Design Patterns

by Versus 2020

Single-serving visitor
Одноразовый посетитель

Поведенческий шаблон

Суть шаблона

Добавляет в программу новые операции, не изменяя классы объектов, над которыми эти операции могут выполняться при этом посетителю не нужно оставаться в памяти.

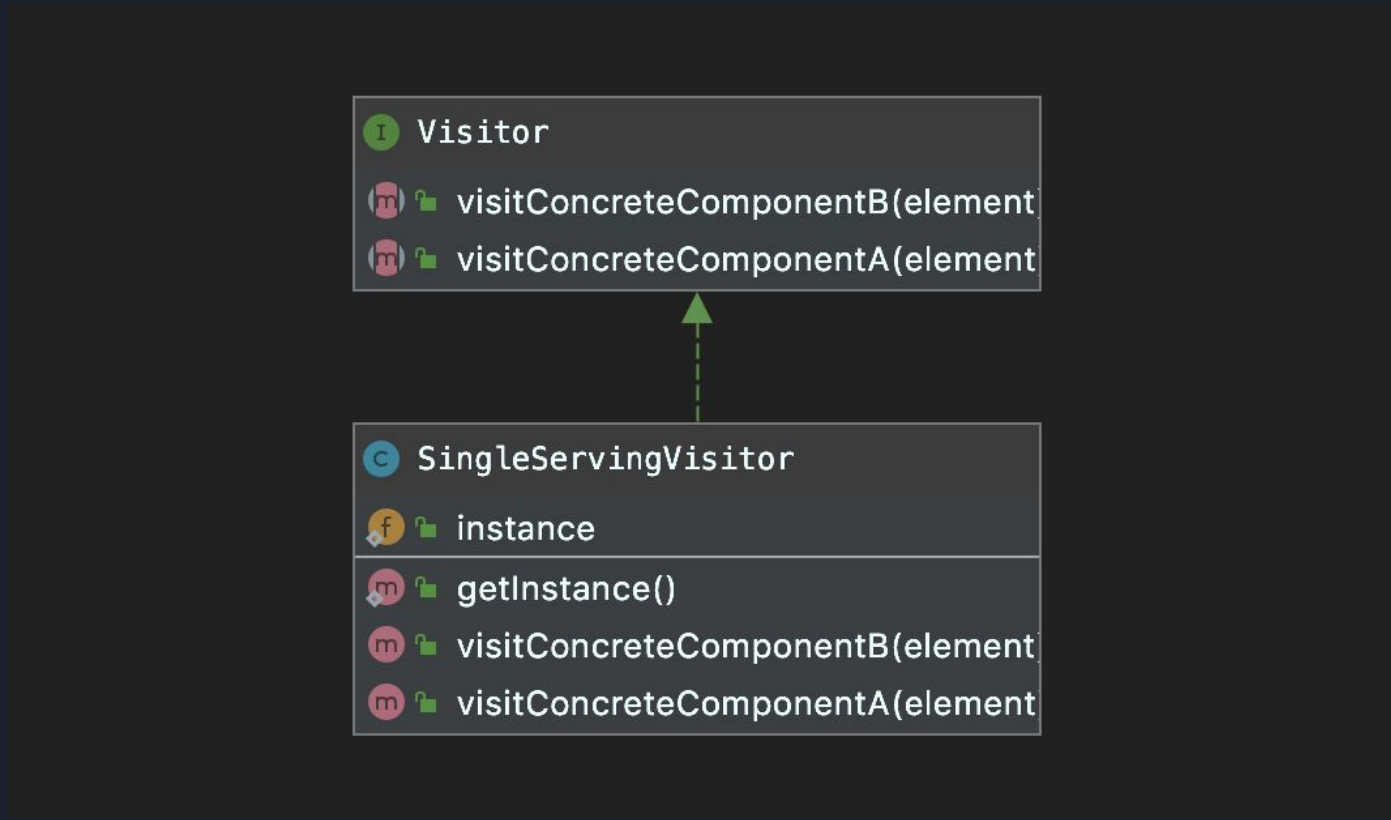
Проблема

Нельзя трогать и изменять класс нужного вам программного узла для расширения функционала и гарантировать, что новые объекты будут удалены из памяти, когда они станут бесполезными.

Решение

Разместить новое поведение в отдельном классе-одиночке.

Структура


Visitor
visitConcreteComponentB(element
visitConcreteComponentA(element
SingleServingVisitor
instance
getInstance()
visitConcreteComponentB(element
visitConcreteComponentA(element

Пример кода

Применимость

Нужно выполнить какую-то операцию над всеми элементами сложной структуры объектов.
Над объектами сложной структуры объектов надо выполнять некоторые не связанные между собой операции.
Новое поведение имеет смысл только для некоторых классов из существующей иерархии.

Шаги реализации

1. Создайте интерфейс посетителя и объявите в нём методы «посещения» для каждого класса элемента.
2. Опишите интерфейс элементов.
3. Реализуйте методы принятия во всех конкретных элементах.
4. Иерархия элементов должна знать только о базовом интерфейсе посетителей.
5. Для каждого нового поведения создайте конкретный класс посетителя.

Преимущества

Никаких "зомби" объектов.
Более простой интерфейс, чем посетитель.
Посетитель создается, используется и освобождается единственным вызовом статического метода apply_to.

Недостатки

Повторное выделение. При каждом вызове метода apply_to создается одноразовый посетитель, который отбрасывается, что отнимает много времени.
Напротив, синглтон выполняет только одно выделение.

Связь с другими шаблонами

Одиночка

Итог