Proxy Pattern

Решеников Илья Алексеевич
Преподаватель: Сартасов С.Ю.

СПбГУ
Программная инженерия
271 группа

22 апреля 2019 г.

Проблема

- Необходимо контролировать доступ к объекту, не изменяя при этом поведение клиента.

Суть паттерна

- Заместитель — это структурный паттерн проектирования, который позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу.

Применимость

- Виртуальный прокси
- Защищающий прокси
- Удаленный прокси
- Логирующий прокси
- ”Умная” ссылка

Виртуальный прокси

- Когда у вас есть тяжёлый объект, грузящий данные из файловой системы или базы данных.
- Вместо того, чтобы грузить данные сразу после старта программы, можно сэкономить ресурсы и создать объект тогда, когда он действительно понадобится.

Защищающий прокси

- Когда в программе есть разные типы пользователей, и вам хочется защищать объект от неавторизованного доступа. Например, если ваши объекты — это важная часть операционной системы, а пользователи — сторонние программы (хорошие или вредоносные).
- Прокси может проверять доступ при каждом вызове и передавать выполнение служебному объекту, если доступ разрешён.

Удаленный прокси

- Когда настоящий сервисный объект находится на удалённом сервере.
- В этом случае заместитель транслирует запросы клиента в вызовы по сети в протоколе, понятном удалённому сервису.

Логирующий прокси

- Когда требуется хранить историю обращений к сервисному объекту.
- Заместитель может сохранять историю обращения клиента к сервисному объекту.

”Умная” ссылка

- Когда нужно кешировать результаты запросов клиентов и управлять их жизненным циклом.
- Заместитель может подсчитывать количество ссылок на сервисный объект, которые были отданы клиенту и остаются активными. Когда все ссылки освобождаются, можно будет освободить и сам сервисный объект (например, закрыть подключение к базе данных).
- Кроме того, Заместитель может отслеживать, не менял ли клиент сервисный объект. Это позволит использовать объекты повторно и здóрово экономить ресурсы, особенно если речь идёт о больших прожорливых сервисах.

Шаги реализации

1. Определите интерфейс, который бы сделал заместитель и оригинальный объект взаимозаменяемыми.
2. Создайте класс заместителя. Он должен содержать ссылку на сервисный объект.
3. Реализуйте методы заместителя в зависимости от его предназначения.
4. Подумайте о введении фабрики, которая решала бы, какой из объектов создавать — заместитель или реальный сервисный объект.
5. Подумайте, не реализовать ли вам ленивую инициализацию сервисного объекта при первом обращении клиента к методам заместителя.

Приемущества

- Позволяет контролировать сервисный объект незаметно для клиента.
- Может работать, даже если сервисный объект ещё не создан.
- Может контролировать жизненный цикл служебного объекта.

Недостатки

- Усложняет код программы из-за введения дополнительных классов.
- Увеличивает время отклика от сервиса.

Благодарность

Спасибо за внимание!