LAB- Using Map , Orderby, filter, if, match and creating reusable functions

In this lab you will understand how to use map, orderBy,filter, if-else and create functions

STEP 1

Create a project with name **03-using-map-mapobject-start**

Create a mule configuration file with name **usingmap.xml**

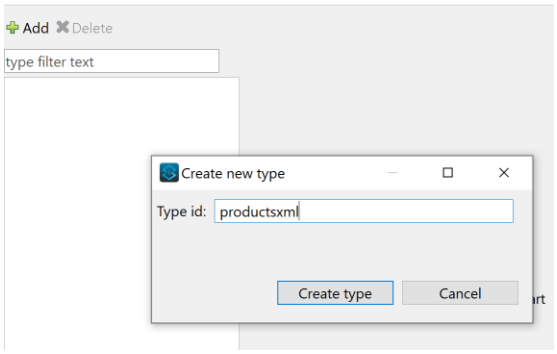
Drag “Transform Message” component in to a new flow

Now copy products.xml given to you inside src/main/resources and observe that it contains list of products in json format.

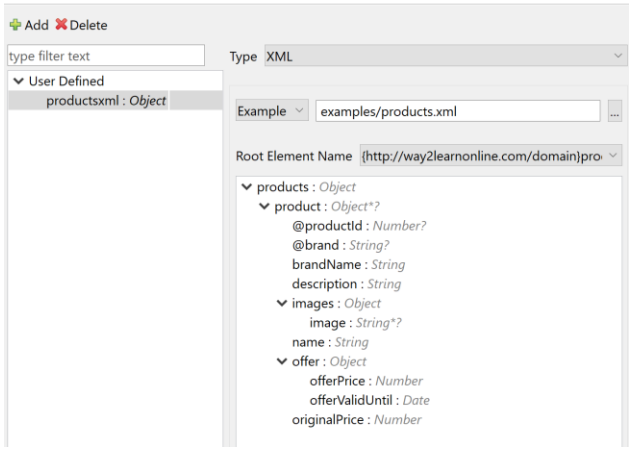
In the Input section of the Transform Message Properties view, click “Payload: Any Define metadata”

In the pop up window, click on “+Add” button . Give they type id as “productsxml”.

Select Type as xml

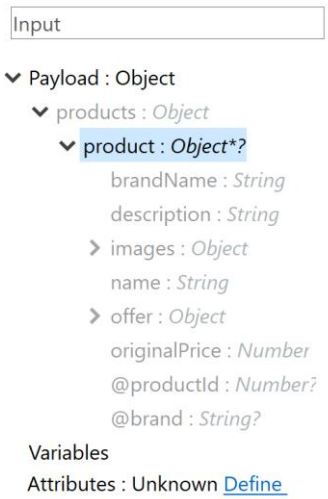


Now select the type as xml and select example in the dropdown then select src/main/resources/products.xml as shown below:

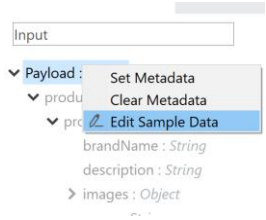


Now click on Select.

In the input metadata, you should see as shown below:



Now, right click on payload and select edit sample data as shown below:



You should see the sample data

Now modify the dataweave expression as "payload" and output as application/dw and observe the preview.

It should look like below :

```
do {
  ns way2learn http://way2learnonline.com/domain
  ---
  {
    way2learn#products: {
      product @(productId: "1"): {
        brandName: "HP",
        description: "Hp Laptop",
        images: {
          image: "image1.jpeg",
          image: "image2.jpeg",
          image: "image3.jpeg"
        },
        name: "Hp Pavilion laptop",
        offer: {
          offerPrice: "1000.0",
          offerValidUntil: "2016-06-29"
        },
        originalPrice: "1000.0"
      },
      product @(productId: "2"): {
        brandName: "Apple",
        description: "Apple Laptop",
        images: {
          image: "image3.jpeg",
          image: "image4.jpeg",
          image: "image5.jpeg"
        },
```

Now change the expression as shown below:

`payload.products.product`

U should see only first product. Do you know the reason?

Now change the expression to `payload.products.*product` and observe that result is a list of products

Create a variable to point to this list of products and change the body expression to the variable as shown below:

Output Payload

```
%dw 2.0
output application/dw

var products= payload.products.*product

---

products
```

```
[
  {
    brandName: "HP",
    description: "Hp Laptop",
    images: {
      image: "image1.jpeg",
      image: "image2.jpeg",
      image: "image3.jpeg"
    },
    name: "Hp Pavilion laptop",
    offer: {
      offerPrice: "1000.0",
      offerValidUntil: "2016-06-29"
    },
    originalPrice: "1000.0"
  },
  {
    brandName: "Apple",
    description: "Apple Laptop",
    images: {
      image: "image3.jpeg",
      image: "image4.jpeg",
      image: "image5.jpeg"
    },
    name: "Macbook Pro laptop",
```

We want to iterate over the list and prepare a list of brandName and name. So,

modify the expression as shown below :

```
products map {

}
```

Observe the preview. You should observe list of empty maps

Now change the expression as shown below:

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

Preview should look like below

Now modify the variable products as shown below

```
%dw 2.0
output application/dw

var products= payload.products.*product map {
    productId:$.@productId,
    name:$.name,
    brandName:$.brandName,
    originalPrice:$.originalPrice
}

---

products
```

Now change the body expression as below:

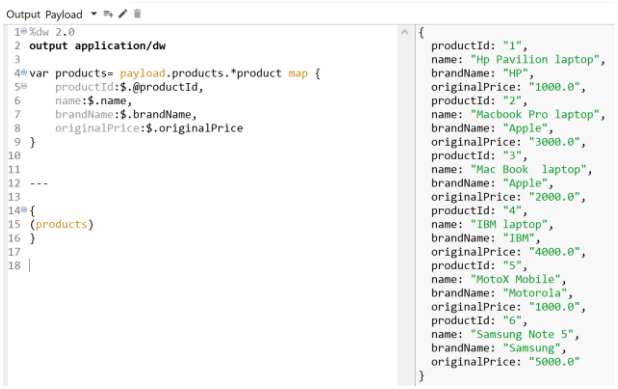
```
{
products

}
```

You should see an error.

Now change it to

```
{
(products)
}
```

You should see the result as shown below:



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: {
   (products)
}
```

You should observe the preview as shown below :

```xml
<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 variable products as shown below:

We can use $$ as index.

Modify the variable products as shown below and observe the changes in preview

```
var products= payload.products.*product map {
       'product$$':{
```

```
            productId:$.@productId,
            name:$.name,
            brandName:$.brandName,
            originalPrice:$.originalPrice
        }
}
```

After observing the preview, remove $$ in the key. Don't forget to do this before u proceed

We want all the products to be sorted by originalPrice.

Create a variable sortedProducts as shown below

```
var sortedProducts= products orderBy $.product.originalPrice
```

modify the dataweave expression as shown below

```
products: {
        (sortedProducts)
}
```

```
Verify that the products are sorted by original price.
```

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

Solution :

```
var sortedProducts= products orderBy -$.product.originalPrice
```

We want the product with least Price. Change the body expression to

```
products minBy $.product.originalPrice
```

Verify if the result is a product with least price

Similarly, try replacing minBy with maxBy

Now we want to eliminate duplicate products by brandName. So, we can use distinctBy .

Change the body expression to `products distinctBy $.product.brandName` and observe that product with id 3 whose brandName is Apple is not displayed as product with id 2 is already having brandName as Apple

We want to apply a filter which selects only products whose brandName is “HP”

```
var filteredProducts= products filter $.product.brandName=='HP'
```

modify the dataweave expression as shown below and observe the preview

```
products: {
(filteredProducts)
}
```

Till now filter looks like an operator. Actually, it is a function.

Press CTRL+Click on filter to see the syntax of filter. It should look like below:

```
fun filter <T>(@StreamCapable items: Array<T> , criteria: (item: T, index: Number)
-> Boolean): Array<T> = native("system::ArrayFilterFunctionValue")
```

Now let us try to use filter like a function. Change the variable as shown below:

```
var filteredProducts=  filter(products, (item,index) ->
item.product.brandName=='HP')
```

verify that the result is same like before.

Now we want all the products whose brandName is not HP. So, we can use not operator as shown below:

```
var filteredProducts= products filter not  $.product.brandName=='HP'
```

Try out and verify the result.

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

```
var filteredProducts= products filter
$.product.brandName==attributes.queryParams.brandName
```

Remember that preview may not be shown when you write expression using attributes or vars.

If you want the preview to be shown, right click on attributes in the input section and select “Edit Sample data” and give the data as

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

We want to group all the products by brandName

Change the body expression to `products groupBy $.product.brandName`

You should observe the preview as shown below:

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

Modify the expression as shown below :

```
var products= payload.products.*product map(product,productindex) -> {
        'product':{
                productId:product.@productId,
                name:product.name,
                brandName:product.brandName,
                originalPrice:product.originalPrice
        }
}
```

we want to display images for each product with <img> tag for each images

Change the products variable as shown below:

```
var products= payload.products.*product map(product,productindex) -> {
        'product':{
                productId:product.@productId,
                name:product.name,
                brandName:product.brandName,
                originalPrice:product.originalPrice,
                images: {

                                (
                                        product.images.*image map {
                                                img: $
```

```
                                        }
                                )

                        }
                }
}
```

You should observe the preview as shown below:

Writing such a big expression makes it difficult to read. Better to divide a bigger expression into small expression by writing functions.

Write a function like below :

```
fun getImages(images) = {
                                (   images map {
                                                img: $
                                                }
                                )
                                }
```

Write Now call this function as shown below :

```
var products= payload.products.*product map(product,productindex) -> {
    'product':{
        productId:product.@productId,
        name:product.name,
        brandName:product.brandName,
        originalPrice:product.originalPrice,
        images: getImages(product.images.*image)
    }
}
```

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

Click the Preview button to see preview. You should see that there is empty <image> tag

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.*image) ) if product.images?
```

Observe the preview and make sure that there is no images tag for first product

We want to convert product name in to upper case . use upper function present in core module as shown below:

```
var products= payload.products.*product map(product,productindex) -> {
    'product':{
        productId:product.@productId as Number,
        name: upper(product.name) ,
        brandName:product.brandName,
        originalPrice:product.originalPrice,
        offerValidUntill: product.offer.offerValidUntil as Date,
        (images: getImages(product.images.*image) ) if product.images?
    }
}
```

Now we want to convert all the keys into upper case.

So, write a function like below:

```
fun convertKeysToUpperUsingif(obj)=
                obj mapObject(value,key) -> {
                      (upper(key)) : value

               }
```

Change the body expression to below:

```
convertKeysToUpperUsingif({products:{(products)}})
```

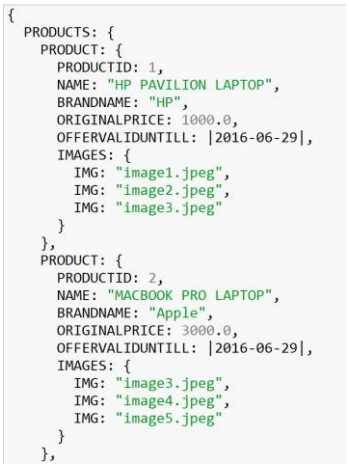
View the preview and observe that only top level key is converted to Upper.

So, we need to call the same function recursively if value is of type Object

Now modify the function as shown below:

```
fun convertKeysToUpperUsingif(obj)=
                    if(typeOf(obj) ~= Object)
                obj mapObject(value,key) -> {
                     (upper(key)) : convertKeysToUpperUsingif(value)

                    }
            else
             obj
```

You should observe that all the keys are un upper case as shown below:



```
{
  PRODUCTS: {
    PRODUCT: {
      PRODUCTID: 1,
      NAME: "HP PAVILION LAPTOP",
      BRANDNAME: "HP",
      ORIGINALPRICE: 1000.0,
      OFFERVALIDUNTILL: |2016-06-29|,
      IMAGES: {
        IMG: "image1.jpeg",
        IMG: "image2.jpeg",
        IMG: "image3.jpeg"
      }
    },
    PRODUCT: {
      PRODUCTID: 2,
      NAME: "MACBOOK PRO LAPTOP",
      BRANDNAME: "Apple",
      ORIGINALPRICE: 3000.0,
      OFFERVALIDUNTILL: |2016-06-29|,
      IMAGES: {
        IMG: "image3.jpeg",
        IMG: "image4.jpeg",
        IMG: "image5.jpeg"
      }
    },
```

Now we want all the values of Date to be in format dd,MMM,yyyy

Modify the function as below:

```
fun convertKeysToUpperUsingif(obj)=
                    if(typeOf(obj) ~= Object)
                obj mapObject(value,key) -> {
                     (upper(key)) : convertKeysToUpperUsingif(value)
```

```
                }
            else
             if(typeOf(obj) ~= Date)
              obj as Date as String {format:"dd,MMM,yyyy"}
              else
              obj
```

Preview must look like below: (Observe the OfferValidUntil value)


```
{
  PRODUCTS: {
    PRODUCT: {
      PRODUCTID: 1,
      NAME: "HP PAVILION LAPTOP",
      BRANDNAME: "HP",
      ORIGINALPRICE: "1000.0",
      OFFERVALIDUNTILL: "29,Jun,2016" as String {format: "dd,MMM,yyyy"},
      IMAGES: {
        IMG: "image1.jpeg",
        IMG: "image2.jpeg",
        IMG: "image3.jpeg"
      }
    },
    PRODUCT: {
      PRODUCTID: 2,
      NAME: "MACBOOK PRO LAPTOP",
      BRANDNAME: "Apple",
      ORIGINALPRICE: "3000.0",
      OFFERVALIDUNTILL: "29,Jun,2016" as String {format: "dd,MMM,yyyy"},
      IMAGES: {
        IMG: "image3.jpeg",
        IMG: "image4.jpeg",
        IMG: "image5.jpeg"
      }
    },
```


We want all the Numbers to be of format #,###.00

Modify the function as shown below:

```
fun convertKeysToUpperUsingif(obj)=
                    if(typeOf(obj) ~= Object)
                obj mapObject(value,key) -> {
                     (upper(key)) : convertKeysToUpperUsingif(value)

                    }
            else
             if(typeOf(obj) ~= Date)
              obj as Date as String {format:"dd,MMM,yyyy"}
              else
              if(typeOf(obj) ~= Number)
               obj as String {format:"#,###.00"}
              else
              obj
```

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

Instead of using many if-else statements, it is better to use **match** operator

Now create a new function as shownBelow:

```
fun convertKeysToUpperusingmatch(obj)=
                obj match {

                     else -> obj mapObject(value,key) -> {
                      (upper(key)) : value

                     }
                }
```

Change the body expression to call this function as shown below:

```
convertKeysToUpperusingmatch({products:{(products)}})
```

You should observe that only root tag is converted to upper.

Now modify the function as shown below:

```
fun convertKeysToUpperusingmatch(obj)=
                obj match {
                           case is String -> obj
                           case is Number -> obj as String {format:"#,###.00"}
                           case is Date -> obj as Date as String {format:"dd,MMM,yyyy"}

                     case myinput is Object -> myinput mapObject(value,key) -> {
                      (upper(key)) : convertKeysToUpperusingmatch(value)

                     }

                     else -> "Incorrect Inputttt"
                }
```

This is the end of the Exercise