More Complex Mappings

Example File (Builder projectAdvanced Modeling/Mapping):

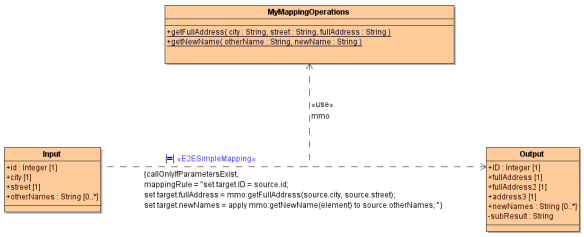
<your example path>\Advanced Modeling\Mapping\uml\mappingContext.xml
<your example path>\Advanced Modeling\Mapping\uml\mappingScriptUsageDependencies.xml

On this Page:

Related Pages:

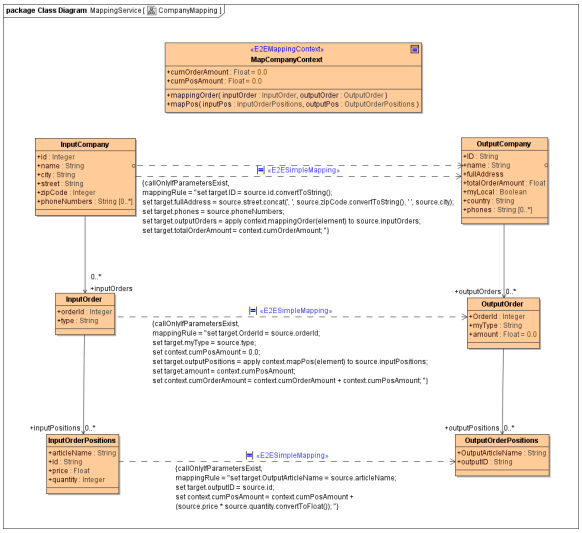
Using Static Operations with Simple Mappings

Sometimes you may not only want to map simple attributes using built-in operations, but it may be necessary to execute self-defined operations, for example because of more complex mapping rules or to do a look-up for a code-value translation. You can do this by defining a class that contains the needed operations and draw a <<use>> dependency between this class and the <<E2ESimpleMapping>> dependency. Note that all operations of this class, that you want to access via a <<use>> dependency, must be **static**.



Mapping with a Mapping Context

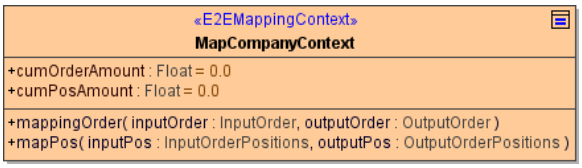
Have a look at the class diagram below showing a mapping between the order structure of two different companies.



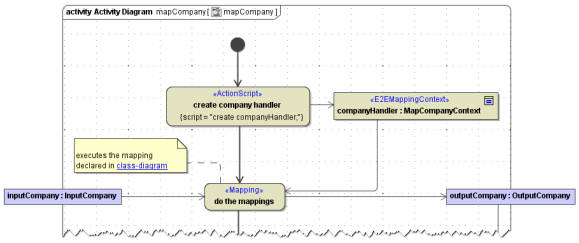
The data structures of the output company contain some fields with calculated amounts, e.g. **totalOrderAmount** and **amount**. This means, that during the mapping process you will have to calculate the positions and order values and accumulate them. Therefore, you will need to persist the interim results of your calculations between the mappings.
This can be done by the use of a mapping context.

Defining and Providing the Mapping Context

Create a class having the stereotype <<E2EMappingContext>> which contains all needed attributes to store the interim data, e.g. **cumOrderAmount** and **cumPosAmount**.



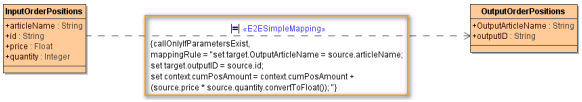
Before calling the mapping adapter, create the mapping context and provide it (see activity diagram below).



Using the Mapping Context in Mapping Scripts

In the mapping script, you can access the attributes of the mapping context class via the prefix **context**, e.g. **context.cumPosAmount**.

Figure: Usage of the Mapping Context a Mapping Script

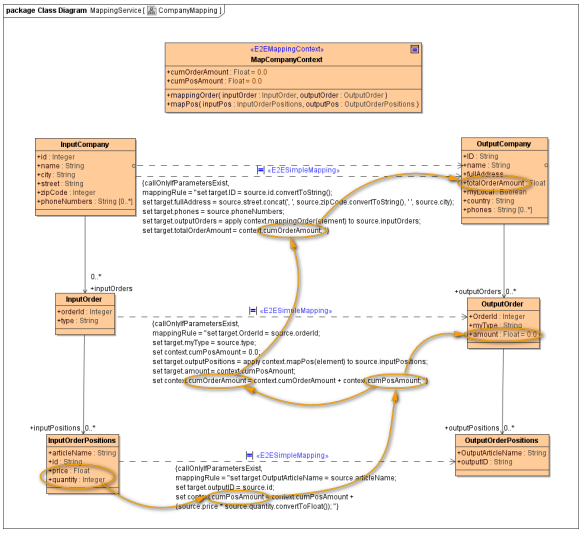


In the picture above, the value of the order position is calculated and stored in the mapping context to calculate the total order value.

```
set context.cumPosAmount = context.cumPosAmount + (source.price * source.
quantity.convertToFloat());
```

Using Mapping Context Operations in Mapping Scripts

The mapping context class may not only contain attributes to persist data, but also operations. These operations can be used to easily resolve hierarchical structures as depicted in the mapping context example.

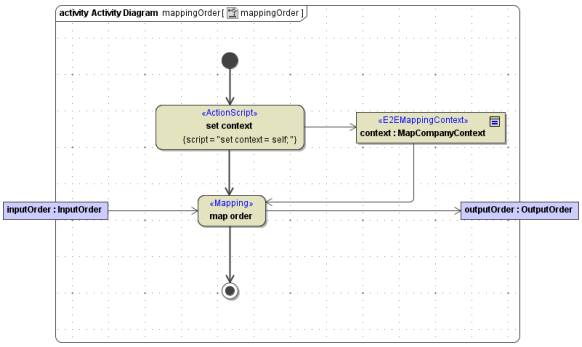


The position value (quantity * price) is accumulated to the order value and all order values are accumulated to a company order value.

To do this, the mappings have to be called in the correct context and order. That means that e.g. the company order value only can be mapped, when all orders are mapped.
So before mapping the **totalOrderAmount** of the company, all orders have to be processed.

```
set target.outputOrders = apply context.mappingOrder(element) to source.
inputOrders;
set target.totalOrderAmount = context.cumOrderAmount;
```

In the mapping script above, the operation **mappingOrder()** of the mapping context class is performed on each order (**element** of the apply operation) of the company. All this operation does is calling the mapping adapter for an input company order and providing the mapping context.



The same applies to the order level where first the position mapping is called before calculating the order value:

```
set context.cumPosAmount = 0.0;
set target.outputPositions = apply context.mapPos(element) to source.
inputPositions;
set target.amount = context.cumPosAmount;
set context.cumOrderAmount = context.cumOrderAmount + context.cumPosAmount;
```