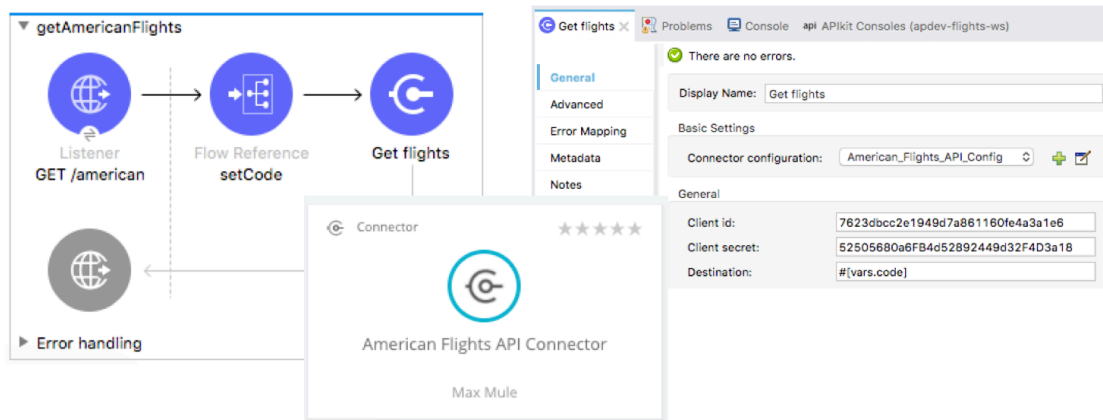


Walkthrough 8-1: Consume a RESTful web service that has a connector in Exchange

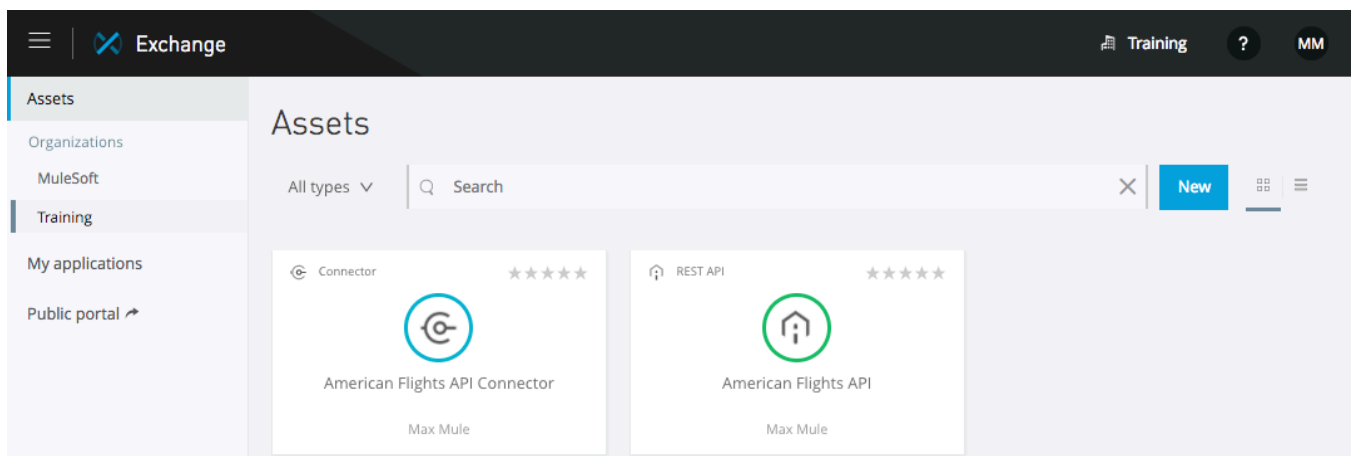
In this walkthrough, you consume the American flights RESTful web service that you built that has an API and a connector in Exchange. You will:

- Create a new flow to call the American RESTful web service.
- Add a REST connector from Exchange to an Anypoint Studio project.
- Configure and use a REST connector to make a call to a web service.
- Dynamically set a query parameter for a web service call.



Review the auto-generated REST connector in Exchange

1. Return to Exchange in Anypoint Platform.
2. Locate the American Flights API Connector that was auto-generated for your American flights API.



3. Select the connector and review its information.

The screenshot shows the MuleSoft Exchange interface for the 'American Flights API Connector'. The left sidebar shows the 'Assets list' with the connector selected. The main area displays the connector's name, a 5-star rating (0 reviews), and a placeholder image with the text 'This page currently doesn't contain a description. Click Edit to add text, images, videos, code blocks, etc...'. The right sidebar shows the 'Overview' section with details: Type (Connector), Created by (Max Mule), and Published on (Apr 18, 2018). Below this is a 'Versions' table:

Version	Runtime version
1.0.2	3.6.0
1.0.1	3.6.0
1.0.0	3.6.0

Make a request to the web service

4. Return to Advanced REST Client.
5. Select the first tab – the one with the request to your American Flights API <http://training4-american-api-{lastname}.cloudhub.io/flights> and that passes a `client_id` and `client_secret` as headers.
6. Send the request; you should still see JSON data for the American flights as a response.

The screenshot shows the Advanced REST Client interface. The 'Method' is set to 'GET' and the 'Request URL' is `http://training4-american-api-mule.cloudhub.io/flights`. The 'Parameters' tab is selected, showing two headers: `client_id` with value `7623dbcc2e1949d7a861160fe4a3a1e6` and `client_secret` with value `52505680a6FB4d52892449d32F4D3a18`. The response status is `200 OK` with a response time of `2125.58 ms`. The response body is a JSON array:

```
[Array[1]]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 787",
    "totalSeats": 200
  }
}
```

Add a new flow with an HTTP Listener

- Return to the apdev-flights-ws project in Anypoint Studio.
- Open implementation.xml
- Drag out an HTTP Listener and drop it in the canvas.
- Rename the flow to getAmericanFlights.
- In the Listener properties view, set the display name to GET /american.
- Set the connector configuration to the existing HTTP_Listener_config.
- Set the path to /american.
- Set the allowed methods to GET.

The screenshot shows the 'Listener' properties view in Anypoint Studio. The 'General' tab is selected in the left sidebar. The 'Display Name' is set to 'GET /american'. Under 'Basic Settings', the 'Connector configuration' is set to 'HTTP_Listener_config'. Under 'General', the 'Path' is set to '/american'. The 'Problems' and 'Console' tabs are also visible at the top.

Add the American Flights API module to Anypoint Studio

- In the Mule Palette, select Search in Exchange.
- In the Add Modules to Project dialog box, enter American in the search field.

The screenshot shows the 'Add Modules to Project' dialog box. The 'Username' is set to 'maxmule4'. The search field contains 'american'. The 'Available modules' table lists two modules: 'Training: American Flights API' and 'American Flights API'. The 'Selected modules' table is empty. The 'Add >' button is visible between the two tables. The 'Cancel' and 'Finish' buttons are at the bottom right.

Name	Publisher
Training: American Flights API	MuleSoft
American Flights API	Training

Name	Version
------	---------

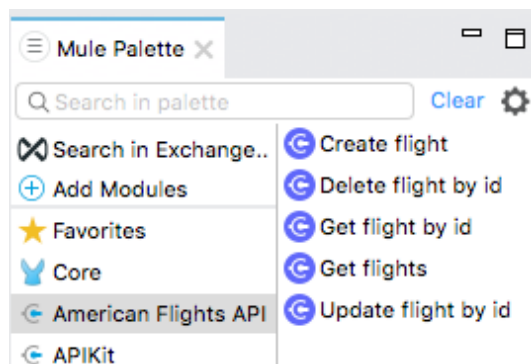
17. Select your American Flights API (**not** the Training: American Flights API) and click Add.

Note: If you did not successfully create the American Flights API and connector in the first part of the course, you can use the pre-built Training: American Flights API connector. This connector does not require client authentication.

18. Click Finish.

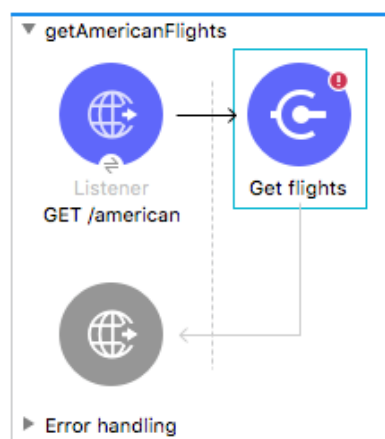
19. Wait for the module to be downloaded.

20. Select the new American Flights API module in the Mule Palette.

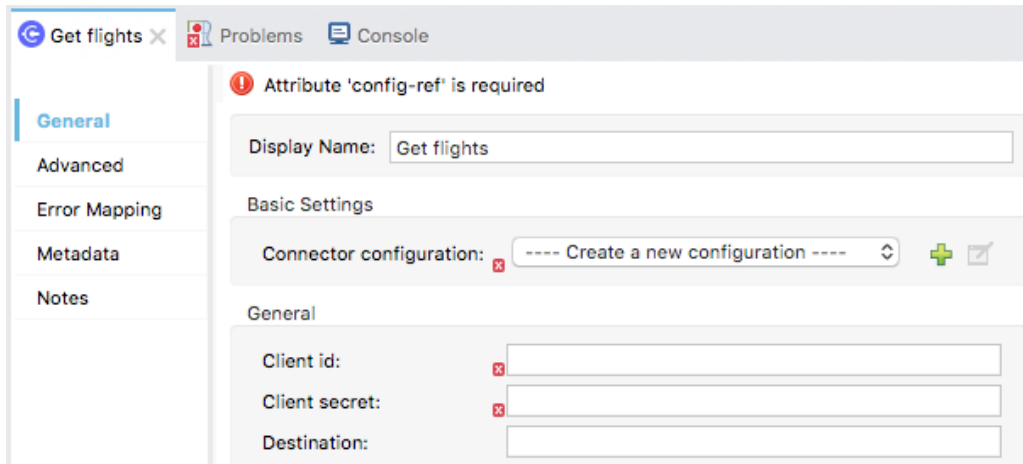


Add a Get all flights operation

21. Select the Get flights operation in the right side of the Mule Palette and drag and drop it in the process section of the flow.

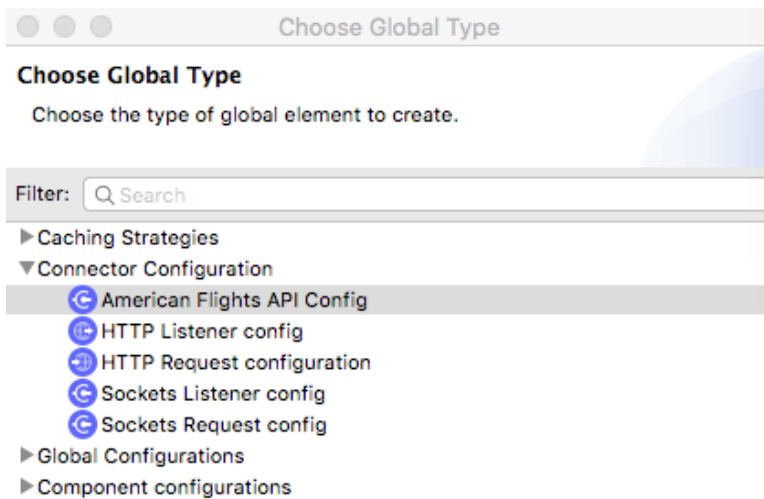


22. Select the Get all flights operation in the canvas and in its properties view, see that you need to need to create a new configuration and enter a client_id and client_secret.



Configure the American Flights API connector

23. Return to global.xml.
24. In the Global Elements view, click Create.
25. In the Choose Global Type dialog box, select Connector Configuration > American Flights API Config and click OK.



26. In the Global Element Properties dialog box, set each field to a corresponding property placeholder (that you will define and set next):

- host: `${american.host}`
- port: `${american.port}`
- basePath: `${american.basepath}`
- protocol: `${american.protocol}`

The screenshot shows a dialog box titled "Global Element Properties" with a subtitle "American Flights API Config". It has three tabs: "General", "Advanced", and "Notes". The "General" tab is selected. Inside the "General" tab, there is a "Name:" field with the value "American_Flights_API_Config". Below this, there is a "General" section with four fields: "host:" with value "\${american.host}", "port:" with value "\${american.port}", "basePath:" with value "\${american.basepath}", and "protocol:" with value "\${american.protocol}". At the bottom of the dialog, there is a question mark icon, a "Cancel" button, and an "OK" button.

27. Click OK.

28. Return to the course snippets.txt file and copy the text for the American RESTful web service properties.

29. Return to config.yaml in src/main/resources and paste the code at the end of the file.



```
1 http:
2   port: "8081"
3
4 american:
5   host: "training4-american-api-{lastname}.cloudhub.io"
6   port: "80"
7   basepath: "/"
8   protocol: "HTTP"
9   client_id:
10  client_secret:
```

30. In the american.host property, replace {lastname} with your application identifier.

31. Return to Advanced REST Client and copy the value for your client_id.

32. Return to config.yaml and paste the value.

33. Repeat for your client_secret.

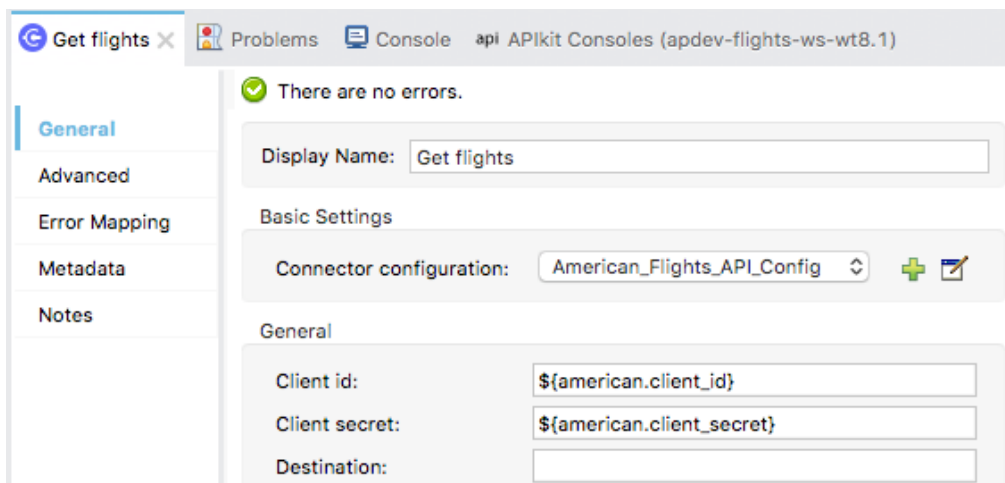
Configure the Get flights operation

34. Return to implementation.xml.

35. In the Get flights properties view, set the Connector configuration to American_Flights_API_Config.

36. Set the client id to the \${american.client_id} property.

37. Leave the destination field blank.



Get flights × Problems Console api APIkit Consoles (apdev-flights-ws-wt8.1)

✓ There are no errors.

Display Name: Get flights

Basic Settings

Connector configuration: American_Flights_API_Config + ✎

General

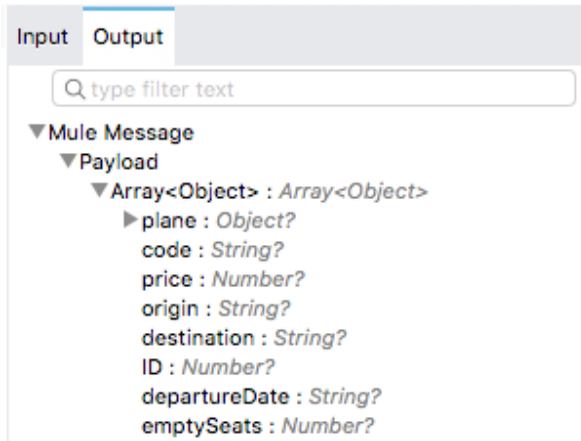
Client id: \${american.client_id}

Client secret: \${american.client_secret}

Destination:

Review metadata associated with the operation

38. Select the output tab in the DataSense Explorer and expand Payload.

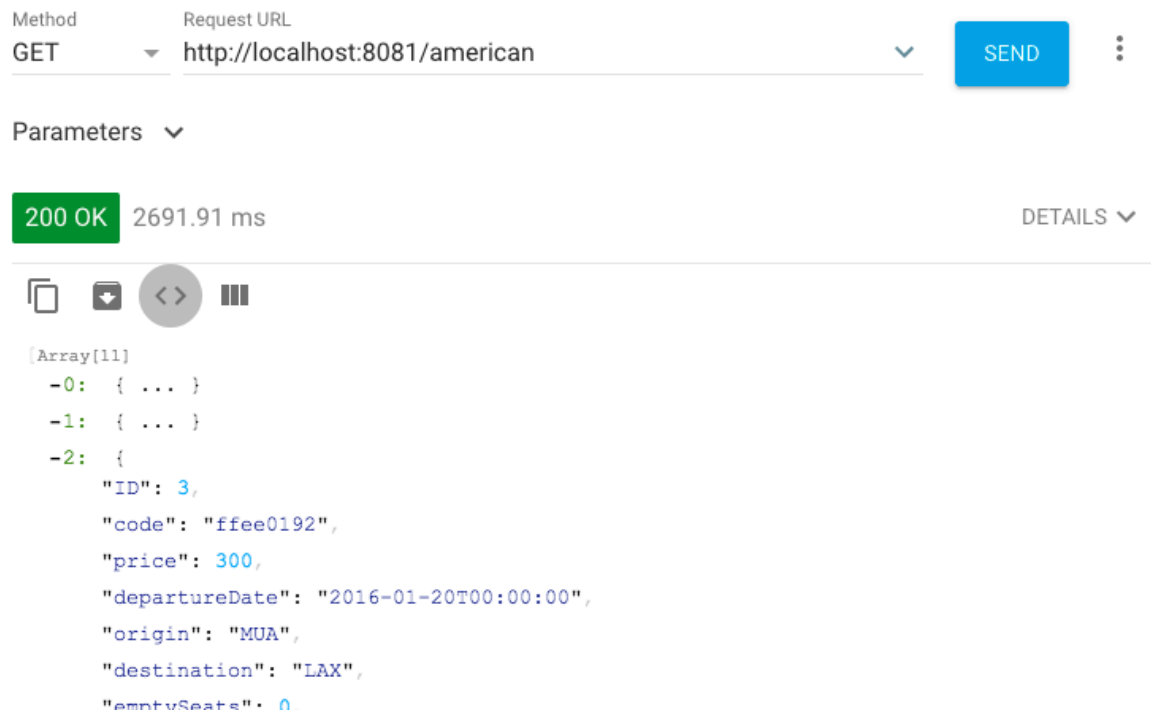


Test the application

39. Run the project.

40. In Advanced REST Client, return to the tab with the localhost requests.

41. Change the URL to make a request to <http://localhost:8081/american>; you should get all the flights.



Review the specification for the API you are building

42. Open mua-flights-api.raml in src/main/resources/api.
43. Review the optional query parameters.

```
/flights:
  get:
    displayName: Get flights
    queryParameters:
      code:
        displayName: Destination airport code
        required: false
        enum:
          - SFO
          - LAX
          - PDX
          - CLE
          - PDF
      airline:
        displayName: Airline
        required: false
        enum:
          - united
          - delta
          - american
```

44. Locate the return type for the get method of the /flights resource.

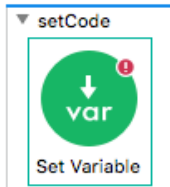
```
responses:
  200:
    body:
      application/json:
        type: Flight[]
        example: !include examples/FlightsExample.raml
```

45. Open FlightsExample.raml in src/main/resources/api/examples and review its structure.

```
##RAML 1.0 NamedExample
value:
-
  airline: United
  flightCode: ER38sd
  fromAirportCode: LAX
  toAirportCode: SFO
  departureDate: May 21, 2016
  emptySeats: 0
  totalSeats: 200
  price: 199
  planeType: Boeing 737
-
  airline: Delta
  flightCode: ER0945
  fromAirportCode: PDX
  toAirportCode: CLE
  departureDate: June 1, 2016
  emptySeats: 24
  totalSeats: 350
  price: 450
  planeType: Boeing 747
```

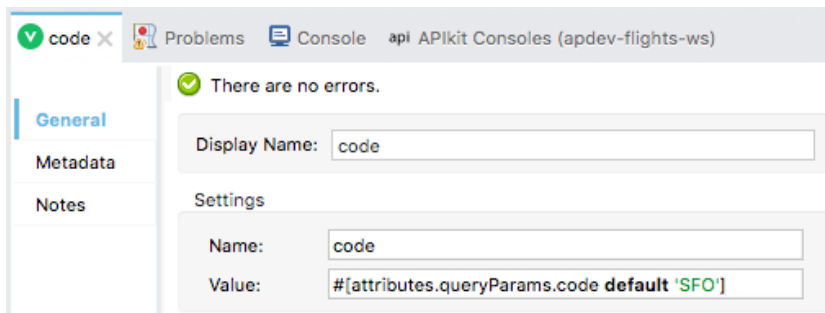
Create a variable to set the destination airport code

46. Return to implementation.xml.
47. Drag a Sub Flow scope from the Mule Palette and drop it at the top of the canvas.
48. Change the name of the flow to setCode.
49. Drag a Set Variable transformer from the Mule Palette and drop it in the setCode subflow.



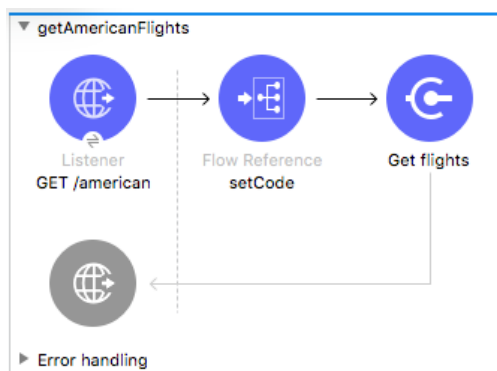
50. In the Set Variable properties view, set the display name and name to code.
51. Set the value to a query parameter called code and give it a default value of SFO if no query parameter is passed to the flow.

```
#[attributes.queryParams.code default 'SFO']
```



Dynamically set a query parameter for the web service call

52. Drag a Flow Reference component from the Mule Palette and drop it after the GET /american Listener in getAmericanFlights.
53. In the Flow Reference properties view, set the flow name to setCode.



54. In the Get flights properties view, set the value of the destination parameter to the value of the variable containing the airport code.

`#[vars.code]`

General

Client id:	<code>\${american.client_id}</code>
Client secret:	<code>\${american.client_secret}</code>
Destination:	<code>#[vars.code]</code>

Test the application

55. Save the file to redeploy the application.
56. In Advanced REST Client, send the same request; this time you should only get flights to SFO.

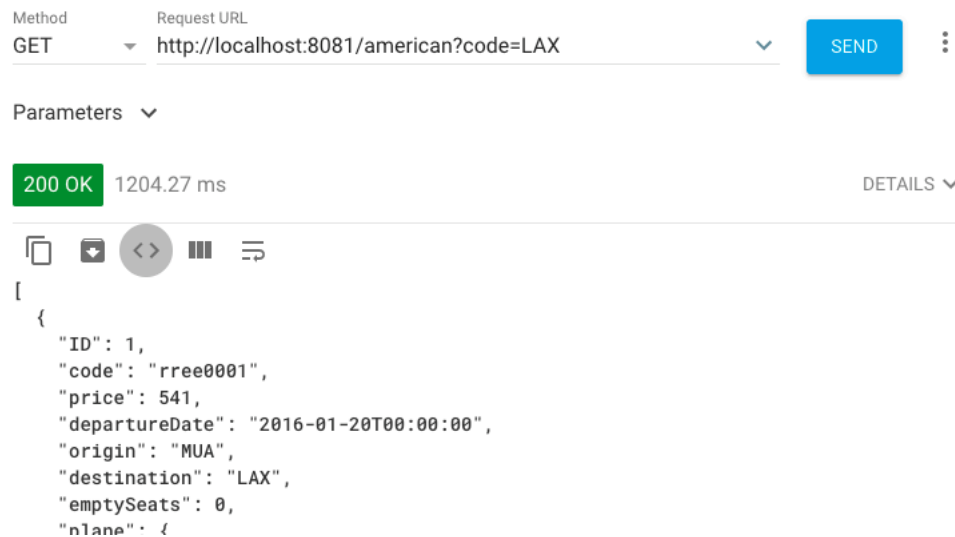
Method GET Request URL `http://localhost:8081/american` SEND

Parameters

200 OK 1743.09 ms DETAILS

```
{
  {
    "ID": 5,
    "code": "rree1093",
    "price": 142,
    "departureDate": "2016-02-11T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 1,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 7,
    "code": "eefd1994",
    "price": 676,
    "departureDate": "2016-01-01T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 0
  }
}
```

57. Add query parameter called code equal to LAX and make the request; you should now only see flights to LAX.



58. Examine the data structure of the JSON response.

Note: You will transform the data to the format specified by the mua-flights-api in a later walkthrough.