



Wrocław MuleSoft

Meetup #1

April 25th, 2024



- Actively participate
- Propose interesting topics
- Present at future meetups
- Propose improvements/changes/ideas
- Have fun!!



Organizer



Kuba Cieplucha

MuleSoft Developer, Competence Lead

Nextview Consulting

- 3+ years experience in MuleSoft
- Started as a Java, SQL Developer



Sponsors



nextview...

Special thanks to:



Philip Poynton
Head of Recruitment
Nextview Consulting



Jarosław Kotuła
Recruiter
Nextview Consulting



AGENDA

- Following ELK on the data integration trail
- DataWeave Libraries
- Wrap-up & Quiz
- Networking time



MuleSoft Certifications transition to the Salesforce Certification program



- Certifications will be administered through Webassessor
- If you already have a Webassessor account no action is necessary. You will receive an email confirmation
- If you do not have a Webassessor account one will be created for you using the email address on your MuleSoft Training profile. You will receive a welcome email with login instructions and next steps
- Dates:
 - ~~April 19, 2024: Last day to purchase MuleSoft exams through MuleSoft Training.~~
 - April 26, 2024: Last day to take a MuleSoft exam on MuleSoft Training
 - May 6, 2024: Start scheduling future exams via Webassessor.
- No maintenance is required for MuleSoft certification holders with pending maintenance as of January 1, 2024. Moving forward, maintenance requirements will be fulfilled through Trailhead, with more details to come in 2025.



Speaker



Leszek Gersztyn

Software Engineer

EG/AS

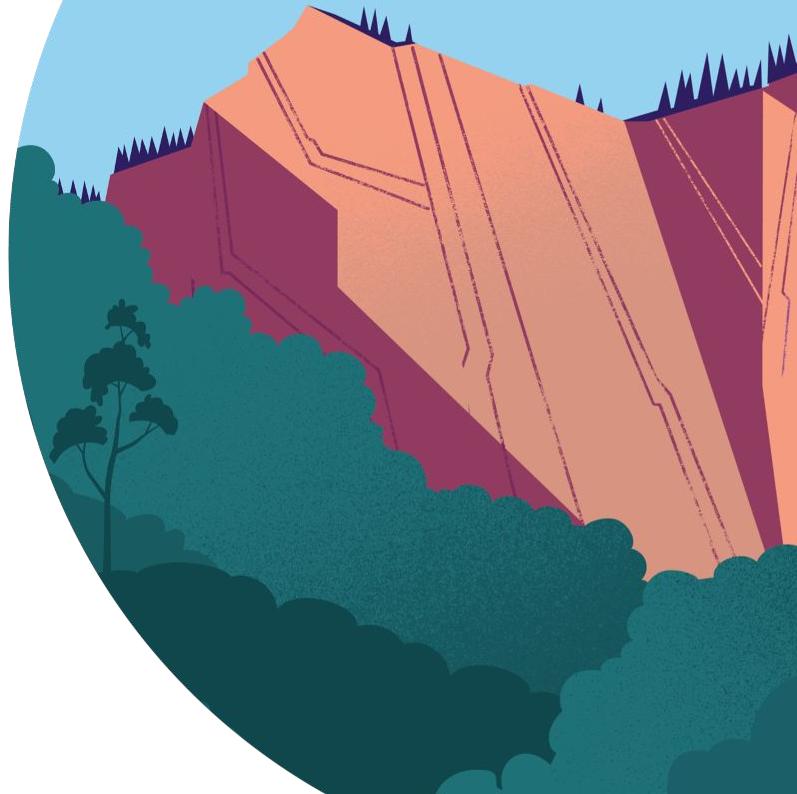
- 5+ years experience in MuleSoft
- Previously a Java Developer
- RabbitMq enthusiast
- PhD in Agronomy (really)

Following ELK on the data integration trail



Looking for...

- Better understanding of the processes in our MuleSoft landscape.
- One place to-go for resolving issues with the flow.
- Keep track of the message flowing in the system.



Solution - use RTF



The screenshot shows the MuleSoft Runtime Manager interface with the following highlights:

- 1 Applications**: A red box highlights the 'Applications' tab in the left sidebar, which is currently selected.
- Logs**: A red box highlights the 'Logs' tab in the top right corner of the main deployment area.
- Forward application logs to a external service**: A red box highlights the checkbox for enabling log forwarding to an external service.
- Deploy**: A red box highlights the large blue 'Deploy' button at the bottom right of the deployment form.

Application Overview:

- Application status: Running
- Configuration: d574c6
- Last updated: 2021-07-15 12:45:41PM
- Replicas: 1 / 1 started
- Application url: [REDACTED]
- Target name: [REDACTED]
- Target type: Runtime Fabric

Deployment Form:

- Application File: hellofabric.jar
- Version: 1.0.3
- Deployment Target (selected): [REDACTED]
- Ingress: [REDACTED]
- Properties: [REDACTED]
- JVM: [REDACTED]
- Log forwarding:
 - Log forwarding is not enabled for this Runtime Fabric. [Learn more](#).
 - Forward application logs to Anypoint Monitoring (requires Titanium)
 - Forward application logs to a external service

Thank you



No RTF - we will manage!

We can use some external tools and libs to manage our logs and send them to a place where we can analyze them.



Looking for ELK

Elastic stack to the rescue!



The Elastic Stack, formerly known as the ELK Stack, is a powerful set of open-source tools designed for a variety of data analytics and visualization tasks.

Comprised of Elasticsearch, Logstash, Kibana, and Beats, this stack offers a comprehensive solution for searching, analyzing, and visualizing large volumes of data in real-time.



Log4j2

A quiet hero of this tale.



Embedded in the mulesoft container is a powerful and flexible logging framework that allows developers to instrument their code to produce detailed logs, which can be useful for debugging, monitoring, and auditing purposes.



What do we need?



- Docker
- Docker compose
- Elastic stack containers - <https://github.com/deviantony/docker-elk>
- MuleSoft apps

Additional components:

- json-logger - <https://github.com/mulesoft-consulting/json-logger> (fork it!)



Let's do it!

Setting up the elastic stack

```
~/docker-elk
> docker compose up -d
[+] Running 7/7
  # Network docker-elk_elk           Created      0.0s
  # Volume "docker-elk_setup"        Created      0.0s
  # Volume "docker-elk_elasticsearch" Created      0.0s
  # Container docker-elk-elasticsearch-1 Started     0.9s
  # Container docker-elk-setup-1      Started     0.7s
  # Container docker-elk-kibana-1      Started     2.0s
  # Container docker-elk-logstash-1    Started     2.2s

~/docker-elk
> # Open your web browser at http://localhost:5601 
```

```
~/docker-elk
> curl http://localhost:9200 -u elastic:changeme
```

Track the ELK



All we need now is to forward some logs

Stash it!

Logstash serves as the data processing pipeline component of the Elastic Stack. It ingests data from multiple sources, transforms it according to user-defined rules, and then sends it to Elasticsearch for indexing and storage. Logstash supports a wide range of input sources, including logs, metrics, and event streams, making it a versatile tool for data ingestion and enrichment.



Logging

What to do, what to do...

- Use the proper level for your logging.
- Use correlationId for tracking your message.
- Payload logging - think about GDPR.



Logging

What to do, what to do...

When thinking about the logs just remember these principles:

- Do I need it?
- Will I understand what I'm trying to tell the user?
- Will the user understand what I'm trying to tell him?
- Do I really need it?
- Is it in the proper level?
- What is the purpose of this message?
- Do i really, really need it?



Over logging

Some consequences

- **Performance Impact:** Excessive logging can impact application performance, especially in high-throughput systems, due to the overhead of generating and writing log messages.
- **Increased Log File Size:** Over logging leads to larger log files, making it more difficult to find relevant information when troubleshooting issues.
- **Difficulty in Log Analysis:** With an abundance of log messages, it becomes challenging to identify important events and diagnose issues effectively.



Logstash plugins

You never know what you will get

<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>

- [beats](#)
- [dead_letter_queue](#)
- [http](#)
- [jdbc](#)
- [jms](#)
- [jmx](#)
- [kafka](#)
- [kinesis](#)
- [log4j](#)
- [rabbitmq](#)
- [salesforce](#)
- [tcp](#)



pom.xml configuration



```
<plugin>
    <groupId>org.mule.tools.maven</groupId>
    <artifactId>mule-maven-plugin</artifactId>
    <version>${mule.maven.plugin.version}</version>
    <extensions>true</extensions>
    <configuration>
        <cloudHubDeployment>
            <properties>
                <elkstack.host>${elk.host}</elkstack.host>
                <elkstack.port>${elk.port}</elkstack.port>
            </properties>
        </cloudHubDeployment>
        (...)
```

Lets format some logs

Log format configuration

```
<Appenders>
    (...)

        <Socket name="ELK_STACK" host="${sys:elkstack.host}" port="${sys:elkstack.port}" protocol="TCP"
    >

        <JsonLayout complete="false" compact="true" eventEOL="true" properties="true"
objectMessageAsJsonObject="true" >
            <KeyValuePair key="appName" value="mule-meetup-logging-app" />
        </JsonLayout>
    </Socket>
    (...)

</Appenders>
```



Log example

```
{  
    "instant" : {  
        "epochSecond" : 1713899793,  
        "nanoOfSecond" : 405830300  
    },  
    "thread" : "[MuleRuntime].uber.01: [mule-meetup-logging].mule-meetup-loggingFlow.CPU_LITE @64d075d2",  
    "level" : "INFO",  
    "loggerName" : "org.mule.runtime.core.internal.processor.LoggerMessageProcessor",  
    "message" : "{user=James Bond, password=*****}",  
    "endOfBatch" : true,  
    "loggerFqcn" : "org.apache.logging.slf4j.Log4jLogger",  
    "contextMap" : {  
        "correlationId" : "fbf3b860-01a5-11ef-a8ae-bcf171a2e298",  
        "processorPath" : "mule-meetup-loggingFlow/processors/1"  
    },  
    "threadId" : 42,  
    "threadPriority" : 5,  
    "appName" : "mule-meetup-logging"  
}
```



logs+ Discover

Filter your data using KQL syntax

Available fields

- @timestamp
- @version
- action
- app
- appName
- contextMap.correlationId
- contextMap.processorPath
- correlationId
- data_stream.dataset
- data_stream.namespace
- data_stream.type
- endOfBatch
- env
- instant.epochSecond
- instant.nanoOfSecond
- level
- loggerFqn
- loggerMessage
- loggerName
- message
- objectID
- priority
- systemTags
- tags
- thread

[Add a field](#)

0 Auto interval No breakdown



Documents (1,308) Field statistics

Get the best look at your search results

Add relevant fields, reorder and sort columns, resize rows, and more in the document table.

[Take the tour](#) [Dismiss](#)

@timestamp	Document
Apr 24, 2024 @ 08:03:44.642	timestamp Apr 24, 2024 @ 08:03:44.642 @version 1 appName mule-meetup-logging contextMap.correlationId 650e87f0-0200-11ef-83ba-bcf171a2e298 contextMap.processorPath mule-meetup-loggingFlow/processors/2 data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,624 instant.nanoOfSecond 640,210,000 level ERROR loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName org.mule.runtime.core.internal.exception.OnErrorHandler message ***** Message : What is it boy? Element : mule-meetup-logging!low/processors/2 @ mule-meetup-logging:mule-meetup-logging.xml!14 (Raise error) Element DSL : <raise-error doc:name="Raise error" doc:id="57473b4-17f7-413a-*****>
Apr 24, 2024 @ 08:03:43.970	timestamp Apr 24, 2024 @ 08:03:43.970 @version 1 appName mule-meetup-logging contextMap.correlationId 650e87f0-0200-11ef-83ba-bcf171a2e298 contextMap.processorPath mule-meetup-loggingFlow/processors/1 data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,623 instant.nanoOfSecond 968,327,400 level INFO loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName org.mule.runtime.core.internal.processor.LoggerMessageProcessor message {user=lesge, password=***} thread [MuleRuntime].uber.08: [mule-meetup-logging].mule-meetup-loggingFlow.CPU_LITE #2482d485 threadId 48 threadPriority 5 _id 9AC2D0B86MvLc2sRhz _index .ds-logs-generic-default-2024.04.24-000001 _score -
Apr 24, 2024 @ 08:01:57.560	timestamp Apr 24, 2024 @ 08:01:57.560 @version 1 appName mule-meetup-logging contextMap.correlationId 2258b876-0200-11ef-83ba-bcf171a2e298 contextMap.processorPath mule-meetup-loggingFlow/processors/2 data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,517 instant.nanoOfSecond 587,179,300 level ERROR loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName org.mule.runtime.core.internal.exception.OnErrorHandler message ***** Message : What is it boy? Element : mule-meetup-logging!low/processors/2 @ mule-meetup-logging:mule-meetup-logging.xml!14 (Raise error) Element DSL : <raise-error doc:name="Raise error" doc:id="57473b4-17f7-413a-*****>
Apr 24, 2024 @ 08:01:53.096	timestamp Apr 24, 2024 @ 08:01:53.096 @version 1 appName mule-meetup-logging contextMap.correlationId 2258b876-0200-11ef-83ba-bcf171a2e298 contextMap.processorPath mule-meetup-loggingFlow/processors/1 data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,513 instant.nanoOfSecond 94,532,400 level INFO loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName org.mule.runtime.core.internal.processor.LoggerMessageProcessor message {user=lesge, password=***} thread [MuleRuntime].uber.05: [mule-meetup-logging].mule-meetup-loggingFlow.CPU_LITE #2402d085 threadId 45 threadPriority 5 _id 8QCy0oBB0MvLc2sPRKw _index .ds-logs-generic-default-2024.04.24-000001 _score -
Apr 24, 2024 @ 08:01:47.127	timestamp Apr 24, 2024 @ 08:01:47.127 @version 1 appName mule-meetup-logging data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,507 instant.nanoOfSecond 124,210,000 level INFO loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName org.mule.runtime.core.internal.logging.LogUtil message ***** Application: mule-meetup-logging + * OS encoding: UTF-8, Mule encoding: UTF-8 * * * * * thread ArtifactDeployer.start.01 threadId 63 threadPriority 5 _id 8QCy0oBB0MvLc2sRzg _index .ds-logs-generic-default-2024.04.24-000001 _score -
Apr 24, 2024 @ 08:01:47.126	timestamp Apr 24, 2024 @ 08:01:47.126 @version 1 appName mule-meetup-logging data_stream.dataset generic data_stream.namespace default data_stream.type logs endOfBatch true instant.epochSecond 1,713,938,507 instant.nanoOfSecond 124,210,000 level WARN loggerFqn org.apache.logging.slf4j.Log4jLogger loggerName com.mulesoft.agent.configuration.descriptor.YamlMuleAgentDescriptor message Descriptor file C:\AnyPointStudio\plugins\org.mule.tooling.server.4.4.0.ee_7.1.0-282402261838\mule\conf\mule-meetup-logging.yml not found. thread ArtifactDeployer.start.01 threadId 63 threadPriority 5 _id 8QCy0oBB0MvLc2sRzg _index .ds-logs-generic-default-2024.04.24-000001 _score -

Rows per page: 100

< 1 2 3 4 5 >

I have my logs in
configured. What now?



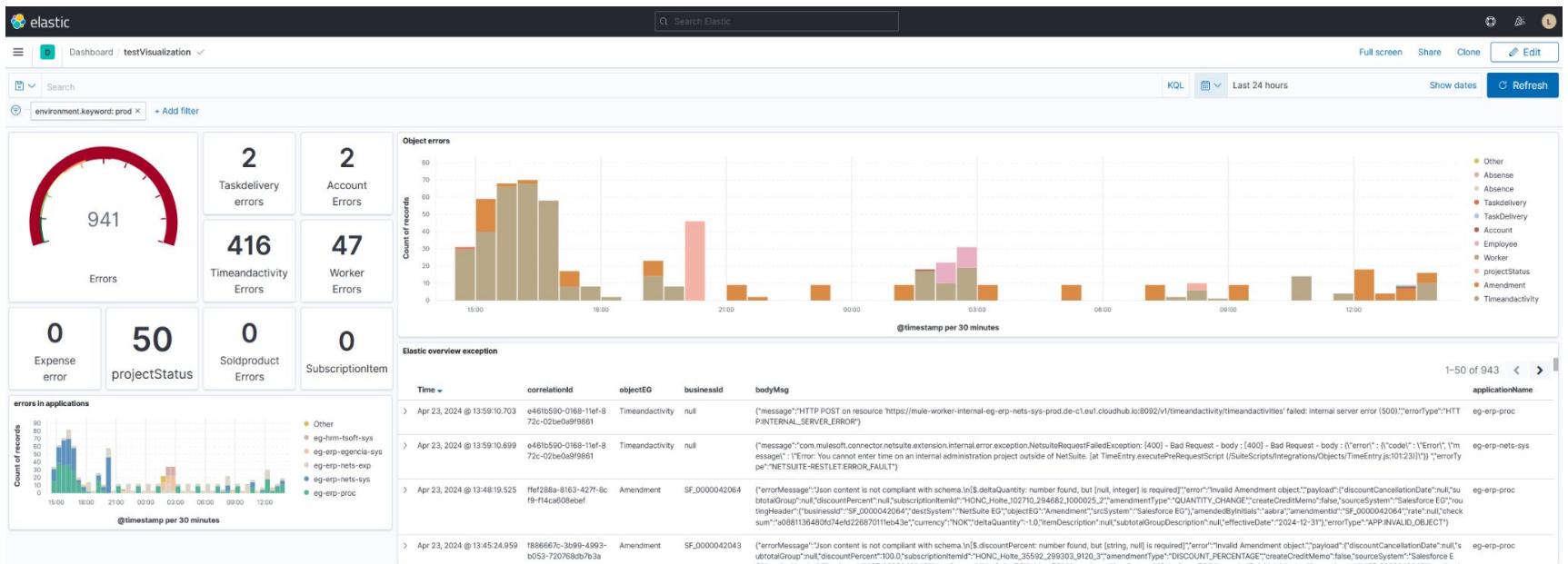
View it!

Kibana to the rescue!

Kibana is the visualization and exploration component of the Elastic Stack. It provides a web-based interface for creating dashboards, charts, and visualizations based on data stored in Elasticsearch. With Kibana, users can interactively explore their data, gain insights through powerful analytics features, and share their findings with others.



Dashboard



Let's add some complexity to our logs

I am GROK!

Grok is a great way to parse unstructured log data into something structured and queryable.

```
filter {  
  
    grok {  
  
        match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request}  
        %{NUMBER:bytes} %{NUMBER:duration}" }  
  
    }  
  
}
```



Log example

```
{  
    "instant" : {  
        "epochSecond" : 1713899793,  
        "nanoOfSecond" : 405830300  
    },  
    "thread" : "[MuleRuntime].uber.01: [mule-meetup-logging].mule-meetup-loggingFlow.CPU_LITE @64d075d2",  
    "level" : "INFO",  
    "loggerName" : "org.mule.runtime.core.internal.processor.LoggerMessageProcessor",  
    "message" : "{user=James Bond, password=*****}",  
    "endOfBatch" : true,  
    "loggerFqcn" : "org.apache.logging.slf4j.Log4jLogger",  
    "contextMap" : {  
        "correlationId" : "fbf3b860-01a5-11ef-a8ae-bcf171a2e298",  
        "processorPath" : "mule-meetup-loggingFlow/processors/1"  
    },  
    "threadId" : 42,  
    "threadPriority" : 5,  
    "appName" : "mule-meetup-logging"  
}
```



Understand and analyze.

I am Grok.

```
filter {  
    date {  
        match => [ "timeMillis", "UNIX_MS" ]  
    }  
    mutate {  
        add_field => {  
            "correlationId" => "%{[contextMap][correlationId]}"  
        }  
    }  
    mutate {  
        remove_field => ["thread", "loggerName", "endOfBatch", "loggerFqcn", "threadId", "threadPriority", "contextMap"]  
    }  
}
```



Easy to understand, easy to analyze

I am Grok?



```
{  
  "@timestamp": [ "2024-04-24T06:28:38.518Z" ],  
  "@version": [ "1" ],  
  "appName": [ "mule-meetup-logging" ],  
  "correlationId": [ "e17c1e60-0203-11ef-ae28-bcf171a2e298" ],  
  "data_stream.dataset": [ "generic" ],  
  "data_stream.namespace": [ "default" ],  
  "data_stream.type": [ "logs" ],  
  "instant.epochSecond": [ 1713940118 ],  
  "instant.nanoOfSecond": [ 515248500 ],  
  "level": [ "INFO" ],  
  "message": [ "{user=James Bond, password=*****}" ],  
  "_id": "BwDKDo8B6MvvLc2svB1n",  
  "_index": ".ds-logs-generic-default-2024.04.24-000001",  
  "_score": null  
}
```



+ Filter your data using KQL syntax

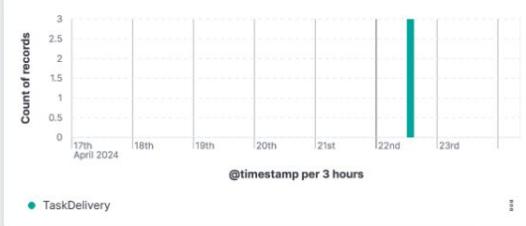
Prepared X

TEST X

Objects in the integration



errors



Object distribution



①

0

API
ERROR

36

Account

0

Amendment

3

TaskDelivery

0

SubscriptionItem

0

Contract

0

ContractLine

salesforce logs

Columns ↓ 1 field sorted

	@timestamp	level	tracePoint	action	objectEG	businessId	correlationId	payload	systemTags	loggerMessage
✓	Apr 24, 2024 ...	INFO	END	UPsert	BillingRateC...	1230.0	aecf5c30-0202-11ef-...	-	NetSuite EG, Salesforce EG	BillingRateCard 1230 send to Salesforce.
✓	Apr 24, 2024 ...	INFO	END	UPsert	BillingRateC...	1226.0	aecf5c30-0202-11ef-...	-	NetSuite EG, Salesforce EG	BillingRateCard 1226 send to Salesforce.
✓	Apr 24, 2024 ...	INFO	END	UPsert	BillingRateC...	1215.0	aecf5c30-0202-11ef-...	-	NetSuite EG, Salesforce EG	BillingRateCard 1215 send to Salesforce.
✓	Apr 24, 2024 ...	INFO	END	UPsert	BillingRateC...	1206.0	aecf5c30-0202-11ef-...	-	NetSuite EG, Salesforce EG	BillingRateCard 1206 send to Salesforce.

Thank you



DataWeave Libraries

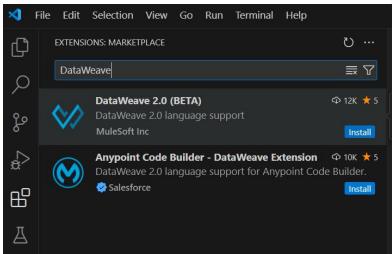


Purpose

- Create functionalities for common use-cases in your integration/project/organization
- Enable organized reusability in different applications
- Version management
- Ensure proper execution by creating tests
- Create documentation to provide explanations and show examples
- Enable discovery through Exchange



Configure



Implement

A screenshot of a code editor displaying a DataWeave script. The script is a complex transformation involving multiple steps of mapping and filtering data from various sources like 'CountryList', 'CountryDetails', 'CountryCodes', and 'CountryTranslations'. It uses functions like 'map', 'filter', and 'join' along with 'let' statements to define variables and process data.

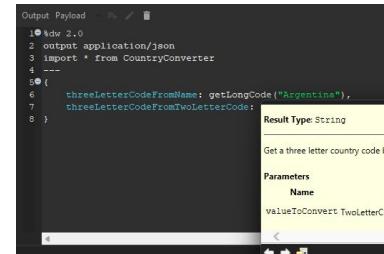
```
%dw 2.0
output application/json
import * from CountryConverter
---
let countryList = [
    {
        "id": "1",
        "name": "Argentina"
    },
    {
        "id": "2",
        "name": "Brazil"
    },
    {
        "id": "3",
        "name": "Canada"
    },
    {
        "id": "4",
        "name": "China"
    },
    {
        "id": "5",
        "name": "France"
    },
    {
        "id": "6",
        "name": "Germany"
    },
    {
        "id": "7",
        "name": "India"
    },
    {
        "id": "8",
        "name": "Italy"
    },
    {
        "id": "9",
        "name": "Japan"
    },
    {
        "id": "10",
        "name": "Mexico"
    },
    {
        "id": "11",
        "name": "Netherlands"
    },
    {
        "id": "12",
        "name": "Norway"
    },
    {
        "id": "13",
        "name": "Russia"
    },
    {
        "id": "14",
        "name": "Spain"
    },
    {
        "id": "15",
        "name": "Sweden"
    },
    {
        "id": "16",
        "name": "United Kingdom"
    },
    {
        "id": "17",
        "name": "United States"
    }
]
let countryDetails = [
    {
        "id": "1",
        "name": "Argentina",
        "code": "AR"
    },
    {
        "id": "2",
        "name": "Brazil",
        "code": "BR"
    },
    {
        "id": "3",
        "name": "Canada",
        "code": "CA"
    },
    {
        "id": "4",
        "name": "China",
        "code": "CN"
    },
    {
        "id": "5",
        "name": "France",
        "code": "FR"
    },
    {
        "id": "6",
        "name": "Germany",
        "code": "DE"
    },
    {
        "id": "7",
        "name": "India",
        "code": "IN"
    },
    {
        "id": "8",
        "name": "Italy",
        "code": "IT"
    },
    {
        "id": "9",
        "name": "Japan",
        "code": "JP"
    },
    {
        "id": "10",
        "name": "Mexico",
        "code": "MX"
    },
    {
        "id": "11",
        "name": "Netherlands",
        "code": "NL"
    },
    {
        "id": "12",
        "name": "Norway",
        "code": "NO"
    },
    {
        "id": "13",
        "name": "Russia",
        "code": "RU"
    },
    {
        "id": "14",
        "name": "Spain",
        "code": "ES"
    },
    {
        "id": "15",
        "name": "Sweden",
        "code": "SE"
    },
    {
        "id": "16",
        "name": "United Kingdom",
        "code": "GB"
    },
    {
        "id": "17",
        "name": "United States",
        "code": "US"
    }
]
let countryCodes = [
    {
        "id": "1",
        "name": "Argentina",
        "code": "AR"
    },
    {
        "id": "2",
        "name": "Brazil",
        "code": "BR"
    },
    {
        "id": "3",
        "name": "Canada",
        "code": "CA"
    },
    {
        "id": "4",
        "name": "China",
        "code": "CN"
    },
    {
        "id": "5",
        "name": "France",
        "code": "FR"
    },
    {
        "id": "6",
        "name": "Germany",
        "code": "DE"
    },
    {
        "id": "7",
        "name": "India",
        "code": "IN"
    },
    {
        "id": "8",
        "name": "Italy",
        "code": "IT"
    },
    {
        "id": "9",
        "name": "Japan",
        "code": "JP"
    },
    {
        "id": "10",
        "name": "Mexico",
        "code": "MX"
    },
    {
        "id": "11",
        "name": "Netherlands",
        "code": "NL"
    },
    {
        "id": "12",
        "name": "Norway",
        "code": "NO"
    },
    {
        "id": "13",
        "name": "Russia",
        "code": "RU"
    },
    {
        "id": "14",
        "name": "Spain",
        "code": "ES"
    },
    {
        "id": "15",
        "name": "Sweden",
        "code": "SE"
    },
    {
        "id": "16",
        "name": "United Kingdom",
        "code": "GB"
    },
    {
        "id": "17",
        "name": "United States",
        "code": "US"
    }
]
let countryTranslations = [
    {
        "id": "1",
        "name": "Argentina"
    },
    {
        "id": "2",
        "name": "Brazil"
    },
    {
        "id": "3",
        "name": "Canada"
    },
    {
        "id": "4",
        "name": "China"
    },
    {
        "id": "5",
        "name": "France"
    },
    {
        "id": "6",
        "name": "Germany"
    },
    {
        "id": "7",
        "name": "India"
    },
    {
        "id": "8",
        "name": "Italy"
    },
    {
        "id": "9",
        "name": "Japan"
    },
    {
        "id": "10",
        "name": "Mexico"
    },
    {
        "id": "11",
        "name": "Netherlands"
    },
    {
        "id": "12",
        "name": "Norway"
    },
    {
        "id": "13",
        "name": "Russia"
    },
    {
        "id": "14",
        "name": "Spain"
    },
    {
        "id": "15",
        "name": "Sweden"
    },
    {
        "id": "16",
        "name": "United Kingdom"
    },
    {
        "id": "17",
        "name": "United States"
    }
]
let countryListWithDetails = countryList.map((country) => {
    let details = countryDetails.find((detail) => detail.id === country.id);
    let codes = countryCodes.find((code) => code.id === country.id);
    let translations = countryTranslations.find((translation) => translation.id === country.id);
    return { ...country, details, codes, translations };
})
let filteredCountryList = countryListWithDetails.filter((country) => country.name === "Argentina");
let result = filteredCountryList[0].details;

```

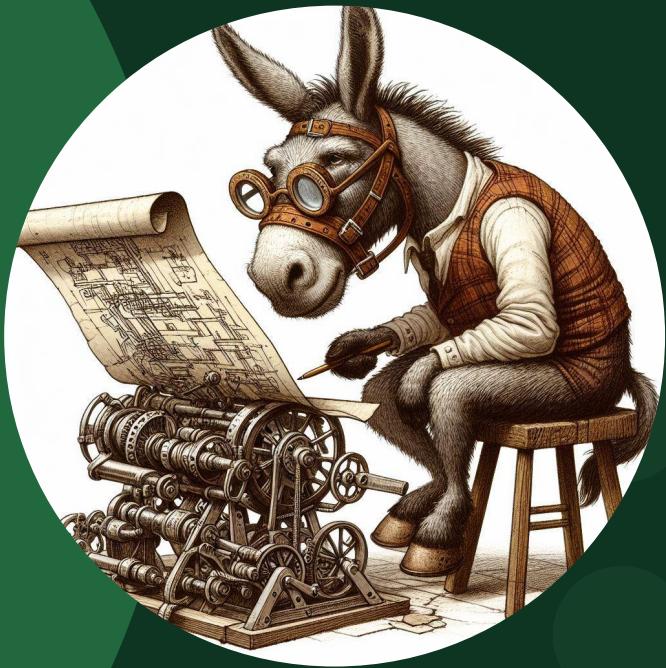
Deploy



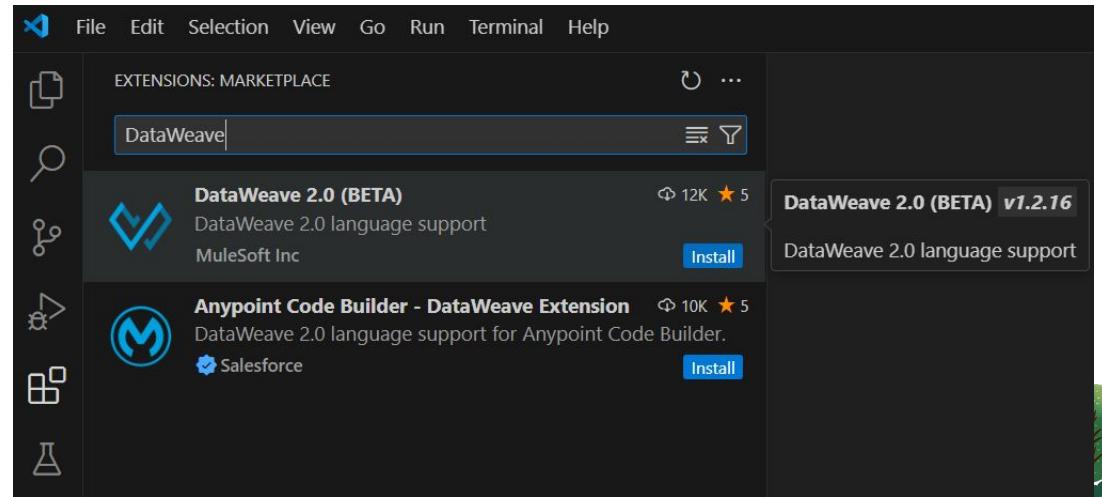
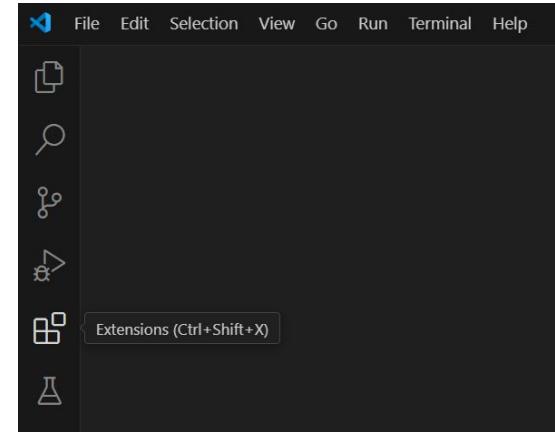
Use



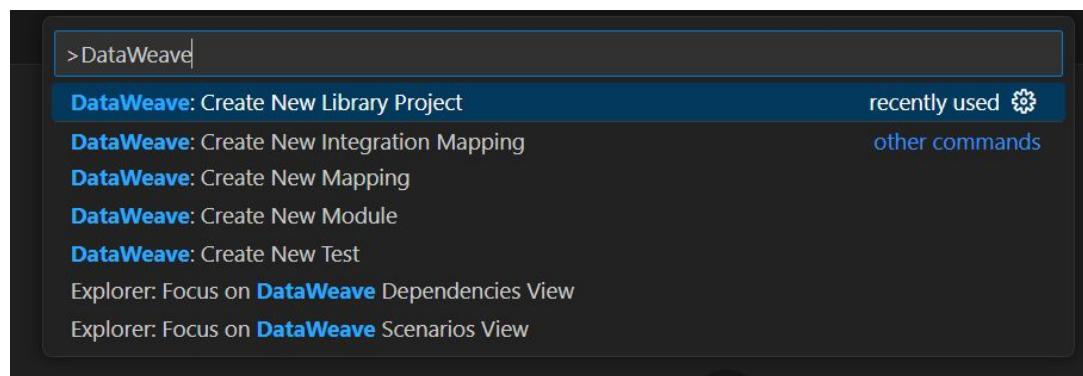
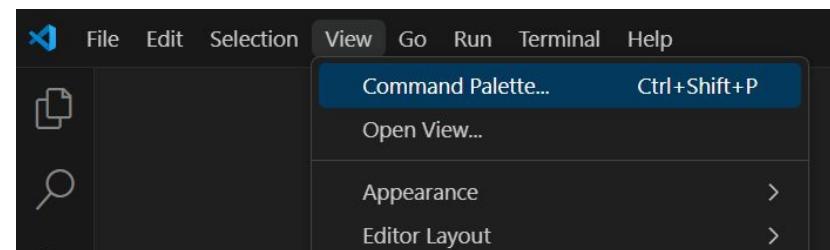
Configure



- Download Visual Studio Code 
- Install the DataWeave extension:
 - Click "Extensions" on the panel on the left
 - Type "DataWeave" into the search bar
 - Click the blue install button next to "DataWeave 2.0 (BETA)"



- Select View -> Command Pallete...
- Type “DataWeave” into the search bar
- Select “DataWeave: Create Library Project”



Create Project (1/5)

Organization ID (Press 'Enter' to confirm or 'Escape' to cancel)

* In order to deploy to Exchange you need to input the Organization ID taken from Anypoint Platform. It can later on be changed in the pom



←

Create Project (2/5)

wro-event-library

Artifact ID (Press 'Enter' to confirm or 'Escape' to cancel)

←

Create Project (3/5)

1.0.0-SNAPSHOT

Version (Press 'Enter' to confirm or 'Escape' to cancel)



← Create Project (4/5)

Event Library

Project name (Press 'Enter' to confirm or 'Escape' to cancel)

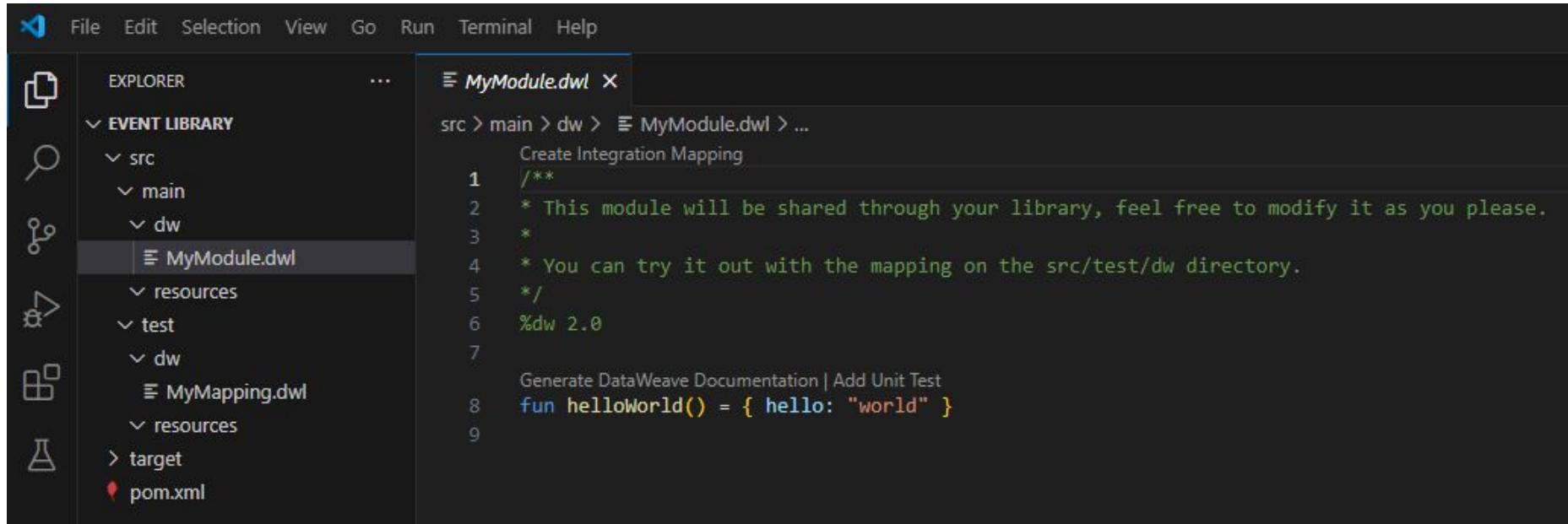
← Choose project path (5/5) 

<PATH>

Press 'Enter' to confirm your input or 'Escape' to cancel

Path to the location in which you want to save your project **without spaces**





The screenshot shows the MuleSoft Anypoint Studio interface. The left sidebar contains icons for File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer pane on the left displays the project structure:

- src
- └ main
 - └ dw
 - └ MyModule.dwl
 - resources
 - test
 - └ dw
 - └ MyMapping.dwl
 - resources
 - target
 - pom.xml

The Editor pane on the right shows the content of the selected file, `MyModule.dwl`:

```
src > main > dw > MyModule.dwl > ...
      Create Integration Mapping
1  /**
2   * This module will be shared through your library, feel free to modify it as you please.
3   *
4   * You can try it out with the mapping on the src/test/dw directory.
5   */
6 %dw 2.0
7
8 fun helloWorld() = { hello: "world" }
9
```

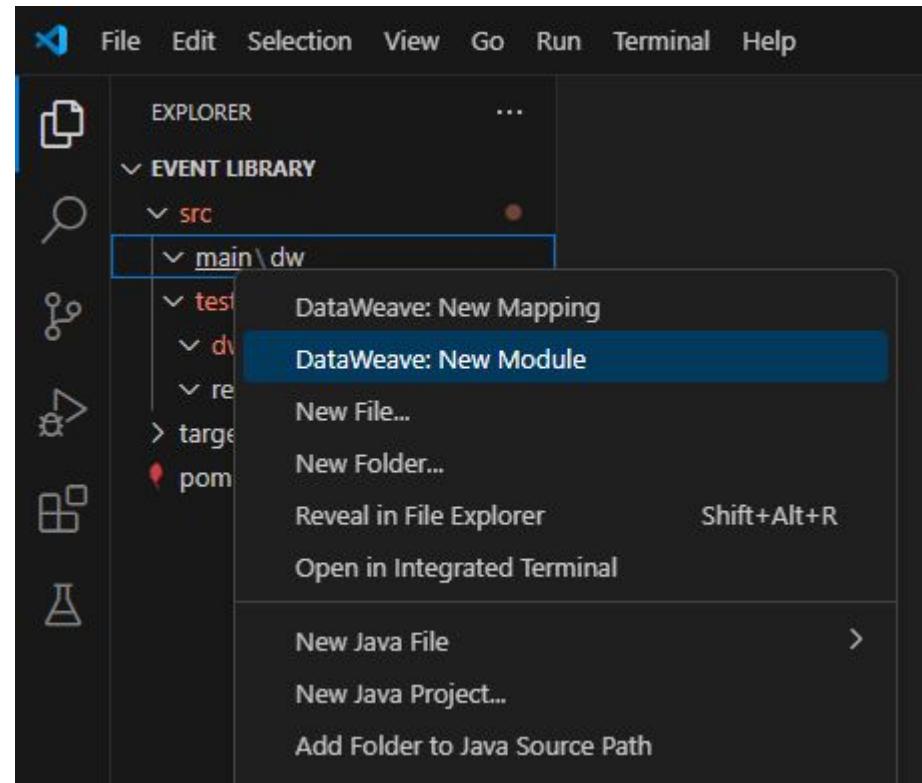
Below the code editor, there are links for "Generate DataWeave Documentation" and "Add Unit Test".



Implement



- Create modules/files in the `src/main/dw` folder. Files created there will be packaged and shared as part of the library
- Use **PascalCase** naming format for the modules/files e.g. `MySuperCoolModule.dwl`



EXPLORER ...

EVENT LIBRARY

src

main\dw

CountryConverter.dwl

test

target

pom.xml

CountryConverter.dwl X

src > main > dw > CountryConverter.dwl

Create Integration Mapping

%dw 2.5

```
<properties>
    <data.weave.version>2.5.0</data.weave.version>
    <data.weave.testing.framework.version>1.2.2</data.weave.testing.framework.version>
    <data.weave.maven.plugin.version>0.3.4</data.weave.maven.plugin.version>
</properties>
```



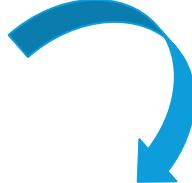
- Use camelCase for function names
- Use PascalCase for type names

```

CountryConverter.dwl | Create Integration Mapping
  1  Schema 2.5
  2  var countryMapping =
  3  [
  4    {
  5      "name": "Afghanistan",
  6      "alpha2": "AF",
  7      "alpha3": "AFG"
  8    },
  9    {
 10      "name": "Albania",
 11      "alpha2": "AL",
 12      "alpha3": "ALB"
 13    },
 14    {
 15      "name": "Algeria",
 16      "alpha2": "DZ",
 17      "alpha3": "DZA"
 18    },
 19    {
 20      "name": "Andorra",
 21      "alpha2": "AD",
 22      "alpha3": "AND"
 23    },
 24    {
 25      "name": "Angola",
 26      "alpha2": "AO",
 27      "alpha3": "AGO"
 28    },
 29    {
 30      "name": "Antigua and Barbuda",
 31      "alpha2": "AG",
 32      "alpha3": "ATG"
 33    },
 34    {
 35      "name": "Argentina",
 36      "alpha2": "AR",
 37      "alpha3": "ARG"
 38  ]
 39 type ThreeLetterCode = "AFG" | "ALB" | "DZA" | "AND" | "AGO" | "ATG" | "ARG"
 40 type TwoLetterCode = "AF" | "AL" | "DZ" | "AD" | "AO" | "AG" | "AR"
 41 type CountryName = "Afghanistan" | "Albania" | "Algeria" | "Andorra" | "Angola" | "Antigua and Barbuda" | "Argentina"
 42
 43 Generate DataWeave Documentation | Add Unit Test
 44 fun getShortCode(valueToConvert: ThreeLetterCode | CountryName): String =
 45 do{
 46   var correctMapping = valueToConvert match {
 47     case longCode if (countryMapping.alpha3 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha3 == valueToConvert))
 48     case name if (countryMapping.name contains valueToConvert) -> (countryMapping filter ((country, index) -> country.name == valueToConvert))
 49   }
 50   correctMapping.alpha2[0]
 51 }
 52
 53 Generate DataWeave Documentation | Add Unit Test
 54 fun getLongCode(valueToConvert: TwoLetterCode | CountryName): String =
 55 do{
 56   var correctMapping = valueToConvert match {
 57     case shortCode if (countryMapping.alpha2 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha2 == valueToConvert))
 58     case name if (countryMapping.name contains valueToConvert) -> (countryMapping filter ((country, index) -> country.name == valueToConvert))
 59   }
 60   correctMapping.alpha3[0]
 61 }
 62
 63 Generate DataWeave Documentation | Add Unit Test
 64 fun getName(valueToConvert: TwoLetterCode | ThreeLetterCode): String =
 65 do{
 66   var correctMapping = valueToConvert match {
 67     case shortCode if (countryMapping.alpha2 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha2 == valueToConvert))
 68     case longCode if (countryMapping.alpha3 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha3 == valueToConvert))
 69   }
 70   correctMapping.name[0]
 71 }

```



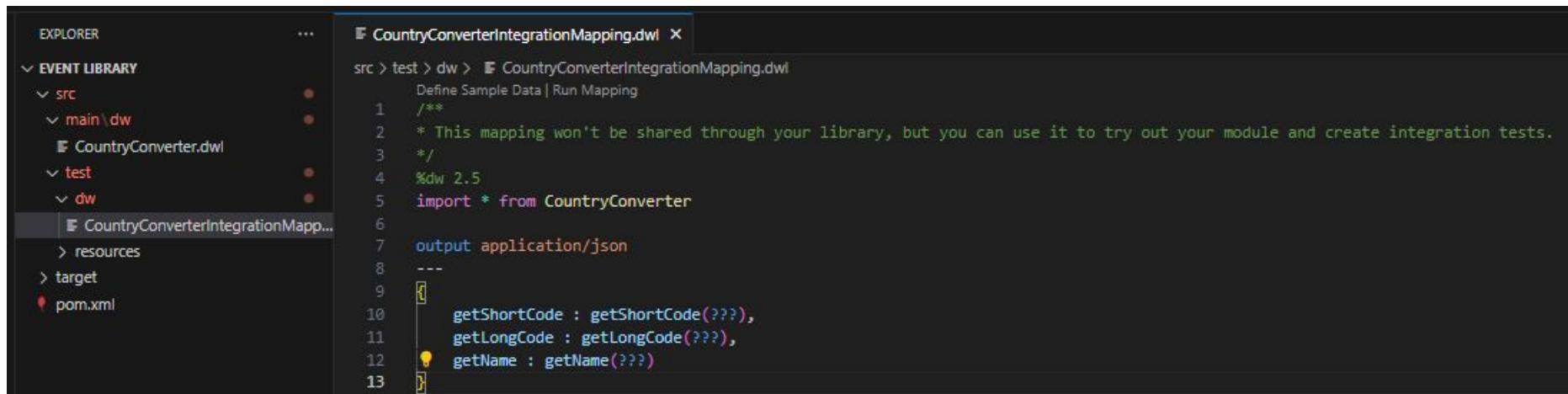


CountryConverter.dwl X

src > main > dw > CountryConverter.dwl > ...

Create Integration Mapping

- Use mapping to try out your code



EXPLORER

EVENT LIBRARY

- src
 - main\dw
 - CountryConverter.dwl
 - test
 - dw
 - CountryConverterIntegrationMapping.dwl
- resources
- target
- pom.xml

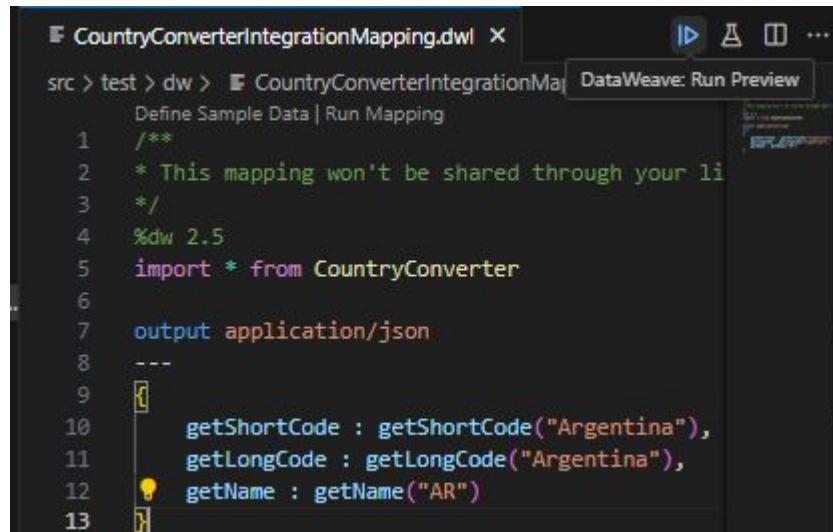
CountryConverterIntegrationMapping.dwl X

src > test > dw > CountryConverterIntegrationMapping.dwl

Define Sample Data | Run Mapping

```
1 /**
2 * This mapping won't be shared through your library, but you can use it to try out your module and create integration tests.
3 */
4 %dw 2.5
5 import * from CountryConverter
6
7 output application/json
8 ---
9 [
10     getCode : getShortCode(???),
11     getName : getName(??),
12     getLongCode : getLongCode(??),
13 ]
```

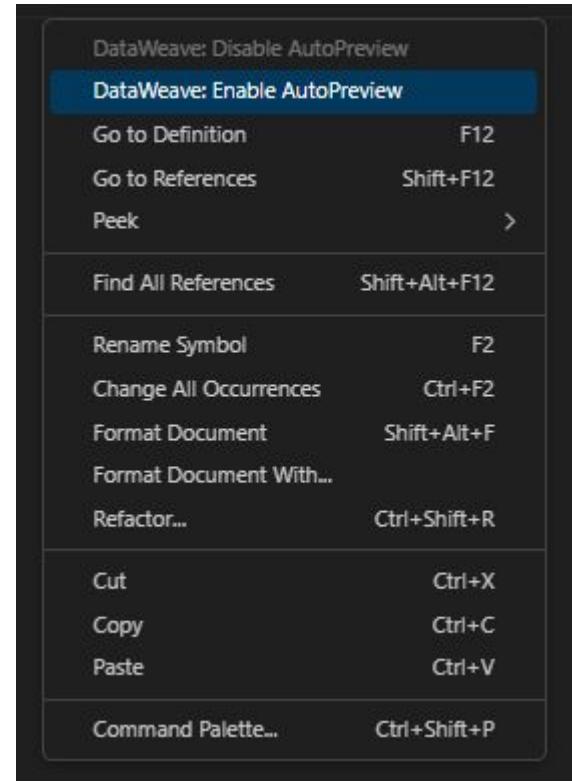
- Using “Run Preview” will execute the current code state
- Enabling AutoPreview will continuously evaluate the script



```

CountryConverterIntegrationMapping.dwl X
src > test > dw > CountryConverterIntegrationMa DataWeave: Run Preview
Define Sample Data | Run Mapping
1 /**
2  * This mapping won't be shared through your li
3 */
4 %dw 2.5
5 import * from CountryConverter
6
7 output application/json
8
9 [
10     getShortCode : getShortCode("Argentina"),
11     getLongCode : getLongCode("Argentina"),
12     getName : getName("AR")
13 ]

```



CountryConverterIntegrationMapping.dwl X

src > test > dw > CountryConverterIntegrationMapping.dwl

Define Sample Data | Run Mapping

```
1 /**
2 * This mapping won't be shared through your library
3 */
4 %dw 2.5
5 import * from CountryConverter
6
7 output application/json
8 ---
9 [
10     {
11         getShortCode : getShortCode("Argentina"),
12         getName : getName("AR")
13     }
]
```

Preview Output X

Preview Output > ...

Preview on: CountryConverterIntegrationMapping.dwl

```
1 {
2     "getShortCode": "AR",
3     "getLongCode": "ARG",
4     "getName": "Argentina"
5 }
```



Add Unit Test

83

```
fun getShortCode(valueToConvert:
```



MuleSoft[®]
from Salesforce

The screenshot shows the MuleSoft Studio interface. On the left, the Explorer view displays the project structure under the EVENT LIBRARY node:

- src
- main\dw
- CountryConverter.dwl
- test
- dw
- CountryConverterIntegrationMapp...
- CountryConverterTest.dwl

The "CountryConverterTest.dwl" file is selected in the Explorer view.

On the right, there are two code editors:

- CountryConverter.dwl
- CountryConverterTest.dwl

The "CountryConverterTest.dwl" editor contains the following DW script:

```
1  %dw 2.0
2  import * from dw::test::Tests
3  import * from dw::test::Asserts
4
5  import * from CountryConverter
6  ---
7  <?xml version="1.0" encoding="UTF-8"?>
8  <testsuite name="CountryConverterTest" tests="1" timestamp="2023-09-18T14:45:23Z" id="1">
9    <!-- Integration test for getShortCode -->
10   <!-- Test for getShortCode -->
11   <!-- Test for getShortCode -->
12   <!-- Test for getShortCode -->
13   <!-- Test for getShortCode -->
14   <!-- Test for getShortCode -->
```



- Use the Asserts modules functions to create test cases validating the behaviour of created functions

```
CountryConverterTest.dwl X
src > test > dw > CountryConverterTest.dwl > ...
1 %dw 2.0
2 import * from dw::test::Tests
3 import * from dw::test::Asserts
4 import * from CountryConverter
5
6 var record = countryMapping[0]
7 ---
8 "CountryConverter" describedBy [
9   "getShortCode" describedBy [
10     "Convert three letter code to two letter code" in do {
11       getShortCode(record.alpha3) must equalTo(record.alpha2)
12     },
13     "Convert name to two letter code" in do {
14       getShortCode(record.name) must equalTo(record.alpha2)
15     }
16   ],
17   "getLongCode" describedBy [
18     "Convert two letter code to three letter code" in do {
19       getLongCode(record.alpha2) must equalTo(record.alpha3)
20     },
21     "Convert name to two letter code" in do {
22       getLongCode(record.name) must equalTo(record.alpha3)
23     }
24   ],
25   "getName" describedBy [
26     "Convert three letter code to country name" in do {
27       getName(record.alpha3) must equalTo(record.name)
28     },
29     "Convert two letter code to country name" in do {
30       getName(record.alpha2) must equalTo(record.name)
31     }
32   ],
33 ]]
```





OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS

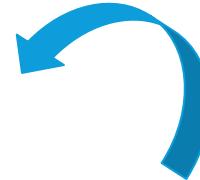
```
{ "event": "testStdOut", "name": "console", "message": "CountryConverterTest"}  
{ "event": "testSuiteStarted", "name": "CountryConverter", "captureStandardOutput": "true", "locationHint": "CountryConverterTest", "nodeId": "0", "parentNodeId": "-1", "location": "{}"}  
{ "event": "testStdOut", "name": "console", "message": "\tCountryConverter"}  
{ "event": "testSuiteStarted", "name": "getShortCode", "captureStandardOutput": "true", "locationHint": "CountryConverterTest", "nodeId": "1", "parentNodeId": "0", "location": "{}"}  
{ "event": "testStdOut", "name": "console", "message": "\t\tgetShortCode"}  
{ "event": "testStarted", "name": "Convert three letter code to two letter code", "locationHint": "CountryConverterTest", "nodeId": "2", "parentNodeId": "1", "location": "{}"}  
{ "event": "testFinished", "name": "Convert three letter code to two letter code", "duration": "13.0", "locationHint": "CountryConverterTest", "nodeId": "2", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert three letter code to two letter code 13.0(ms) ?"}  
{ "event": "testStarted", "name": "Convert name to two letter code", "locationHint": "CountryConverterTest", "nodeId": "3", "parentNodeId": "1", "location": "{}"}  
{ "event": "testFinished", "name": "Convert name to two letter code", "duration": "2.0", "locationHint": "CountryConverterTest", "nodeId": "3", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert name to two letter code 2.0(ms) ?"}  
{ "event": "testSuiteFinished", "name": "getShortCode", "nodeId": "1", "duration": "51.0"}  
{ "event": "testSuiteStarted", "name": "getLongCode", "captureStandardOutput": "true", "locationHint": "CountryConverterTest", "nodeId": "4", "parentNodeId": "0", "location": "{}"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\tgetLongCode"}  
{ "event": "testStarted", "name": "Convert two letter code to three letter code", "locationHint": "CountryConverterTest", "nodeId": "5", "parentNodeId": "4", "location": "{}"}  
{ "event": "testFinished", "name": "Convert two letter code to three letter code", "duration": "2.0", "locationHint": "CountryConverterTest", "nodeId": "5", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert two letter code to three letter code 2.0(ms) ?"}  
{ "event": "testStarted", "name": "Convert name to two letter code", "locationHint": "CountryConverterTest", "nodeId": "6", "parentNodeId": "4", "location": "{}"}  
{ "event": "testFinished", "name": "Convert name to two letter code", "duration": "1.0", "locationHint": "CountryConverterTest", "nodeId": "6", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert name to two letter code 1.0(ms) ?"}  
{ "event": "testSuiteFinished", "name": "getLongCode", "nodeId": "4", "duration": "10.0"}  
{ "event": "testSuiteStarted", "name": "getName", "captureStandardOutput": "true", "locationHint": "CountryConverterTest", "nodeId": "7", "parentNodeId": "0", "location": "{}"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\tgetName"}  
{ "event": "testStarted", "name": "Convert three letter code to country name", "locationHint": "CountryConverterTest", "nodeId": "8", "parentNodeId": "7", "location": "{}"}  
{ "event": "testFinished", "name": "Convert three letter code to country name", "duration": "2.0", "locationHint": "CountryConverterTest", "nodeId": "8", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert three letter code to country name 2.0(ms) ?"}  
{ "event": "testStarted", "name": "Convert two letter code to country name", "locationHint": "CountryConverterTest", "nodeId": "9", "parentNodeId": "7", "location": "{}"}  
{ "event": "testFinished", "name": "Convert two letter code to country name", "duration": "1.0", "locationHint": "CountryConverterTest", "nodeId": "9", "status": "OK"}  
{ "event": "testStdOut", "name": "console", "message": "\t\t\t\t\tConvert two letter code to country name 1.0(ms) ?"}  
{ "event": "testSuiteFinished", "name": "getName", "nodeId": "7", "duration": "10.0"}  
{ "event": "testSuiteFinished", "name": "CountryConverter", "nodeId": "0", "duration": "92.0"}  
  
All test passed (ignored: 0 total: 6)
```



```
{ "event" :"testStarted", "name":"Convert two letter code to country name", "locationHint":"CountryConverterTest", "nodeId": "9", "parentNodeId": "7", "location": "{}"}  
{ "event" :"testFailed", "name":"Convert two letter code to country name", "message": "Expecting `(root)` to be `\\\"Albania\\\"` but was `\\\"Afghanistan\\\"`", "duration": "2.0", "locationHint": "CountryConverterTest", "nodeId": "9", "status": "FAIL"}  
{ "event" :"testStdOut", "name": "console", "message": "\\t\\t\\t\\tConvert two letter code to country name 2.0(ms) ?"}  
{ "event" :"testStdOut", "name": "console", "message": "\\t\\t\\t\\t\\tExpecting `(root)` to be `\\\"Albania\\\"` but was `\\\"Afghanistan\\\"`"}  
{ "event" :"testSuiteFinished", "name": "getName", "nodeId": "7", "duration": "13.0"}  
{ "event" :"testSuiteFinished", "name": "CountryConverter", "nodeId": "0", "duration": "90.0"}  
  
Summary ignored: 0 failures: 1 errors: 0 total: 6
```



```
43 /**
44 * Describes the `getShortCode` function purpose.
45 *
46 * === Parameters
47 *
48 * [%header, cols="1,1,3"]
49 * | ===
50 * | Name | Type | Description
51 * | `valueToConvert` | ThreeLetterCode &#124; CountryName |
52 * | ===
53 *
54 * === Example
55 *
56 * This example shows how the `getShortCode` function behaves under different inputs.
57 *
58 * ===== Source
59 *
60 * [source,DataWeave,linenums]
61 * -----
62 * %dw 2.0
63 * output application/json
64 * ---
65 *
66 *
67 * -----
68 *
69 * ===== Output
70 *
71 * [source,Json,linenums]
72 * -----
73 *
74 * -----
75 *
76 */
77 Add Unit Test
77 fun getShortCode(valueToConvert: ThreeLetterCode | CountryName): String =
```



Generate DataWeave Documentation | Add Unit Test
fun getShortCode(valueToConvert: ThreeLetterCode | CountryName): String =

- Generate documentation to describe functions, parameters, provide examples of use



```

//TYPES
/**
 * Set of allowed values for three letter country codes
 */
type ThreeLetterCode = "AFG" | "ALB" | "DZA" | "AND" | "AGO" | "ATG" | "ARG"

//TYPES
/**
 * Set of allowed values for two letter country codes
 */
type TwoLetterCode = "AF" | "AL" | "DZ" | "AD" | "AO" | "AG" | "AR"

//TYPES
/**
 * Set of allowed values for country names
 */
type CountryName = "Afghanistan" | "Albania" | "Algeria" | "Andorra" | "Angola" | "Antigua and Barbuda" | "Argentina"

/** 
 * Get a two letter country code based on the country name or it's three letter country code.
 */
<%>
<%== Parameters
<%
<% [ %header, cols="1,1,3" ]
<% | ===
<% | Name | Type | Description
<% | `valueToConvert` | ThreeLetterCode #124; CountryName | ThreeLetterCode: ["AFG", "ALB", "DZA", "AND", "AGO", "ATG", "ARG"] <br/>CountryName: ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina"]
<% | ===
<%>
<%== Example
<%
<% This example shows how the `getShortCode` function behaves under different inputs.
<%
<%==== Source
<%
<% [source,DataWeave,linenums]
<% ----
<% %dw 2.0
<% output application/json
<% import getShortCode from CountryConverter
<% ---
<%{
<%   twoLetterCodeFromName: getShortCode("Argentina"),
<%   twoLetterCodeFromThreeLetterCode: getShortCode("ARG")
<%}
<% ----
<%==== Output
<%
<% [source,Json,linenums]
<% ----
<%{
<%   twoLetterCodeFromName: "AR",
<%   twoLetterCodeFromThreeLetterCode: "AR"
<%}
<% ----
<%
<%/
Add Unit Test
fun getShortCode(valueToConvert: ThreeLetterCode | CountryName): String =
  do{
    var correctMapping = valueToConvert match {
      case longCode if (countryMapping.alpha3 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha3 == valueToConvert))
      case name if (countryMapping.name contains valueToConvert) -> (countryMapping filter ((country, index) -> country.name == valueToConvert))
    }
  ...
  correctMapping.alpha2[0]
}

```



- You can get a preview of the documentation by hovering over a function/type/variable name



```

/*
 * Type: (valueToConvert: ThreeLetterCode | CountryName) -> String
 * Get a two letter country code based on the country name or it's three letter country code.
 */
Parameters
Name          Type          Description
valueToConvert ThreeLetterCode | CountryName ThreeLetterCode: ["AFG", "ALB", "DZA", "AND", "AGO", "ATG", "ARG"] CountryName: ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina"]
Example
This example shows how the getShortCode function behaves under different inputs.
Source
%dw 2.0
output application/json
import getShortCode from CountryConverter
---
{
    twoLetterCodeFromName: getShortCode("Argentina"),
    twoLetterCodeFromThreeLetterCode: getShortCode("ARG")
}
Add U
fun getShortCode(valueToConvert: ThreeLetterCode | CountryName): String =
do{
    var correctMapping = valueToConvert match {
        case longCode if (countryMapping.alpha3 contains valueToConvert) -> (countryMapping filter ((country, index) -> country.alpha3 == valueToConvert))
        case name if (countryMapping.name contains valueToConvert) -> (countryMapping filter ((country, index) -> country.name == valueToConvert))
    }
    correctMapping.alpha2[0]
}

```

- By packaging the library you can check the generated Exchange pages in
..../target/data-weave-docs/exchange_markdown



File Edit Selection View Go Run Terminal Help

Event Library

EXPLORER

EVENT LIBRARY

- > .vscode
- < src
 - < main\dw
 - CountryConverter.dwl
 - < test
 - < dw
 - CountryConverterIntegrationMap...
 - CountryConverterTest.dwl
 - > resources
- < target
- > classes
- < data-weave-docs
 - < exchange_markdown
 - CountryConverter.md
 - home.md
- > exchange_markdown_asciidocs
- > data-weave-test-reports
- > maven-archiver
- > test-classes
- wro-event-library-1.0.0-dw-library.jar
- pom.xml

CountryConverter.md

```

1 ##### _CountryConverter_
2
3
4 This module contains logic for converting country representation values
5
6 # Index
7
8 ### Functions
9 | Name | Description|
10|-----|-----|
11|[getLongCode](#getlongcode-index ) | Get a three letter country code based on the country name or it's two letter country code.|
12|[getName](#getname-index ) | Get the country name based on it's two or three letter country code.|
13|[getShortCode](#getshortcode-index ) | Get a two letter country code based on the country name or it's three letter country code.|
14
15 ### Variables
16 | Name | Description|
17|-----|-----|
18|[countryMapping](#countrymapping-index ) | Constant for country definitions|
19
20
21
22
23 ### Types
24 | Name | Description|
25|-----|-----|
26|[CountryName](#countryname-index ) | Set of allowed values for country names|
27|[ThreeLetterCode](#threelettercode-index ) | Set of allowed values for three letter country codes|
28|[TwoLetterCode](#twolettercode-index ) | Set of allowed values for two letter country codes|
29
30

```

Deploy



- In Anypoint Platform, go to to Access Management -> Business Groups -> Select the intended Business Group -> Settings tab -> copy the **Business Group ID**
- Go to the pom.xml file of the DataWeave library and change the groupId to the value from Anypoint



pom.xml x

```
 pom.xml
  pom.xml
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3   |   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   |   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
5
6   <modelVersion>4.0.0</modelVersion>
7   <!-- Set your ORGANIZATION_ID in the groupId section to publish your DataWeave library to Exchange -->
8   <!-- You can find more reference at https://docs.mulesoft.com/exchange/to-publish-assets-maven#publish-an-asset-to-exchange-using-maven -->
9   <!-- <groupId>ORGANIZATION_ID</groupId> -->
10  <!-- NOTE: Remember to add your Anypoint Platform credentials in ~/.m2/settings.xml file -->
11  <groupId>wro.meetup.company</groupId>
12  <artifactId>wro-event-library</artifactId>
13  <version>1.0.0-SNAPSHOT</version>
14  <packaging>dw-library</packaging>
15  <name>Event Library</name>
```

- Scroll to the bottom of the pom.xml file, and uncomment the
 - distributionManagement tag
 - repository for exchange

```
116
117      <!-- Add Exchange repository to publish your DataWeave library to Exchange -->
118      <!-- You can find more reference at https://docs.mulesoft.com/exchange/to-publish-assets-maven#publish-an-asset-to-exchange-using-maven -->
119      <!--
120      <distributionManagement>
121          <repository>
122              <id>exchange</id>
123              <name>Exchange Repository</name>
124              <url>https://maven.anypoint.mulesoft.com/api/v3/organizations/ORGANIZATION\_ID/maven</url>
125              <layout>default</layout>
126          </repository>
127      </distributionManagement>
128      -->
129      <repositories>
130          <!-- Add Exchange repository to consume DataWeave library from Exchange -->
131          <!-- You can find more reference at https://docs.mulesoft.com/exchange/to-publish-assets-maven#consume-an-exchange-asset-with-maven -->
132          <!--
133          <repository>
134              <id>exchange</id>
135              <name>Exchange Repository</name>
136              <url>https://maven.anypoint.mulesoft.com/api/v3/organizations/ORGANIZATION\_ID/maven</url>
137              <layout>default</layout>
138          </repository>
139          -->
```

- Change the **ORGANIZATION_ID** in both URL's to the **Business Group ID** from Anypoint Platform



```
116      <!-- Add Exchange repository to publish your DataWeave library to Exchange -->
117      <!-- You can find more reference at https://docs.mulesoft.com/exchange/to-publish-assets-maven#publish-an-asset-to-exchange-using-maven -->
118
119      <distributionManagement>
120          <repository>
121              <id>exchange</id>
122              <name>Exchange Repository</name>
123              <url>https://maven.anypoint.mulesoft.com/api/v3/organizations/ORGANIZATION\_ID/maven</url>
124              <layout>default</layout>
125          </repository>
126      </distributionManagement>
127
128
129      <repositories>
130          <!-- Add Exchange repository to consume DataWeave library from Exchange -->
131          <!-- You can find more reference at https://docs.mulesoft.com/exchange/to-publish-assets-maven#consume-an-exchange-asset-with-maven -->
132          <repository>
133              <id>exchange</id>
134              <name>Exchange Repository</name>
135              <url>https://maven.anypoint.mulesoft.com/api/v3/organizations/ORGANIZATION\_ID/maven</url>
136              <layout>default</layout>
137          </repository>
```



- Configure your maven settings.xml file with credentials for Anypoint Platform.
- The id of the distributionManagement, repository from pom.xml should correspond to the id of the server in settings.xml
- A good way to organise the credentials is to create a **Connected App**, for which the server credentials in settings.xml need to follow this format:

```
<server>
    <id>exchange</id>
    <username>~~~Client~~~</username>
    <password>clientId~?~clientSecret</password>
</server>
```



- Use the ***mvn clean deploy*** command to publish the library to Exchange
- The created test will be ran and validated

```
[INFO] --- data-weave-maven-plugin:0.3.4:test (default-test) @ wro-event-library ---
CountryConverterTest
    CountryConverter
        getCode
            Convert three letter code to two letter code 16.0(ms) ?
            Convert name to two letter code 2.0(ms) ?

        getLongCode
            Convert two letter code to three letter code 2.0(ms) ?
            Convert name to two letter code 2.0(ms) ?

        getName
            Convert three letter code to country name 2.0(ms) ?
            Convert two letter code to country name 1.0(ms) ?

All test passed (ignored: 0 total: 6)
Check out the test report at C:\Users\jakub\OneDrive\Dokumenty\Meetups\meetup-1\dw-library\Event Library\target\data-weave-test-reports\data-weave-testing-framework-summary.html
```

```
[INFO] -----
[INFO] Publication status: completed
[INFO] -----
[INFO] Steps:
[INFO]   - Description: Publishing asset
[INFO]   - Status: completed
[INFO] -----
[INFO] -----
[INFO] Your asset has been successfully published to Exchange.
```



- Go to your Exchange
- The library will be available under the specified **Name**
- The Exchange documentation will be generated with elements that were not specified as empty/pre-populated



← Go to assets list

Event Library

DataWeave Library · Demo Company · Updated 35 seconds ago

Jakub Cieplucha published 51 seconds ago

[Edit documentation](#) | [Share](#) | [Download](#) | [Dependency Snippets](#) | [⋮](#)

Manage versions | 1.0.x | Latest: 1.0.0 · [C Stable](#) | [↗](#)

+ Add tags

PAGES	DataWeave Version	
Home	This library requires DataWeave version 2.5 or higher.	
CountryConverter		

Modules

Name	Description
CountryConverter	This module contains logic for converting country representation values

Reviews

JC Be the first to review Event Library 1.0.x



PAGES

[Home](#)[CountryConverter](#)[!\[\]\(3e5c8028dfbd33327033568c6df503f9_img.jpg\) Edit documentation](#)

CountryConverter

CountryConverter

This module contains logic for converting country representation values

Index

Functions

Name	Description
getLongCode	Get a three letter country code based on the country name or it's two letter country code.
getName	Get the country name based on it's two or three letter country code.
getShortCode	Get a two letter country code based on the country name or it's three letter country code.

Variables

Name	Description
countryMapping	Constant for country definitions

Types

Name	Description
CountryName	Set of allowed values for country names
ThreeLetterCode	Set of allowed values for three letter country codes
TwoLetterCode	Set of allowed values for two letter country codes



Functions

getLongCode ↑↑

getLongCode(valueToConvert: TwoLetterCode | CountryName): String

Get a three letter country code based on the country name or it's two letter country code.

Parameters

Name	Type	Description
valueToConvert	TwoLetterCode CountryName	TwoLetterCode: ["AF", "AL", "DZ", "AD", "AO", "AG", "AR"] CountryName: ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina"]

Example

This example shows how the `getLongCode` function behaves under different inputs.

Source

```
%dw 2.0
output application/json
import getLongCode from CountryConverter
---
{
  threeLetterCodeFromName: getLongCode("Argentina"),
  threeLetterCodeFromTwoLetterCode: getLongCode("AR")
}
```

Output

```
{
  threeLetterCodeFromName: "ARG",
  threeLetterCodeFromTwoLetterCode: "ARG"
}
```



getName ↑↑

getName(*valueToConvert: TwoLetterCode | ThreeLetterCode*): String

Get the country name based on it's two or three letter country code.

Parameters

Name	Type	Description
valueToConvert	TwoLetterCode ThreeLetterCode	TwoLetterCode: ["AF", "AL", "DZ", "AD", "AO", "AG", "AR"] ThreeLetterCode: ["AFG", "ALB", "DZA", "AND", "AGO", "ATG", "ARG"]

Example

This example shows how the `getName` function behaves under different inputs.

Source

```
%dw 2.0
output application/json
import getName from CountryConverter
---
{
  nameFromTwoLetterCode: getName("AR"),
  nameFromThreeLetterCode: getName("ARG")
}
```

Output

```
{
  nameFromTwoLetterCode: "Argentina",
  nameFromThreeLetterCode: "Argentina"
}
```



getShortCode ↑↑

getShortCode(*valueToConvert*: ThreeLetterCode | CountryName): String

Get a two letter country code based on the country name or it's three letter country code.

Parameters

Name	Type	Description
<i>valueToConvert</i>	ThreeLetterCode CountryName	ThreeLetterCode: ["AFG", "ALB", "DZA", "AND", "AGO", "ATG", "ARG"] CountryName: ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina"]

Example

This example shows how the `getShortCode` function behaves under different inputs.

Source

```
%dw 2.0
output application/json
import getShortCode from CountryConverter
---
{
  twoLetterCodeFromName: getShortCode("Argentina"),
  twoLetterCodeFromThreeLetterCode: getShortCode("ARG")
}
```

Output

```
{
  twoLetterCodeFromName: "AR",
  twoLetterCodeFromThreeLetterCode: "AR"
}
```



Variables

countryMapping ↑↑

Constant for country definitions

Types

CountryName ↑↑

Set of allowed values for country names

Definition

```
"Afghanistan" | "Albania" | "Algeria" | "Andorra" | "Angola" | "Antigua and Barbuda" | "Argentina"
```

ThreeLetterCode ↑↑

Set of allowed values for three letter country codes

Definition

```
"AFG" | "ALB" | "DZA" | "AND" | "AGO" | "ATG" | "ARG"
```

TwoLetterCode ↑↑

Set of allowed values for two letter country codes

Definition

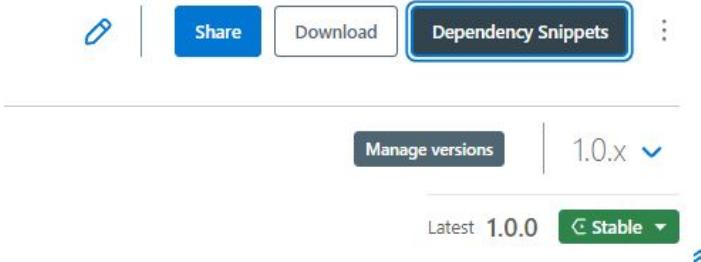
```
"AF" | "AL" | "DZ" | "AD" | "AO" | "AG" | "AR"
```



Use

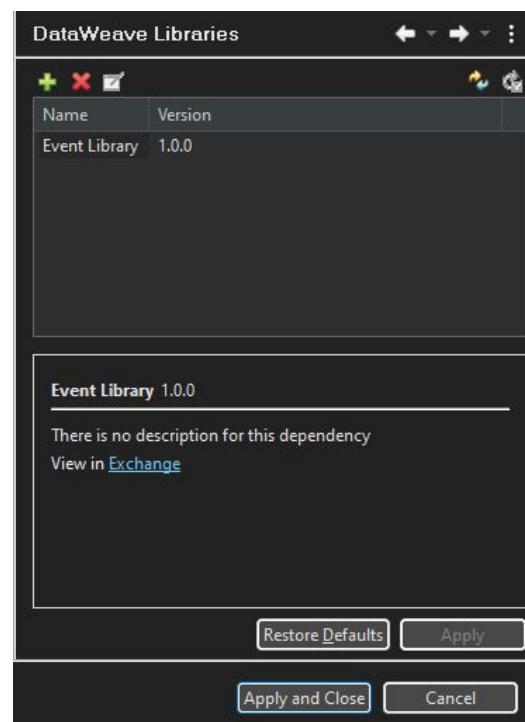
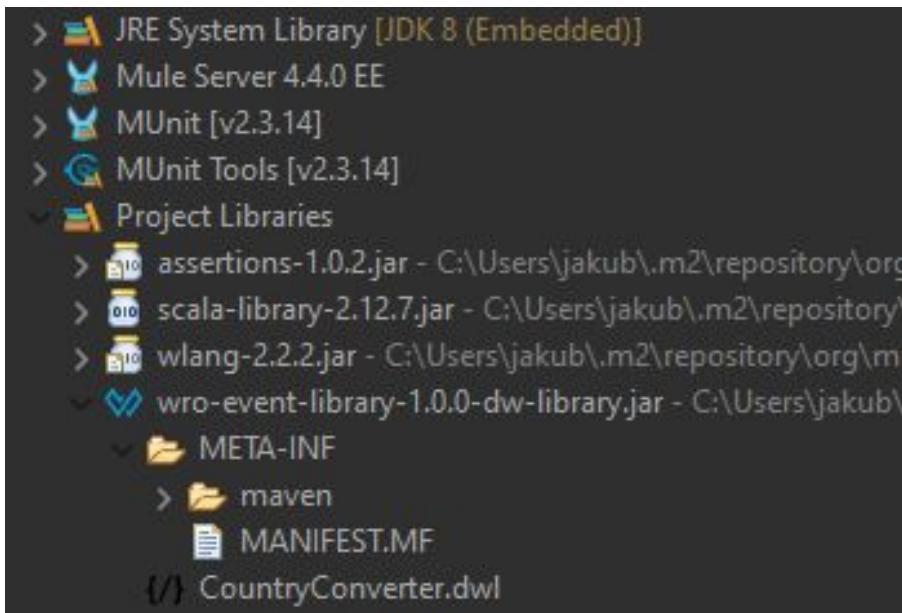


- You can get the dependency snippet for your DataWeave library in Exchange
- Add it in the dependencies section of your applications pom.xml



```
<dependency>
<groupId>ORGANIZATION ID</groupId>
<artifactId>wro-event-library</artifactId>
<version>VERSION</version>
<classifier>dw-library</classifier>
</dependency>
```

- You can also add it using by right clicking on the project -> Anypoint Platform -> Manage DataWeave Libraries -> Clicking the plus to search for it in your Exchange
- Once it's added you can find in your projects ***Project Libraries***



- To use the created functionality simply import the created module using:
import */nameOfTheFunction from ModuleName

Output Payload      

```
1@output application/json
2 import * from CountryConverter
3 ---
4 {
5     threeLetterCodeFromName: getLongCode("Argentina"),
6     threeLetterCodeFromTwoLetterCode: getLongCode("AR")
7 }
```

Result Type: String

Get a three letter country code based on the country name or it's two letter country code.

Parameters

Name	Type	Description
valueToConvert	TwoLetterCode CountryName	TwoLetterCode: ["AF", "AL", "DZ", "AD", "AO", "AG", "AR"] CountryName: ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina"]

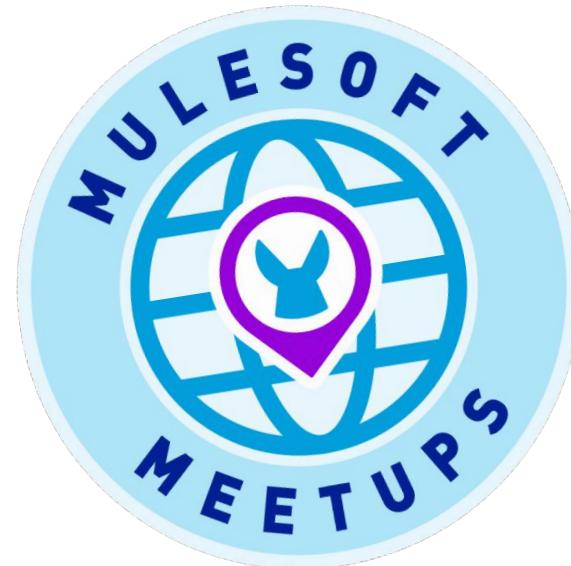
Thank you



Quiz



- Actively participate
- Propose interesting topics
- Present at future meetups
- Propose improvements/changes/ideas
- Please complete the survey





Wrocław MuleSoft Meetup #1

nextview...

Design-Led
Salesforce
Consulting



Thank You
and
see you next time!

