WAY2LEARN

LAB- Dataweave basics

In this lab you will understand

a) how to write data weave expressions

b) How to use transform message component

STEP 1

Create a new project with name 01-dw-basics-start.

Copy the product.json given to you into src/main/resources.

We want to make a Http POST request with json content. We want to transform it to XML and Java

1)create a mule configuration file with name "transformerdemo"

2)Drag a Http Endpoint and configure it to listen for requests at URL http://localhost:8081/transform

3) Drag "transform message" component and a logger after it. Configure logger to log the payload.

Double click on transform message. In the Transform Message Properties view, look at the Input, Transform, and Output sections.

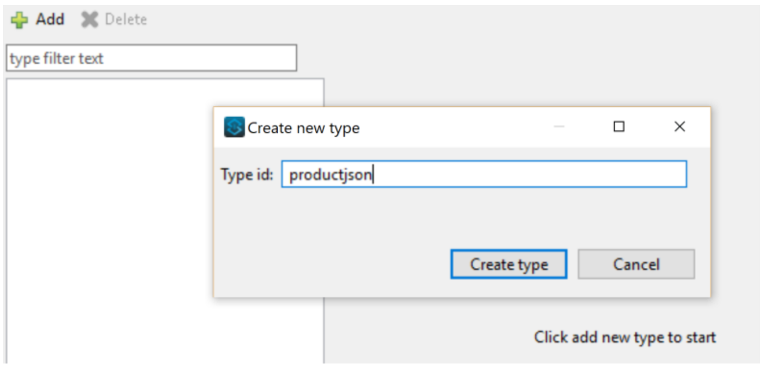
In the Transform section(the place where we write dw expressions), locate the drop-down menu at the top of the view that sets the output type; it should be set to Payload.

Below it, locate the directive that sets the output to application/java.

Delete the existing DataWeave expression (the curly braces) and set it to payload.

In the Input section of the Transform Message Properties view, click "Payload: Any Define metadata"

In the pop up window, click on “+Add” button . Give the type id as “productjson”. Select Type as json

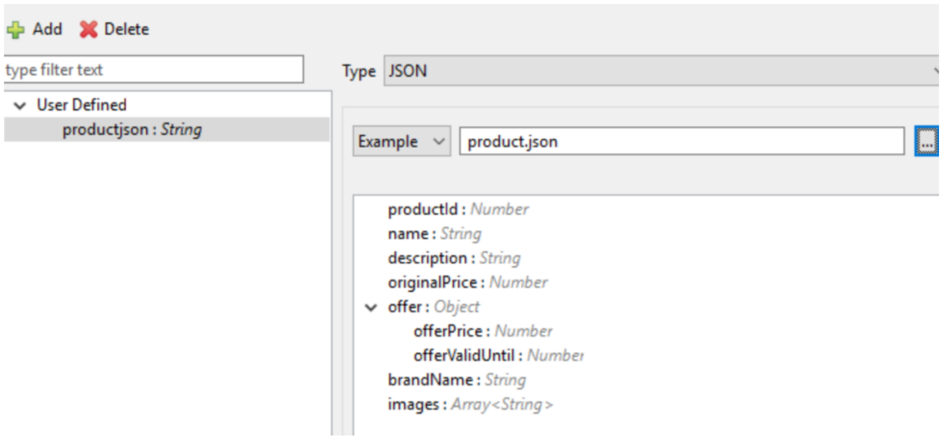


Type id: productjson

Select type as json

Now click on schema dropdown and select “example”

Select src/main/resources/product.json.



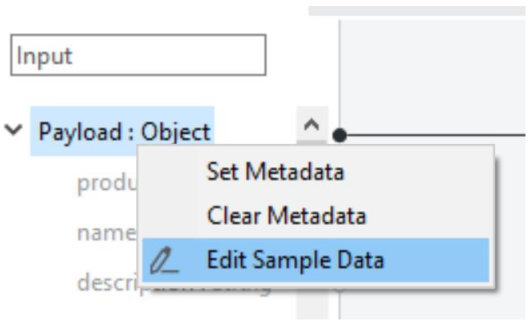
Type: JSON

Example: product.json

```
productId : Number
name : String
description : String
originalPrice : Number
offer : Object
    offerPrice : Number
    offerValidUntil : Number
brandName : String
images : Array<String>
```

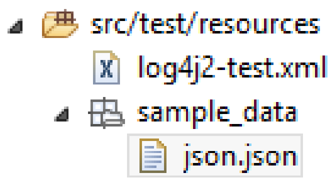
Click on Select. To close the window.

Now right click on payload in the input section and select “Edit Sample data” as shown below:



You should able to see the sample data in the input section.

Observe that there is a file with name json.json created under src/test/resources/sample_data folder

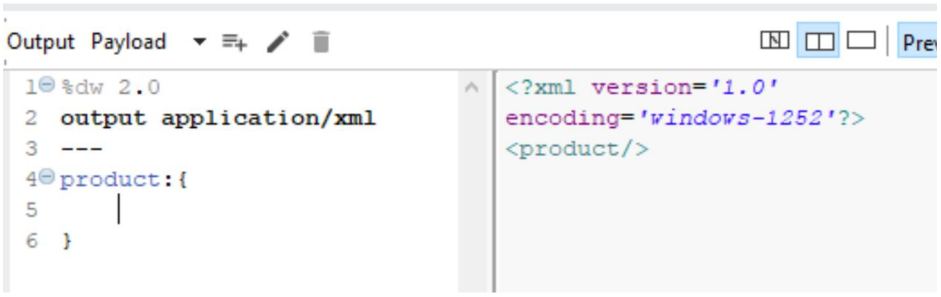


Now click on the preview button in the output section. you should see the output preview

4) In the Transform section, change the output type from application/java to application/json. You should observe the same json in preview

5) Now change the output type to xml. It shows error as root tag is required for xml.

Now change the expression as product :{} . it looks like below:



```
%dw 2.0
output application/xml
---
product:{

}
```

```
<?xml version='1.0'
encoding='windows-1252'?>
<product/>
```

Change the expression as below   and observe the preview

```
product:{
	pid:payload.productId,
	productName:payload.name,
	discountPercentage: payload.offer.discountPercentage,
	image1 :payload.images[0]
}
```

Change the expression as below   and observe the preview

```
product:{
	pid:payload.productId,
	productName:payload.name,
	offer:{
		discountPercentage: payload.offer.discountPercentage,
		offerValidUntil:payload.offer.offerValidUntil
	},
	image1 :payload.images[0]
}
```

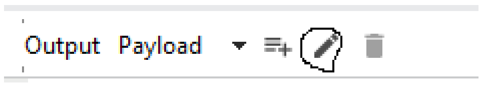
we want "pid" to be attribute of product tag. So, change the expression as shown below :

```
product @(pid:payload.productId):{

	productName:payload.name,
	offer:{
		discountPercentage: payload.offer.discountPercentage,
		offerValidUntil:payload.offer.offerValidUntil
	},
	image1 :payload.images[0]
}
```

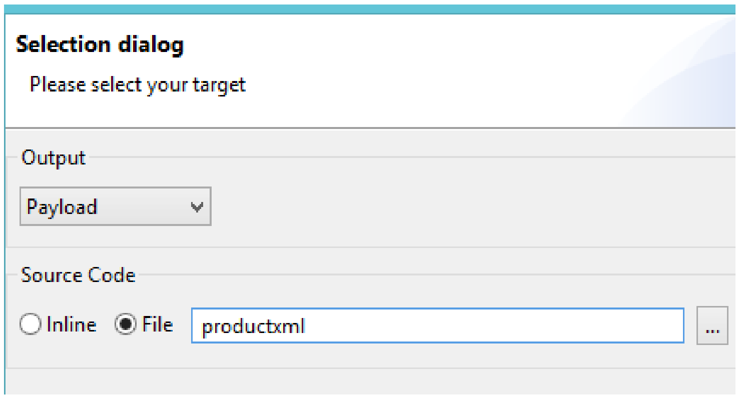
Now switch to xml view and observe that all dwl expressions are inline in xml which makes xml difficult to read.

So, we want to externalize the dwl into separate files.

Now click on edit button highlighted below to edit the current target for dwl .



Select target as a file with name productxml as shown below and press ok :


Selection dialog

Please select your target

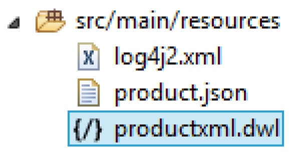
Output

Payload

Source Code

( ) Inline (●) File productxml ...

You should observe that a file with name productxml.dwl is created as shown below

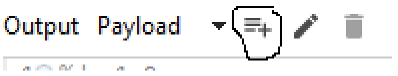


Now switch to xml view and observe that the transform-message configuration looks like below :

```
        <ee:transform doc:name="Transform Message"
doc:id="daf964b2-30e3-420b-bac5-03122c3f7952" >
                        <ee:message >
                                <ee:set-payload resource="productxml.dwl" />
                        </ee:message>
</ee:transform>
```

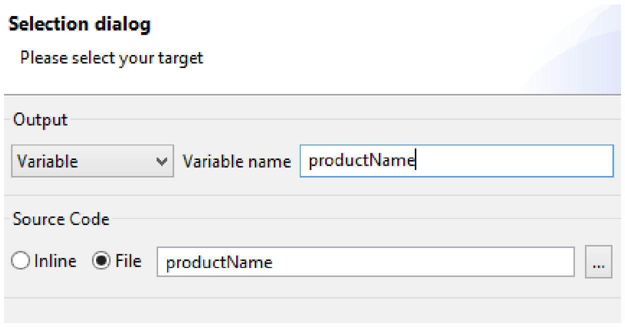
We want to write an expression which will evaluate to productName and store it in a variable with name productName.

So, Create a new target by clicking on "+" highlighted below :



Then Select Variable from Dropdown and give the variable name as productName.

Select the radio button "File" and give the filename as productname as shown below and press OK:



Selection dialog

Please select your target

Output

Variable | Variable name productName

Source Code

Inline (● File) productName ...

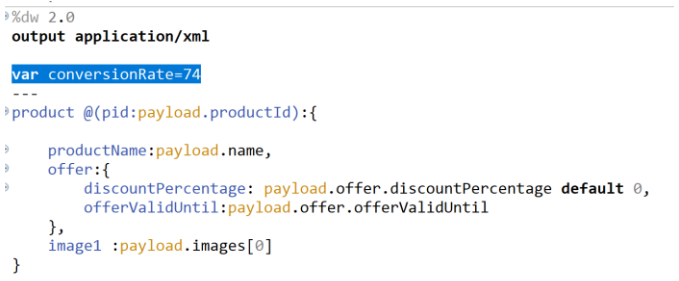
Write the expression as payload.name

Similarly create another variable with name brandName and write the corresponding expression. Make sure that you store that expression in a new file "brand.dwl"

The variables which we created are variables which are part of Mule Event

We can create global variables which will be available only in the current dataweave file

Create a variable in the headers part as shown below:

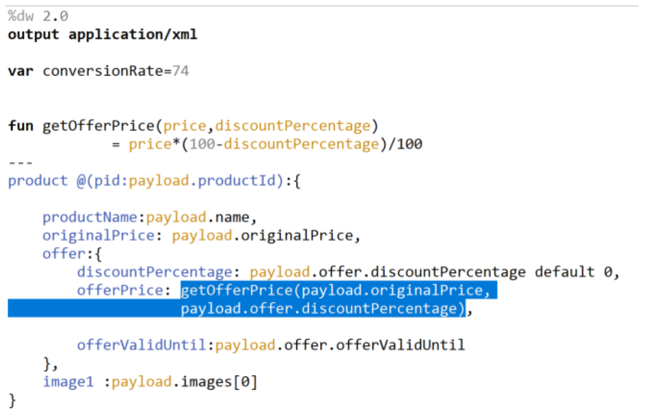


```
%dw 2.0
output application/xml

var conversionRate=74
---
product @(pid:payload.productId):{

    productName:payload.name,
    offer:{
        discountPercentage: payload.offer.discountPercentage default 0,
        offerValidUntil:payload.offer.offerValidUntil
    },
    image1 :payload.images[0]
}
```

In the header section, Create a function to return offerPrice based on currentPrice and discountPercentage as shown below:

```
fun getOfferPrice(price,discountPercentage)
                    = price*(100-discountPercentage)/100
```

Call the function as shown below:



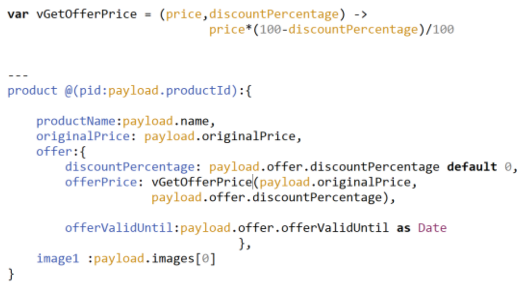
```
%dw 2.0
output application/xml

var conversionRate=74


fun getOfferPrice(price,discountPercentage)
            = price*(100-discountPercentage)/100
---
product @(pid:payload.productId):{

    productName:payload.name,
    originalPrice: payload.originalPrice,
    offer:{
        discountPercentage: payload.offer.discountPercentage default 0,
        offerPrice: getOfferPrice(payload.originalPrice,
                    payload.offer.discountPercentage),

        offerValidUntil:payload.offer.offerValidUntil
    },
    image1 :payload.images[0]
}
```

Observe the preview and make sure that it is as expected.

We want to create a variable and initialize it to a function definition.So, create a variable as shown below:

```
var vGetOfferPrice = (price,discountPercentage) ->
                                   price*(100-discountPercentage)/100
```

Use this variable just like a function as shown below:



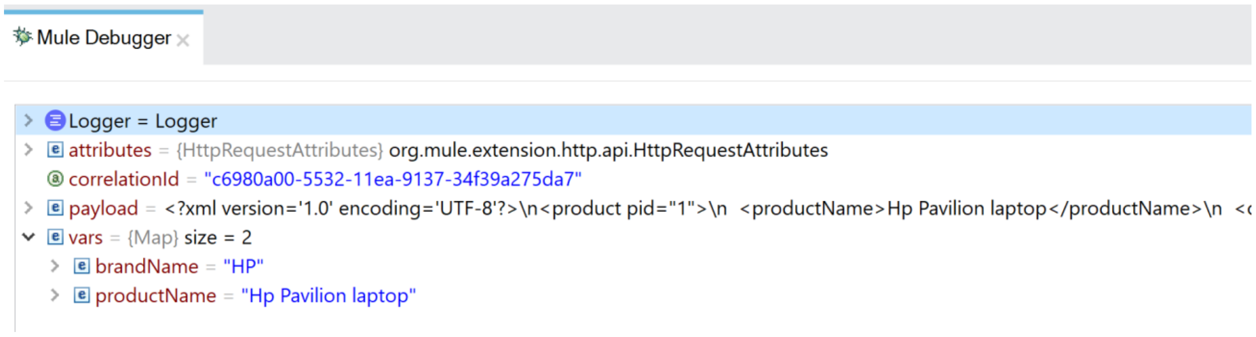
```
var vGetOfferPrice = (price,discountPercentage) ->
                            price*(100-discountPercentage)/100


---
product @(pid:payload.productId):{

    productName:payload.name,
    originalPrice: payload.originalPrice,
    offer:{
        discountPercentage: payload.offer.discountPercentage default 0,
        offerPrice: vGetOfferPrice(payload.originalPrice,
                    payload.offer.discountPercentage),

        offerValidUntil:payload.offer.offerValidUntil as Date
                                },
    image1 :payload.images[0]
}
```

Keep a break point on Logger run the application in debug mode.

Open POSTMAN and give Http POST request to http://localhost:8081/transform and send the content of src/main/resources/product.json in body.

Dont forget to add a request header "Content-Type" with value "application/json".

Once you switch to debug perspective after making request, observe that variables productname and brandname will be created as shown below:



```
> Logger = Logger
> attributes = {HttpRequestAttributes} org.mule.extension.http.api.HttpRequestAttributes
  correlationId = "c6980a00-5532-11ea-9137-34f39a275da7"
> payload = <?xml version='1.0' encoding='UTF-8'?>\n<product pid="1">\n  <productName>Hp Pavilion laptop</productName>\n  <
v vars = {Map} size = 2
  > brandName = "HP"
  > productName = "Hp Pavilion laptop"
```

This is the end of the Exercise