Структурните шаблони Composite и Flyweight се прилагат към системи които имат много информационни обекти (data object). Шаблонът Composite има широко приложение и неговите съставни списъци (composite lists) могат да използват шаблона Flyweight. А шаблонът Flyweight споделя идентични обекти „зад кулисите“ с цел да се спести място. За имплементацията на тези шаблони се използват някои от известните технологии в C#, например : Generics типове, пропъртита, структури, индексатори, имплицитно типизиране (Implicit typing), инициализатори и анонимни типове (Anonymous types).

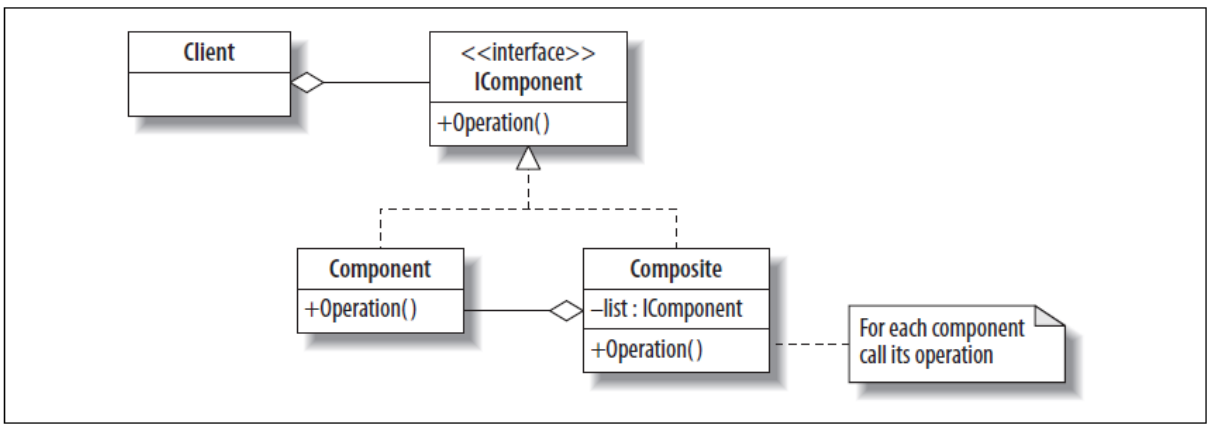
Шаблон Composite

Описание:

Шаблонът Composite подрежда структурните йерархии, така че единични компоненти и групи от компоненти да могат да се третират по еднакъв начин. Типични операции над компонентите са добавяне, премахване, извеждане, търсене и групиране. Могат да се посочат много примери за съвременни приложения, които са специализирани в групиране на данни. Например плейлистите на някои сайтове като iTunes, фотоалбуми към Flickr и много други. Елементите се поставят в голям списък, които се разделят по определен критерий. Например ако разгледаме един фотоалбум, може да изберем различен начин за преглеждане на снимките, които са добавени: в хронологичен ред , само по година, по име на събитие и т.н. По този начин една снимка може да се появи в няколко албума. Създаването на албум формира композитен (т.е. съставен) обект, като това не води до копиране на снимките. В този контекст същественото за шаблона Composite, е че операциите извършени за единични снимки или албуми от снимки трябва да имат едни и същи имена и ефекти, въпреки че реализациите са различни. Например потребителя трябва да може да покаже (изведе) снимка или албум (със снимки). По същия начин изтриването на една снимка да става по същия начин както и изтирването на цял албум.

Дизайн:

Шаблонът Composite е лесен за реализация. Той трябва да се справя с два типа: Components и Composites от тези компоненти. И двата типа ще са съгласувани с интерфейс за общите операции. Composite обектите са съставени от Components и в повечето случаи операциите над Composite са имплементирани чрез извикването на еквивалентните операции за своите Component обекти. Ето и UML диаграмата:



Основните елементи са:

IComponent – дефинира операциите за обектите в композицията и всяко поведение по подразбиране, което ще е приложимо за обектите от всеки тип

Operation – Извършва операция над обектите съгласувани с IComponent

Component – Имплементира операциите, приложими за единични обекти, които не могат да бъдат разделени занапред.

Composite – Имплементира операциите които са приложими към съставните обекти, използвайки (или предоставяйки) списък на компонентите съхранявани локално и най-вероятно ще ползва еквивалентни Component операции.

Client - работи с интерфейса IComponent, който опростява своите задачи.

Пример:

За споменатата фотогалерия от по-горе може да съпоставим следните обекти и действия:

IComponent - Един елемент в галерията

Component – Една единствена снимка

Composite – Един албум от снимки

Operation – Отваряне, разглеждане и т.н.

Реализация:

За реализацията може да се използват някои от познатите колекции, заедно с използването на generic типове. Тези типове (още шаблони) са разширение на системата от типове посредством структури, класове, интерфейси, делегати и методи, които могат да бъдат параметризирани с типове. За пример generic типа List<T> е генератор за конструкции като List<string> и List<Person>

==========

Като пример е реализирана структура, която може да има снимки или множество от снимки и албуми (от снимки). Входа се взима от файла Composite.dat , който съдържа команди за добавяне на снимки или албуми, изтриване и др. Ето описание на командите от файла:

AddSet – Добавя множество с дадено име и оставаме там (в тази „директория“).

AddPhoto – Добавя ново име за снимка посочено след командата и оставаме там.

Find – Намира съответния компонент (множество или снимка) или връща null ако не го намери.

Remove – Премахва компонент с дадено име (множество или снимка) и оставаме там където е премахнат.

Display – Извежда цялата структура

Quit – Изход от програмата

* едно множество (за дадената директория) може да има снимки и албуми.