknąć ich ponownego wykonywania.

Przykładowe zastosowania

• Systemy autoryzacji i uwierzytelniania (kontrola dostępu).
• Aplikacje sieciowe – np. pobieranie obrazów z serwera dopiero, gdy są potrzebne.
• Zdalne wywoływanie metod (np. w RMI, CORBA, gRPC).
• Buforowanie wyników w systemach o dużym obciążeniu.
• Monitorowanie i logowanie operacji bez modyfikowania kodu aplikacji.
• Optymalizacja pamięci w grach lub systemach graficznych (leniwa inicjalizacja zasobów).

Struktura UML

Client --> Subject ← Proxy --> RealSubject

• Subject – interfejs wspólny dla RealSubject i Proxy.
• RealSubject – właściwy obiekt wykonujący zadanie.
• Proxy – pośrednik, który kontroluje dostęp do RealSubject.
• Client – korzysta z interfejsu Subject, nie wiedząc, czy pracuje z Proxy czy RealSubject.

Zalety

• Umożliwia kontrolę dostępu do rzeczywistego obiektu.
• Pozwala dodawać nowe funkcjonalności bez zmiany istniejącego kodu.
• Poprawia wydajność – dzięki leniwemu tworzeniu lub buforowaniu.
• Zwiększa bezpieczeństwo i modularność.
• Ułatwia zdalną komunikację i ukrywa szczegóły implementacyjne.

Wady

• Wprowadza dodatkowy poziom pośrednictwa – spadek wydajności.
• Może skomplikować strukturę kodu, szczególnie przy wielu typach Proxy.
• Wymaga utrzymania spójnego interfejsu między Proxy a RealSubject.
• Trudniejsze debugowanie – rzeczywisty obiekt jest ukryty za warstwą pośrednią.

Przykład implementacji (Java)

```
interface Image {
   void display();
}

class RealImage implements Image {
   private String filename;

   public RealImage(String filename) {
      this.filename = filename;
      loadFromDisk();
   }

   private void loadFromDisk() {
      System.out.println("Ładowanie obrazu: " + filename);
   }

   public void display() {
      System.out.println("Wyświetlanie obrazu: " + filename);
   }
}

class ProxyImage implements Image {
   private RealImage realImage;
   private String filename;

   public ProxyImage(String filename) {
      this.filename = filename;
   }

   public void display() {
      if (realImage == null) {
         realImage = new RealImage(filename);
      }
      realImage.display();
   }
}
```

Użycie

```
Image img = new ProxyImage("obraz1.png");
img.display();
img.display();
```

Podsumowanie

Wzorzec Proxy to potężne narzędzie projektowe pozwalające na:

• kontrolowanie dostępu,

• optymalizację kosztownych operacji,

• bezpieczne zarządzanie zasobami.

Jest szeroko stosowany w nowoczesnych frameworkach, takich jak Spring, Hibernate, gRPC czy CORBA.
Dzięki niemu możliwe jest utrzymanie czystej architektury i oddzielenie logiki biznesowej od aspektów technicznych (np. bezpieczeństwa, sieci, pamięci).