1.1 Partial Population Pattern

1.1.1 Name and Source

Partial Population Pattern
Page 297-310 in the book "Web Service Patterns: Java Edition" [WSP 03]

1.1.2 Also Known As

1.1.3 Type

1.1.4 Intent

To provide a generic and reusable “Data Transfer Object” that lets the client choose which data to be returned, while avoiding to clutter the web service interface with a lot of methods that return different data transfer object classes.

1.1.5 Problem

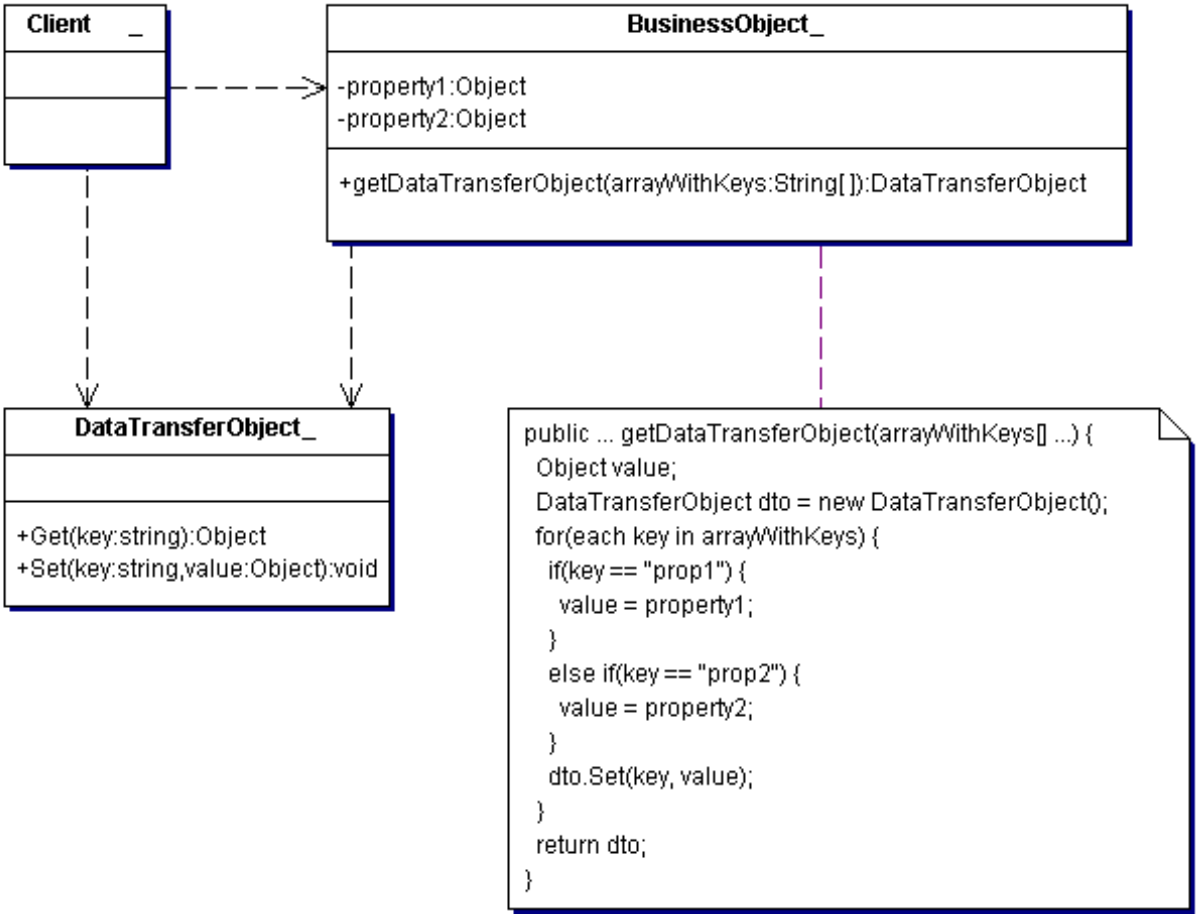
It is difficult to predict what groups of related data the clients will want to receive as data transfer objects, and there is a risk that there will be a lot of different data transfer object classes and a web service with a lot of methods that return instances of these classes.

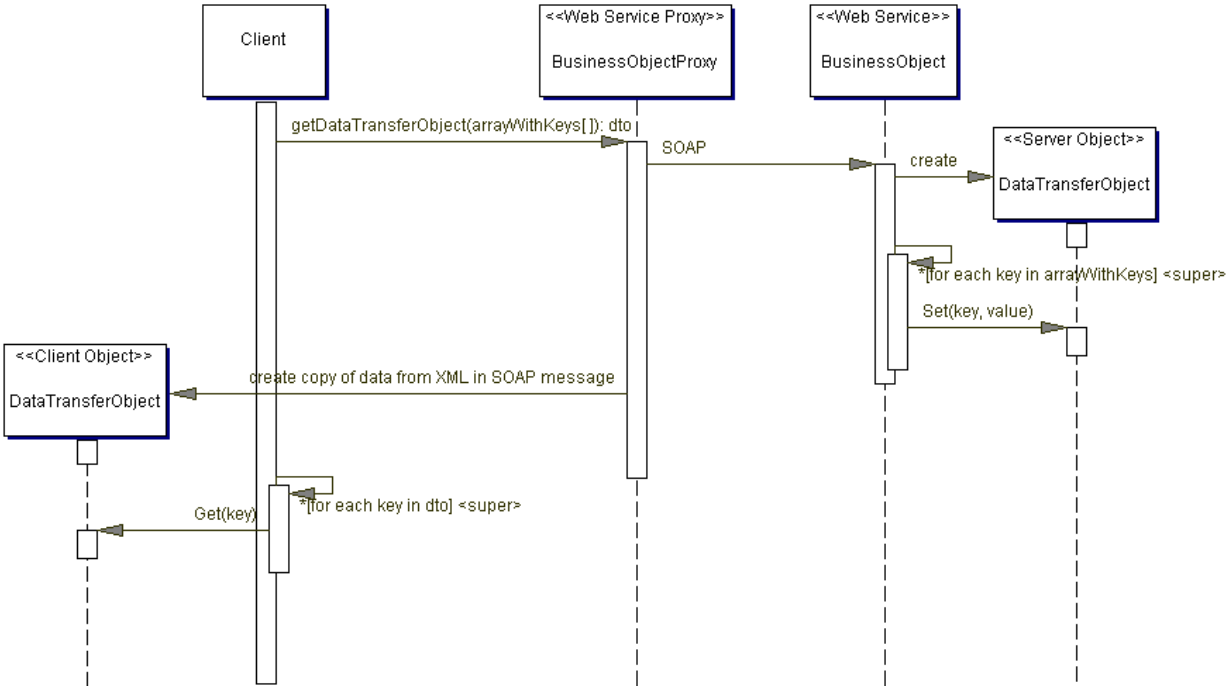
How can a single, generic and reusable data transfer object be constructed while also letting the client choose the data to be included in the object ?

1.1.6 Forces

1.1.7 Solution

One part of the solution is to create a data transfer object that essentially is a hash table with get and set methods. These two methods include key parameters with names that represent the data attributes of interest. The other part of the solution is to provide a web service method that receives an array of strings that contains the names of the keys corresponding to the data attributes of interest.





[the occurrence of "<super>" in the diagram is not intentional and will be removed in the final version]

DataTransferObject <<Client>> – A client sided copy of the generic server sided data transfer object.

Client – A consumer of the web service.

BusinessObjectProxy – A proxy to the web service.

BusinessObject – The web service implementation. It might contain a lot of conditional statements, as illustrated in the UML note in the class diagram.

DataTransferObject<<Server>> – A data transfer object that will contain a subset of the data in the business object web service. What subset it will contain is flexibly chosen by the client in the parameter that is an array with key names.

1.1.8 Consequences

You lose semantic in the interface for this kind of generic, partially populated Data Transfer Object (DTO), compared with a regular DTO that have typed accessor methods or public attributes for each data value of interest. For example, the partial populated DTO will have this kind of code: *dto.get("FirstName")* instead of *dto.getFirstName()* as in the regular DTO.

The conditional statements in the business object may have a negative impact to the performance, and therefore it can be a good idea to try to predict the most commonly requested data values and implement these in the first conditional branches.

1.1.9 Related patterns

The "Partial Population pattern" (PP) is a kind of "Data Transfer Object pattern" (DTO). The structure of the classes as illustrated in the UML class diagram is the same, and the only differences between these patterns are the method signatures in the objects and also the fact that the PP will only have one generic DTO class, while you may use many different kind of data transfer objects if the regular DTO pattern is used instead.

One part of the solution to this pattern is to provide an object with generic get and set methods, and that concept can also be found in the "Dynamic Attributes pattern" described in the book "Corba Design Patterns" [CORBA 97].