Lab 1

Implement the Order API in Studio
Overview

In this lab, we will create the skeleton implementation of our Order API that will process requests to a SOAP Web Service. You will use Anypoint Studio to create a Mule application where there will be one flow for each method of each resource (i.e. GET orders). Additionally you will use APIKit (as part of your Mule application) to process REST requests, transform them to messages to be handled and processed by each flow.

The implementation will consist of a couple of steps:

1. Create a new Mule Project in Anypoint Studio from the Order API RAML
2. Run the Mule API(Application) from Anypoint Studio

Then we will test this new API using the API Console, in Studio. In this application we will create an Order as the test.

Step 1: Create a new Mule Project from the RAML

In this step we will create a new Mule application in Anypoint Studio from the Order API RAML. This will be the implementation of our REST API.

1. If it’s not already opened, start Anypoint Studio from the desktop icon and select the same workspace you’ve been working (example: C:\workspaces\myworkspace).
2. From Anypoint Studio’s menu, select File > New > Mule Project to create a New Mule project. A Window will pop-up to define the details of this new application.
3. Give the project the name api-order
4. Select the Mule Server 4.1.4 EE.

5. Under API Implementation settings, check Specify API definition file location or URL and choose Design Center... by clicking in the (...) button on the right of the Location textbox.

6. Once you clicked there, it should take you to the Browse Design Center for APIs window, if it asks you to login, use your Anypoint Platform account.

7. Under your default Business Group, and you will see APIs associated to this Business Group under the Project Name/Branch table.

8. To easily find the correct project, you can filter for it by typing Order in the search box. Then select Order API Project name and hit the OK button.
WARNING If you don't see the Order API, press Load more projects. This link only appears when there are too many projects on the API Designer.

9. Click Finish on the next screen. This will create a skeleton project that implements your API.

Let's explore what was automatically generated.

api-main This is the main flow. It exposes an HTTP service and processes the requests using the APIKit Router.

The HTTP request will be converted to a mule message, and redirected to the requested flow by the APIKit router.
**get:`orders`**
This flow returns all the products from the database.

**post:`orders`**
This flow creates a new product in the database.

**get:`orders`(id)**
This flow is used for getting the detail of a product based on the product id.

**put:`orders`(id)**
This flow is used for updating an order based on the order id.

**delete:`orders`(id)**
This flow is used for deleting an order based on the order id.

When APIkit detects example data in the response of a method in the RAML it inserts a Transform Message into the flow which returns the static response specified by the example data reference.
If you do not see the `src/main/resources/api` folder with the `api.raml` (name may differ) file and corresponding JSON files (as shown in the screenshot above), the project generation was not successful. Try deleting/re-generating your project or alert your instructor to troubleshoot.

The static response returned by the auto-generated Transform Message allows you to test the stub immediately after generation. Obviously these flows can be enhanced to provide more advanced mock services as well as evolve into the full API implementation.

**Step 2: Run the Mule API(Application) from Anypoint Studio**

1. Right-click the application (or on the top project folder in Package Explorer).
2. Select Run As > Mule Application
Anypoint Studio deployed this application to an embedded Mule run-time. There is no need to deploy to a separate Mule server environment. The developer will be able to develop and test the application locally until it's ready to be deployed to a shared development or upper environments.

3. If the application is deployed successfully, you should see a message like the one in the picture below. If this is the case, continue.
4. Test the application using the **APIkit Console**. You will see a link to the console in a new tab. Click on the link to bring up the console in a browser.

The console will look like this in the browser.
5. Click the POST link on the left then the TRY IT link on the top right. This will allow you to test the Orders resource (/orders) using the Create an Order (POST) method. Make sure you explore the Parameters, Headers, and Body tabs on this page to see its contents.

6. You will notice the request body is a pre-populated payload shown in the Body tab. Now Click on the SEND link.
7. After hitting the SEND request of a POST method, you should see an "example" response below. Remember this is no longer a mocking service responding. It is a live application deployed to a local Mule run-time server. You just tested your end-to-end API running in your environment.
8. Go back to Anypoint Studio, under the Console tab and press the red button to stop the Mule run-time server.

Summary

In this lab, you completed the following steps:

- Step 1: Create a new Mule Project from the RAML
- Step 2: Run the Mule API(Application) from Anypoint Studio

We used the RAML specification to define our API and auto-generated a skeleton project to implement the API. This lab shows how APIkit quickly allows developers to import the API designed REST API (RAML and supporting artifacts) to enable MuleSoft flow development and testing. We also ran the Order API by using RAML specification designed to GET Orders info, POST (create) Order(s), GET Order by ID, PUT (update) Order by ID and DELETE Order by ID. We leveraged RAML for a design first approach. This significantly speeds up end to end development. Additionally, we tested the API from a browser to simulate a real client of the API. To learn more
about RAML follow this tutorial.

See the APIkit doc for more information.

Congratulations! You have completed Lab 1.

Please proceed to Lab 2

Take me to the TOP