Assembler

F. Balaguer - G. Dombiak
LIFIA-Departamento de Informática, Facultad de Ciencias Exactas, UNLP
CC 11 (1900) La Plata, Buenos Aires, Argentina
[gaston,fede]@sol.info.unlp.edu.ar
Tel/Fax : (54) (21) 228252

1.-Pattern Name
Assembler

2.-Also Known As
Grouper / Packager

3.-Intent
Provide an *object artifact* responsible to group a collection of objects, using a domain-based criterion. The resulting object or collection of objects may be part of a new collection to assemble again.

4.-Motivation
It is common in financial applications to find the requirement to group Transactions, TransactionFundSplits (TransactionFundSplits represents the split payment to different counterpart accounts) or Instruments in other objects.
Sometimes the user of a trade support application needs to group Transactions with the same Counterpart and maturityDate in order to make the settlement. The result is a new financial object that encapsulates the bulk set of elements. Since these new financial objects are polymorphic with the initial ones, the settlement operation makes no difference between them. There exists a well-defined selection criterion in order to group these objects; in the example above the criteria was by **counterpart + maturityDate**.
Usually the user needs to be able to specify the grouping criteria. At other times, this criteria is hard coded into some method of a domain object.
We propose to model the selection criteria as a component that will be used by an *assembler* object in order to group the collection. This component can be replaced by other selection criteria without affecting the other objects. In our financial model[1] (see Fig 1.) instances of class Netter acts like an *assembler*, while instances of NetTransactionFundSplit play the *assembled object* role.

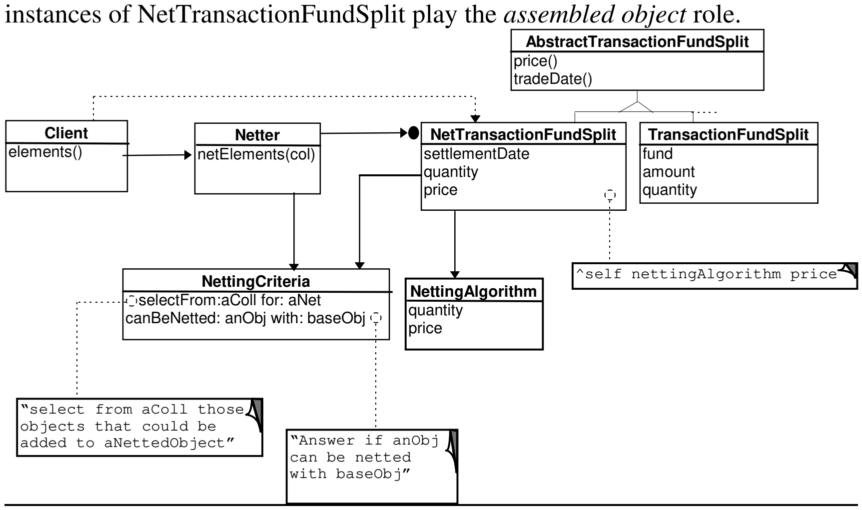

Figure 1. The structure of the financial implementation of the Assembler pattern

[1]The model is the result of three years of development at J. P. Morgan doing financial applications (trade, confirmation & settlement of transactions) implemented in VisualWorks with ENVY/Developer and Sybase RDBMS

Part of the knowledge of the NetTransactionFundSplit is based on the computing of a given aspect of the grouped elements, such as: price, quantity, etc. Thus, it is possible to define operations to compute these transformations.
We usually find that this knowledge is placed in methods of the *assembled object*. The problem with this approach is that it is difficult to reuse the a*ssembled object* since it is tightly coupled with the context where it was defined. We propose to split this knowledge in a new object called CalculationAlgorithm which is a private collaborator of the *assembled object* (in our model TransactionFundSplit). The *assembled object* delegates to the CalculationAlgorithm the responsibility to respond to some messages.

5.-Applicability
Use the Assembler pattern when:

- There are lots of objects that have to be grouped based on some recurrent aspect. The resulting set is a new abstraction, That is based on its components.
- The selection criteria's algorithm should be independent and may be replaced by another one.
- The AssembledObject wants to be used in many contexts without having to modify it.

6.-Structure

Figure 2. Shows the general structure of the Assembler pattern.

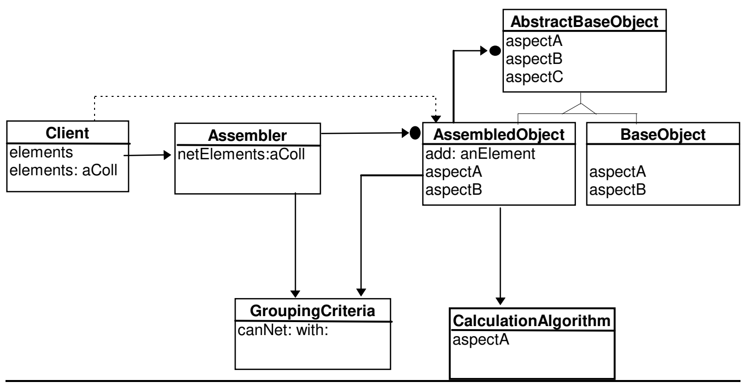

Figure 2. The general structure of the Assembler pattern

7.-Participants

- Assembler (Netter)
  - Defines an interface for receiving the objects to group.
  - Defines an interface for receiving the GroupingCriteria.
  - Constructs and assembles AssembledObjects from an initial collection of objects according to the GroupingCriteria.

- GroupingCriteria (NettingCriteria)
  - Specifies which objects could be grouped according to the criteria.

- AssembledObject (NettedObject)
  - Implements the default behavior for its interface according to the hierarchy where it belongs.
  - Implements the Collection interface (#add:, #addAll:, #includes:, etc.).
  - Holds a reference to the GroupingCriteria used to build itself.
  - Holds a reference to the CalculationAlgorithm to use in order to respond to some messages.

- CalculationAlgorithm (NettingAlgorithm)
  - Implements the behavior necessary to calculate some aspecs of the AssembledObject

8.-Collaborations

- The Client creates the Assembler object and configures it with the GroupingCriteria to use. This may be accomplished with a Builder (refer to the Builder pattern for more information [Gamma95]). The Client must also provide the collection of objects to group.
- The Assembler asks the GroupCriteria to select which objects of the collection could be grouped in an AssembledObject.
- The Assembler requests the AssembledObject class to create a new instance which will contain a given collection of objects according to a GroupingCriteria.
- The AssembledObject collaborates with the CalculationAlgorithm in order to respond to some messages.

9.-Known Uses

Building a geographical application that shows different temperature and pressure measures, we had to represent isopleths based on these measures. An isopleth is a curve which joint points with equal characteristics. We used a solution based on the Assembler pattern in order to model an Isopleth object. Fig 3 shows the structure of the geographical implementation. In this case the Isopleth hierarchy has a Composite [Gamma95]. A SingleIsopleth is instantiated when the first two points are found; afterwards only ComposedIsopleth instances are used to generate the temperature or pressure isopleth.

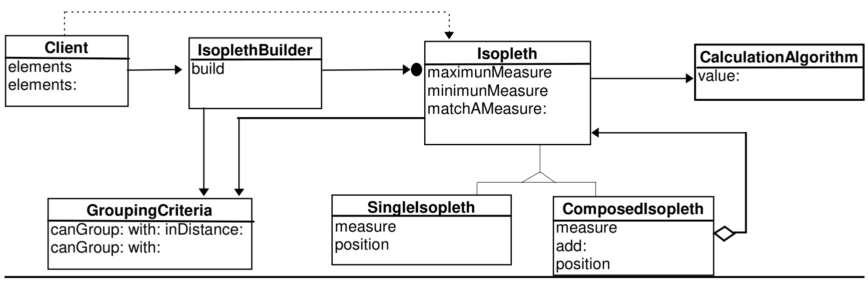

Figure 3. The geographical implementation of the Assembler pattern

10.-Implementations

We have seen two different implementations of this pattern, the first one in the financial domain was presented in section 4 “motivation”, the second one was the geographical example that was presented in the known uses (section 9). The main difference between both cases is the specific implementation of AssembledObject. This class has to be implemented following the rules given by the domain.
In the financial case the result of the whole netting operation is a collection of NetTransactionFundSplit, while the source is a collection of TransactionFundSplit and NetTransactionFundSplit.
In the geographical case, the IsoplethBuilder produce one Isopleth as result, while the source is a collection of measure and Isopleth (Single or Composed).
The GrouppingCriteria and the CalculationAlgorithm may be implemented as strategies since any of them may vary independently from the clients that use them. Refer to the Strategy pattern for more information [Gamma95]

11.-Consequences

- The creation of AssembledObject is based on a GroupingCriteria (a logic based on the domain), which could change, or applied again.
- The Assembler object can be easily reused in many different contexts just by replacing the GroupingCriteria.
- For small collections of objects where the GroupingCriteria is too silly, it may be wiser to extend Collection in order to respond to #groupBy: instead of implementing the Assembler pattern.

12.- Related Patterns

Actually, Assembler is a pattern that is composed of smaller patterns. It could be structurally described in terms of a Composite which interacts with two Strategies; one for building and the other for calculating. It could be part of a second order pattern classification.

- The AssembledObject is similar to Composite, because it groups objects with a defined structure.
- The GroupingCriteria provides the domain based logic to select objects that will be grouped, it could be implemented as a Strategy.
- The CalculationAlgorithm implements operations to compute or represent the AssembledObject's internal state. It could be implemented as a Strategy.
- Assembler, like a Builder, implements the construction of the AssembledObject. It uses the GroupingCriteria in order to accomplish this task.

13.- References

[Gamma95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *"Design Patterns. Elements of reusable Object-Oriented Software"*. Addison Wesley, 1995.