Json to xml

Example:1

Input:-{

"title": "Java 8 in Action: Lambdas, Streams, and functional-style programming",

"author": "Mario Fusco",

"year": 2014

}

Output:-<?xml version='1.0' encoding='UTF-8'?>

<order>

<type>Book</type>

<title>Java 8 in Action: Lambdas, Streams, and functional-style programming</title>

<details>By Mario Fusco (2014)</details>

</order>

Logic:-{

order: {

type: "Book",

title: payload.title,

details: "By $(payload.author) ($(payload.year))" }}

Note:- we can concordant values in this way also and in this method of Concorde nation we add special characters as we see above <details>By Mario Fusco (2014)</details>

=====================================================================

i.Example2:

Input:-

<users>

<user>

<name>Mariano Achaval</name>

<phone>152235465654</phone>

<street>Laprida 924</street>

</user>

<user>

<name>Martin Alejandro Cousido</name>

<phone>15332255555</phone>

<street>Acassuso 2280</street>

</user>

</users>

Output:-

{"contacts": [

{

"Name": "Mariano",

"LastName": "Achaval",

"SecondName": null,

"Email": "Mariano.Achaval@gmail.com",

"Phone": "152235465654",

"Street": "Laprida 924"

},

{

"Name": "Martin",

"LastName": "Alejandro",

"SecondName": "Cousido",

"Email": "Martin.Alejandro.Cousido@gmail.com",

"Phone": "15332255555",

"Street": "Acassuso 2280" }]}

Logic:-

contacts:

(payload.users.\*user map {

Name:($.name splitBy (" ")) [0],

LastName:($.name splitBy (" ")) [1],

SecondName:($.name splitBy (" ")) [2],

Email:$.name replace" " with "." ++ "@gmail.com",

Phone:$.phone,

Street:$.street})

Need to improve :-

is key is not available need to remove the tag

logic:- payload.users.\*user map ((item, index) ->{

Name:(item.name splitBy (" "))[0],

(lastname:(item.name splitBy(" "))[1]) if isBlank((item.name splitBy(" "))[1]),

Email:(item.name++"@gmail.com")replace " " with "."} )

(Or)

payload.users.\*user map ((item, index) ->{

Name:(item.name splitBy (" "))[0],

(lastname:(item.name splitBy(" "))[1]) if ((item.name splitBy(" "))[1]?),

Email:(item.name++"@gmail.com")replace " " with "."

} )

need to remove the contacts in out put

logic:-filter ((item, index) -> !isBlank(item.lastname) )

Input:- [

{

"Type" :"book",

"label" : "Java Curiosities",

"author" : "Luis Miguel",

"image": "1",

"url": "s"

},

{

"Type" :"CD",

"label" : "No culpes a la noche",

"author" : "Luis Miguel",

"image":"w"

},

{

"Type" :"book",

"label" : "Mule in Action",

"author" : "Victor Romero"

},

{

"Type" :"CD",

"label" : "Yesterday",

"author" : "The Beatles"

}

]

Logic:-%dw2.0

output application/xml

---

(order:

order1:payload map (value) -> {id:value.Type,

label:value.label,

Author:value.author,

Image:value.image,

Url:value.url

})

Output:-<?xmlversion='1.0'encoding='UTF-8'?>

<order>

<order1>

<id>book</id>

<label>Java Curiosities</label>

<Author>Luis Miguel</Author>

<Image>1</Image>

<Url>s</Url>

</order1>

<order1>

<id>CD</id>

<label>No culpes a la noche</label>

<Author>Luis Miguel</Author>

<Image>w</Image>

<Url/>

</order1>

<order1>

<id>book</id>

<label>Mule in Action</label>

<Author>Victor Romero</Author>

<Image/>

<Url/>

</order1>

<order1>

<id>CD</id>

<label>Yesterday</label>

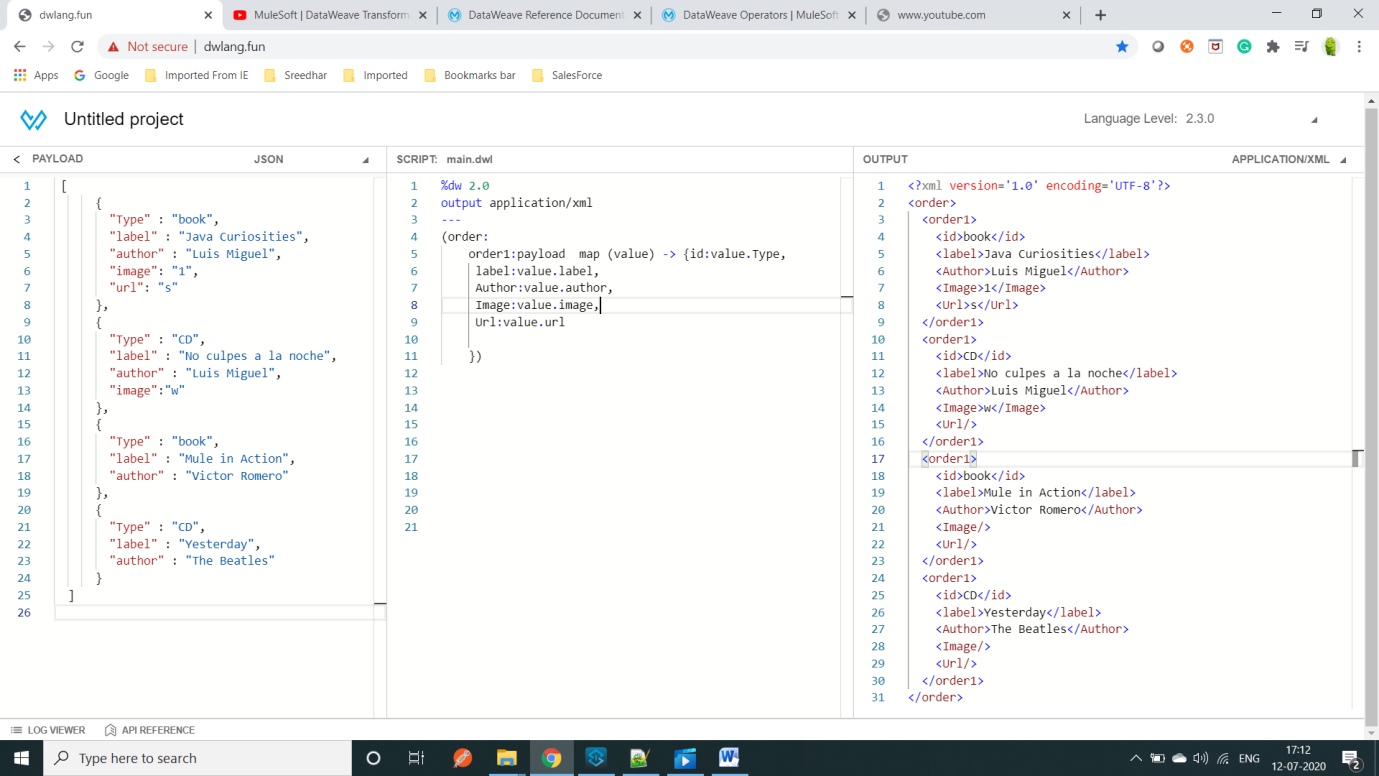
<Author>The Beatles</Author>

<Image/>

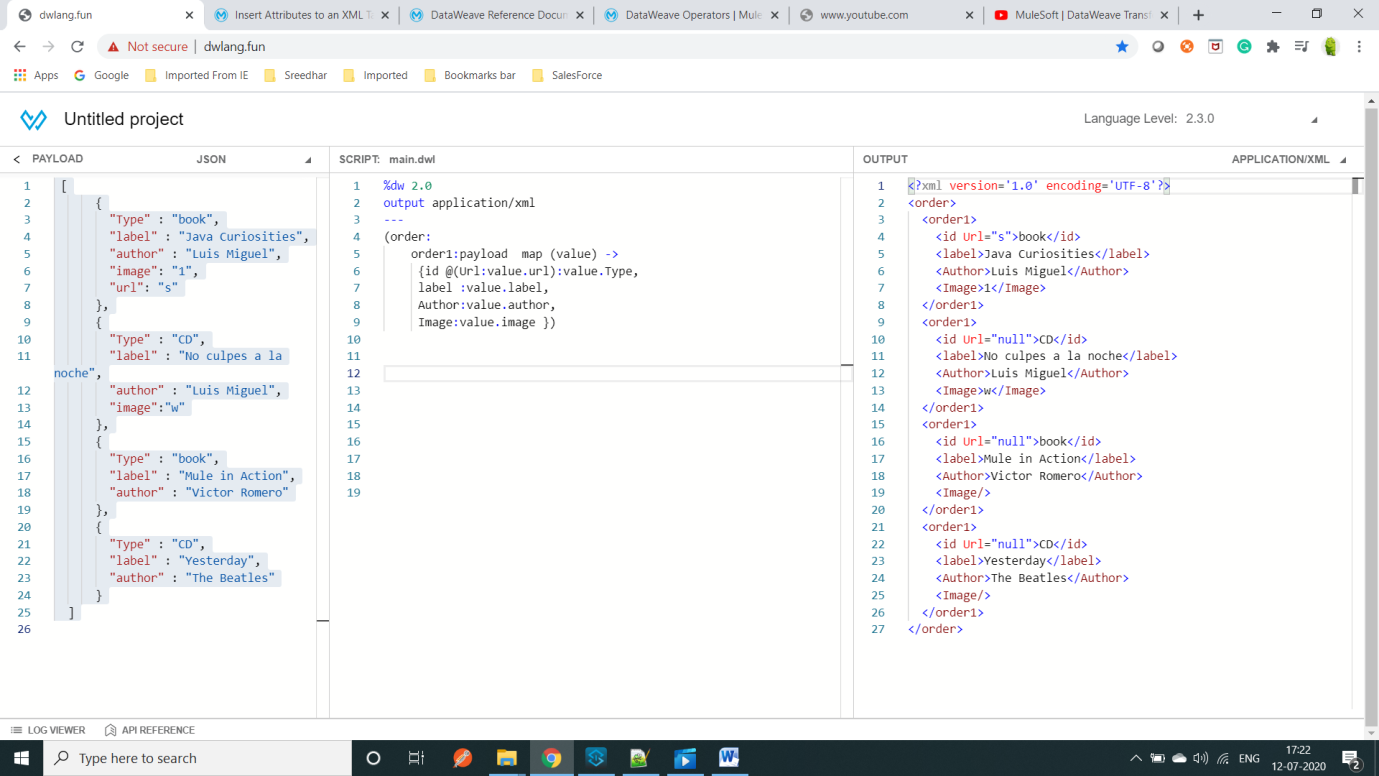
<Url/>

</order1>

</order>

Pick:-

Example:- how To write attributes @(Url:value.url)



Input:-[

{

"Type" :"book", "label" : "Java Curiosities", "author" : "Luis Miguel", "image": "1","url": "s"},{

"Type" :"CD","label" : "No culpes a la noche","author" : "Luis Miguel","image":"w"},{

"Type" :"book","label" : "Mule in Action","author" : "Victor Romero"},{

"Type" :"CD","label" : "Yesterday","author" : "The Beatles"}]

Logic:-(order: order1:payload map (value) ->

{id@(Url:value.url):value.Type,

label:value.label,

Author:value.author,

Image:value.image })

Output:-<?xmlversion='1.0'encoding='UTF-8'?>

<order>

<order1>

<idUrl="s">book</id>

<label>Java Curiosities</label>

<Author>Luis Miguel</Author>

<Image>1</Image>

</order1>

<order1>

<idUrl="null">CD</id>

<label>No culpes a la noche</label>

<Author>Luis Miguel</Author>

<Image>w</Image>

</order1>

<order1>

<idUrl="null">book</id>

<label>Mule in Action</label>

<Author>Victor Romero</Author>

<Image/>

</order1>

<order1>

<idUrl="null">CD</id>

<label>Yesterday</label>

<Author>The Beatles</Author>

<Image/>

</order1>

</order>

Exampe:- converting payload Json to xml

Input:-[ {

"Type": "book",

"label": "Java Curiosities",

"author": "Luis Miguel",

"image": "1",

"url": "s","offer": {"offerprice": 1000,

"offerValidunit": 1313133 },

"brandName": "Hp",

"images": [ "image.jpeg","image1.jpeg","image2.jpeg","image3.jpeg"]},

{

"Type": "CD",

"label": "No culpes a la noche",

"author": "Luis Miguel",

"image": "w",

"offer": { "offerprice": 1000,

"offerValidunit": 13133 },

"brandName": "Hp",

"images": [ "image.jpeg", "image1.jpeg", "image2.jpeg", "image3.jpeg" ] },

{

"Type": "book",

"label": "Mule in Action",

"author": "Victor Romero",

"offer": {

"offerprice": 1000,

"offerValidunit": 1313133 },

"brandName": "Hp",

"images": [ "image.jpeg", "image1.jpeg", "image2.jpeg"],

"phone":[

{"phone":"Home",

"phoneNumber":123132},

{"phone":"Type",

"phoneNumber":123132} ] },

{

"Type": "CD",

"label": "Yesterday",

"author": "The Beatles",

"offer": {

"offerprice": 1000,

"offerValidunit": 1313133 },

"brandName": "Hp",

"images": [ "image.jpeg","image1.jpeg","image2.jpeg","image3.jpeg"]}]

Logic:-

%dw2.0

output application/xml

---

prodect:

prodect:payload map (value,index) -> {

Type:value.Type,

label:value.label,

Author:value.author,

OFFer:{

Offer:value.offer.offerprice,

offerValiduniT:value.offer.offerValidunit,

IMG:{

(value.images map (item,index) -> {

IMAGES:item}),

Phone:{(value.phone map (value1,index) -> {

Phone:value1.phone,

Phonenumber:value1.phoneNumber})}}}}

Logic-2:-users:

userid: payload map(value,index) -> { Type:value.Type,

Label:value.label,

Author:value.author,

Images:value.image,

URL:value.url,

Offer:{ offerPrice:value.offer.offerprice,

offerValiduniT:value.offer.offerValidunit},

imag: {( value.images map {imag:$})},

(phone:{(value.phone map (item,index) -> {

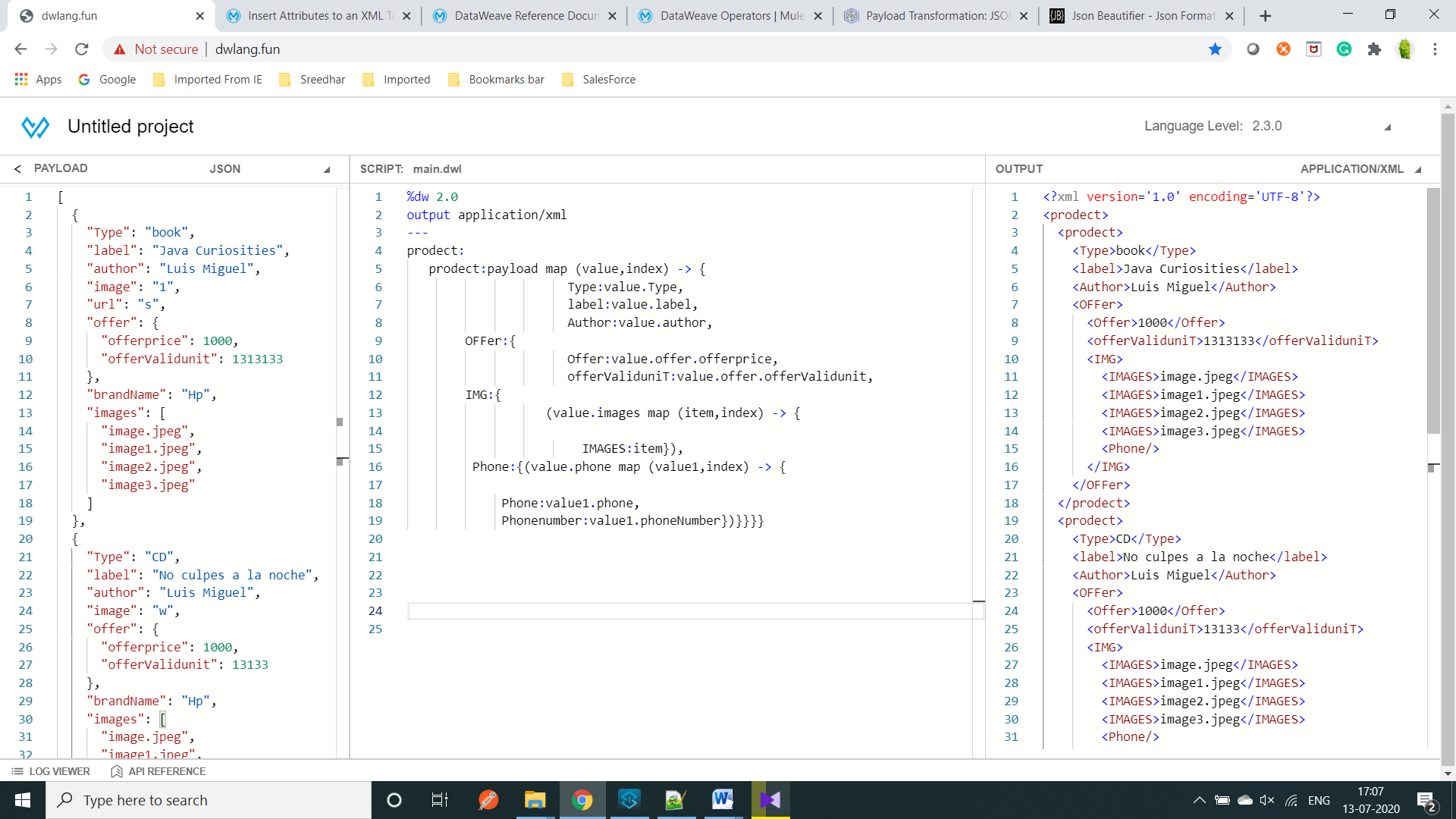
PHONE:item.phone,

PhoneNumber:item.phoneNumber})})

ifvalue.phone?

}

Note:- for this above logic if key is not present in output the key is not going to print

Pic:-

OutPut:-

<?xmlversion='1.0'encoding='UTF-8'?>

<prodect>

<prodect>

<Type>book</Type>

<label>Java Curiosities</label>

<Author>Luis Miguel</Author>

<OFFer>

<Offer>1000</Offer>

<offerValiduniT>1313133</offerValiduniT>

<IMG>

<IMAGES>image.jpeg</IMAGES>

<IMAGES>image1.jpeg</IMAGES>

<IMAGES>image2.jpeg</IMAGES>

<IMAGES>image3.jpeg</IMAGES>

<Phone/>

</IMG>

</OFFer>

</prodect>

<prodect>

<Type>CD</Type>

<label>No culpes a la noche</label>

<Author>Luis Miguel</Author>

<OFFer>

<Offer>1000</Offer>

<offerValiduniT>13133</offerValiduniT>

<IMG>

<IMAGES>image.jpeg</IMAGES>

<IMAGES>image1.jpeg</IMAGES>

<IMAGES>image2.jpeg</IMAGES>

<IMAGES>image3.jpeg</IMAGES>

<Phone/>

</IMG>

</OFFer>

</prodect>

<prodect>

<Type>book</Type>

<label>Mule in Action</label>

<Author>Victor Romero</Author>

<OFFer>

<Offer>1000</Offer>

<offerValiduniT>1313133</offerValiduniT>

<IMG>

<IMAGES>image.jpeg</IMAGES>

<IMAGES>image1.jpeg</IMAGES>

<IMAGES>image2.jpeg</IMAGES>

<Phone>

<Phone>Home</Phone>

<Phonenumber>123132</Phonenumber>

<Phone>Type</Phone>

<Phonenumber>123132</Phonenumber>

</Phone>

</IMG>

</OFFer>

</prodect>

<prodect>

<Type>CD</Type>

<label>Yesterday</label>

<Author>The Beatles</Author>

<OFFer>

<Offer>1000</Offer>

<offerValiduniT>1313133</offerValiduniT>

<IMG>

<IMAGES>image.jpeg</IMAGES>

<IMAGES>image1.jpeg</IMAGES>

<IMAGES>image2.jpeg</IMAGES>

<IMAGES>image3.jpeg</IMAGES>

<Phone/>

</IMG>

</OFFer>

</prodect>

</prodect>

==============================================================================

Input:- {

"drivers": [

{

"id": "pcR:11109",

"firstName": "ALM#3771",

"lastName": "IN0605",

"dateOfBirth": "01/09/1092",

"addresses": [ {

"type": "primary","addressLine1": "2503 N DELAWARE ST","city": "Test","state": "TN",

"postalCode": "43211" },

{

"type": "office", "addressLine1": "10, Nadesan Street","city": "Chennai", "state": "TN",

"postalCode": "600017" }],

"license": {"state": "IN","number": "S234567890"} },

{

"id": "pcR:11110",

"firstName": "ALM#3771",

"lastName": "IN0605",

"dateOfBirth": "31/10/1993",

"addresses": [{

"type": "primary","addressLine1": "123 N VIVEKANANTHA ST","city": "Test2", "state": "TN",

"postalCode": "12345" },

{

"type": "office","addressLine1": "100, GN Chetty Road","city": "Chennai","state": "TN",

"postalCode": "600017" }],

"license": {"state": "IN", "number": "S234567890" }}]}

Logic:-

%dw 2.0

output application/xml

---

"order":{

"products":{

"clue\_auto" : {

"type" @(attachments:"no",currentCarrier:"no", driverDiscovery:"no",subjectSearch:"y",vinSearch:"no"): { },

        "units":{ (payload.drivers map (v,i) -> {  "subject" @(ref:"S" ++ (i + 1)):{}

})}}},

"dataset":{

"subjects":{

(payload.drivers map (value,index) -> {

"subject" @(id:"S" ++ (index+1), quoteback: value.id) :{

"name":{

"first" : value.firstName,

"last" : value.lastName },

"birthdate" : value.dateOfBirth,

(value.addresses map (value1,index1) -> {

"address" @(ref:"A" ++ (index1+1) ++ "S" ++ (index+1),

"type": value1."type") : {}}),

"driverslicense" @(number: value.license.number, state:value.license.state): {}}})},

"addresses" :{

( payload.drivers map (value3,index3) -> {

(value3.addresses map (value4,index4) -> {

"address" @(ref:"A" ++ (index4+1) ++ "S" ++ (index3+1)) :{

"street1": value4.addressLine1,

"city": value4.city,

"state" : value4.state,

"postalcode" : value4.postalCode }})})}}}

Output:- <?xml version='1.0' encoding='UTF-8'?>

<order>

<products>

<clue\_auto>

<type attachments="no" currentCarrier="no" driverDiscovery="no" subjectSearch="y" vinSearch="no"/>

<units>

<subject ref="S1"/>

<subject ref="S2"/>

</units>

</clue\_auto>

</products>

<dataset>

<subjects>

<subject id="S1" quoteback="pcR:11109">

<name>

<first>ALM#3771</first>

<last>IN0605</last>

</name>

<birthdate>01/09/1092</birthdate>

<address ref="A1S1" type="primary"/>

<address ref="A2S1" type="office"/>

<driverslicense number="S234567890" state="IN"/>

</subject>

<subject id="S2" quoteback="pcR:11110">

<name>

<first>ALM#3771</first>

<last>IN0605</last>

</name>

<birthdate>31/10/1993</birthdate>

<address ref="A1S2" type="primary"/>

<address ref="A2S2" type="office"/>

<driverslicense number="S234567890" state="IN"/>

</subject>

</subjects>

<addresses>

<address ref="A1S1">

<street1>2503 N DELAWARE ST</street1>

<city>Test</city>

<state>TN</state>

<postalcode>43211</postalcode>

</address>

<address ref="A2S1">

<street1>10, Nadesan Street</street1>

<city>Chennai</city>

<state>TN</state>

<postalcode>600017</postalcode>

</address>

<address ref="A1S2">

<street1>123 N VIVEKANANTHA ST</street1>

<city>Test2</city>

<state>TN</state>

<postalcode>12345</postalcode>

</address>

<address ref="A2S2">

<street1>100, GN Chetty Road</street1>

<city>Chennai</city>

<state>TN</state>

<postalcode>600017</postalcode>

</address>

</addresses>

</dataset>

</order>