LAB- Transformers- complex DWL

In this lab, you will be working on **01-transformers-basics-start** project under **05-transformers** section

In this lab you will understand

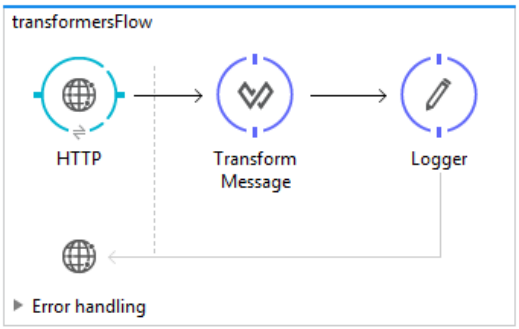
a) how to write iterate over an array or list using map operator and generate xml and java list

STEP 1

In this step, you will understand how to do transformations on json data.

Open transformers.xml given to you in src/main/app. Observe the code in it.

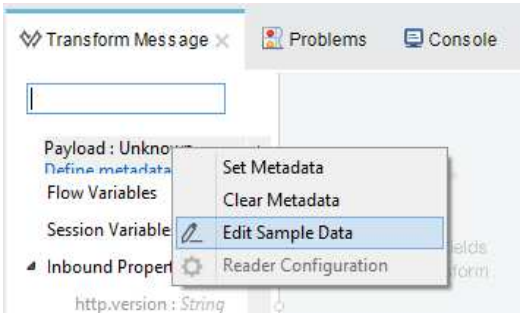
It is already conigured as shown below :



Now Open src/main/resources/products.json and observe that it contains list of products in json format.

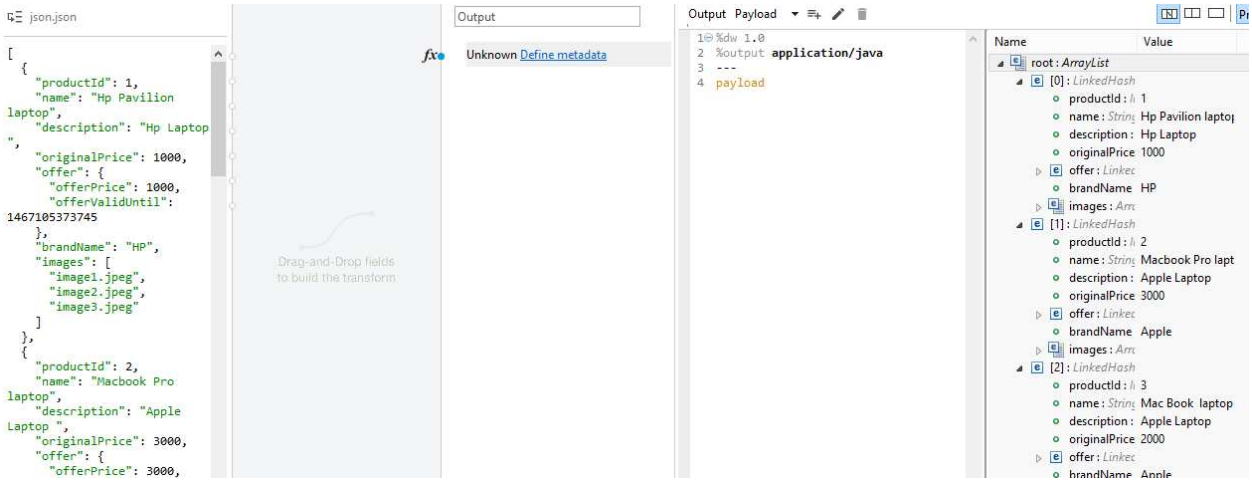
We want to convert this json to xml using dataweave.

Double click on Transform message. Right click on payload on left side as shown below and select Edit Sample data

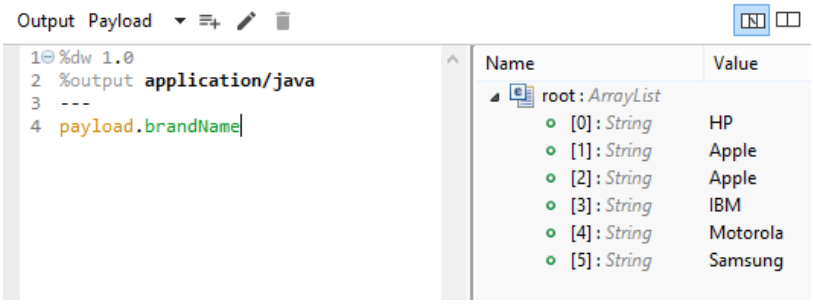


Select Json and copy the content of src/main/resources/products.json .

Now modify the dataweave expression as "payload" and observe the preview. It should look like below :



Change the expression to payload.brandName. You should observe the the list of All brandNames as shown below :



Now change the expression as `[payload.brandName,payload.name]`

Observe the preview.

We want to iterate over the list and prepare a list of brandName and name. So, modify the expression as shown below :

```
payload map {

}
```

Observe the preview. You should observe list of empty maps

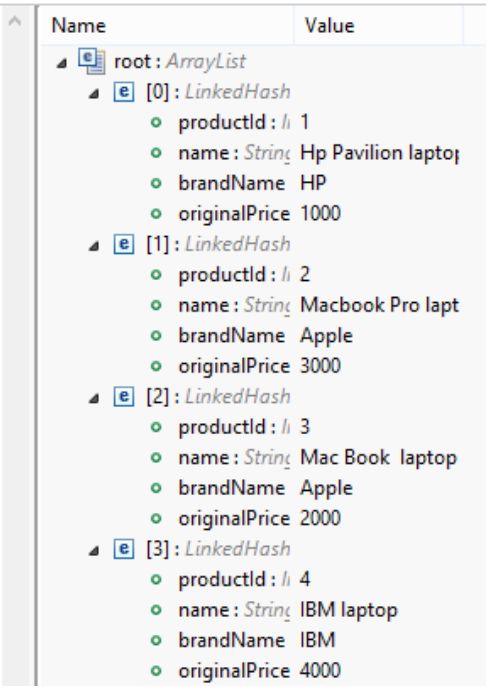
Now change the expression as shown below:

```
payload map {
	productId:$.productId,
	name:$.name,
	brandName:$.brandName,
	originalPrice:$.originalPrice
}
```

Preview should look like below

```
%dw 1.0
%output application/java
---
payload map {
    productId:$.productId,
    name:$.name,
    brandName:$.brandName,
    originalPrice:$.originalPrice
}

```



Now we want to convert it to xml

So, change the mime to application/xml. You should observe error as there is no root tag.

Now change the expression as show below:

```
products: {
       payload map {
      productId:$.productId,
      name:$.name,
      brandName:$.brandName,
      originalPrice:$.originalPrice
}

}
```

You should still observe the error. Can you tell why? . If not see my video again.

Now modify the expression as below by wrapping the expression inside paranthesis :

```
products: {
	(
		payload map {
			productId:$.productId,
			name:$.name,
			brandName:$.brandName,
			originalPrice:$.originalPrice
		}

	)
}
```

You should observe the preview as shown below :

```
<products>
  <productId>1</productId>
  <name>Hp Pavilion laptop</name>
  <brandName>HP</brandName>
  <originalPrice>1000</originalPrice>
  <productId>2</productId>
  <name>Macbook Pro laptop</name>
  <brandName>Apple</brandName>
  <originalPrice>3000</originalPrice>
  <productId>3</productId>
  <name>Mac Book  laptop</name>
  <brandName>Apple</brandName>
  <originalPrice>2000</originalPrice>
  <productId>4</productId>
  <name>IBM laptop</name>
  <brandName>IBM</brandName>
  <originalPrice>4000</originalPrice>
  <productId>5</productId>
  <name>MotoX Mobile</name>
  <brandName>Motorola</brandName>
  <originalPrice>1000</originalPrice>
  <productId>6</productId>
  <name>Samsung Note 5</name>
  <brandName>Samsung</brandName>
  <originalPrice>5000</originalPrice>
</products>
```

Modify the expression as below :

```
products: {
	(
		payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice:$.originalPrice
			}
		}

	)
}
```

You should Observe the preview as shown below:

```
<products>
  <product>
    <productId>1</productId>
    <name>Hp Pavilion laptop</name>
    <brandName>HP</brandName>
    <originalPrice>1000</originalPrice>
  </product>
  <product>
    <productId>2</productId>
    <name>Macbook Pro laptop</name>
    <brandName>Apple</brandName>
    <originalPrice>3000</originalPrice>
  </product>
  <product>
    <productId>3</productId>
    <name>Mac Book  laptop</name>
    <brandName>Apple</brandName>
    <originalPrice>2000</originalPrice>
  </product>
  <product>
    <productId>4</productId>
    <name>IBM laptop</name>
    <brandName>IBM</brandName>
    <originalPrice>4000</originalPrice>
  </product>
</products>
```

We can use $$ as index.

Modify the expression as shown below and observe the changes in preview

```
products: {
	(
		payload map {
			'product$$' :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice:$.originalPrice
			}
		}

	)
}
```

We want all the products to be sorted by originalPrice.

Modify the expression as shown below :

```
products: {
	(
		payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice:$.originalPrice
			}
		} orderBy $.product.originalPrice

	)
}
```

See the preview. You should observe that all the products are sorted in ascending order.

Can you modify the expression so that we can see the products sorted by originalPrice in descending order?

[Hint : Use Range Selector [-1..0] . But make sure that you wrap the before expression using ().]

Solution :

```
products: {
	(
		(payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice:$.originalPrice
			}
		} orderBy $.product.originalPrice)[-1..0]

	)
}
```

WAY2LEARN

We want to apply a filter which selects only products whose brandName is same as query parameter "brandName"

```
products: {
	(
		( (payload map (product,indexOfProduct)->
			{
			product : {
				productId:product.productId,
				name:product.name,
				brandName:product.brandName,
				originalPrice: getConvertedOriginalPrice(product.originalPrice)


			}
		} orderBy $.product.originalPrice
		) filter $.product.brandName==inboundProperties."http.query.params".brandName)
	)
}
```

Remember that preview may not be shown when you write expression using inboundProperties.

If you want the preview to be shown, right click on http.query.params in the input section and write a dwl which evaluates to a map as shown below :

```
{
	brandName :'Apple'
}
```

(Please see my my video or ask me if you are confused.)

Now deploy the application and give a request to http://localhost:8081/transformjson?brandName=HP and observe the result

We want the original price to be converted using a conversionrate.

Decalre a variable called conversionRate in headers section and use it in expression as shown below :

```
%dw 1.0
%var conversionRate=65
%output application/xml
---
products: {
	(
		(payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice:$.originalPrice *conversionRate
			}
		} orderBy $.product.originalPrice)[-1..0]

	)
}
```

Decalre We want to define a function called as getConvertedOriginalPrice which returns the converted originalPrice

Modify the expression as shown below :

```
%dw 1.0
%var conversionRate=65
%function getConvertedOriginalPrice(price) price*conversionRate
%output application/xml
---
products: {
	(
		(payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice: getConvertedOriginalPrice($.originalPrice)
			}
		} orderBy $.product.originalPrice)[-1..0]

	)
}
```

Instead of declaring a function, we want to declare a variable pointing to a

function.

Modify the expression as shown below :

```
%dw 1.0
%var conversionRate= 65
//%function getConvertedOriginalPrice(price) price*conversionRate
%var getConvertedOriginalPrice= (price)-> price*conversionRate
%output application/xml
---
products: {
	(
		(payload map {
			product :{
				productId:$.productId,
				name:$.name,
				brandName:$.brandName,
				originalPrice: getConvertedOriginalPrice($.originalPrice)
			}
		} orderBy $.product.originalPrice)[-1..0]

	)
}
```

We want to use aliases for index and value parameters of map operator.

Modify the expression as shown below :

```
%dw 1.0
%var conversionRate= 65
%var getConvertedOriginalPrice= (price)-> price*conversionRate
%output application/xml
---
products: {
	(
		( payload map (product,indexOfProduct)->
			{
			product : {
				productId:product.productId,
				name:product.name,
				brandName:product.brandName,
				originalPrice:
getConvertedOriginalPrice(product.originalPrice)
			}
		} orderBy $.product.originalPrice
		)[-1..0]

	)
}
```

we want to display images for each product.

to get images, create a function as shown below :

```
%var getImages = (images) -> {(
                                images  map {
                                        img : $
                                        }
                                )
                               }
```

Now modify the expression as shown below :

```
products: {
        (
                ( payload map (product,indexOfProduct)->
                        {
                        product : {
                                productId:product.productId,
                                name:product.name,
                                brandName:product.brandName,
                                originalPrice:
getConvertedOriginalPrice(product.originalPrice),
                                images : getImages(product.images)



                        }
                } orderBy $.product.originalPrice
                )[-1..0]

        )
}
```

Observe the preview.

In the sample data, remove images for product with id 1.

Click the Preview button to see preview. You should see that there is an error in the dwl.

We want to display images tag conditionally only when product.images is present in the input.

So, we can use ? operator to check for presence like shown below :

```
(images : getImages(product.images) ) when product.images?
```

Modify the expression and observe product with id 1 in preview.

we want to look up for conversion rate from another flow.

Observe that there is one flow with name "getConversionRateFlow" already created for you. It return conversion rate based on country.

In data weave we want to get the conversion rate from this flow.

Modify the conversionrate variable as shown below :

```
%var conversionRate=
lookup('getConversionRateFlow',inboundProperties."http.query.params".country)
```

(Remember that if you try to preview when you are using lookup() , there will be errors.)

Deploy the application and give a request to http://localhost:8081/transformjson?country=us

You should see the originalPrice is converted .

Try giving requests to http://localhost:8081/transformjson?country=uk and http://localhost:8081/transformjson?country=india

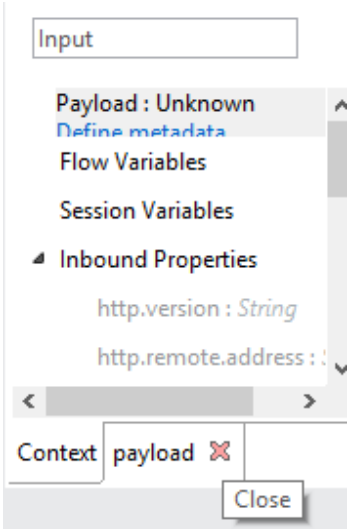
STEP 2

In this step, you will understand how to apply transformations on xml data.

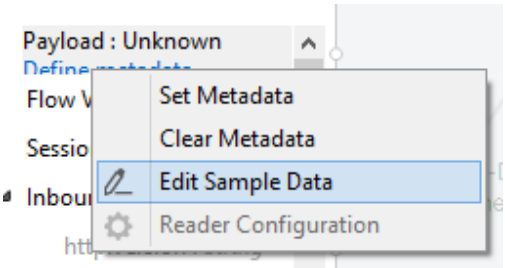
Open transformxmldata.xml which is already given to it and observe that this file is same as the one you developed in step1.

But the Http Listener URL is /transformxml

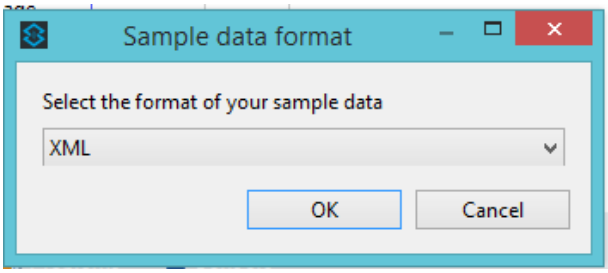
Double click on transform message component and in input section, close the payload tab. Select the option "close tab and delete file" when prompted.



Right click on payload in the input section and select "Edit Sample Data" .

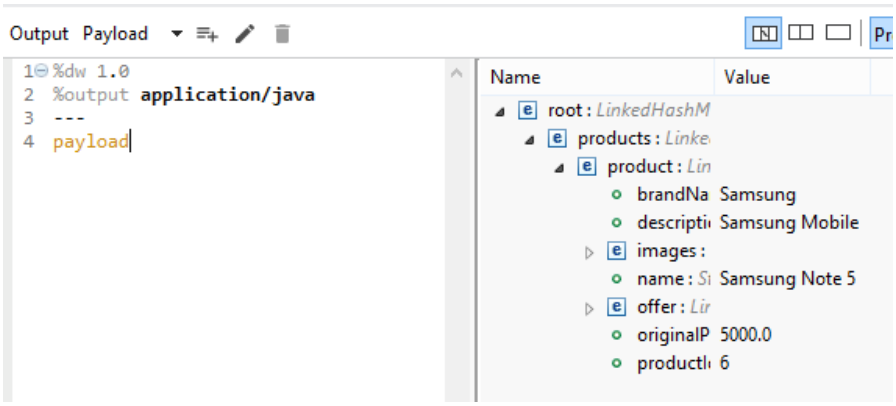


Select xml from the drop down



Copy the contents of src/main/resources/products.xml into sample payload

Modify the output to application/java and dataweave expression as payload and observe the preview. It should look like below:



Modify the expression as payload.products. It should display only first product.

Modify the expression as payload.products.product. It should display the first products properties as MAP.

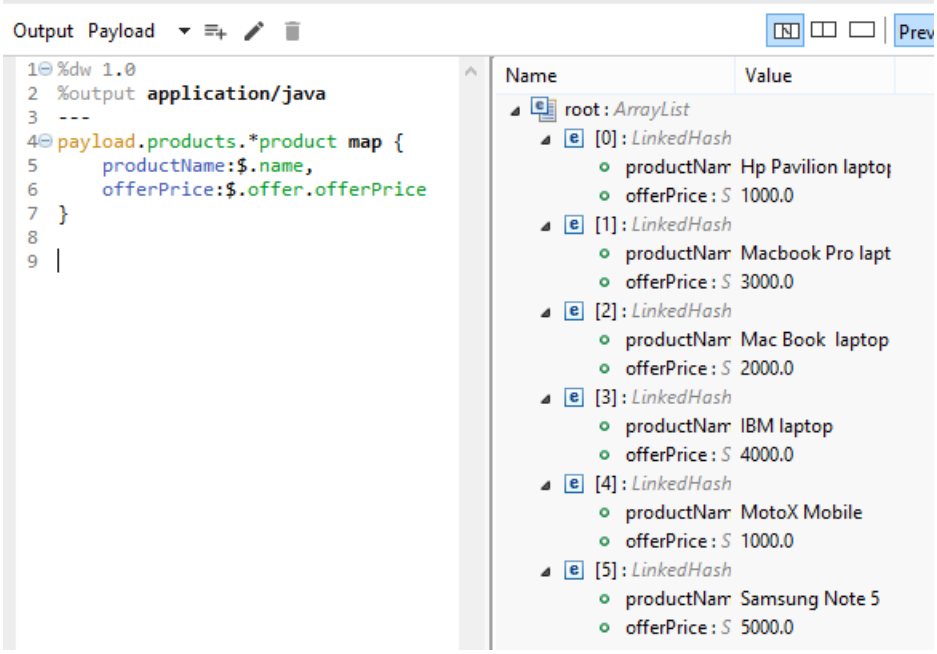
Modify the expression as payload.products.*product . you should observe the list of products

Now modify the expression as below such that it displays only list of product name and its offer price as shown below :

```
payload.products.*product map {
```

```
        productName:$.name,
        offerPrice:$.offer.offerPrice
}
```

The preview should look like below :



Now modify the output as application/xml. it should show an error as root tag is required.

Modify the expression as below :

```
products :{

            payload.products.*product map {
        productName:$.name,
        offerPrice:$.offer.offerPrice
}


}
```

You will still see that there is as error. Can you try to fix it?

Modify as shown below to fix it.

```
products :{
	(
		payload.products.*product map {
	productName:$.name,
	offerPrice:$.offer.offerPrice
}

	)
}
```

You should see the preview as shown below :

Can you modify the expression such that productName and offerPrice are wrapped in a tag with name product?

Modify the expression as shown below :

```
products :{
	(
		payload.products.*product map {
			product:{
				productName:$.name,
				offerPrice:$.offer.offerPrice
			}
		}

	)
}
```

STEP 3

In this step, you will understand how to use dataWeave operators.

Open transform to Java and observe the dataweave expression which displays List of Maps.

If you observe all the data types are String by default.

We want to type coerse productId as number and offerPrice as number

Modify the expression as below :

```
payload.products.*product map {

                        productId:$.productId as :number,
                        productName:$.name,
                        brandName:$.brandName,
                        offer:{
                               offerPrice:$.offer.offerPrice as :number,
                               offerValidUntil:$.offer.offerValidUntil
                        },

                        images: $.images.*image,
                        originalPrice:$.originalPrice


            }
```

we want to convert offerValidUntil to java.util.Date type. Modify the offerValidUntil as shown below :

```
offerValidUntil:$.offer.offerValidUntil as :date {format:"yyyy-MM-dd"}
```

If you observe, offer is of Type Map.

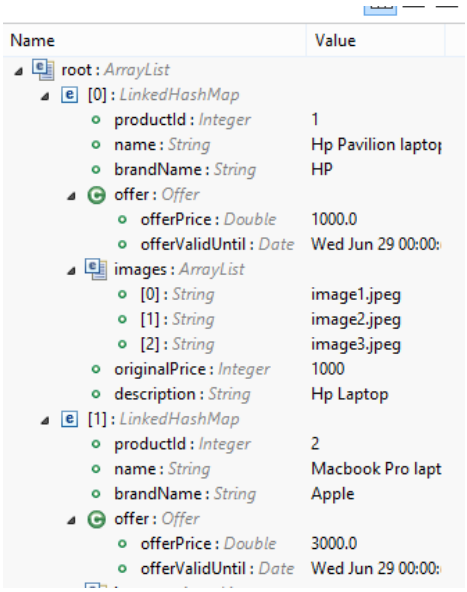
We want it to be mapped as com.way2learnonline.Offer .

Open Offer.java and observe it.

Now modify the expression for offer as shown below:

```
offer:{
       offerPrice:$.offer.offerPrice as :number,
       offerValidUntil:$.offer.offerValidUntil as :date {format:"yyyy-MM-dd"}
       } as :object {class:"com.way2learnonline.Offer"}
```

Now Observe the preview. It should look like below :



Now We want List of Product instead of List of Map

Modify the expression as below :

```
payload.products.*product map {

                        productId:$.productId as :number,
                        name:$.name,
                        brandName:$.brandName,
                        offer:{
                               offerPrice:$.offer.offerPrice as :number,
                               offerValidUntil:$.offer.offerValidUntil as :date
{format:"yyyy-MM-dd"}
                        } as :object {class:"com.way2learnonline.Offer"},

                        images: $.images.*image,
                        originalPrice: $.originalPrice as :number,
                        description: $.description

             } as :object {class:"com.way2learnonline.Product"}
```

We want to define offer and product as custom types.

You can define custom types as shown below:

```
%type myoffer= :object {class:"com.way2learnonline.Offer"}
%type myproduct= :object {class:"com.way2learnonline.Product"}
```

So, modify the expression as below

```
%dw 1.0
%output application/java
%type myoffer= :object {class:"com.way2learnonline.Offer"}
%type myproduct= :object {class:"com.way2learnonline.Product"}

---
payload.products.*product map {

                            productId:$.productId as :number,
                            name:$.name,
                            brandName:$.brandName,
                            offer:{
                                  offerPrice:$.offer.offerPrice as :number,
                                  offerValidUntil:$.offer.offerValidUntil as :date
{format:"yyyy-MM-dd"}
                            } as :myoffer,

                            images: $.images.*image,
                            originalPrice: $.originalPrice as :number,
                            description: $.description

              } as :myproduct
```

This is the end of the Exercise