WEATHER API DOCUMENT

Sumeet Kumar

Introduction

This project explains the technicalities, software's used, process followed to build the REST API for weather SOPA WSDL.

Audience

All software team who will use the API, business user who will consume the API, support team for giving support once the project is deployed to server.

Scope

This can be considered as a project that will take a SOAP wsdl and transform that into a rest web service call in Mule platform.

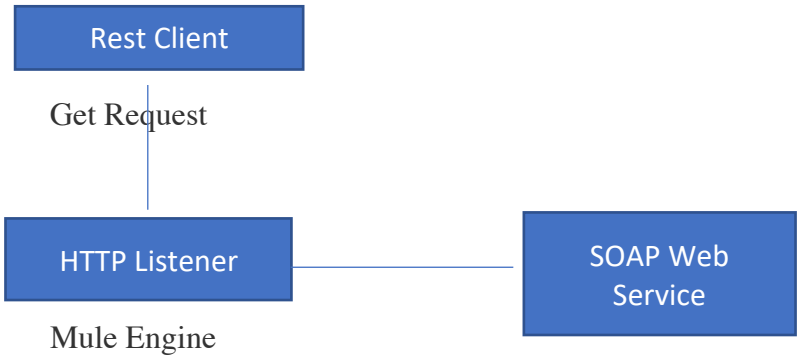
Why MuleSoft

- Reusable APIs save your organization money by reducing the number of connections you need to be operational.
- The faster the deployment, the sooner you're able to start utilizing your new solutions, saving your company money on implementation.
- MuleSoft's plug-and-play solutions are also easy to install, freeing up your IT team to focus on other business-critical tasks like customer experience, security and regulation.
- Moreover, its free to use.

Software Used

- Anypoint Studio - 7.8
- Mule 4.3.0
- RAML 1.0
- Node 15.4.0

Architecture



The HTTP Listener resides within the Mule Engine runtime where the REST adapter resides is Mule EE 4.3.0, and, through a series of components given by the compile and runtime libraries, there is component that listens to HTTP requests on a particular port in the local machine, capturing the input as query Params parameters or path variables to set the input of another component.

Weather service resides in SOAP webservice. The HTTP call is done via the Rest client to the Mule component to see the response from the SOAP web service.

How to Run the project:

- As the provided URL in the specification does not work, so we need to download the Docker Node project separately and run the project
- Note: As the project is build on MAC, hence all the steps to build, execute will be according to the MAC OS.

NODE

- Download the weatherExcerciseDockerFile from the GIT Repository.
- Run npm install
- Then go to the terminal till where the folder weatherExcerciseDockerFile folder and run node server.js to start the weather SOAP service.
- Now go to Postman and trigger the below URL to check whether the service is up and running: http://localhost:8080/GlobalWeather?WSDL
- Once the service is up and running, then you can start with the Mule development flow and other process.

MULE Runtime Environment SETUP

- Go to https://www.mulesoft.com/lp/dl/studio and download Mule latest version. I have used 4.3.0.
- Now you can setup the Mule Anypoint Studio in your local for development.

MULE APPLICATION DEVELOPMENT

- Firstly, go to Design Centre https://anypoint.mulesoft.com/designcenter/ and build your RAML specification for the weather API.
- Once the raml specs are build, you are ready to start for building the Mule project.
- There is another GIT commit https://github.com/sumeetkumar090/weather-api-interface for raml specs also done where the RAML are usually put in under the interface folder.

START BUILDING

- Go to Anypoint Studio and click on File -> Mule Project with the latest mule runtime selected.
- Make sure Java 8 is installed on your computer and is set in the path variable.
- Make sure Mule project points to Java 8 when running the project in Mule.

Weather-API Project Details

- The raml is present under src/main/resources/api.
- The Mule .xml files are present under src/main/mule.
- The munit test cases are present under src/test/munit.

The project details are as below which follow the rest guidelines principles:

- config.xml - All project configs are setup like HTTP, WebService call etc.
- get-cities-by-country.xml - Subflow to get cities by Country.
- get-weather-by-city-country.xml - Subflow to get weather by city and Country.
- util.xml - Utility package.
- global-weather-api.xml - Main flow of the application
- get-cities-by-country-test-suite.xml - Munit test case to get cities by Country.
- get-weather-by-city-country-test-suite.xml - Munit test case to get weather by city and country.

The project has been followed with best design principles for development:

- Modularized RAML with types, traits.
- Configuration properties files.
- Segregation of the flows with sub-flows, utility components.
- Unit test cases completed in Munit.
- Data Transformation done using Dateweave 2.0.
- Error handling done with different cases.

Set Up the Weather-API Project

Prerequisite

- SOAP web service should be running on the server.

Download the project from the Github and import as Mule Project in your Anypoint studio.

- If you find any error just go the project in local and run mvn eclipse:eclipse or mvn clean install to fix any issues locally.
- Once this is all setup, Run the project - Right Click -> Run as Mule Project.
- Now the project is deployed successfully.

Running it in Postman

- Go to postman and run the below commands to run get the response from the API:

  http://localhost:8081/api/getWeather/getCitiesByCountry?countryName=India

  http://localhost:8081/api/getWeather/getWeatherByCityAndCountry?cityName=Melbourne&countryName=Australia

Exception Handling

- If the city, country name is missing HTTP 404 Status code will print on the postman response.
- Any call to the URL registered in the RAML specs, will print 400, 404,405 406, 415, 500 status code in the postman call.

DESIGN DECISIONS

- All the flows, sub-flows, should confirm to the Mule design principles.
- Utility component is created so that it can be reused again in other places.
- The .properties is setup such that we can access the properties directly from the properties file.
- The config is setup to access the configurations from one place.
- The input and output folders with examples are created.
- Munit are created to make sure all the sub-flows are tested and there is no issues in the code.

CHALLENGES

Overall, the weather-api task was a fun one. I faced a couple of challenges and learned also. I have worked with Mule 3.9 extensively and mostly use to working with REST API, DB Call when building REST APIs via MULE. I recently started with Mule 4. Working and setting Mule 4.3.0 in my local and then familiarizing with different components. Setting the

weather docker instance in local took a little bit of time. Need to read a few articles but got to it eventually. Getting the response from the wsdl file in Mule flow took some time to understand and work out. Data sanitization (removing extras CDATA, Lt, gt) from the wsdl response.

LEARNING

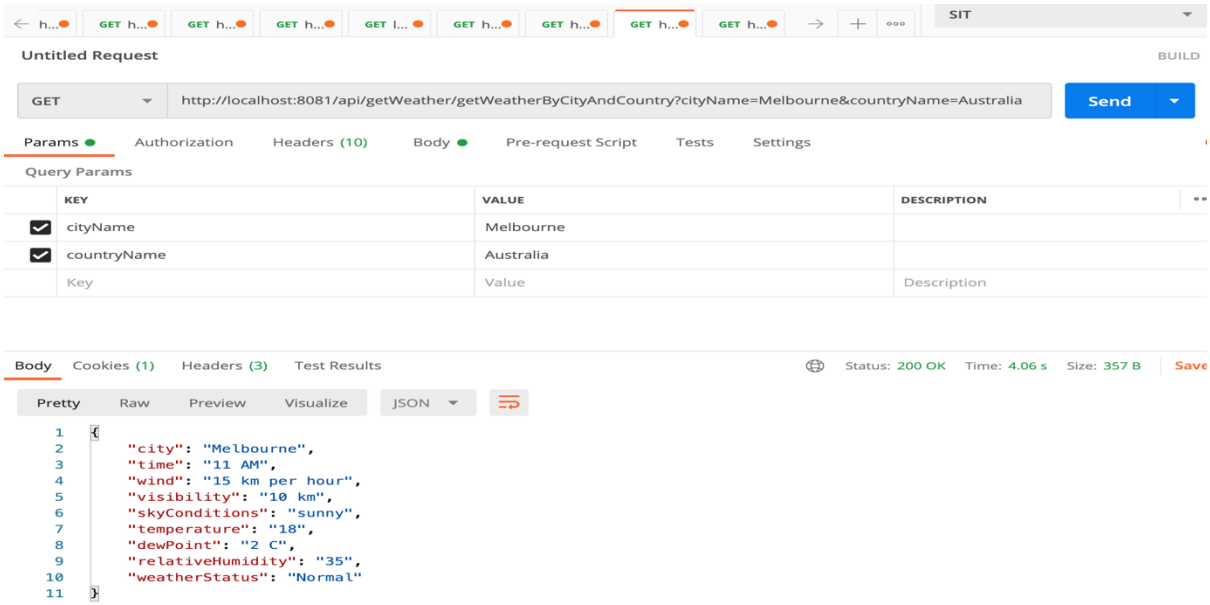
I am comfortable with Data weave as I use it extensively throughout the projects. Learned new things in data weave transformation as setting payload directly as string. Getting .wsdl responses directly in Mule flow. Difference between flowVars and vars in Mule 4. Added extra choice condition just in case there is no response from the SOAP API.

ENHANCEMENTS:

- We can have different .properties files for DEV, SIT, PROD if we need to deploy the projects in different environment.
- The files can be setup in one config and can be used as per the deployment environment.
- Currently the SOAP service does not have username and password, if it has we can keep the secrets under the secrets folders and encrypt it so that it is not seen by the outside world.
- The logging can be improved using Mule Utility package from Mule (Currently using Logger).

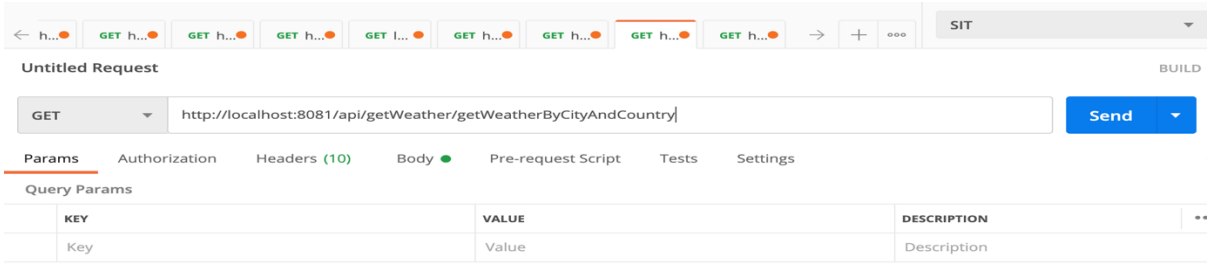
Postman Responses

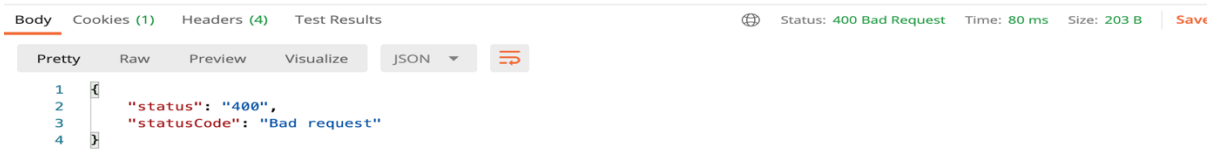
http://localhost:8081/api/getWeather/getWeatherByCityAndCountry?cityName=Melbourne&countryName=Australia



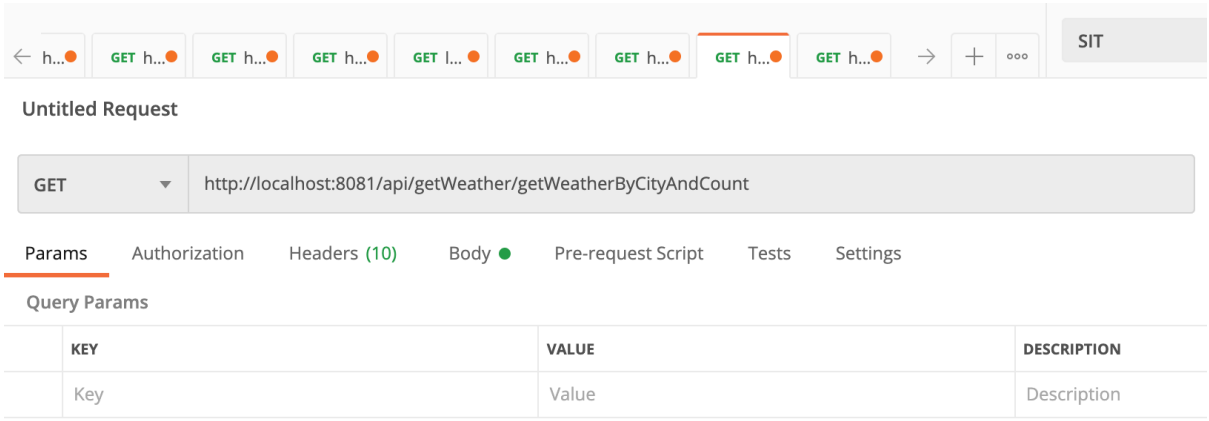
Sumeet Kumar – Weather API Design Specs

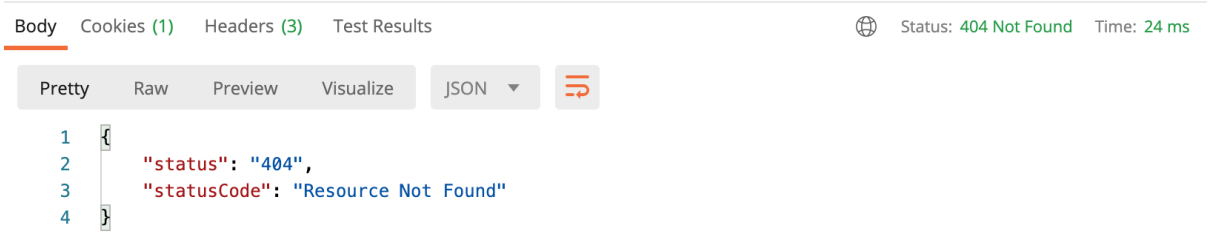
http://localhost:8081/api/getWeather/getWeatherByCityAndCountry





http://localhost:8081/api/getWeather/getWeatherByCityAndCount





http://localhost:8081/api/getWeather/getCitiesByCountry?countryName=Australia