Weather Service REST Adapter

4th November 2020

Author: Benson Coelho

Contents

[Introduction 2](#_Toc57990074)

[Purpose 2](#_Toc57990075)

[Intended functionality 2](#_Toc57990076)

[Description 2](#_Toc57990077)

[Architecture 2](#_Toc57990078)

[Usage 3](#_Toc57990079)

[Mock Weather Service 3](#_Toc57990080)

[Application 3](#_Toc57990081)

[Challenges 3](#_Toc57990082)

# Introduction

## Purpose

This document explains the technical implementation of a REST adapter for a SOAP weather service.

## Intended functionality

* The API must follow RESTful design principles
* The API must use DataWeave code to transform the message
* The API must provide suitable error handling (e.g. bad input message)
* RAML resources must be included in the application source
* Unit tests using Munit should be developed.

# Description

The application enables a user to make http requests to a SOAP service using a RESTful API GET request. It has been built using Mule ESB and the Anypoint IDE available from Mulesoft. A user may be a person or another software that wishes to communicate with the SOAP weather service.

At the time of developing this application, the web service was not accessible. A mock web service provided by the client was used in its place.

# Architecture

The application uses http listeners to receive a request from the user. It then makes a the relevant request to the SOAP web service and delivers the response to the user.

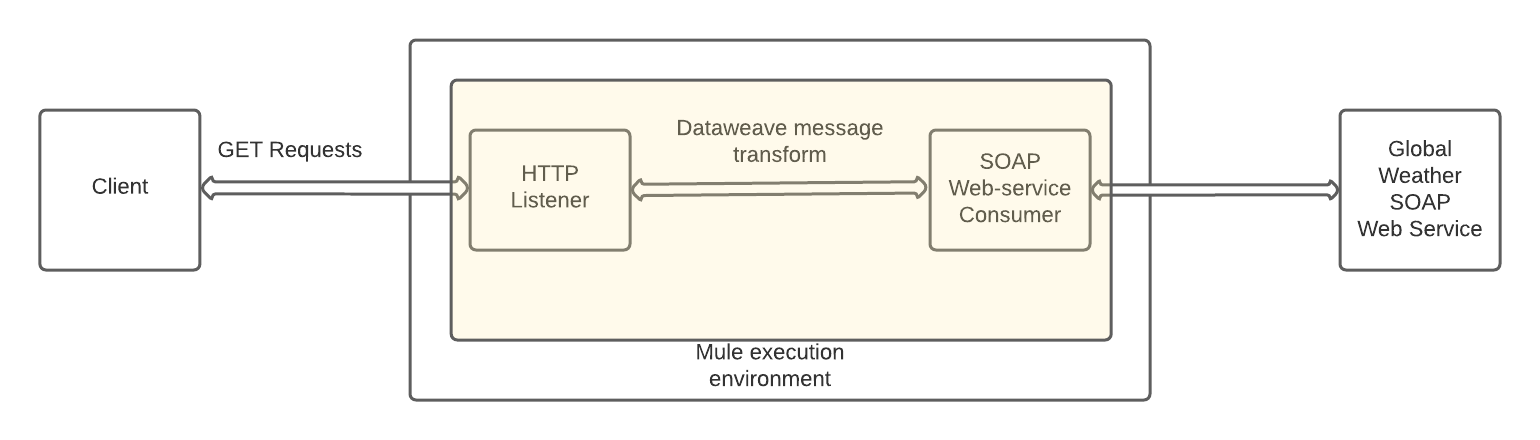


Fig 1: System architecture

# Usage

Clone the repository located at <https://github.com/bdcoelho/WeatherRAML> to your local machine.

## Mock Weather Service

The mock weather service requires some setup before it can be used. The user can set it up using an IDE such as Visual Studio Code. Once downloaded, navigate to the folder ./WeatherRAML/weatherExcerciseDockerFile/weatherExcerciseDockerFile in the integrated terminal and run npm install. This will install the required packages. Next run npm start to run the mock weather service.

## Application

* If you do not have Mule installed, you can download it with the Anypoint IDE from <https://www.mulesoft.com/> .
* You will require to have JDK 8 installed on your machine to use Mule runtime
* Open the application in Anypoint Studio and run the application from the Run menu.
* Once deployed, the service will now be live for access through a browser or Postman.
* You can make request to the GetWeather service using the URL

<http://0.0.0.0:8081/get-weather?CityName=Melbourne&CountryName=Australia>

* Replace Melbourne and Australia with a desired city and country for other locations when using the live weather service. However, Melbourne is the only location available in the mock weather service.
* You can make request to the GetCitiesByCountry service using the URL

http://0.0.0.0:8081/GetCitiesByCountry?CountryName=Australia

* The mock service only has data for Australia while using the above mentioned method.

# Challenges

Both the languages and the tools were unfamiliar to me. I managed to set up the mock weather service without much trouble. I also was able to make requests to the service in the early stages. However, I struggled to parse the response which came back in CDATA format and that prevented me from moving on to the next stages. Additional knowledge of the workings of the Anypoint platform or Java would have been useful to me from what I understand.

My approach after successfully parsing the message would be to add error handling. This would include having an error message where users:

* Entered an invalid country or city
* Did not enter a country or city where required

The unit tests would include tests such as:

* Temperature and Dewpoint can be coerced to a numerical value
* Humidity can be coerced to a percentage
* Status is ‘Normal’