LAB- Using reduce function

In this lab you will understand how to use reduce function

Create a project with name **06-using-reduce-start**

Create a new Mule configuration file with name “usingreduce.xml”

Drag a “Transform Message” component into a new flow and rename the flow as “usingreduce”

Create a variables as shown below:

```
var numbers =1 to 5
var nestedArray = [ [1], [2,3], ["hello"] ]
var objArray = [
                          {one: 1},
                          {two: 2, three: 3}
                          ]
var stringArray = ["one", "two", "three", "four"]
```

Change the body to

```
numbers reduce (number,acc) -> number
```

You should observe the last element of the number

Change the body to

```
{
a: numbers reduce (element, acc) -> element,
b: numbers reduce $
}
```

Observe that both expressions results are same

Use the below expression as body to get sum of all elements in array

```
numbers reduce (element,acc) -> acc+element
```

In this case when we use +, default value of accumulator is 0

Use the below expression as body to get all number numbers in array multiplied

```
numbers reduce (element,acc) -> acc*element
```

In this case accumulator value is 1

We can change the default value of accumulator as shown below:

```
numbers reduce (element,acc=2) -> acc*element
```

We can use below expression to concatenate elements of array as string

```
numbers reduce (element,acc) ->acc ++ element
```

We can use below expression to concatenate elements of array as string in reverse order

```
numbers reduce (element,acc) -> element++ acc
```

We can accumulate the elements in an array into an object by using {} as accumulator

```
objArray reduce (element,acc={}) ->acc ++ element
```

We can flatten a given array into a new array as shown below:

```
nestedArray reduce (element,acc=[]) -> acc++ element
```

Observe that output of above expression is same as `flatten(nestedArray)`

We want to get the count of numbers in array which has more than 3 letters

```
stringArray reduce (element, acc=0) ->
                                       (if(sizeOf(element) < 4) 1 else 0) + acc
```

Observe that result above expression is same as

```
stringArray dw::core::Arrays::countBy(sizeOf($) < 4)
```

Open products.xml and observe that there are 12 products with different brand names like "HP","Apple", "Samsung","IBM" and "Motorola"

We want to calculate total price and counts of all the products grouped by brandNames

Observe the below dwl

```
var products = payload.products.*product

var myacc ={
      hp: {totalPrice: 0,count:0},
      apple: {totalPrice: 0,count:0},
      ibm: {totalPrice: 0,count:0},
      samsung: {totalPrice: 0,count:0},
      motorola: {totalPrice: 0,count:0},
}

var accumulatedValues = products  reduce (product, acc =myacc) -> acc -
lower(product.brandName) ++
                                          (lower(product.brandName)):{
                                                totalPrice:
acc[lower(product.brandName)].totalPrice + product.originalPrice,
                                                count:
acc[lower(product.brandName)].count+1
                                          }

var averages= accumulatedValues mapObject (V,K,I) -> (K):V.totalPrice/V.count
---
products map (item)-> item ++ avgPrice : averages[lower(item.brandName)]
```

This is the end of the Exercise