

LAB- Using Batch

STEP1

Create a new Project with name **01-batch-start**

Before you proceed, execute tables.sql on your database so that a table with name sales will be created

1) create a new mule configuration file with name batch.xml .

We want to poll for files in input folder.

We are expecting sales.csv files in the input directory.

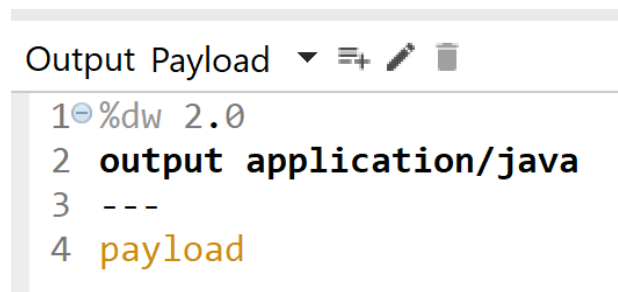
After sales.csv is received, we want to process each record in csv and write them to database concurrently using batch module

Add File Module.

Drag “On New or Upload File”. Configure File connector with working directory as c:\files

Configure directory as “input”.

Drag Transform Message Component and configure it as shown below:




```
Output Payload ▼ ⚙ ✎ 🗑
1 %dw 2.0
2 output application/java
3 ---
4 payload
```

This will Create a List as payload

We want to store the size of the list in a variable with name “size”.

Drag set-variable transformer and configure it as shown below :

Display Name:	size
Settings	
Name:	size
Value:	 <code>#[sizeOf(payload)]</code>

Drag batch job after “Set Variable” transformer.

Click on Step and rename it as “Step1”

Drag another step into Batch Job and rename it as step2.

In Step1, we want to check if there is a record in database with same name and city.

If record is existing in database, we want to create a record variable with name “ispresent” with value “true”. Else “false”

To extract values of city and name in the record into variables, drag and configure “Transform Message” as shown below to create variables name and city. (**Remember to create 2 variables name and city in transform message**)

```
Output Variable - name ▼ ⚙️ ✎ 🗑️
1 %dw 2.0
2 output application/java
3 ---
4 payload.Name as String
```

```
Output Variable - city ▼ ⚙️ ✎ 🗑️
1 %dw 2.0
2 output application/java
3 ---
4 payload.City as String
```

Now to check if this record is present in database, Drag “Select” operation in database module.

Configure Database Connector and give the query as `select * from sales where Name=:myname and City=:mycity`

In the Input Parameters , configure as shown below:

```
%dw 2.0
output application/java
---
{
    mycity: vars.city,
    myname: vars.name
}
```

Click on Advanced tab of “Select” component and configure a variable `isPresent` as shown below:

General	Fetch size: <input type="text" value="100"/>
Advanced	Max rows: <input type="text" value="100"/>
Error Mapping	Parameter types: None
Metadata	Output
Notes	Target Variable: ispresent
Help	Target Value: <input type="text" value="#[sizeof(payload) > 0]"/>
	Connection

In Step2, drag a logger and configure it to log "in step2---- before aggregator".

Click on Step2 Configure accept expression as shown below:

Name:	<input type="text" value="step2"/>
Accept Expression:	<input type="text" value="#[not vars.ispresent]"/>
Accept Policy:	NO_FAILURES (Default)

Drag a batch aggregator after logger.

Click on Batch aggregator and configure Aggregator size as 50

Inside batch aggregator, the payload will be a list.

Drag a logger and configure it with below expression:

'In Aggregator=== size is' ++ sizeof(payload)

Drag "Bulk Insert" component of Database module inside Batch Aggregator

Configure Input Parameters of "Bulk Insert " as shown below:

```
payload map (record,indexOfRecord) ->{  
    mycity: record.City as String,  
    myname: record.Name as String
```

```
}
```

Configure query as

```
insert into sales(Name, City) values(:myname, :mycity )
```

In “On Complete” section of Batch Job , Drag a logger to log `payload.processedRecords`

Run the application and keep sales.csv in the input folder .

Observe that the records are inserted into database.

Observe that each record is processed concurrently by multiple threads

This is the end of the Exercise