Training Report: Mulesoft

10.2021

—

Juan Domínguez
juan_dm93@hotmail.com
M1CH Solutions
Chetumal, Mexico

Content

Overview

This document follows a summary report with specific topics for each introductory concept from the official MuleSoft tutorials pages. Project files hosted on GitHub.

Specifications

Local resources

- *Hardware*: Macbook Pro 13" (Late 2013), i7 (2 * 2.8 GHz), 16 RAM, 512 SSD
- *Software*: MacOS 11, AnypointStudio 7, Postman 9, Chrome 94

Topics

I. *Getting started with MuleSoft* - Best practices and development steps.

II. *Anypoint Development* - Design, build and deploy APIs.

III. *DataWeave* - Data transformation.

IV. *MUnit* - Unit and functional tests in local and CI/CD environments.

Reports

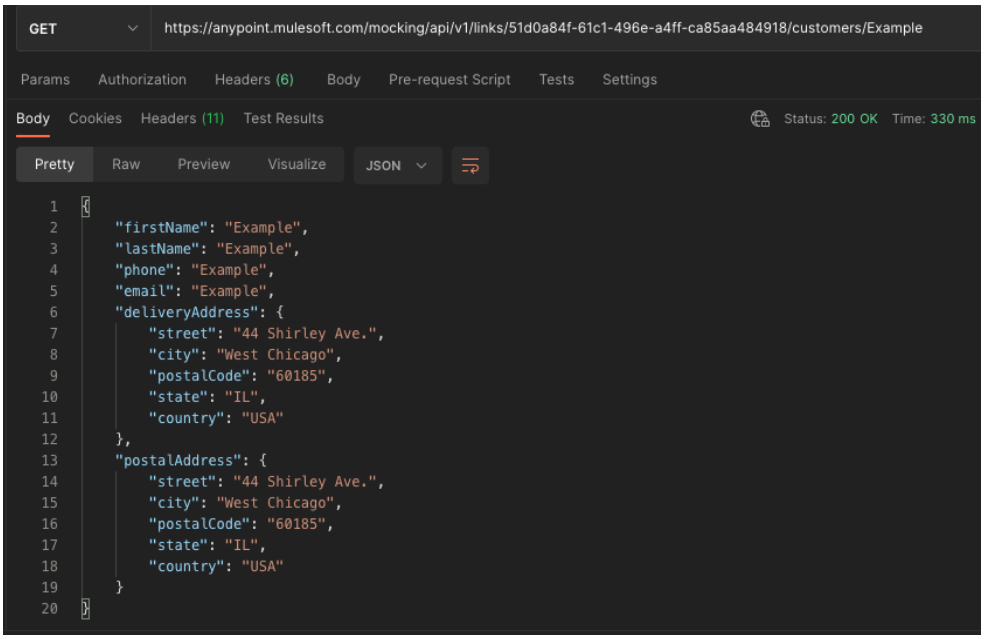
I. Getting started with MuleSoft

Follow a couple basic Mule Application development use cases using Anypoint Studio and applying governance from Anypoint Platform.

NTO Customer Database API

Build, test and publish a basic API with Anypoint Platform

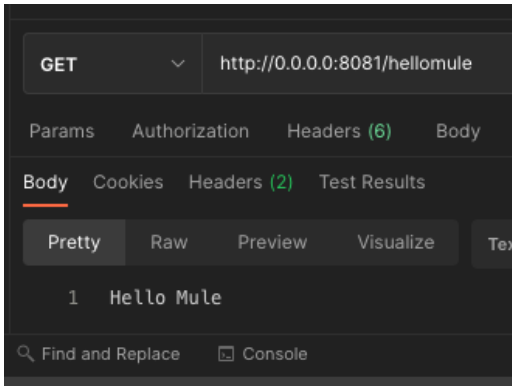
1. Create API Spec
2. Test mocking service

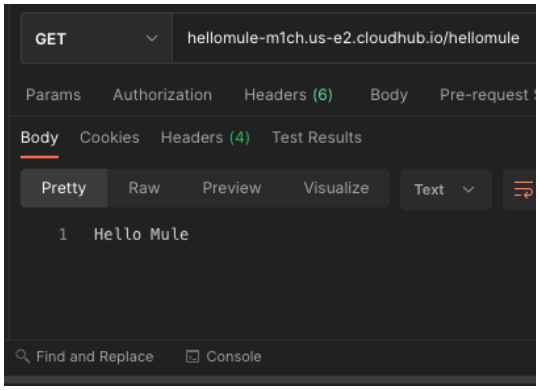


Hello Mule

Build a basic Mule App with Anypoint Studio

1. Setup local environment
    - [ ] Create Anypoint Platform Account
    - [ ] Install Anypoint Studio
2. Create basic Mule Application
    - [ ] Create basic project
    - [ ] Setup HTTP Listener connector
    - [ ] Set payload
3. Deploy and test
    - [ ] Locally
    - [ ] CloudHub

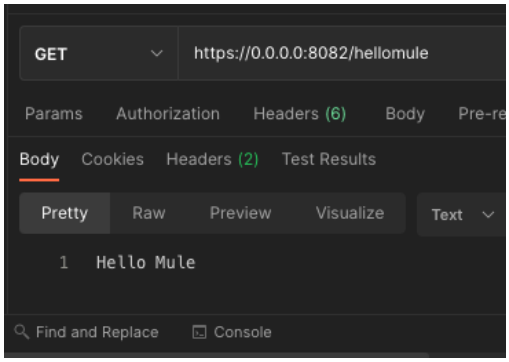




HTTPs endpoint

Use secured protocol connections via *keystore.jks* file for deployment and testing.

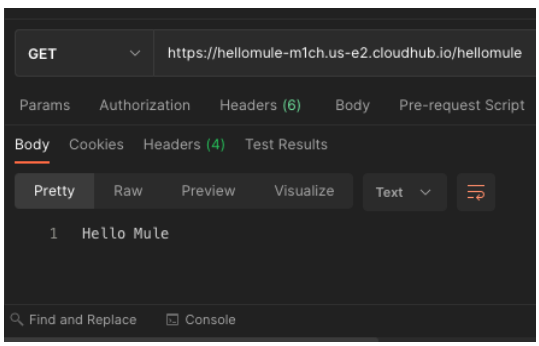
1. Setup
    - [ ] Create *JKS* file
    - [ ] Setup HTTPs configuration
2. Deploy
    - [ ] Use SSL connection

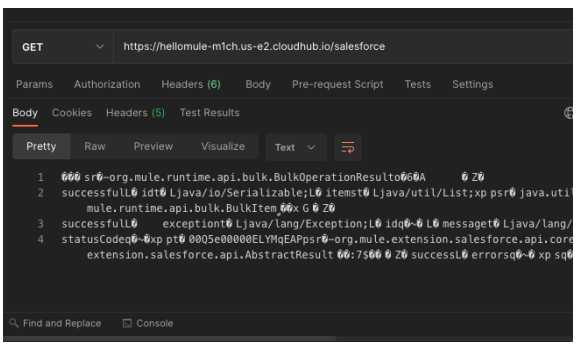


Secure properties

Use security best practices and encryption for local properties

1. Setup
    - [ ] Create *Salesforce* Account
    - [ ] Create *local.properties* file
2. Secure
    - [ ] Add Anypoint Enterprise Security repository
    - [ ] Config Salesforce connector
    - [ ] Implement secured properties and encryption
3. Redeploy and test
    - [ ] Set *secure.key* property
    - [ ] Cloudhub redeployment

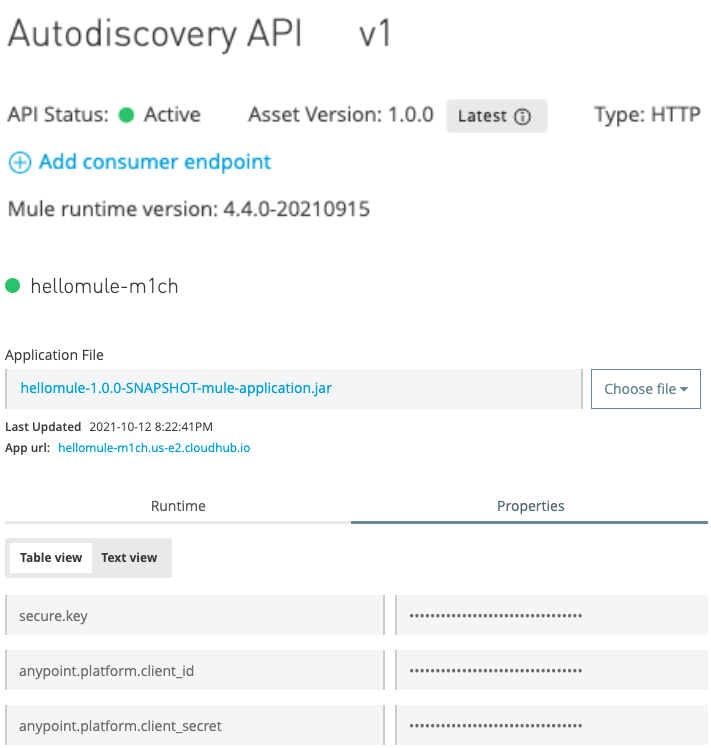




API Autodiscovery

Enable API Autodiscovery

1. Create API in API Manager
2. Setup Mule app on Anypoint Studio
3. Redeploy to CloudHub

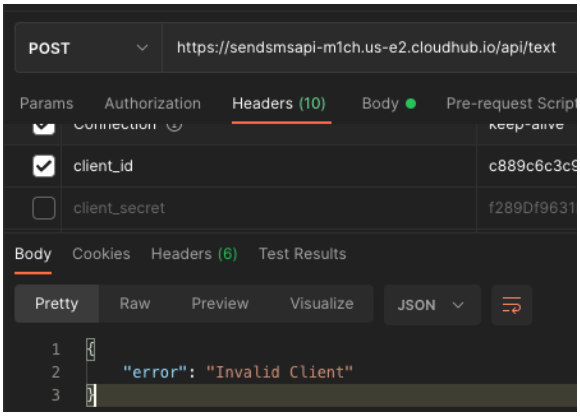


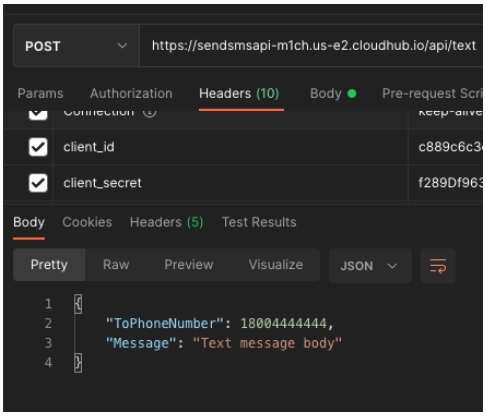
Client ID enforcement

Design API specification, build from Exchange and deploy with secure policies

1. Build API
   - ☐ API Spec in API Designer
   - ☐ Build from Exchange into Anypoint Studio
2. Deploy with API Autodiscovery
   - ☐ Enable API Autodiscovery
   - ☐ CloudHub
3. Apply policies
   - ☐ Setup API Manager

- [ ] Test secured POST





Maven deployment

1. Install *maven*
   - [ ] Download and install
   - [ ] Set up Anypoint Studio
2. Deploy Mule App
   - [ ] Create basic project
   - [ ] Deploy from *maven* to CloudHub

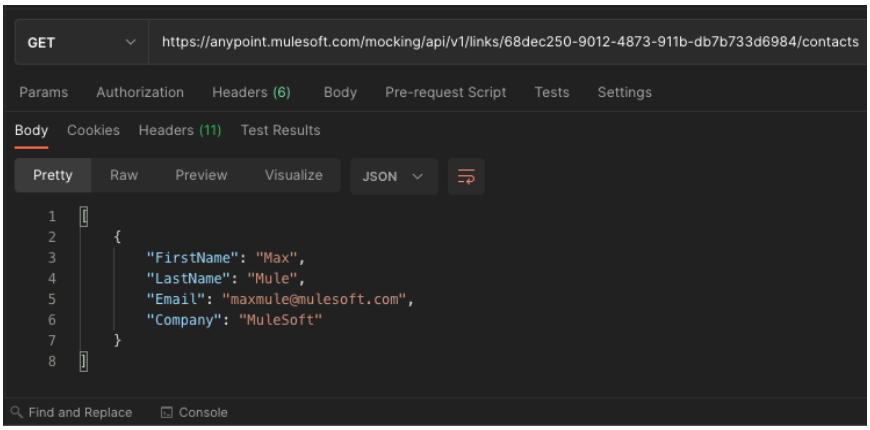
I. Anypoint Development

Follow a full lifecycle for API development and governance.

Design Center

Create an API specification

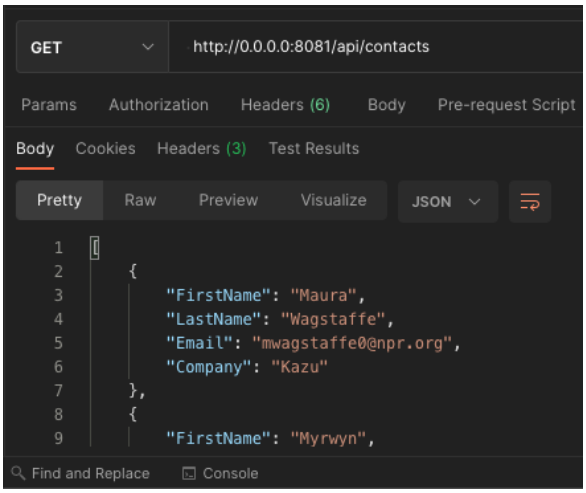
1. Setup API Spec
   - [ ] Setup endpoints and responses
   - [ ] Define data types
   - [ ] Try mocking service
2. Publish API in Exchange



Development

Build an app from spec and deploy

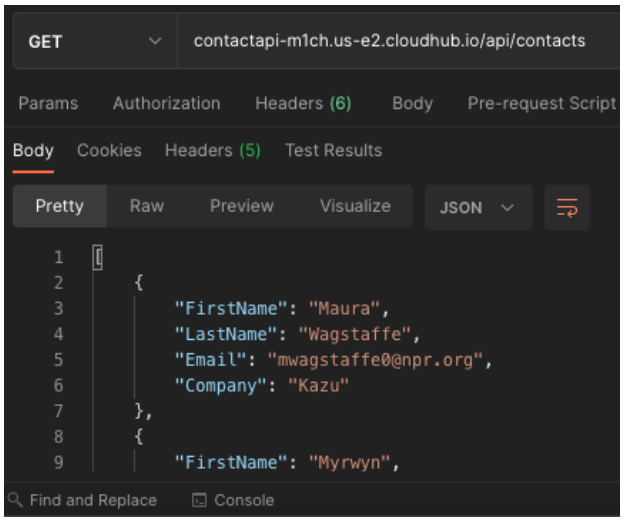
1. Create Mule App
    - [ ] Import from Exchange
    - [ ] Setup DB connector
    - [ ] Transform payload after Select
2. Deploy locally



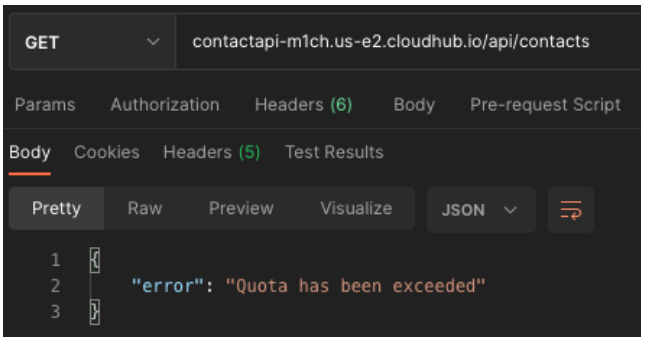
Manage

Setup app for API management

1. Setup Mule App
    - [ ] Enable API Autodiscovery
    - [ ] Secure properties
2. Deploy
    - [ ] CloudHub



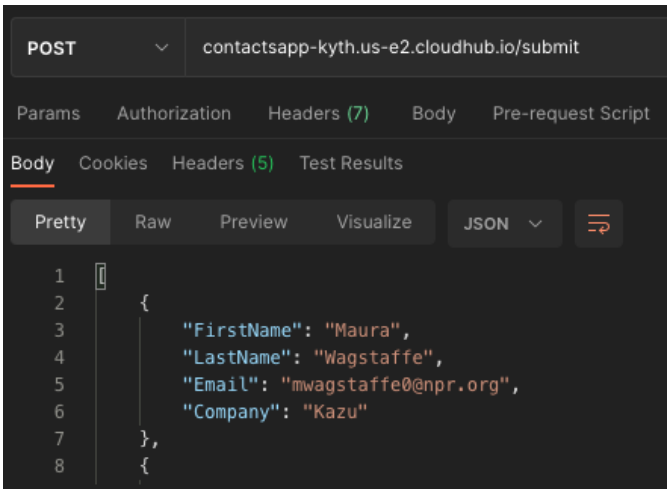
3. Configure API Manager
    - [ ] Setup rate limiting policies



SaaS to Salesforce

Use API from external app

1. Create Mule App in Flow Designer
    - [ ] Setup listener
    - [ ] Request Contact API
    - [ ] Lead to Salesforce
2. Deploy
    - [ ] Test
    - [ ] CloudHub



II. DataWeave

Following *playground* tutorial

Name	MIME Type	ID
XML	application/xml	`xml`
JSON	application/json	`json`
CSV	application/csv	`csv`

Creating Data

Basic data types examples and exploration

- `Number`
- `String`
- `Boolean`
- `Array`
- `Object` or `{}`

Reading Data

DataWeave basic selectors usage and examples

- Single-value selector: `.`
- Index selector: `[n]`
- Range selector: `[n to m]`
- Multi-value selector: `.*`
- Descendants selector: `..`

Variables and Logical Operators

Defining variables and using operators

```
var <var_name> = <expression>
```

Expression	Label
`A > B`	Greater than
`A < B`	Less than
`A >= B`	Greater than or equal to
`A <= B`	Less than or equal to
`A == B`	Equal to
`A != B`	Not equal to
`A ~= B`	Similar to
`not A`	Logical negation
`!A`	Logical negation
`A and B`	Logical and
`A or B`	Logical or

Flow Control

If/else constructs and the basics of pattern matching

```
if (<criteria_expression1>)
  <return_if_true>
else if (<criteria_expression2>)
  <return_if_true>
else
  <return_if_no_other_match>
```

```
<input_expression> match {
  case <condition> -> <execute_if_condition_pass>
  case <condition> -> <execute_if_condition_pass>
  else -> <execute_if_no_condition_pass>
}
```

Functions

Create and reuse *functions*, explore *lambdas* and discover $, $$, $$$ syntax and *infix* notation

```
fun <function_name>(<arg1>, <arg2>, …, <argN>) = <body>
```

```
(<arg1>, <arg2>, …, <argN>) -> body
```

```
<arg1> <function_name> <arg2>
```

Working with Arrays

Basic array functions: *filter, map, disctingBy, groupBy and reduce*

```
fun filter <T>(items: Array<T> , criteria: (item: T, index: Number) -> Boolean):
Array<T>
```

```
map(Array<T>, ((T, Number) -> R)): Array<R>
```

```
distinctBy(Array<T>, ((T, Number) -> Any)): Array<T>
```

```
groupBy(Array<T>, ((T, Number) -> R)): Object<(R), Array<T>>
```

```
reduce(Array<T>, ((T, R) -> R)): R
```

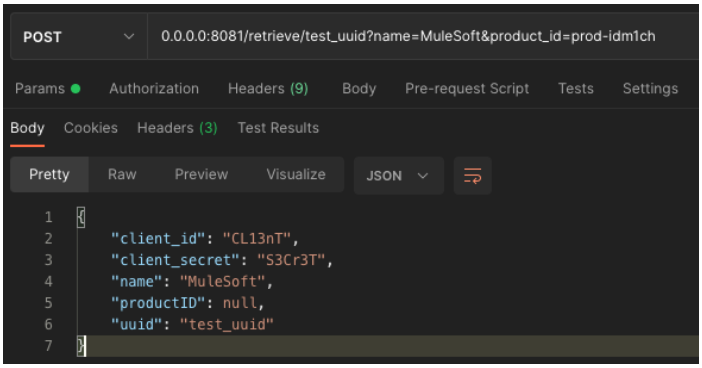
Working with Objects

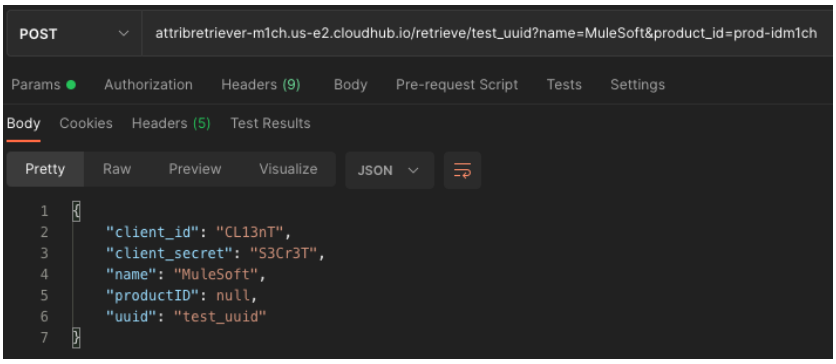
Basic object functions: *filterObject, mapObject, pluck and update (!)*

```
filterObject(Object<K,V>, ((V,K,Number) -> Boolean)): Object<K,V>
mapObject(Object<K,V>, (V,K,Number) -> Object): Object
pluck(Object<K,V>, (V,K,Number) -> T): Array<T>
```

Retrieve attributes

Use *attributes* namespace to retrieve http parameters from connection data (CloudHub).





III. MUnit

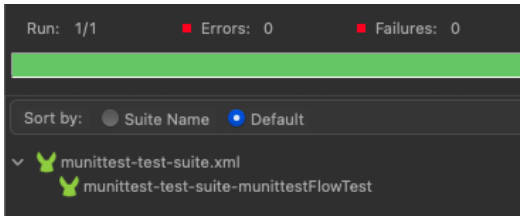
Create a test

Build a mule app setting db properties and use MUnit DB Server to test it with a CSV.

1. Create Mule App
   - ☐ Setup listener
   - ☐ Configure DB Select connector
   - ☐ Run locally
2. Create MUnit Test

- [ ] Setup MUnit DB Server
- [ ] Create db.properties files accordingly

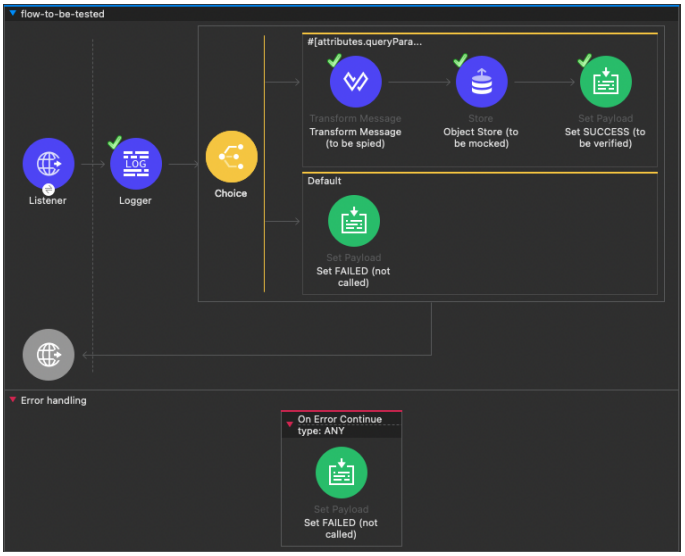
3. Run test



Test Recorder

Use Test Recorder utility to automatically build meaningful tests for a basic project.

1. Create Example mule app
   - [ ] Import ObjectStore
   - [ ] Validate XML file
2. Run test recorder
   - [ ] Start recorder
   - [ ] Capture request
3. Test recorder config
   - [ ] Follow wizard for test suite creation
   - [ ] Verify test



Conclusions

Topics covered in this report represent some basic requirements needed to succeed as a mule developer. From achieving a complete mule application lifecycle using Anypoint Platform, as well as with Anypoint Studio, performing data transformations with DataWeave fundamentals and following developing best practices and testings.