Politechnika Wrocławska

Praktyka Programowania

Autor instrukcji:

Oskar Krupski, nr albumu 272511

Tytuł instrukcji:

Wzorce projektowe - Pełnomocnik

1. Nazwa

Opisywanym w instrukcji wzorcem projektowym będzie

PEŁNOMOCNIK / PROXY

2. Problem

Pełnomocnik jest strukturalnym wzorcem projektowym, którego celem jest utworzenie obiektu zastępującego inny obiekt. Zasadniczo nadzoruje on dostęp do pierwotnego obiektu, pozwalając na wykonanie jakiejś czynności przed lub po przekazaniu do niego żądania.

Pełnomocnik znajduje zastosowanie w sytuacjach, gdy bezpośredni dostęp do jakiegoś obiektu jest kosztowny (nie potrzebujemy go ciągle, lecz co jakiś czas), niebezpieczny lub wymaga dodatkowej kontroli.

Przykładowe zastosowania pełnomocnika:

- obiekt zużywa wiele zasobów i nie powinien być tworzony od razu (np. obraz w aplikacji graficznej),
- potrzeba kontroli dostępu do obiektu (np. zabezpieczenia lub autoryzacja – karta kredytowa jako pełnomocnik konta bankowego),
- konieczność rejestrowania operacji na obiekcie (np. logowanie wywołań)
- potrzeba zdalnego dostępu do obiektu (np. obiekt znajduje się na innym serwerze).

Warunki wstępne:

- istnieje już interfejs lub klasa, której działania chcemy opakować lub kontrolować,
- potrzebna jest dodatkowa logika pośrednicząca między klientem a właściwym obiektem.

3. Rozwiązanie

Pełnomocnik polega na utworzeniu nowego obiektu, który implementuje ten sam interfejs co obiekt docelowy (tzw. obiekt rzeczywisty) i przekazuje do niego wywołania, dodając przy tym własną logikę (np. sprawdzanie uprawnień, opóźnione tworzenie obiektu, logowanie).

Elementy pełnomocnika:

- Interfejs wspólny / Interfejs usługi – definiuje metody dostępne dla klienta,
- Realny obiekt / Usługa – implementacja rzeczywista, która wykonuje konkretne operacje,
- Pełnomocnik (Proxy) – klasa implementująca ten sam interfejs co realny obiekt i deleguje do niego wywołania, dodając własną logikę.
- Klient (Client) – powinien móc pracować zarówno z obiektami, jak i pełnomocnikami za pośrednictwem tego samego interfejsu. Dzięki temu można umieścić pełnomocnika w dowolnym kodzie, który ma współdziałać z obiektem usługowym.

Wskazówka: Pełnomocnik może zawierać instancję obiektu rzeczywistego jako prywatne pole i tworzyć ją w razie potrzeby (tzw. leniwa inicjalizacja).

Można zaimplementować różne typy pełnomocników: zdalny, wirtualny, zabezpieczający, logujący itd.

4. Konsekwencje

Zalety:

- kontrola dostępu do obiektu,
- opóźnione tworzenie kosztownych obiektów,
- transparentność dla klienta – klient nie musi wiedzieć, czy korzysta z pełnomocnika, czy z obiektu rzeczywistego,
- możliwość rozszerzenia funkcjonalności bez zmiany oryginalnego obiektu.

Wady:

- zwiększona złożoność kodu – wprowadzenie dodatkowego poziomu abstrakcji,
- możliwość opóźnień w działaniu – np. przy zdalnym pełnomocniku lub przy dynamicznym tworzeniu obiektu,
- rudność debugowania – dodatkowa warstwa może utrudniać śledzenie błędów.