Question
Tom Philippi · Aug 22, 2017

How to include process context in data transformations

I have been building Business Processes in Ensemble for a few months and in a process i'd use a combination of data from the request that started the process and context variables that where filled by earlier calls in the process as input for a data transformation. I do not know how to achieve that.

What I do currently is that in my process I first call a data transformation and then make a call to an operation. For the data transformation I use as input the request that started the business process and as output the request that I will use to initiate the call to the operation. The latter I define as a context-object in my process. Thus I end up with something like this:

Process-parameters:

- Request: MyProcess.StartProcessRequest
- Response: Ens.Response
- Context-fields
    - Context.CallOperationRequest = MyOperation.CallOperationRequest

Process-flow:

Step 1: Data transformation

- Input: MyProcess.StartProcessRequest
- Output: Context.CallOperationRequest
- Annotation: This step prepares the request for step 2.

Step 2: Call MyOperation

- Request of type: MyOperation.CallOperationRequest
- Mapping:
    - Context.CallOperationRequest -> target.CallRequest
- Annotation 2: This step calls MyOperation using the request prepared in step 1.

Insofar this works fine. But with more complex process I want to prepare operation-callrequests which use data both from MyProcess.StartProcessRequest and output from previous operations which are stored in context-variables. However I do not know how I setup properly a single data transformation that transforms data from both the request and the context to my operation-callrequest. According to the comment posted below. Something like this should be possible. Can someone explain. I.e. I am looking for something along the following lines:

Process-parameters:

- Request: MyProcess.StartProcessRequest
- Response: Ens.Response
- Context-fields
    - Context.CallOperationRequest = MyOperation.CallOperationRequest
    - Context.StringOutputOperationX = %String

Process-flow:

Step X: Call operationX

- Response mapping:
  - -> Fills Context.StringOutputOperationX

Step X+1: Data transformation

- Input: Somehow MyProcess.StartProcessRequest + Context.StringOutputOperationX
- Output: Context.CallOperationRequest
- Annotation: This step prepares the request for step X+2.

Step X+2: Call Data transformation

- Request of type: MyOperation.CallOperationRequest
- Mapping:
  - Context.CallOperationRequest -> target.CallRequest
- Annotation: This step calls MyOperation using the request prepared in step X+1.

~~I have been building Business Processes in Ensemble for a few months and I was wondering what the best design pattern is for complex multistep process. For instance, imagine you start a process with request A and in that process you call operations B, C, and D. Now operation B returns variable X which is used later as input for operations C and D. Now, the main challenge I have is that intuituively you would define a context in the process to put variable X in. However, when mappings for callrequests to operations C and D grow complex you'd like to prepare them with a data transformation (as they somehow have more elaborate functionality than the callrequest mappings), and ideally you like to put the complete mapping in a single datatransformation. However, I think that in Ensemble this is not possible if you have data from request A and from the process context as input for callrequests of subsequent operations (e.g. C and D). Thus, preparing such a call request quickly results in a multi-step process where you first run a data transformation on the initial process request A and then add in all context variables. To me this feels unnecessary cumbersome, so I have thinking about patterns that could make this a bit more clean. However, none of the patterns I have thought up are entirely satisfactory so I was really wondering what the approach was of the more experienced Ensemble process modellers.~~

~~These are the patterns I have come up with:~~

~~Multi-process pattern:~~

~~Here I design a twostep process. The first process starts with request A and calls operation B and then calls a second process which is responsible for calling operations C and D. The advantage here is that you can initiate the second process with a request A+ which contains the variable X from operation B. Consequently, you can do a single data transform from request A+ to C and a single data transform from A+ to B. The disadvantage is that this works nicely for this case but if you have a case where operation B gives input for operation C which in turn gives input for operation D, etc. you end up with a number or processes equal to the number of operations. In my opinion this is far from clean.~~

~~Super request pattern:~~

~~The idea here is that instead of returning inputs from operations to a context, you return them to the request that started the process. Of course, the request that started the process should be expanded to hold these variables. I hoped this should allow you to use a single data transformation to initiate each subsequent call, however, I just found out that this solution doesn't work at all because you can update the initial request with the output from an operation in that process and pass it immediately to the next step, the updated request message isn't stored in the database so when the process goes to sleep again the request "reverts back" to the original request that started the process and losses the output from operation B.~~

~~Two-step data transformation~~

~~Prior to calling an operation (e.g. C, or D) call a data transformation from request A to the request for the operation. Then use a code block to add in all context variables from previous operations in one step. This should keep preperation of all callrequests down to two steps.~~

~~Other options~~

~~As far as I know it is not possible to use the context of a process as input for a data transformation. If I am wrong i'd like to know.~~

#Ensemble #Business Process (BPL) #Mapping