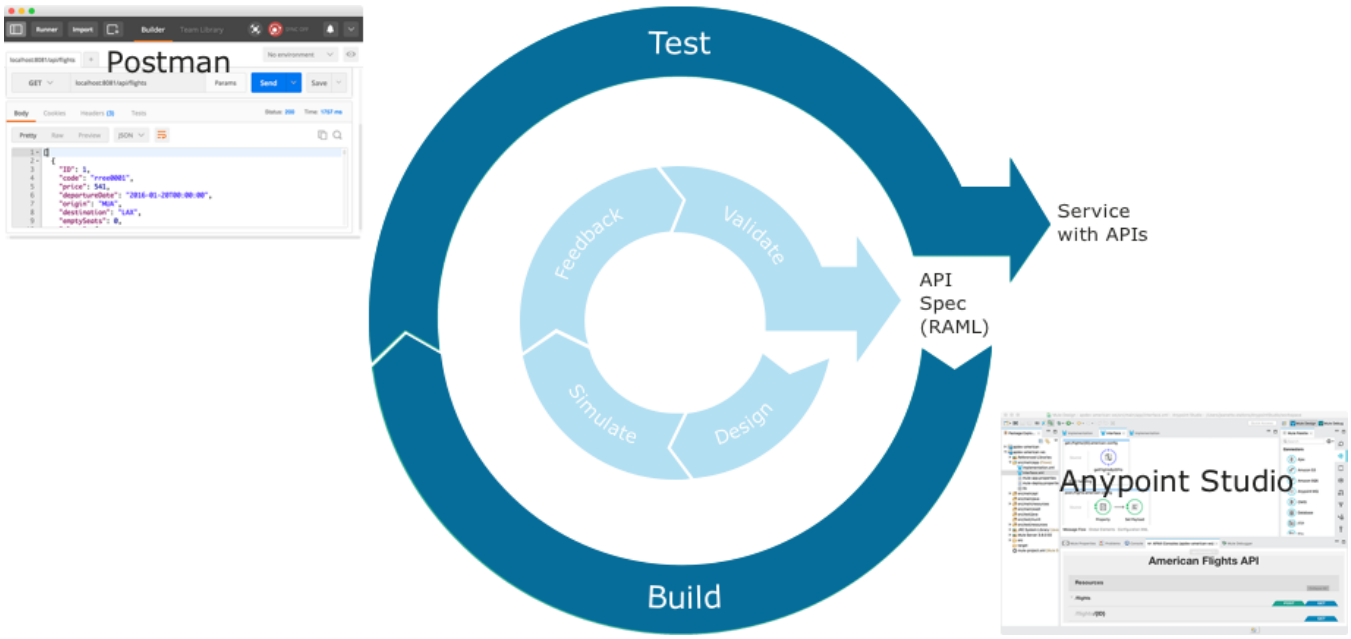
Module 3: Building APIs



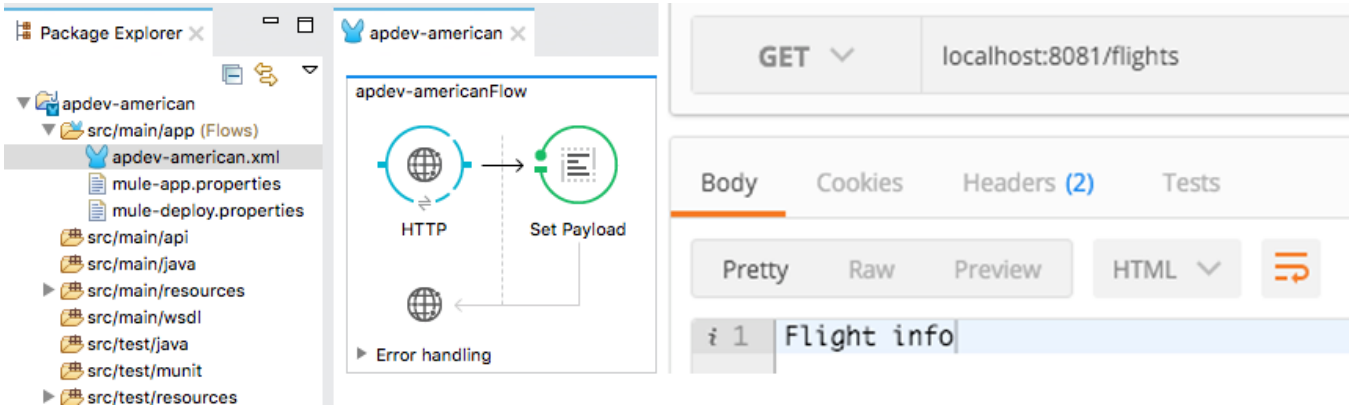
Objectives:

- Introduce Mule applications, flows, messages, and message processors.
- Use Anypoint Studio to create a flow graphically.
- Build, run, and test a Mule application.
- Use a connector to connect to a database.
- Use the graphical DataWeave editor to transform data.
- Create a RESTful interface for an application from a RAML file.
- Connect an API interface to the implementation.

Walkthrough 3-1: Create a Mule application with Anypoint Studio

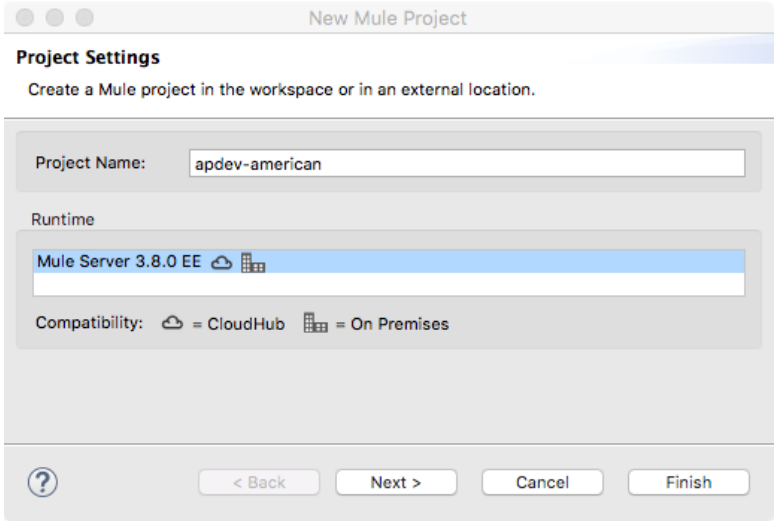
In this walkthrough, you build a Mule application. You will:

- Create a new Mule project with Anypoint Studio.
- Add a connector to receive requests at an endpoint.
- Set the message payload.
- Run a Mule application using the embedded Mule runtime.
- Make an HTTP request to the endpoint using Postman.



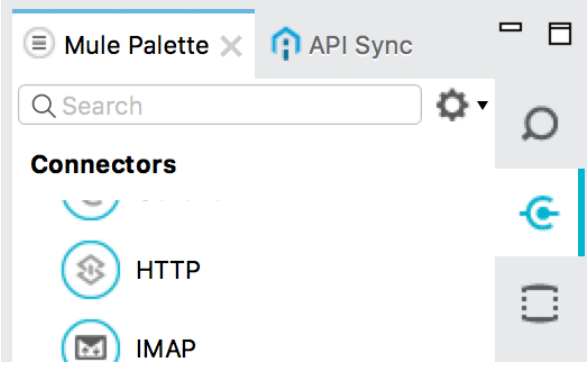
Create a Mule project

1. Open Anypoint Studio.
2. Select File > New > Mule Project.
3. Set the Project Name to apdev-american.
4. Ensure the Runtime is set to the latest version of Mule.
5. Click Finish.

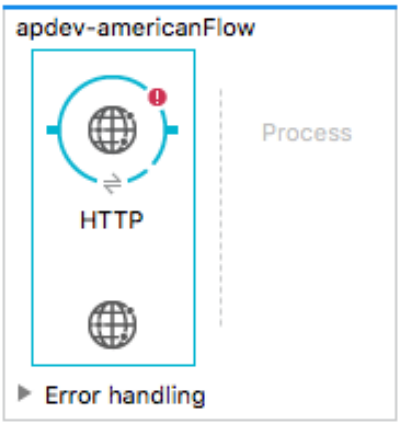


Create an HTTP connector endpoint to receive requests

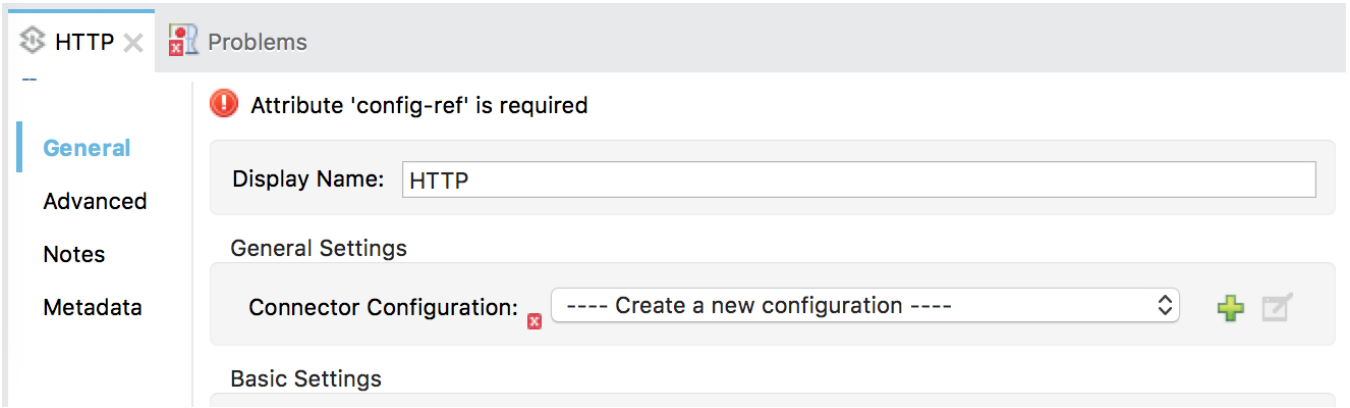
6. In the Mule Palette, select the Connectors tab.



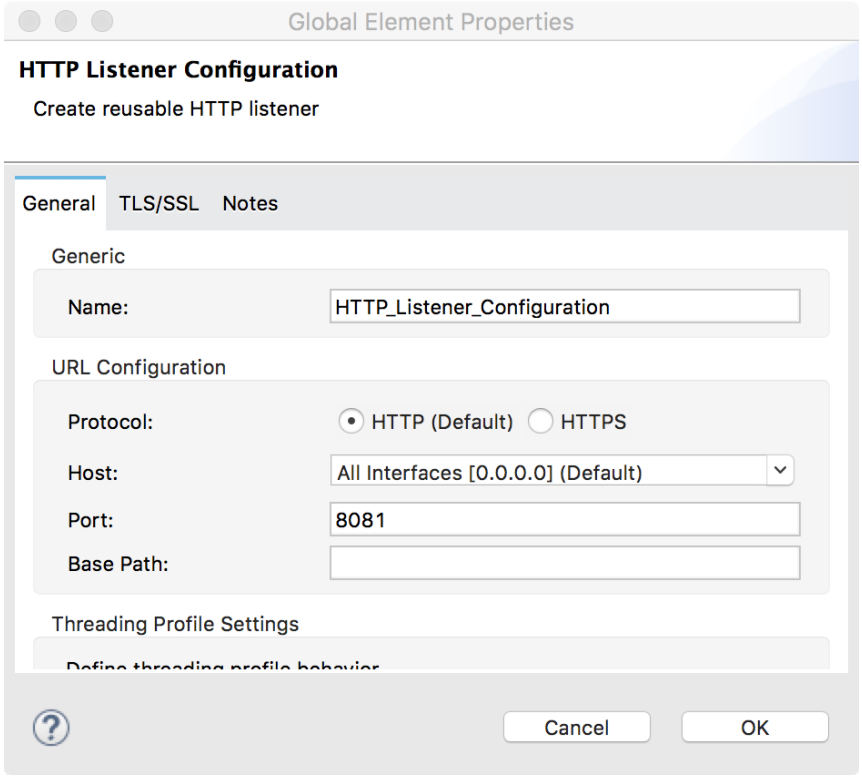
7. Drag an HTTP connector from the Mule Palette to the canvas.



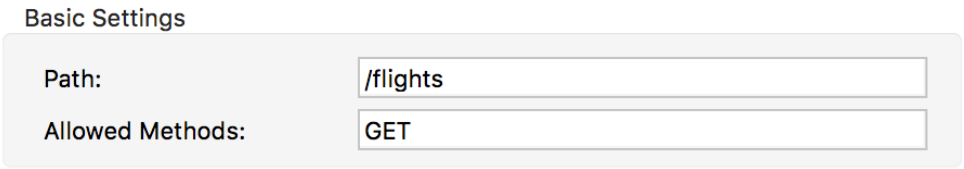
8. In the HTTP properties view that opens at the bottom of the window, click the Add button next to connector configuration.



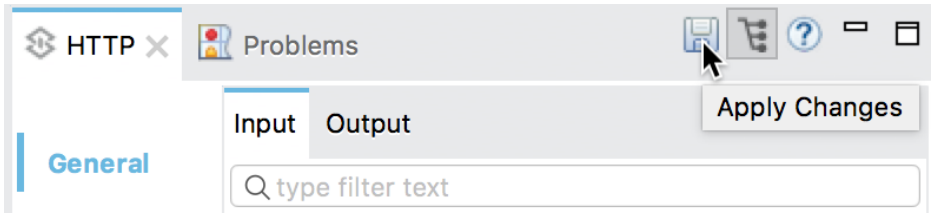
9. In the Global Element Properties dialog box, look at the default values and click OK.



10. In the HTTP properties view, set the path to /flights.
11. Set the allowed methods to GET.

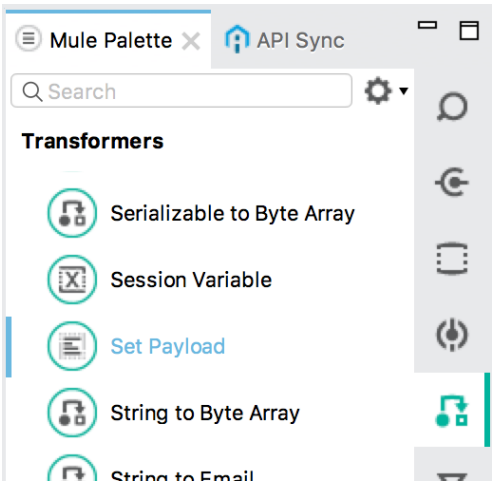


12. Click the Apply Changes button; the errors in the Problems view should disappear.

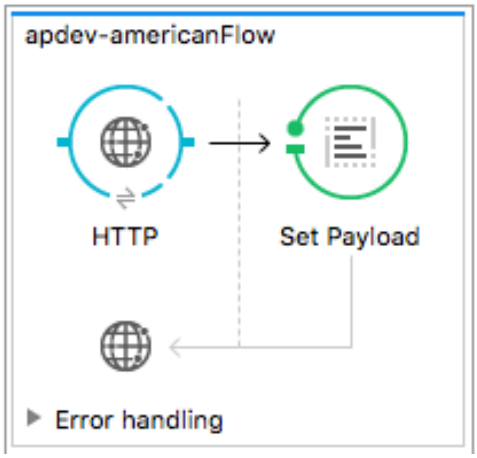


Display data

13. In the Mule Palette, select the Transformers tab.

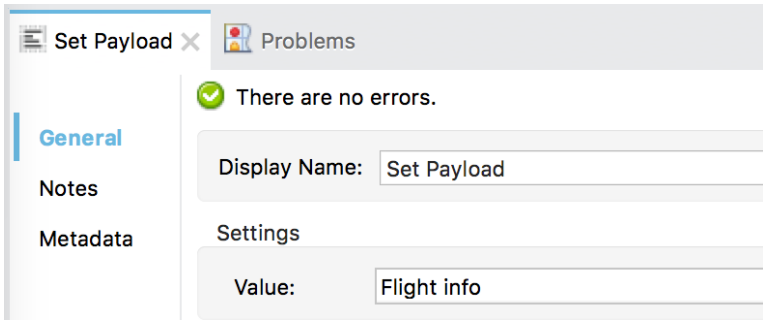


14. Drag a Set Payload transformer from the Mule Palette into the process section of the flow.

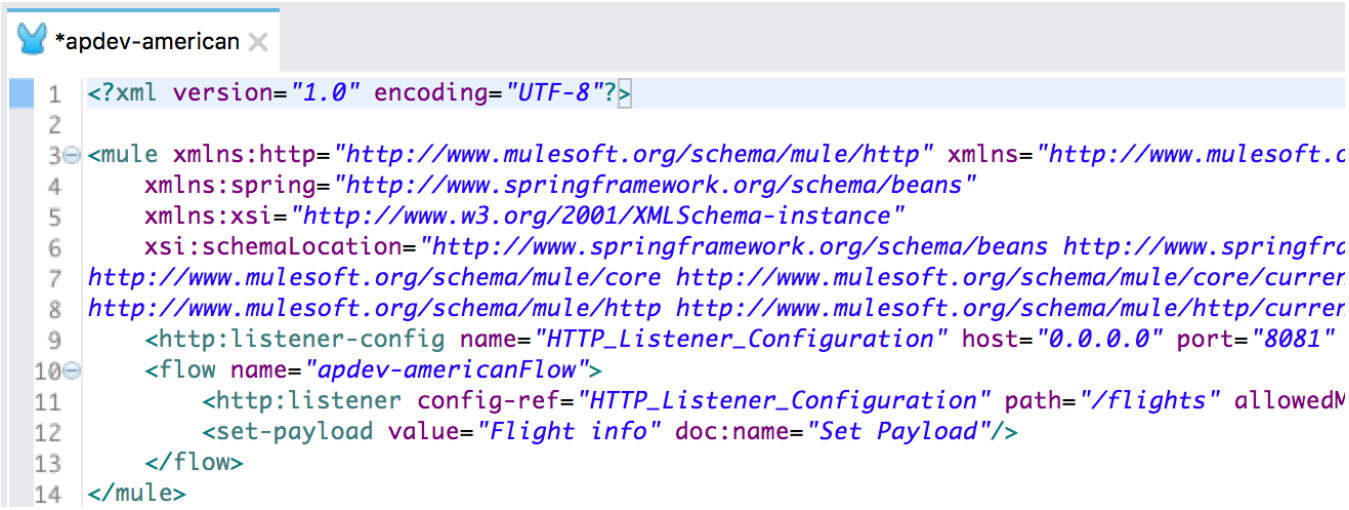


Configure the Set Payload transformer

15. In the Set Payload properties view, set the value field to Flight info.



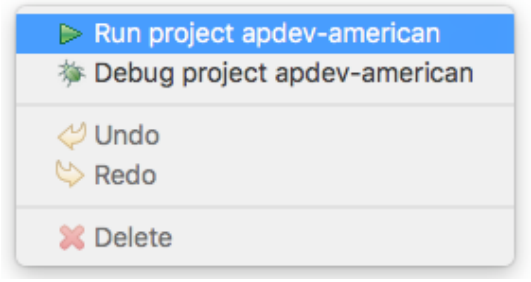
16. Click the Configuration XML link at the bottom of the canvas and examine the corresponding XML.



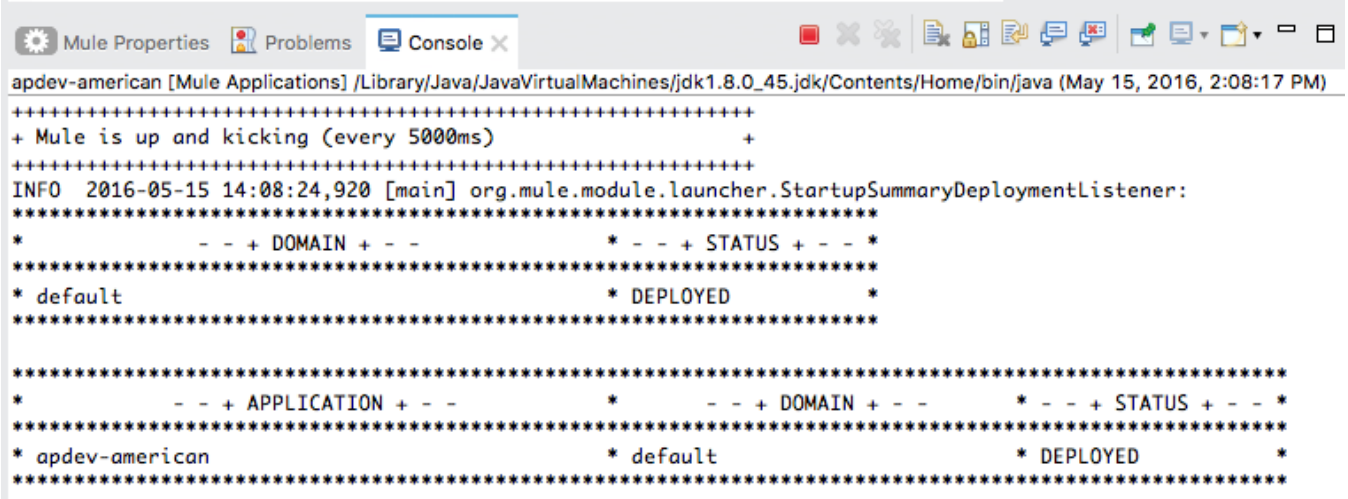
17. Click the Message Flow link to return to the canvas.
18. Click the Save button or press Cmd+S or Ctrl+S.

Run the application

19. Right-click in the canvas and select Run project apdev-american.

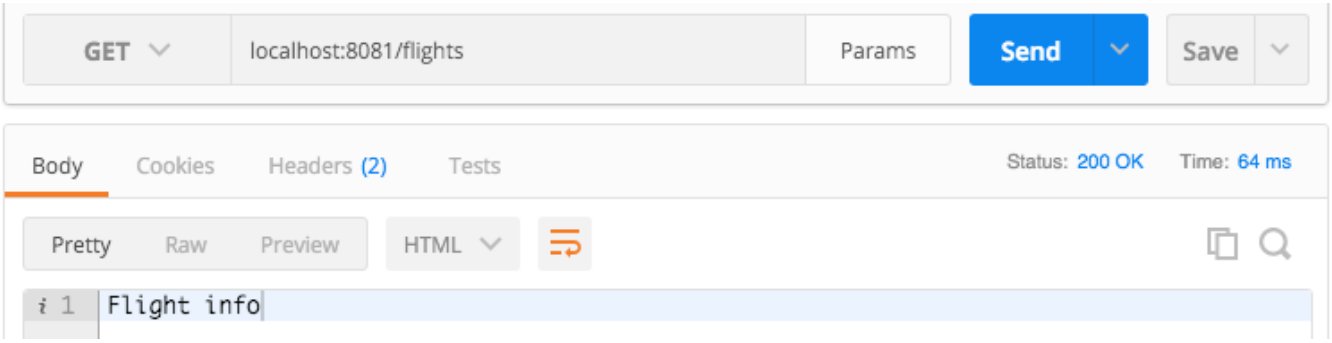


20. Watch the Console view; it should display information letting you know that both the Mule runtime and the apdev-american application started.

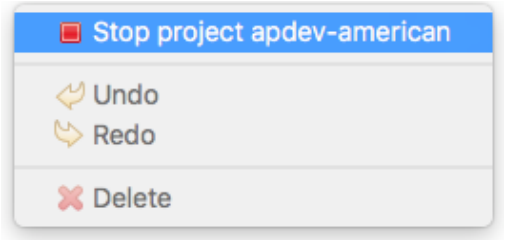


Test the application

21. Return to Postman.
22. Make sure the method is set to GET and that no headers or body are set for the request.
23. Make a GET request to http://localhost:8081/flights; you should see Flight info displayed.



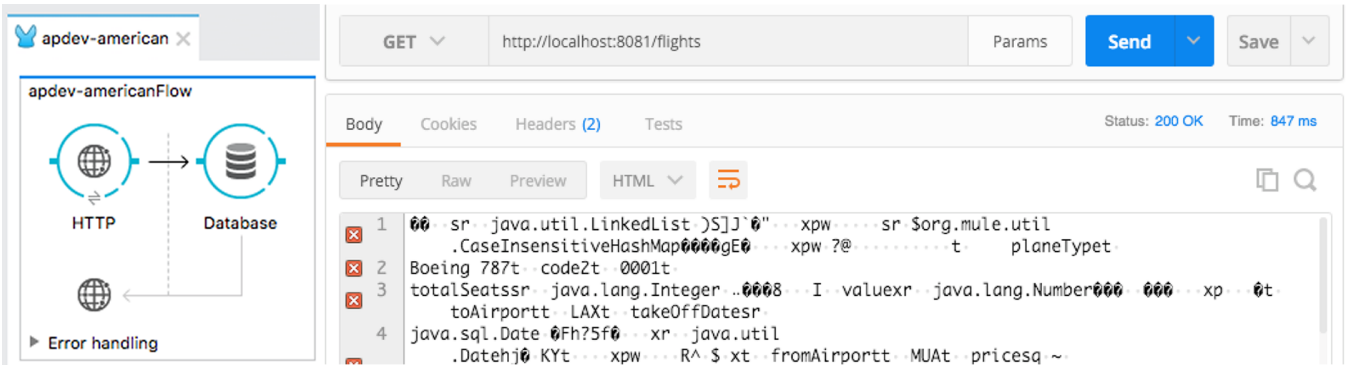
24. Return to Anypoint Studio.
25. Right-click in the canvas and select Stop project apdev-american.



Walkthrough 3-2: Connect to data (MySQL database)

In this walkthrough, you connect to a database and retrieve data from a table that contains flight information. You will:

- Add a Database connector endpoint.
- Configure a Database connector that connects to a MySQL database (or optionally an in-memory Derby database if you do not have access to port 3306).
- Configure the Database endpoint to use that Database connector.
- Write a query to select data from a table in the database.



Locate database information

1. Return to the course snippets.txt file and locate the MySQL database information.

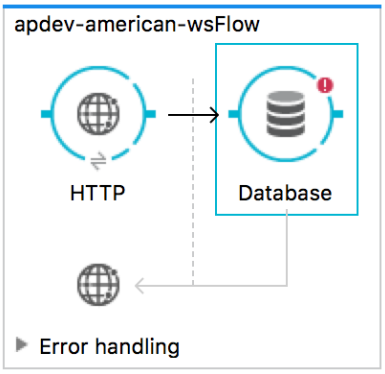
```
64    * MODULE 3 ****************************************
65
66    * MySQL database
67    db.host = mudb.mulesoft-training.com
68    db.port = 3306
69    db.user = mule
70    db.password = mule
71    db.database = training
72    American table: american
```

Note: The database information you see may be different than what is shown here; the values in the snippets file differ for instructor-led and self-paced training classes.

Add a Database connector endpoint

2. Return to Anypoint Studio.
3. Right-click the Set Payload message processor and select Delete.
4. In the Mule Palette, select the Connectors tab.

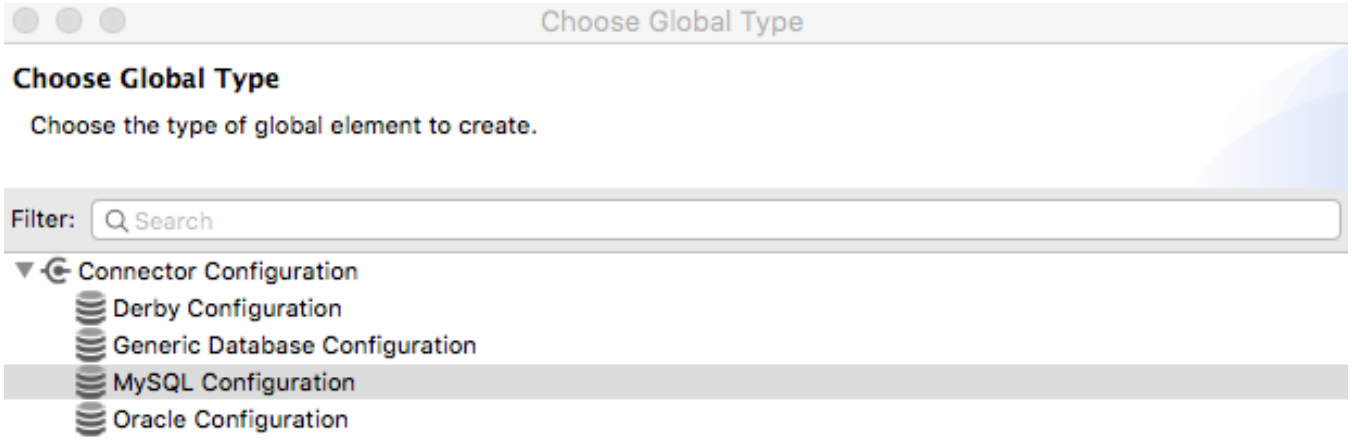
5. Drag a Database connector to the process section of the flow.



6. Double-click the Database endpoint.

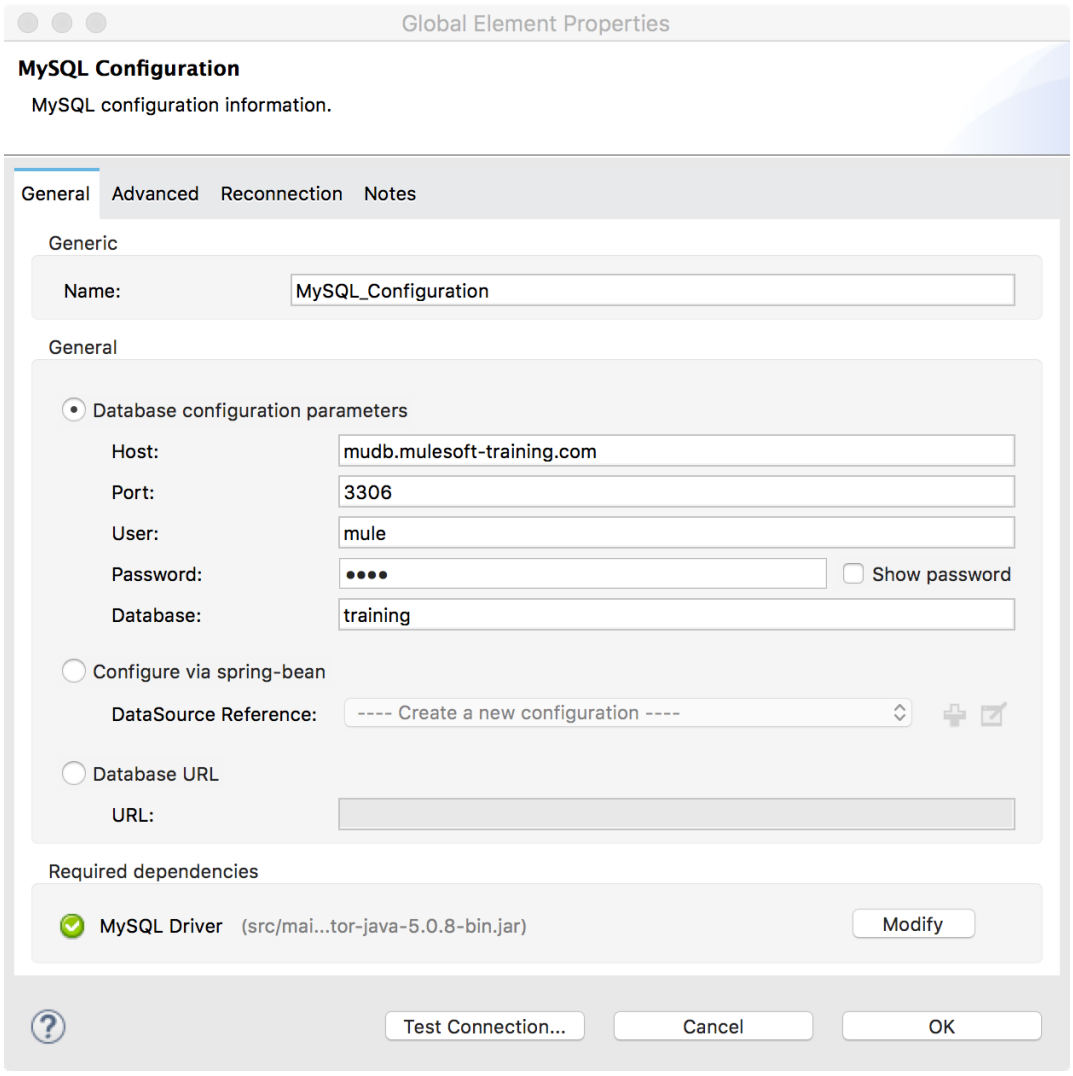
Option 1: Configure a MySQL Database connector (if you have access to port 3306)

7. In the Database properties view, click the Add button next to connector configuration.
8. In the Choose Global Type dialog box, select Connector Configuration > MySQL Configuration and click OK.



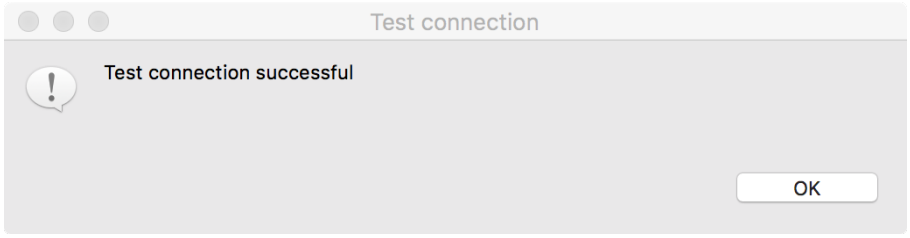
9. In the Global Element Properties dialog box, set the server, port, user, password, and database values to the values listed in the course snippets.txt file.
10. Under Required dependencies, click the Add File button next to MySQL Driver.

11. Navigate to the student files folder, select the MySQL JAR file located in the resources folder, and click Open.



12. Back in the Global Element Properties dialog box, click the Test Connection button; you should get a successful test dialog box.

    Note: Make sure the connection succeeds before proceeding.



    Note: If the connectivity test fails, make sure you are not behind a firewall restricting access to port 3306. If you cannot access port 3306, use the instructions in the next section for option 2.

13. Click OK to close the dialog box.
14. Click OK to close the Global Element Properties dialog box.

Option 2: Configure a Derby Database connector (if no access to port 3306)

15. In a command-line interface, use the cd command to navigate to the folder containing the jars folder of the student files.
16. Run the mulesoft-training-services.jar file.

    ```
    java –jar mulesoft-training-services-X.X.X.jar
    ```

    Note: Replace X.X.X with the version of the JAR file, for example 1.3.0.

    Note: The application uses ports 1527, 9090, 9091, and 61616. If any of these ports are already in use, you can change them when you start the application as shown in the following code.

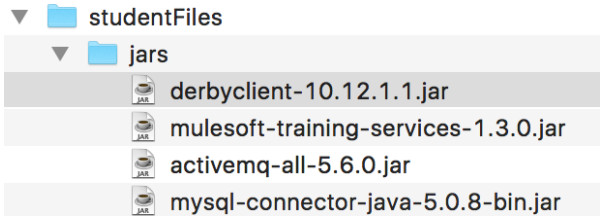
    ```
    java -jar mulesoft-training-services-X.X.X.jar --database.port=1530 --
    ws.port=9092 --spring.activemq.broker-url=tcp://localhost:61617 --
    server.port=9193
    ```

17. Look at the output and make sure all the services started.

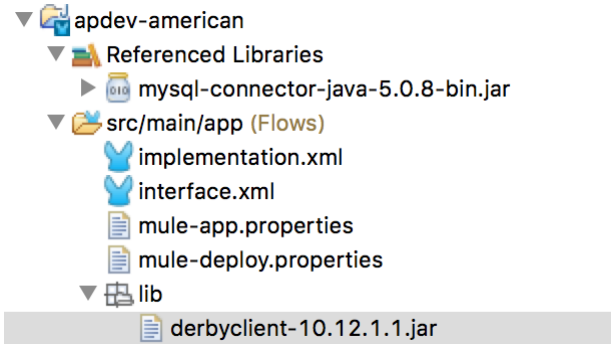
    ```
    1. java
    [MULESOFT TRAINING SERVICES]

    Starting resources:
    - American flights database started
    - American flights database ready
    - Delta flights web service started
    - Message Broker started
    - Banking REST API published
    - JMS API published
    - United flights web service started

    +-------------------------------------------------------------------------------------+
    | MuleSoft training services - Press CTRL-C to terminate this application             |
    |-------------------------------------------------------------------------------------|
    | Welcome page          : http://localhost:9090                                       |
    | Derby database URL    : jdbc:derby://localhost:1527/memory:training                 |
    | JMS broker URL        : tcp://localhost:61616                                       |
    | JMS topic name        : apessentials                                                |
    | United REST service   : http://localhost:9090/essentials/united/flights/{destination} |
    |                         NB: use http://0.0.0.0:9090/essentials/united in Mule apps  |
    | Delta SOAP service    : http://localhost:9191/essentials/delta                      |
    | Delta SOAP WSDL       : http://localhost:9191/essentials/delta?wsdl                 |
    | Delta WS operations   : listAllFlights, findFlights                                 |
    | Banking API           : http://localhost:9090/api/...                               |
    | Banking API RAML      : http://localhost:9090/api/banking-service.raml              |
    | Accounts form         : http://localhost:9090/essentials/accounts/show.html         |
    +-------------------------------------------------------------------------------------+
    ```

    Note: When you want to stop the application, return to this window and press Ctrl+C.

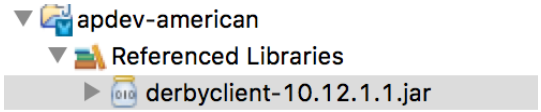
18. In the computer's file explorer, return to the student files folder and locate the derbyclient.jar file in the jars folder.



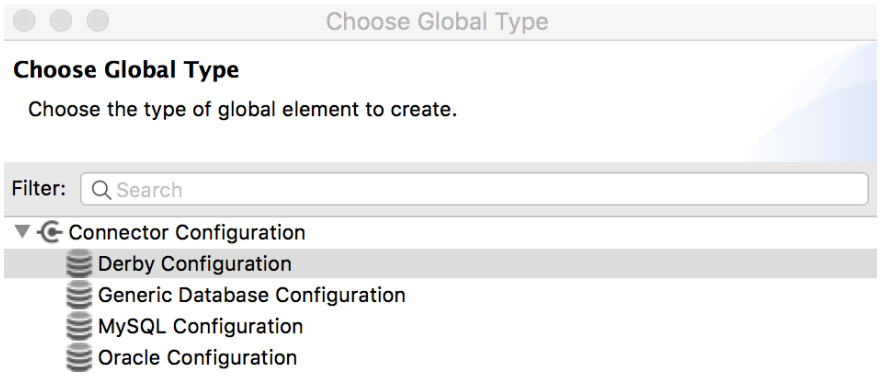
19. Copy and paste or drag this JAR file into the src/main/app/lib folder of the project in Anypoint Studio.



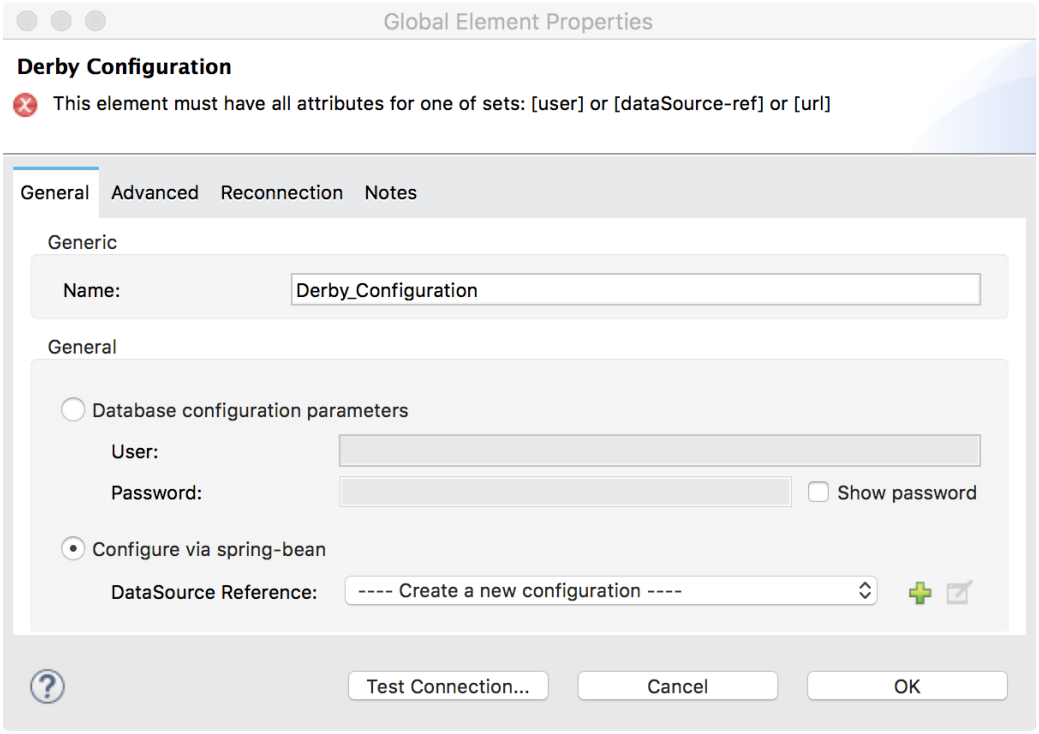
20. Right-click the JAR file and select Build Path > Add to Build Path; you should now see the JAR file in the project's Referenced Libraries.



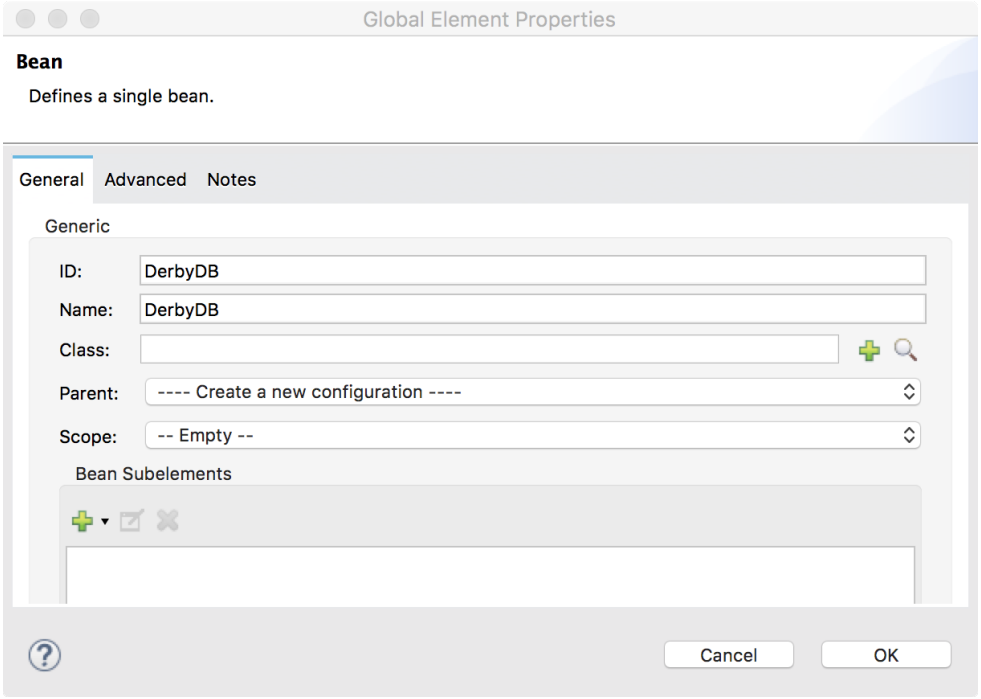
21. Double-click the Database endpoint in the canvas.
22. In the Database properties view, click the Add button next to connector configuration.
23. In the Choose Global Type dialog box, select Connector Configuration > Derby Configuration and click OK.



24. In the Global Element Properties dialog box, select Configure via spring-bean.
25. Click the Add button next to DataSource Reference.

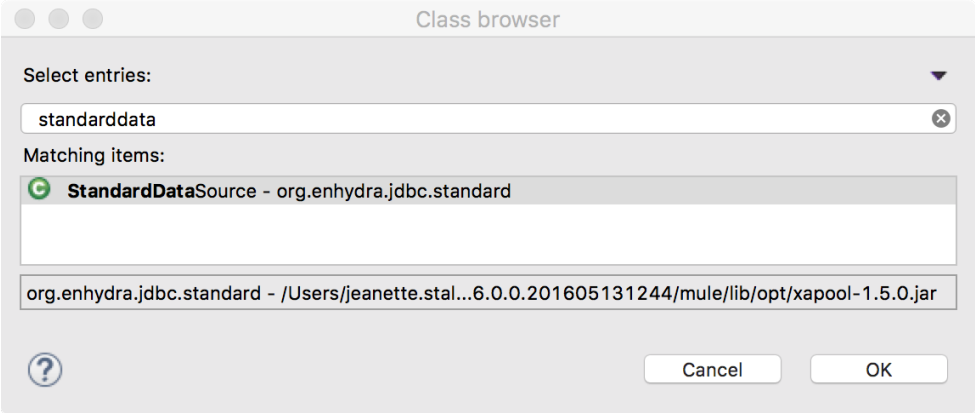


26. On the Bean page of the Global Element Properties dialog box, set the ID and name to DerbyDB.



27. Click the Browse for Java class button next to class.

28. In the Class browser dialog box, start typing StandardDataSource and select the matching item for StandardDataSource – org.enhydra.jdbc.standard.



29. Click OK.
30. On the Bean page of the Global Element Properties dialog box, click the Add button under Bean Subelements and select Add Property.
31. In the Property dialog box, set the following values:

    - Name: driverName
    - Value: org.apache.derby.jdbc.ClientDriver

    Note: You can copy this value from the course snippets.txt file.

32. Click Finish.
33. Click the Add button under Bean Subelements again and select Add Property.
34. In the Property dialog box, set the following values:

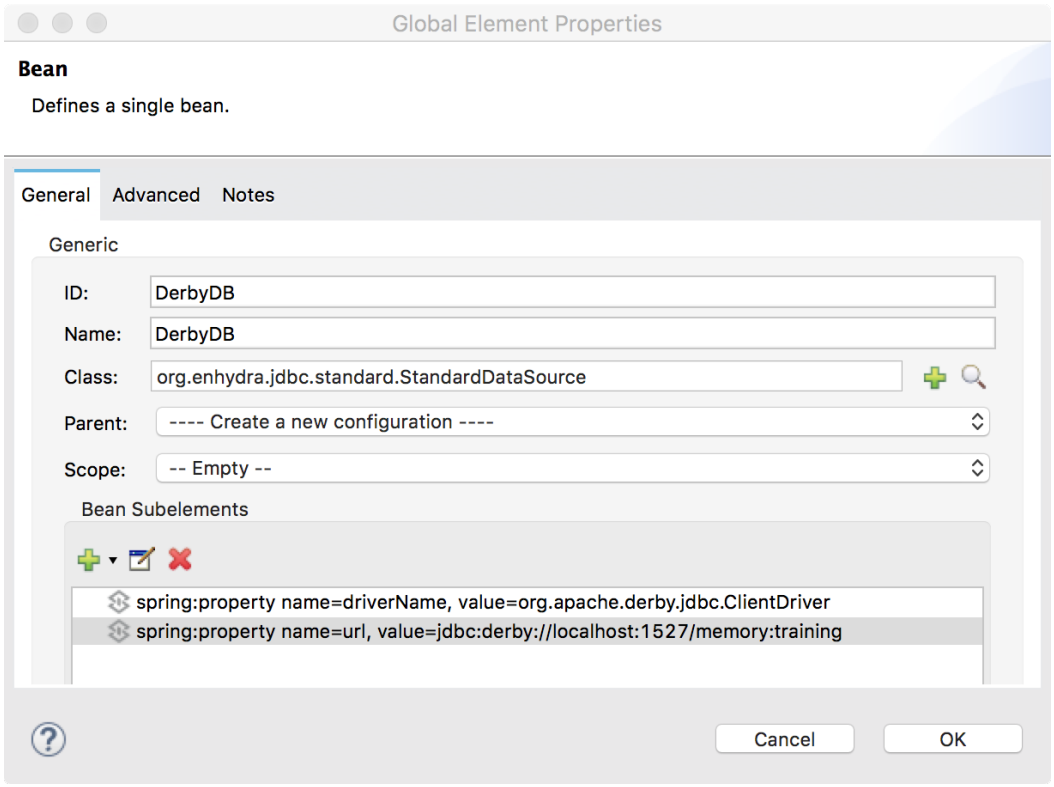
    - Name: url
    - Value: jdbc:derby://localhost:1527/memory:training

    Note: You can copy this value from the course snippets.txt file.

    Note: If you changed the database port when you started the mulesoft-training-services application, be sure to use the new value here.

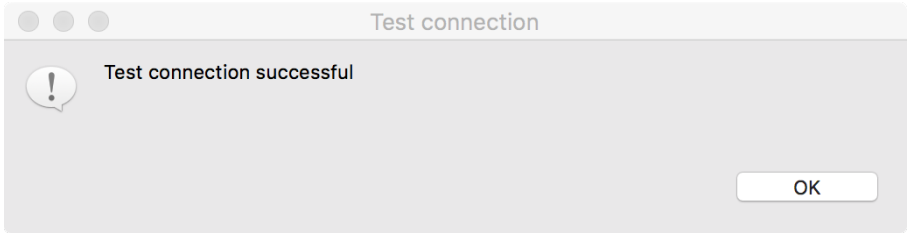
35. Click Finish.

36. On the Bean page of the Global Element Properties dialog box, click OK.



37. On the Derby Configuration page of the Global Element Properties dialog box, click Test Connection; you should get a successful test dialog box.

    Note: Make sure the connection succeeds before proceeding.

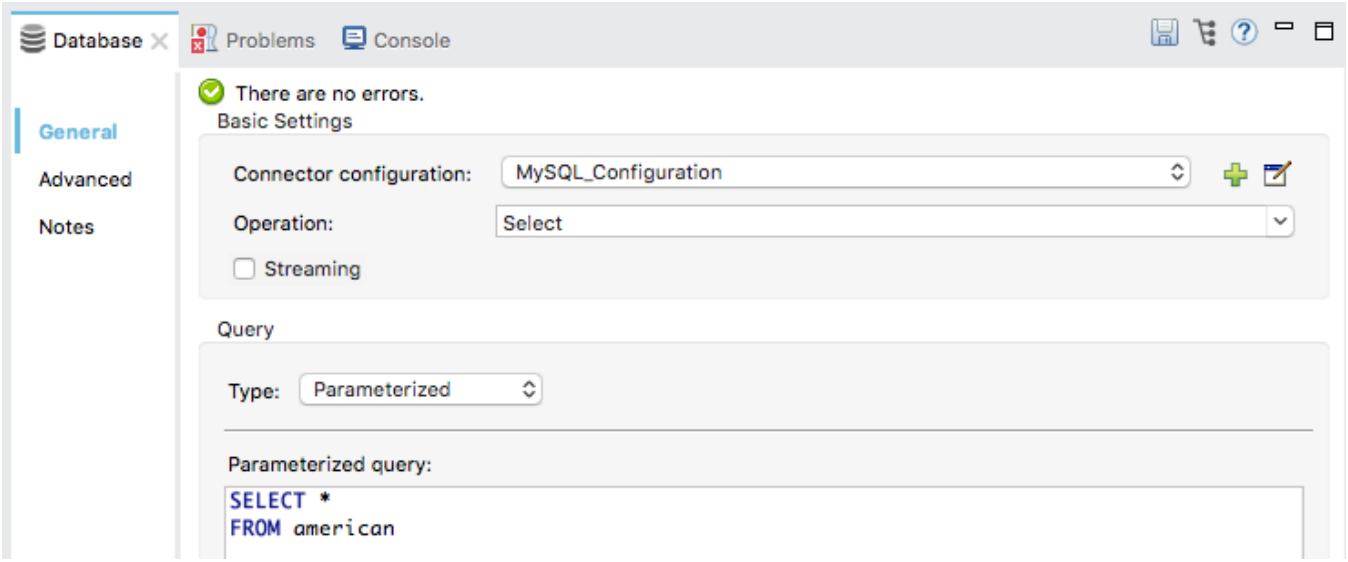


38. Click OK to close the dialog box.
39. Click OK to close the Global Element Properties dialog box.

Write a query to return all flights

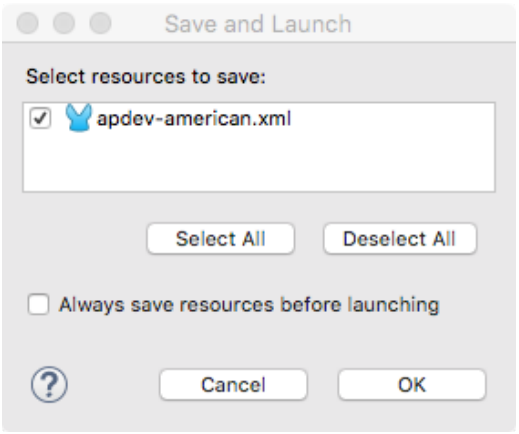
40. In the Database properties view, set the operation to Select.
41. Add a query to select all records from the american table.

```
SELECT *
FROM american
```

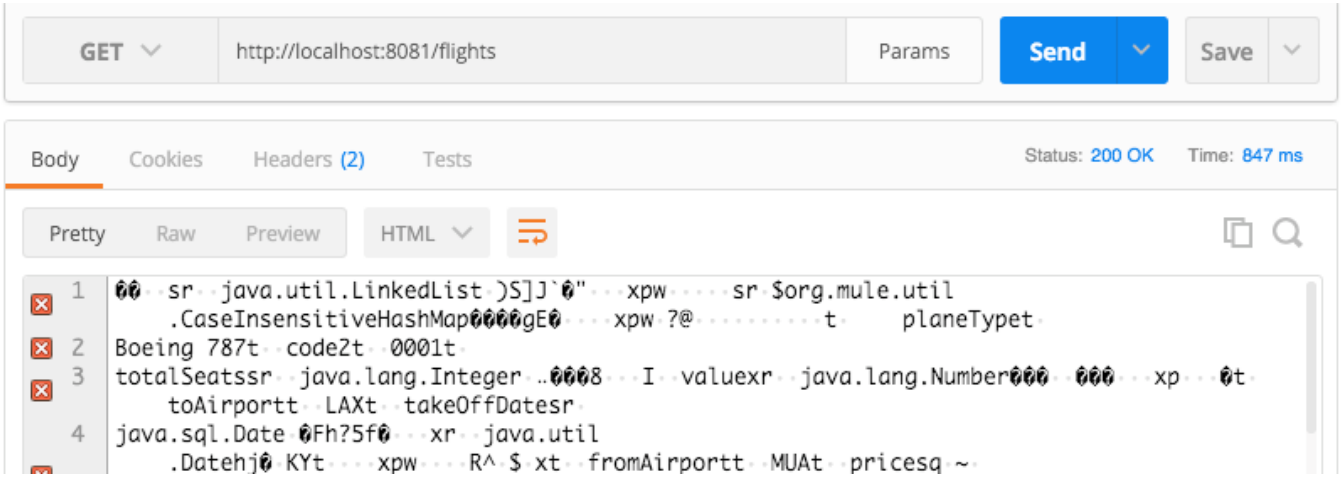


Test the application

42. Run the project.
43. In the Save and launch dialog box, select Always save resources before launching and click OK.



44. Watch the console, and wait for the application to start.
45. Once it has started, return to Postman.
46. In Postman, make another request to http://localhost:8081/flights; you should get some garbled plain text displayed – the tool's best representation of Java objects.

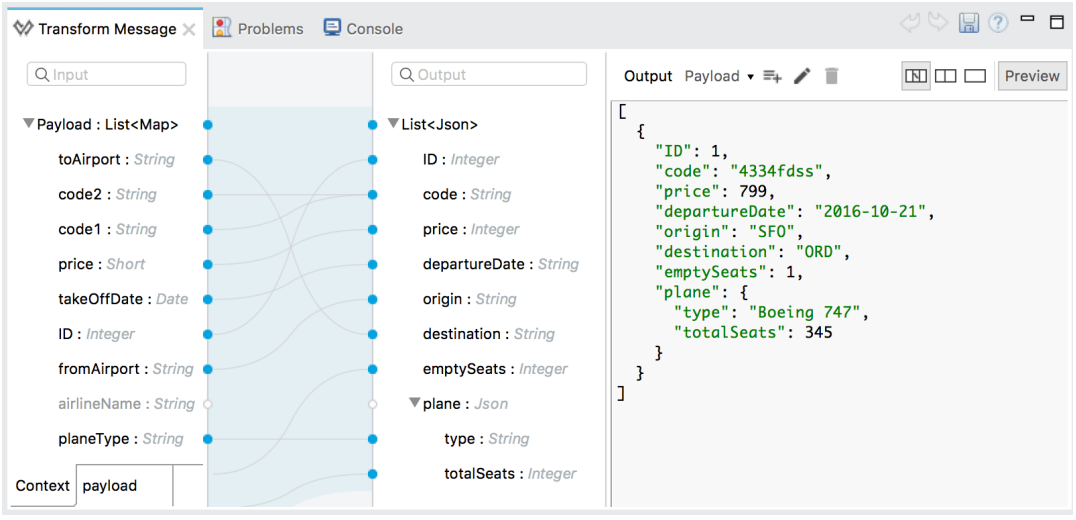


47. Return to Anypoint Studio.
48. Stop the project.

Walkthrough 3-3: Transform data

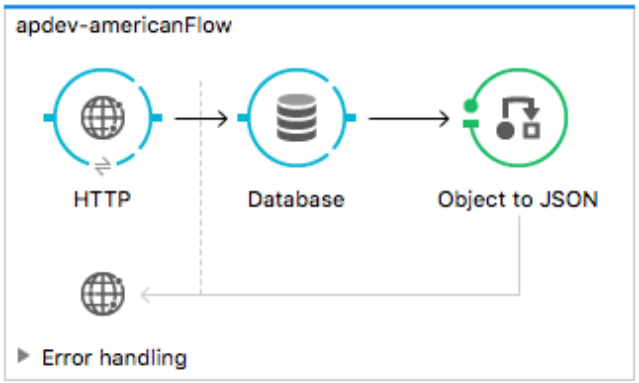
In this walkthrough, you transform and display the account data into JSON. You will:

- Use the Object to JSON transformer.
- Replace it with a Transform Message component.
- Use the DataWeave graphical editor to change the response to a different JSON structure.



Add an Object to JSON transformer

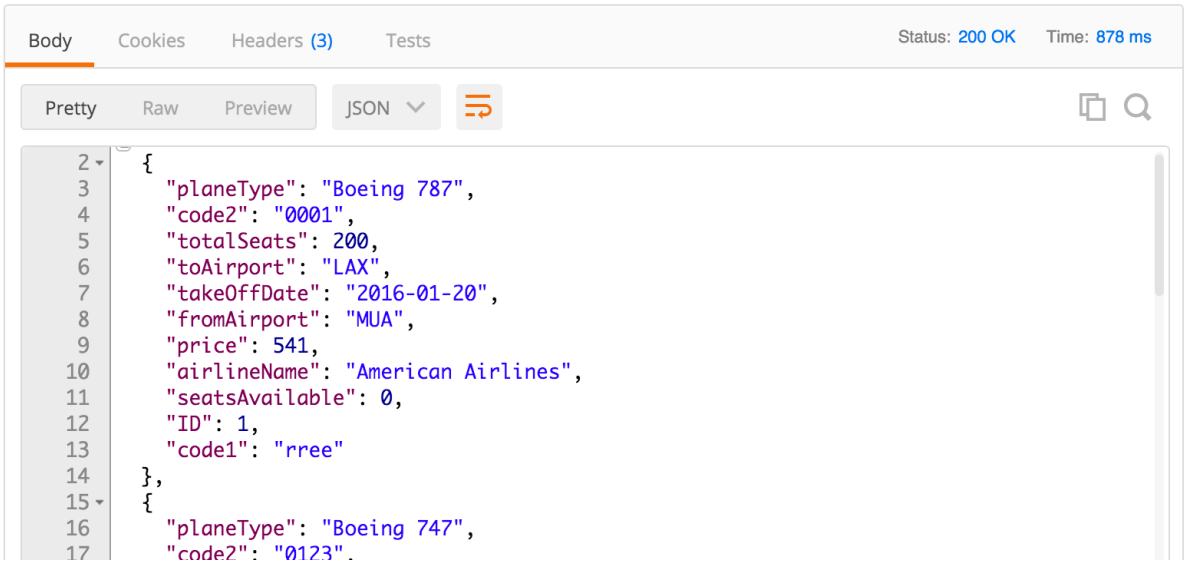
1. In the Mule Palette, select the Transformers tab.
2. Drag an Object to JSON transformer from the Mule Palette and drop it after the Database endpoint.



Test the application

3. Run the project.

4. In Postman, send the same request; you should see the American flight data represented as JSON.



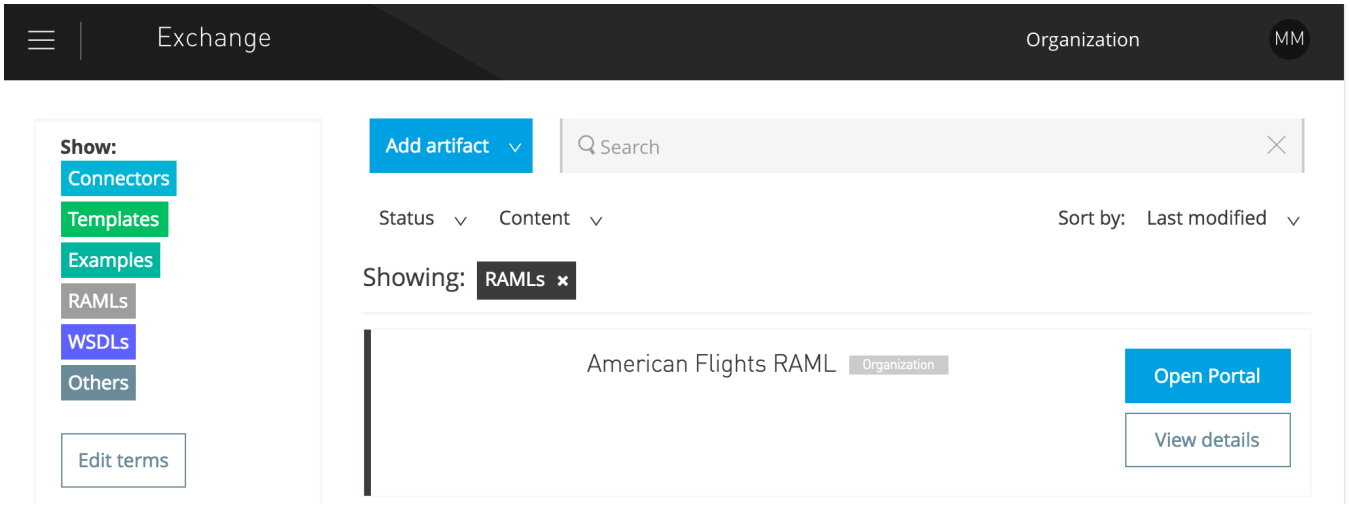
Note: If you are using the local Derby database, the properties will be uppercase instead.

Review the data structure to be returned by the American flights API

5. Return to your Anypoint Exchange open in a web browser.

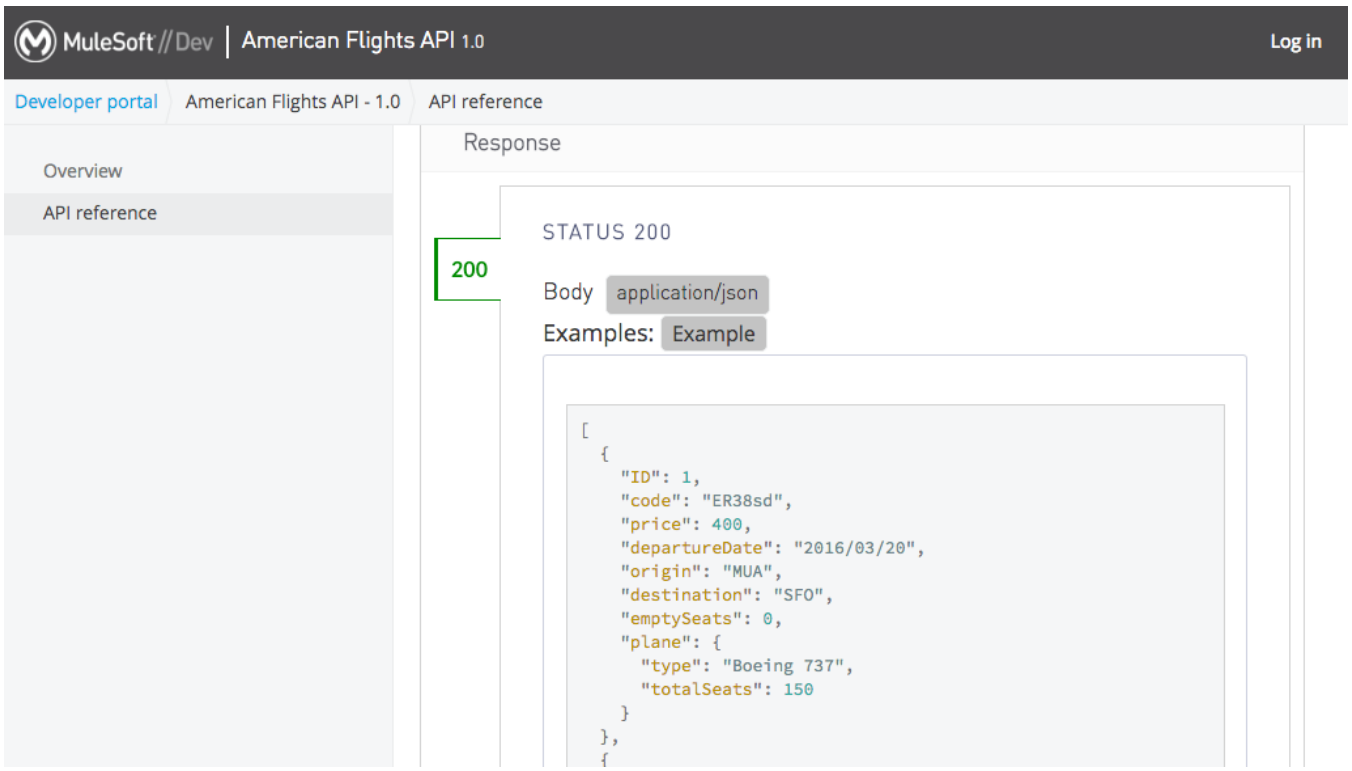
   Note: If you closed it, return to http://anypoint.mulesoft.com and from the main menu, select Exchange.

6. Locate your American Flights RAML and click Open Portal.



7. Click the API reference link.

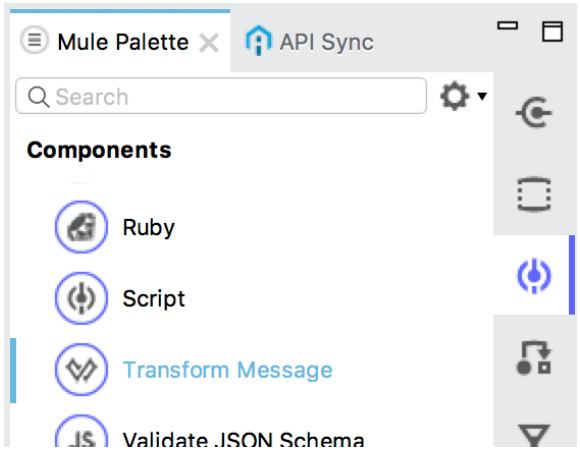
8. Look at the example data returned for the /flights get method.



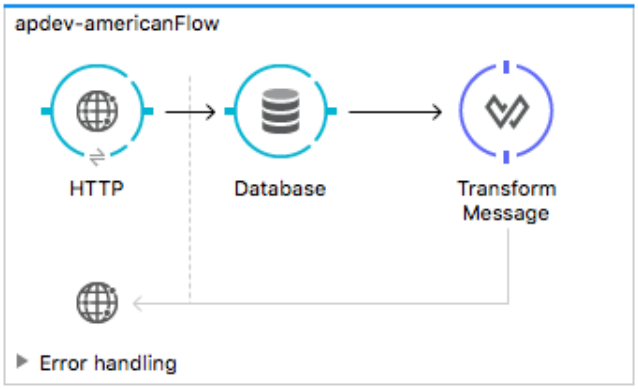
9. Notice that the structure of the JSON being returned by the Mule application does not match this JSON.

Add a Transform Message component

10. Return to Anypoint Studio and stop the project.
11. Right-click the Object to JSON transformer and select Delete.
12. In the Mule Palette, select the Components tab.

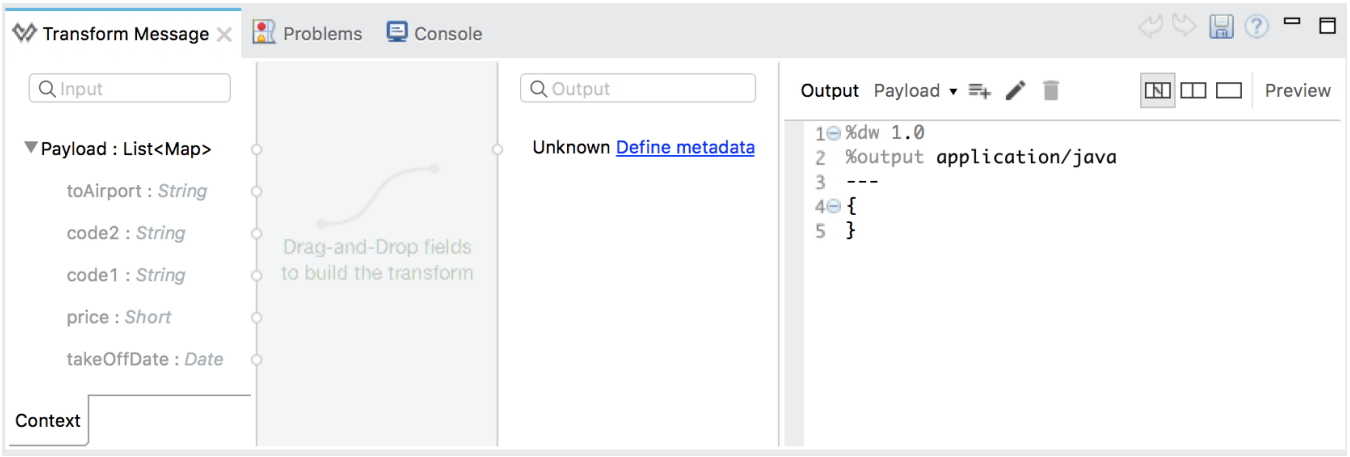


13. Drag a Transform Message component from the Mule Palette and drop it after the Database endpoint.



Review metadata for the transformation input

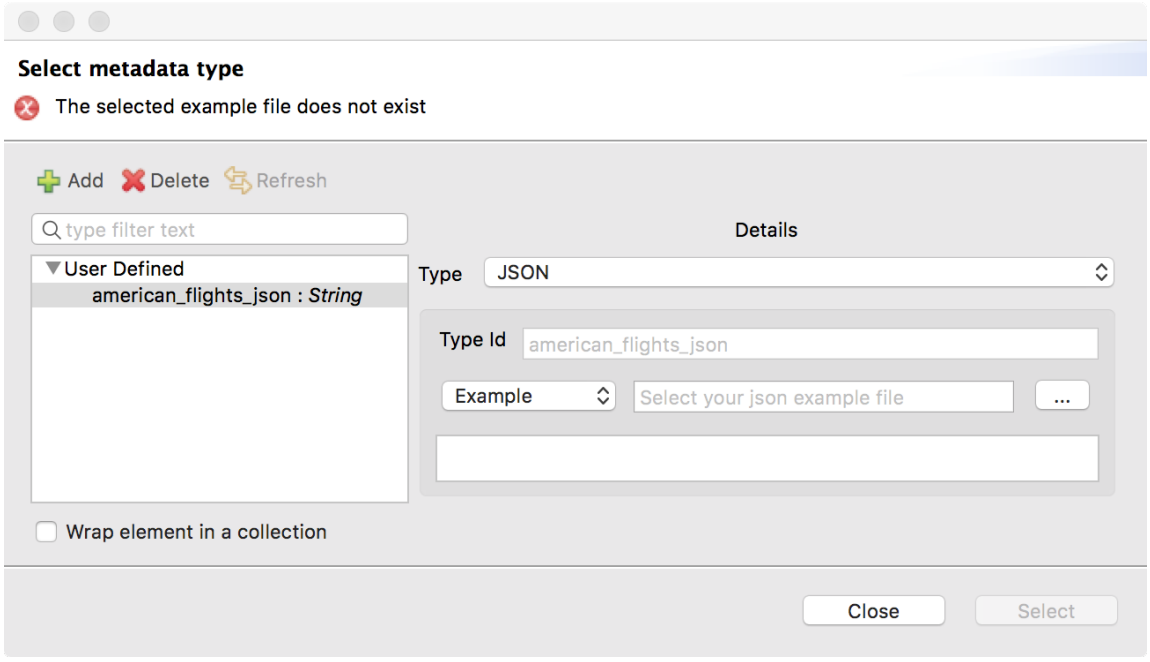
14. Double-click the Transform Message component in the canvas.
15. In the Transform Message properties view, look at the input section and review the payload metadata; it should match the data returned by the Database endpoint.



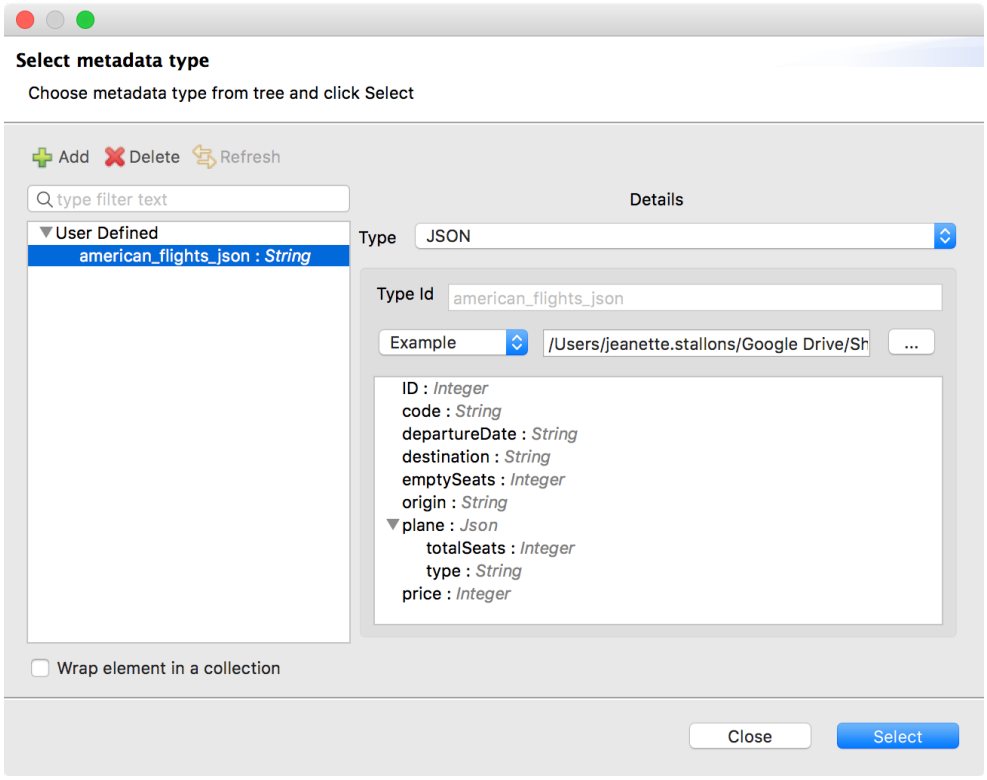
Add metadata for the transformation output

16. Click the Define metadata link in the output section.
17. In the Select metadata type dialog box, click the Add button.
18. In the Create new type dialog box, set the type id to american_flights_json.
19. Click Create type.

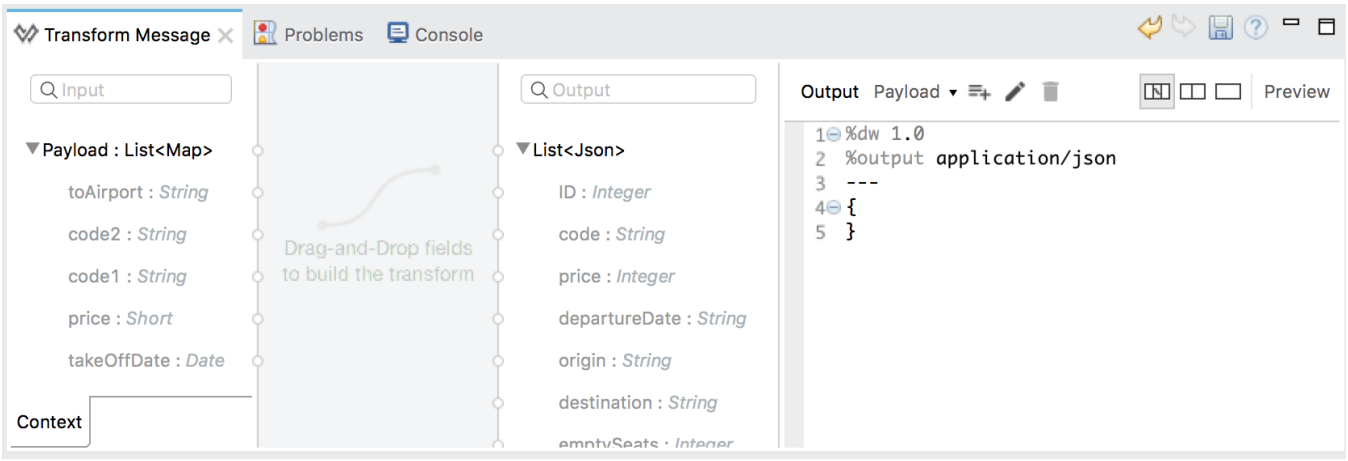
20. Back in the Set metadata type dialog box, set the type to JSON.
21. Change the Schema selection to Example.



22. Click the browse button and navigate to the course student files.
23. Select american-flights-example.json in the schemas-and-examples folder and click Open; you should see the example data for the metadata type.



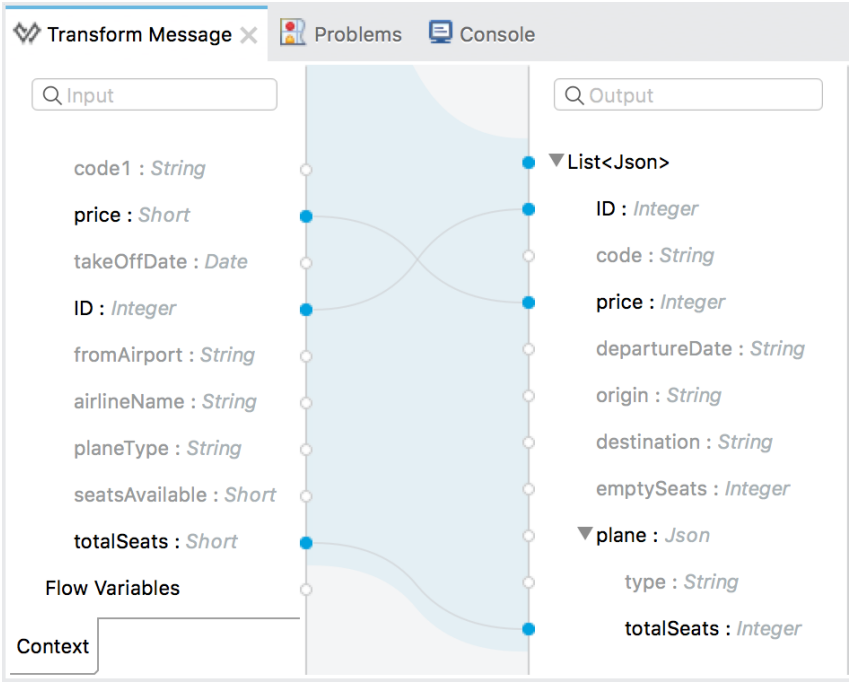
24. Click Select; you should now see output metadata in the output section of the Transform Message properties view.



Create the transformation

25. Map fields with the same names by dragging them from the input section and dropping them on the corresponding field in the output section.

    - ID to ID
    - price to price
    - totalSeats to plane > totalSeats

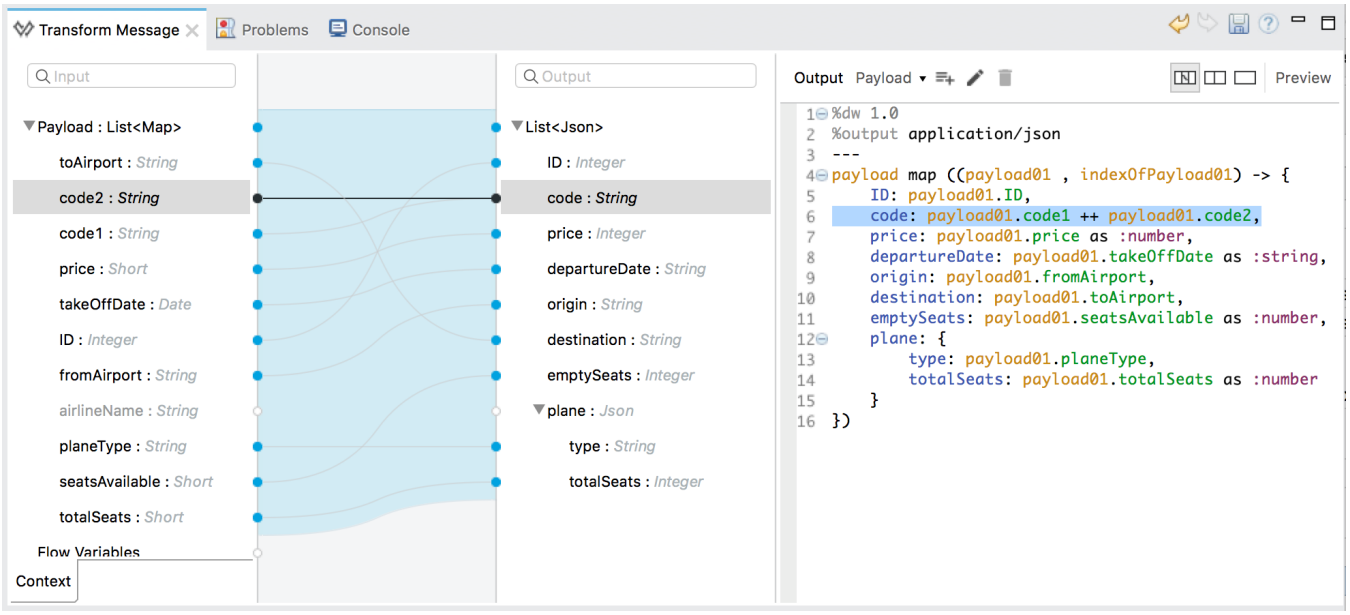


26. Map fields with different names by dragging them from the input section and dropping them on the corresponding field in the output section.

    - toAirport to destination
    - takeOffDate to departureDate
    - fromAirport to origin
    - seatsAvailable to emptySeats
    - planeType to plane > type

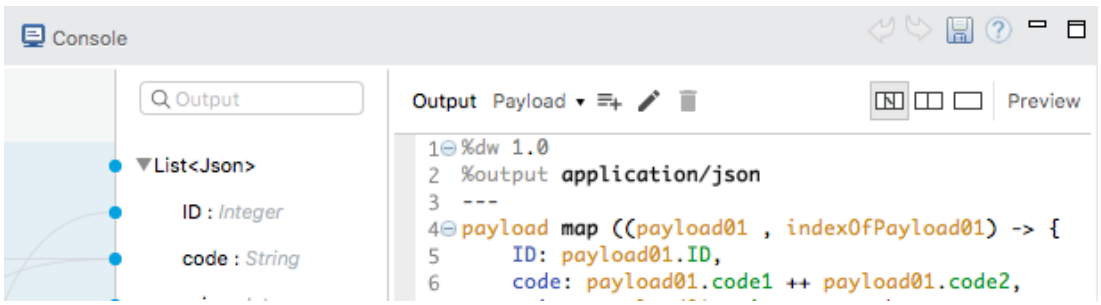
27. Concatenate two fields by dragging them from the input section and dropping them on the same field in the output section.

    - code1 to code
    - code2 to code

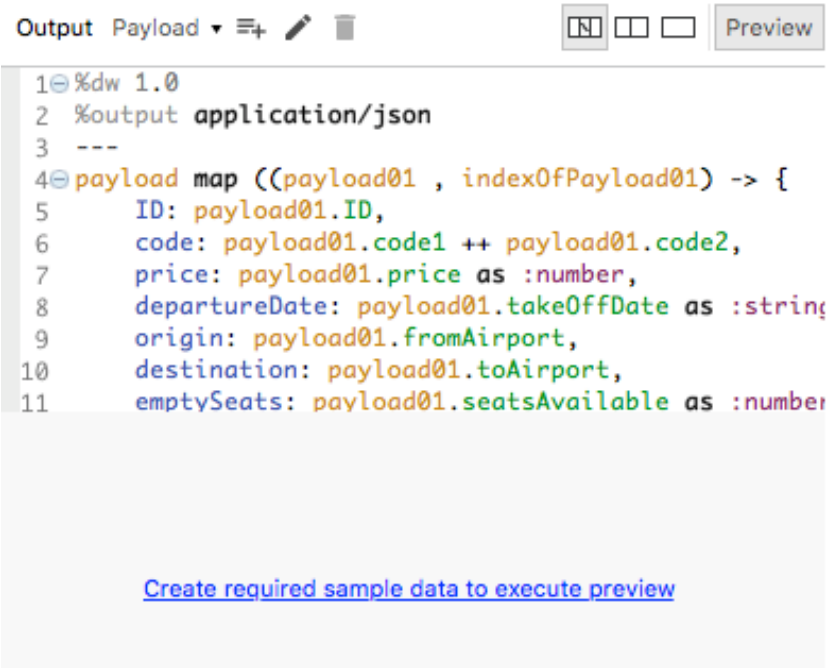


Add sample data

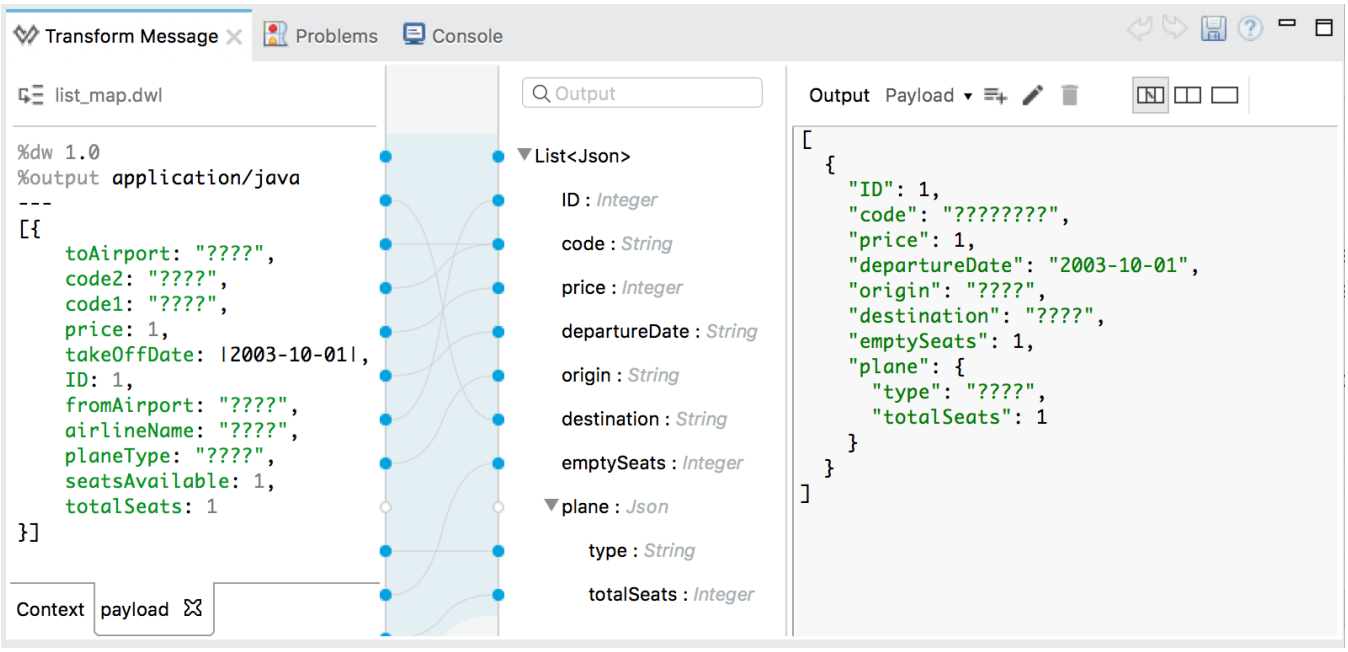
28. Click the Preview button in the output section.



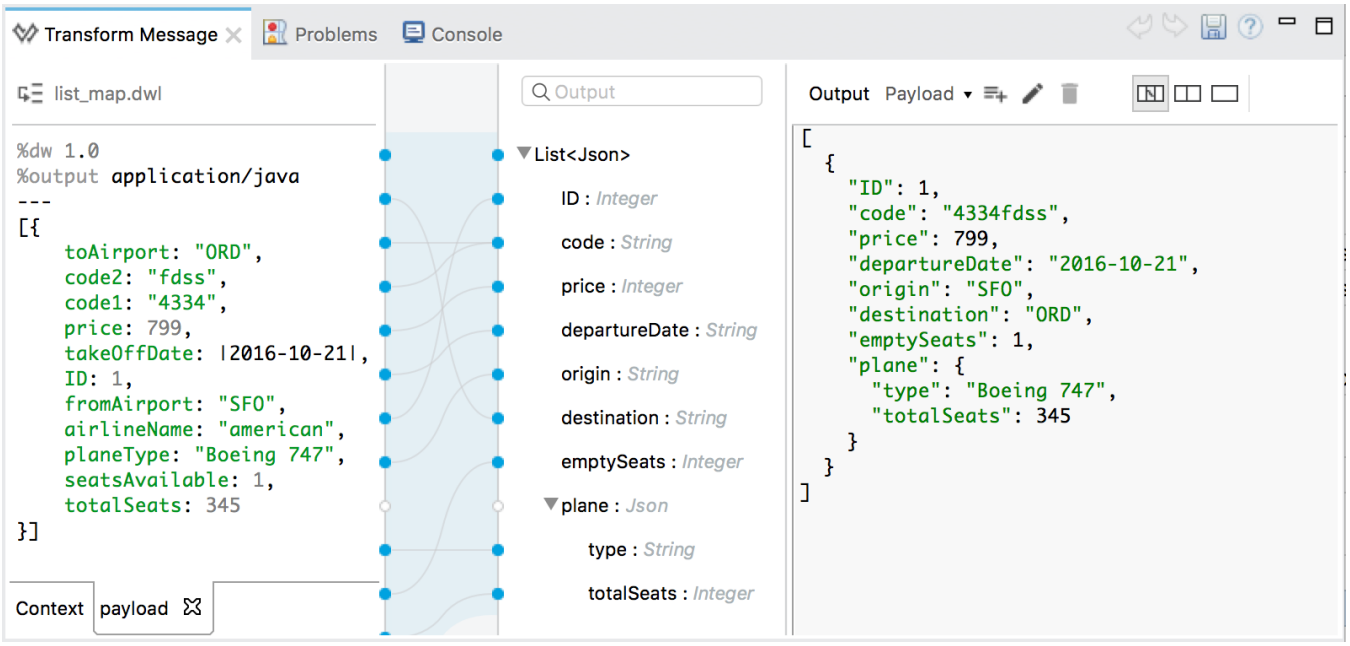
29. In the preview section, click the Create required sample data to execute preview link.



30. Look at the input section, you should see a new tab called payload with sample data generated from the input metadata.
31. Look at the output section, you should see a sample response for the transformation.

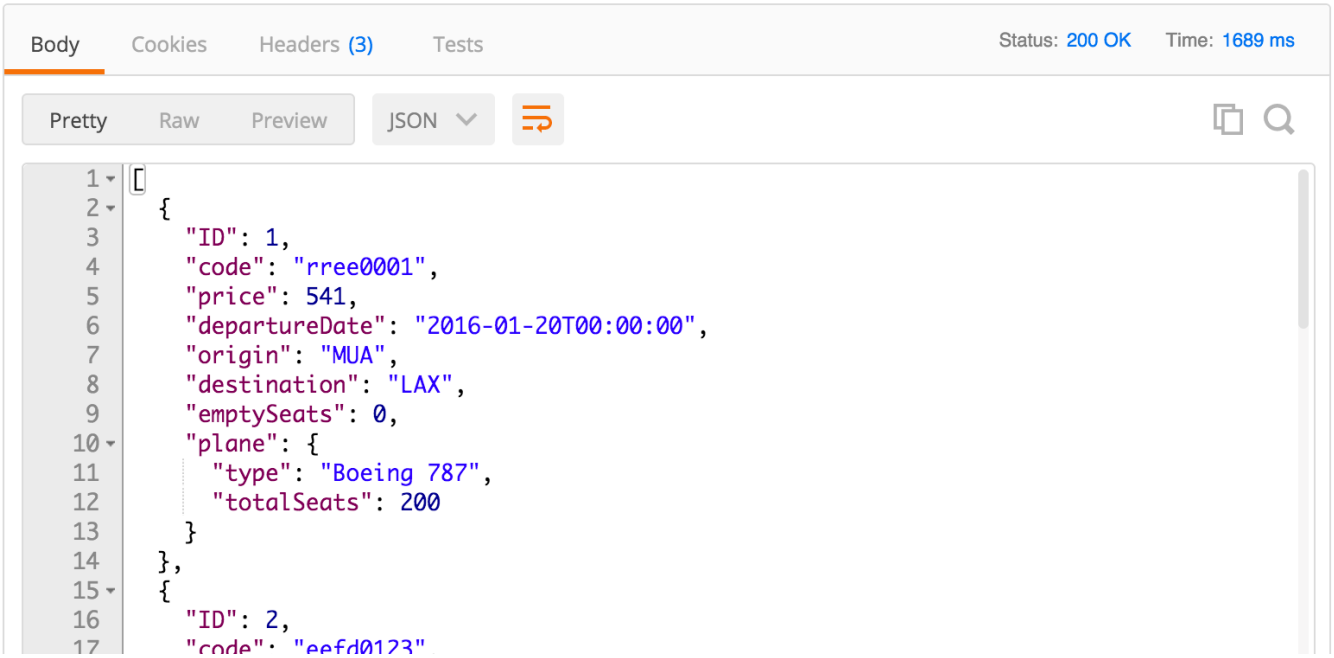


32. In the input section, replace all the ???? with sample values.
33. Look at the output section, you should see the sample values in the transformed data.



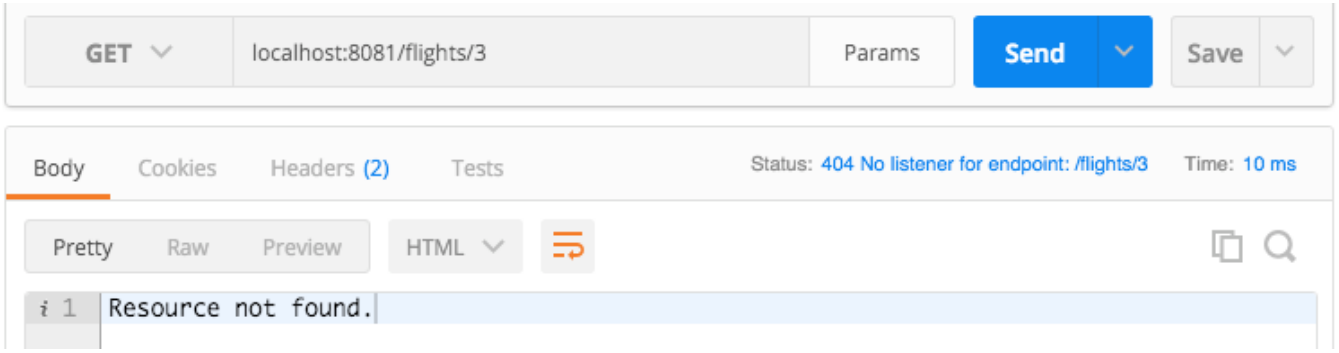
Test the application

34. Run the project.
35. In Postman, make another request to http://localhost:8081/flights; you should see all the flight data as JSON again but now with a different structure.



Try to retrieve information about a specific flight

36. Add a URI parameter to the URL to make a request to http://localhost:8081/flights/3; you should get a 404 response with a resource not found message.

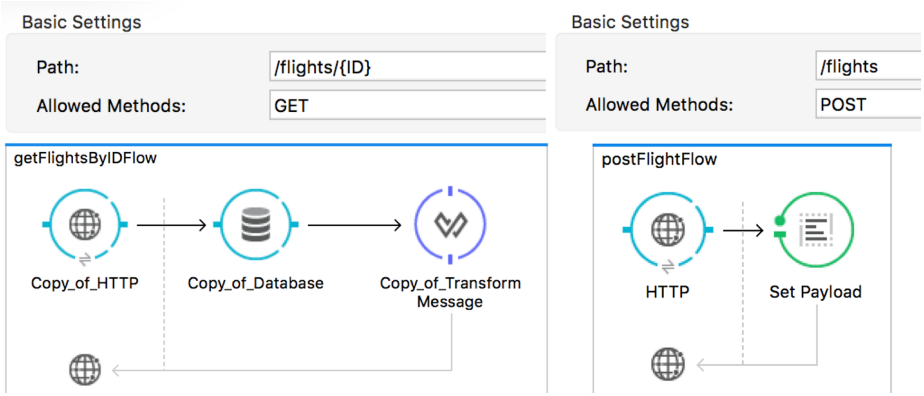


37. Return to Anypoint Studio.
38. Look at the console; you should get a no listener found for request (GET)/flights/3.
39. Stop the project.

Walkthrough 3-4: Create a RESTful interface for a Mule application

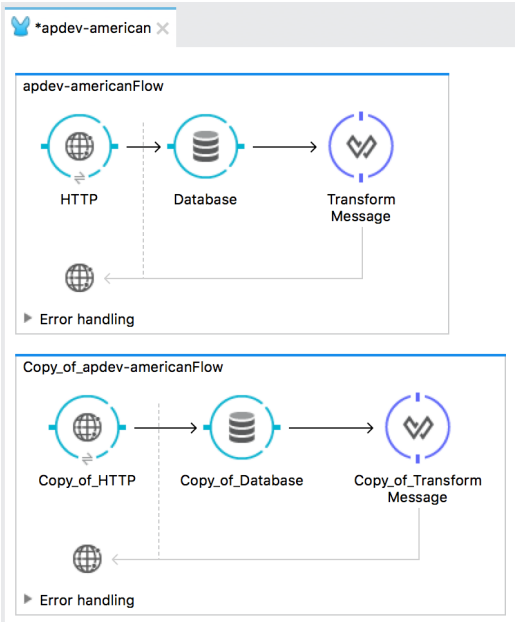
In this walkthrough, you continue to create a RESTful interface for the application. You will:

- Route based on path.
- Add a URI parameter to a new HTTP Listener endpoint path.
- Route based on HTTP method.



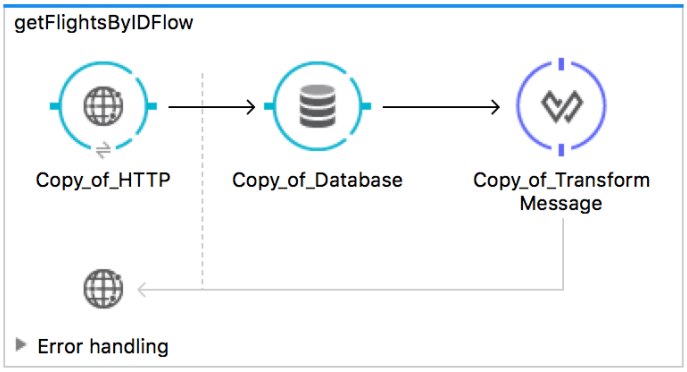
Make a copy of the existing flow

1. Return to apdev-american.xml.
2. Click the flow in the canvas to select it.
3. From the main menu bar, select Edit > Copy.
4. Click in the canvas beneath the flow and select Edit > Paste.



Rename the flows

5. Double-click the name of the first flow.
6. In the Properties view, change its name to getFlightsFlow.
7. Change the name of the second flow to getFlightsByIDFlow.



Note: If you want, change the name of the message source and message processors.

Specify a URI parameter for the new HTTP Listener endpoint

8. Double-click the HTTP Listener endpoint in getFlightsByIDFlow.
9. Change the path to have a URI parameter called ID.

Modify the Database endpoint

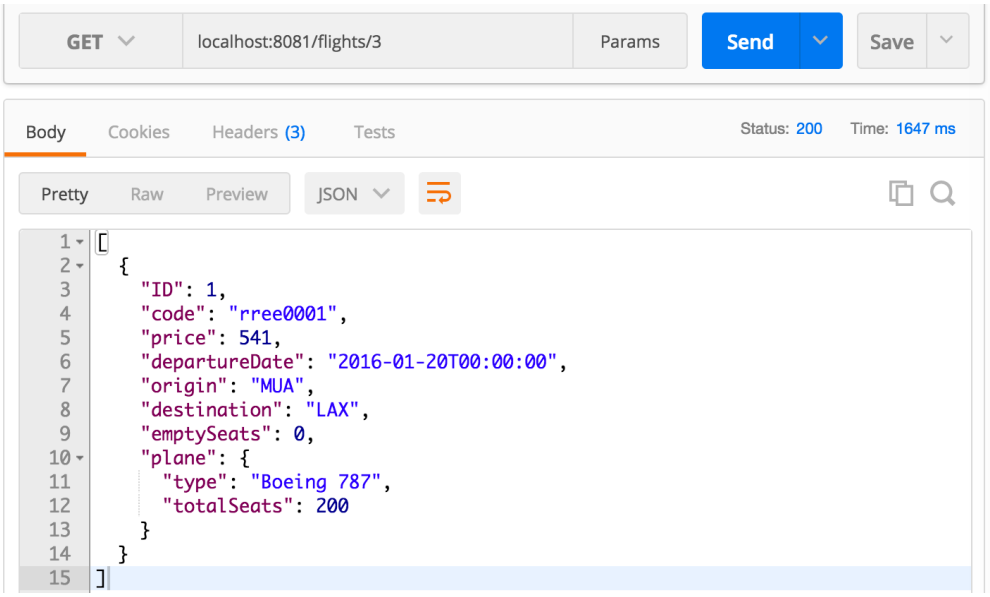
10. Double-click the Database endpoint in getFlightsByIDFlow.
11. Modify the query WHERE clause, to select flights with the ID equal to 1.

```
SELECT *
FROM american
WHERE ID = 1
```

Test the application

12. Run the project.

13. In Postman, make another request to http://localhost:8081/flights/3; you should see details for the flight with an ID of 1.



Modify the database query to use the URI parameter

14. Return to the course snippets.txt file and copy the SQL expression for American Flights API.
15. Return to Anypoint Studio and stop the project.
16. In the Database properties view, replace the existing WHERE clause with the value you copied.

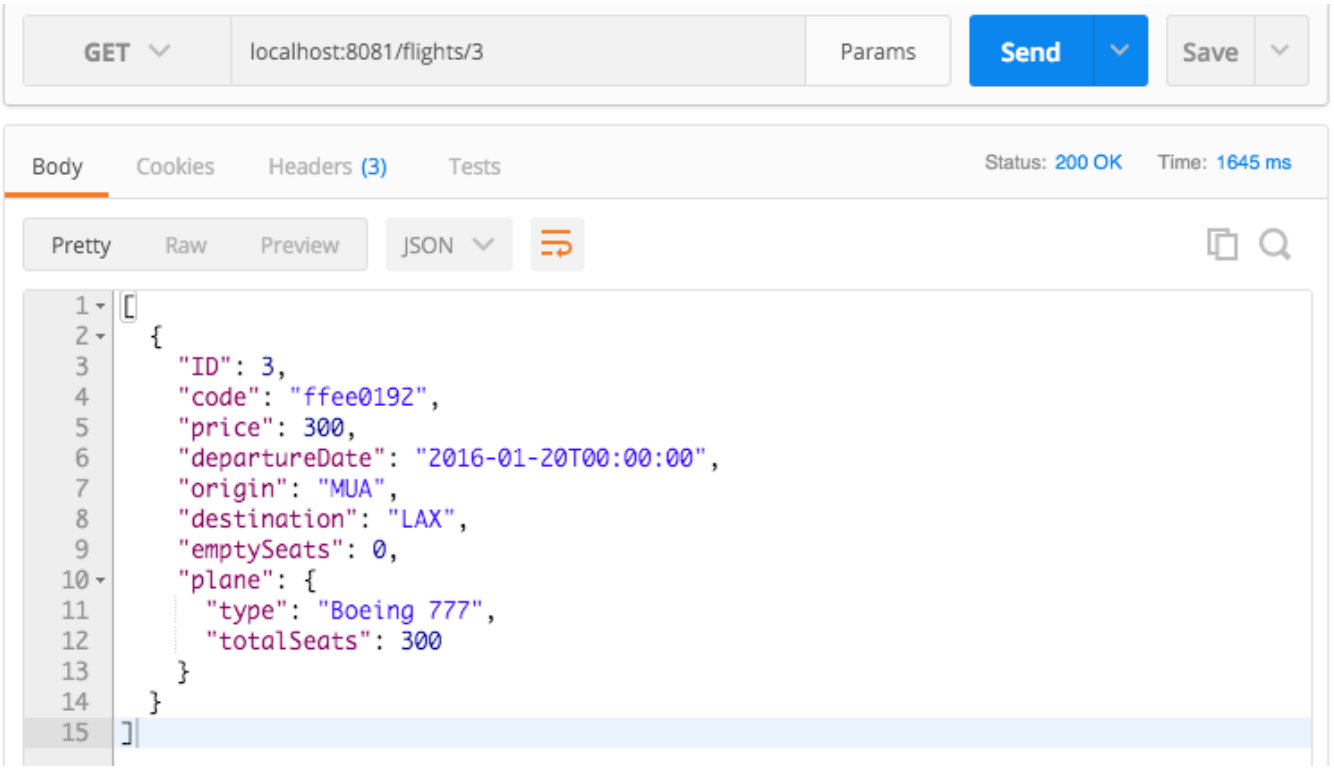
```
SELECT *
FROM american
WHERE ID = #[message.inboundProperties.'http.uri.params'.ID]
```

Note: You learn about reading and writing properties and variables in a later module in the developer courses.

Test the application

17. Run the project.

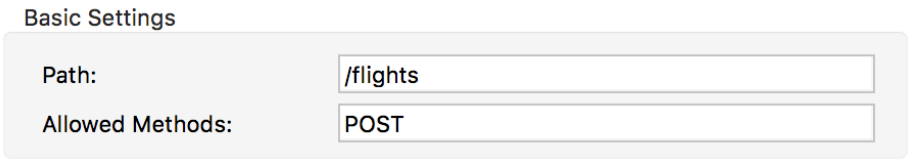
18. In Postman, make another request to http://localhost:8081/flights/3; you should now see the info for the flight with an ID of 3.



19. Return to Anypoint Studio and stop the project.

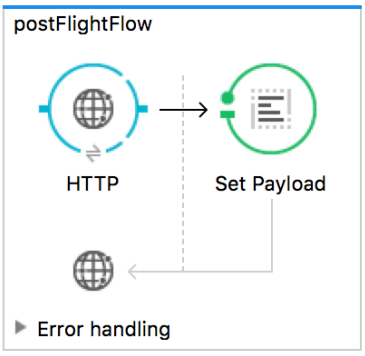
Make a new flow to handle post requests

20. In the Mule Palette, select the Connectors tab.
21. Drag out an HTTP connector from the Mule Palette and drop it in the canvas below the two existing flows.
22. Change the name of the flow to postFlightFlow.
23. Double-click the HTTP Listener endpoint.
24. In the HTTP properties view, set the path to /flights and the allowed methods to POST.



25. In the Mule Palette, select the Transformers tab.

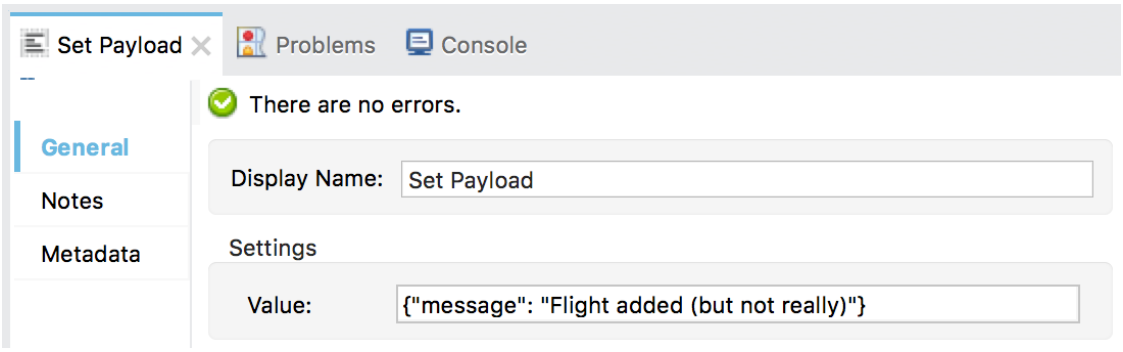
26. Drag out a Set Payload transformer from the Mule Palette and drop it in the process section of the flow.



27. Double-click the Set Payload processor.
28. Return to the course snippets.txt file and copy the Module 2 American Flights API - /flights post response example.

```
{"message": "Flight added (but not really)"}
```

29. Return to Anypoint Studio and in the Set Payload properties view, set value to the value you copied.

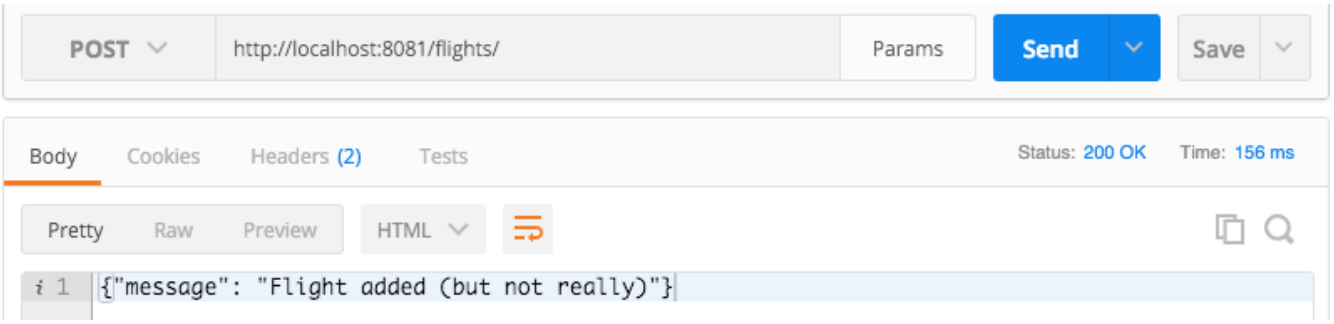


Note: This flow is just a stub. For it to really work and add data to the database, you would need to add logic to insert the request data to the database.

Test the application

30. Run the project.
31. In Postman, change the request type from GET to POST.
32. Remove the URI parameter from the request URL: http://localhost:8081/flights.

33. Send the request; you should now see the message the flight was added – even though you did not send any flight data to add.

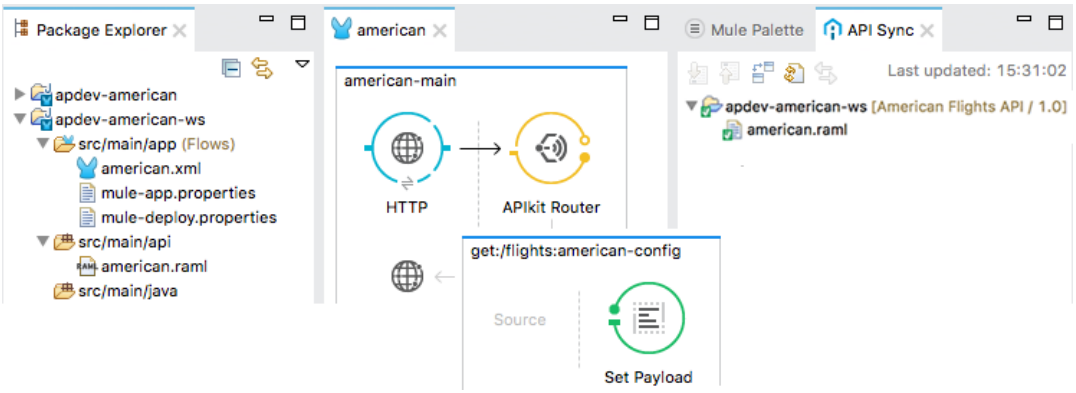


34. Return to Anypoint Studio and stop the project.

Walkthrough 3-5: Use Anypoint Studio to create a RESTful API interface from a RAML file

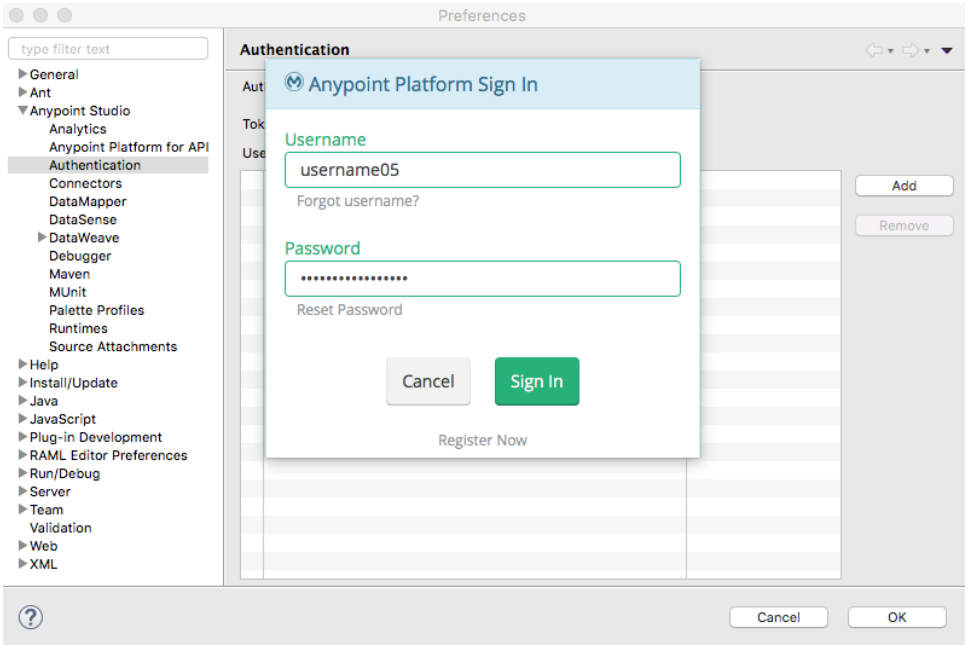
In this walkthrough, you generate a RESTful interface from the RAML file. You will:

- Add Anypoint Platform credentials to Anypoint Studio.
- Add a RAML file from Anypoint Platform to an Anypoint Studio project.
- Use Anypoint Studio and APIkit to generate a RESTful web service interface from a RAML file.
- Test the web service in the APIkit Consoles view and Postman.



Add Anypoint Platform credentials to Anypoint Studio

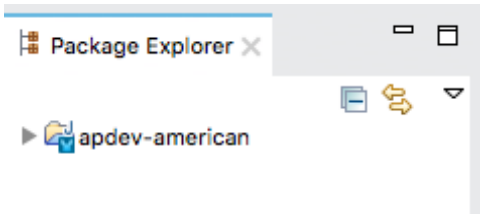
1. In Anypoint Studio, right-click apdev-american and select Anypoint Platform > Configure.
2. In the Authentication page of the Preferences dialog box, click the Add button.
3. In the Anypoint Platform Sign In dialog box, enter your username & password and click Sign In.



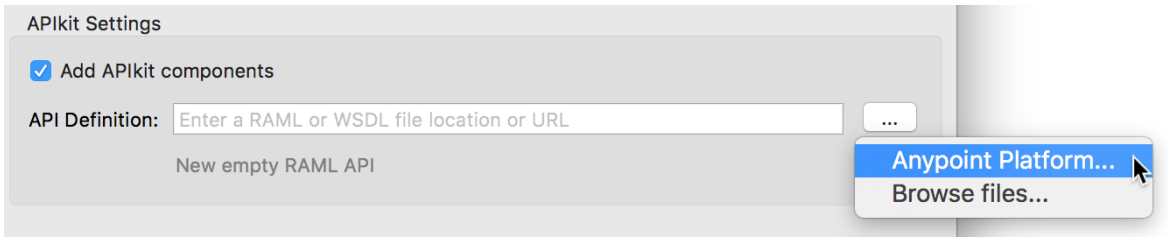
4. On the Authentication page, make sure your username is listed and selected.
5. Click OK.

Create a new project

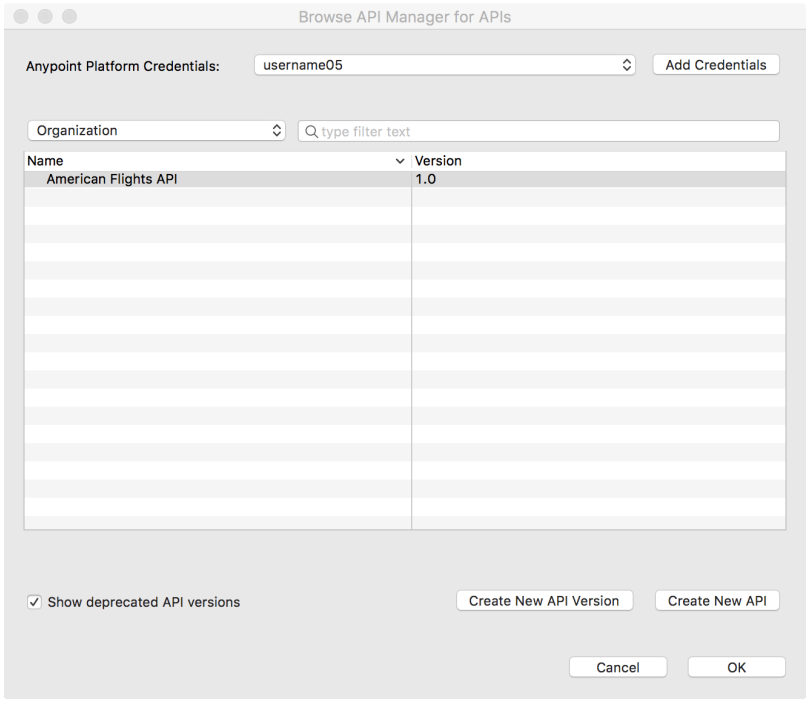
6. In the Package Explorer, click the arrow to the left of the apdev-american project to collapse it.



7. Right-click in the Package Explorer and select New > Mule Project.
8. In the New Mule Project dialog box, set the project name to apdev-american-ws.
9. Check Add APIkit components.
10. Click the browse button next to API Definition and select Anypoint Platform.



11. In the Browse API Manager for APIs dialog box, select the American Flights API and click OK.

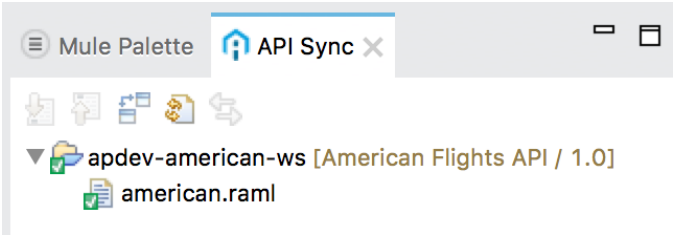


12. In the New Mule Project dialog box, click Finish.

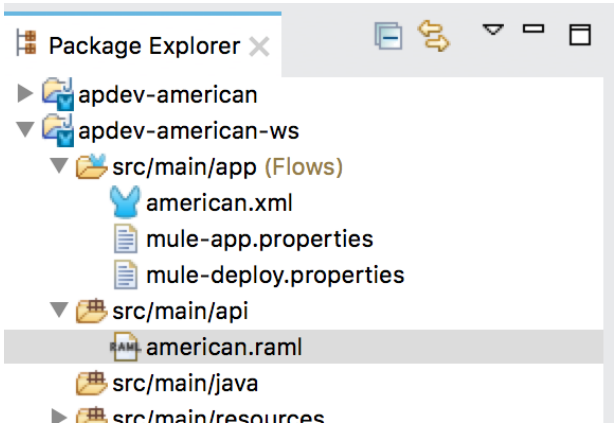
    Note: You are creating a new project because in Anypoint Studio 6.0.0, you cannot connect an existing project to an API on Anypoint Platform. You have to make this connection when you create the project. This will be possible in a future release.

Locate the RAML file added to the project

13. Click the API Sync tab next to Mule Palette tab.
14. Expand the apdev-american-ws folder; you should see the API and the RAML file it contains.



15. In the Package Explorer, locate and expand the project's src/main/api folder; it should contain american.raml.

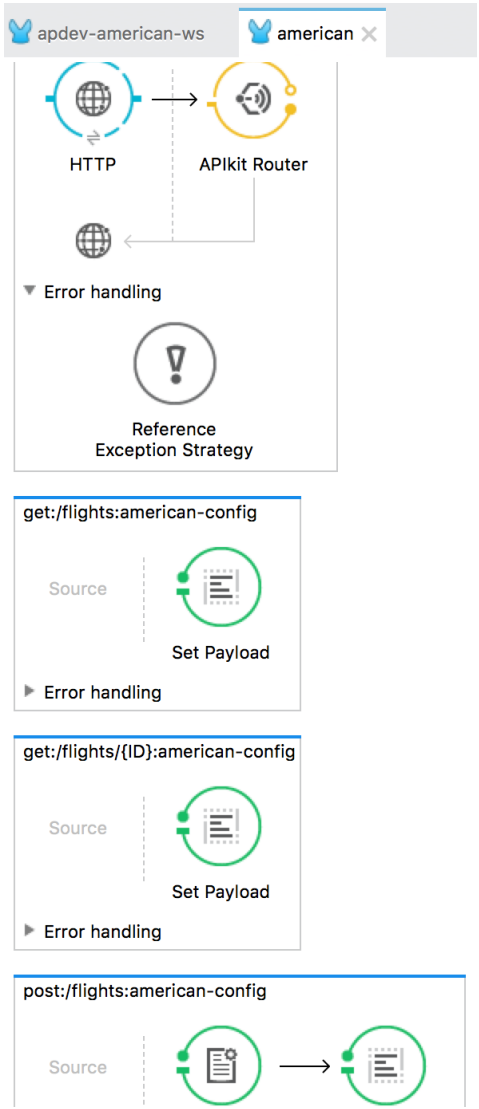


    Note: If you did not successfully create the RAML file earlier, you can use the one included in the solution folder of the student files. Open it in a text editor, copy all the code, open american.raml in Anypoint Studio, replace the existing code, save the file, then click the Upload button in the API Sync view to upload it to your Anypoint Platform account.

Examine the XML file created

16. In american.xml, locate the following three flows:

    - get:/flights
    - get:/flights/{ID}
    - post:/flights

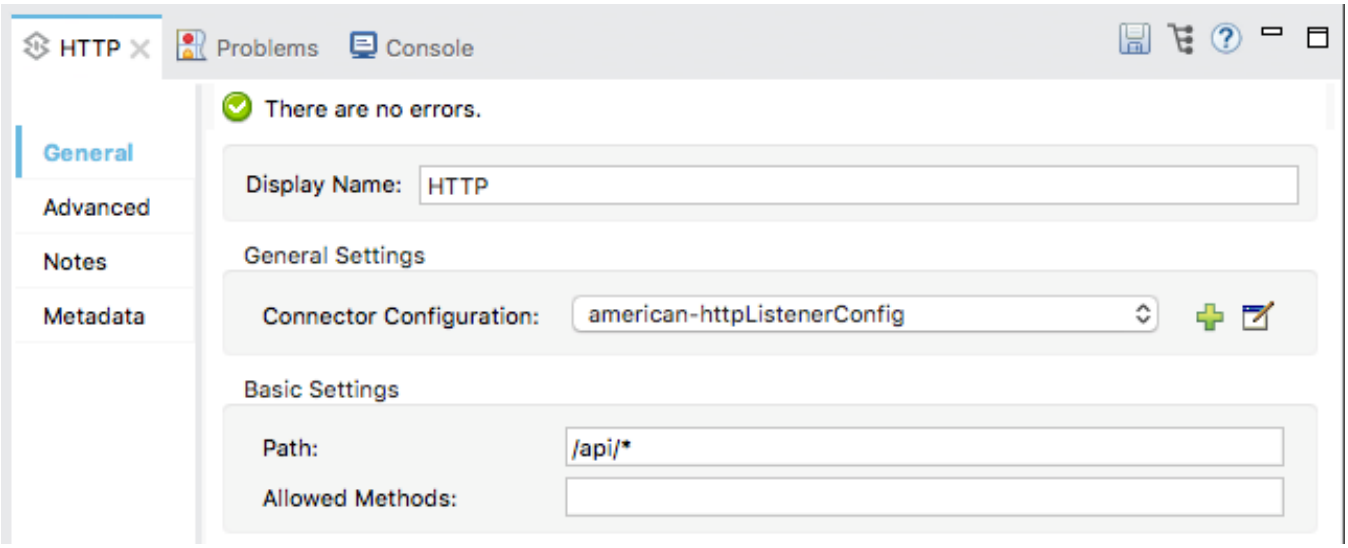


Examine the new HTTP Listener endpoint

17. Double-click the HTTP Listener endpoint in the american-main flow.

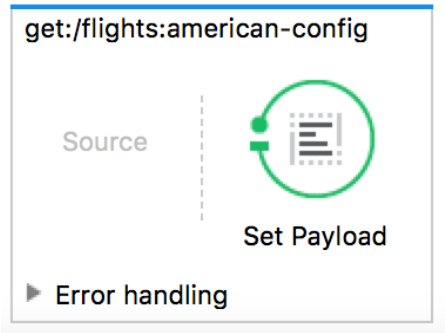
18. In the HTTP properties view, notice that the path is set to /api/*.

    *Note: The * is a wildcard allowing any characters to be entered after /api/.*

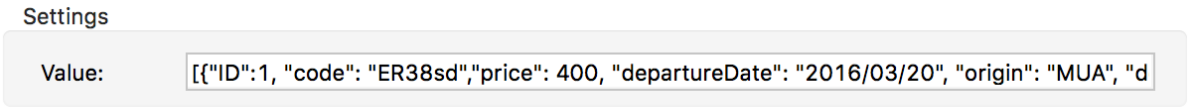


Look at the generated resource flow

19. Look at the get:/flights flow.



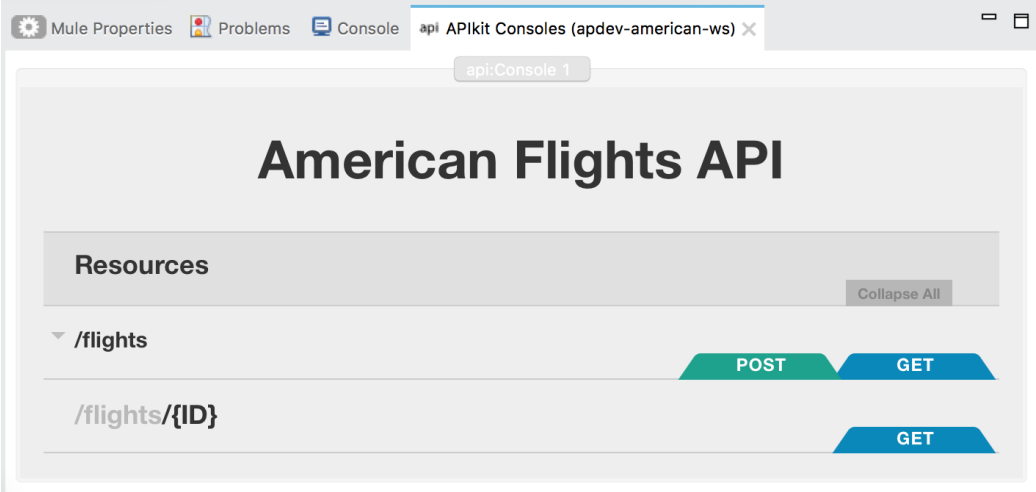
20. Double-click the Set Payload transformer and look at the value in the Set Payload properties view.



Test the web service in the APIkit Consoles view

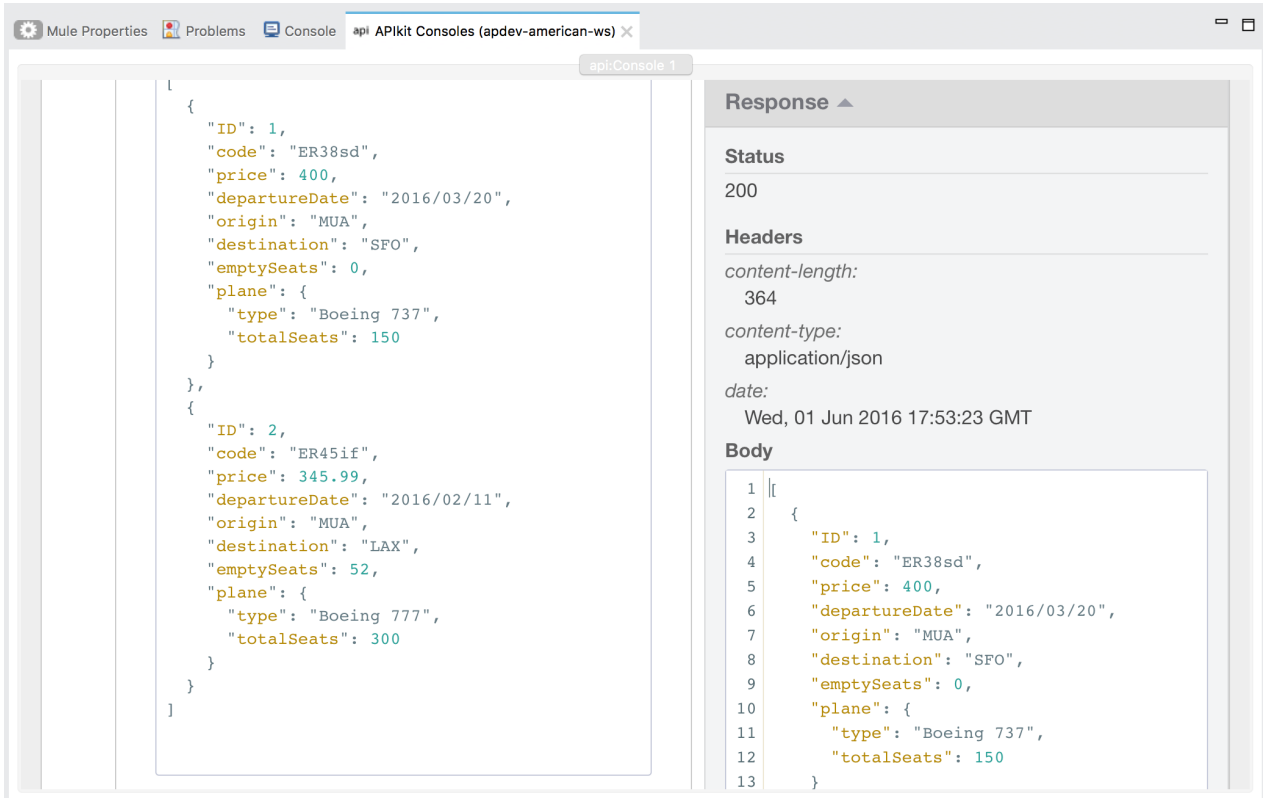
21. Run the project.

22. Look at the APIkit Consoles view that opens.



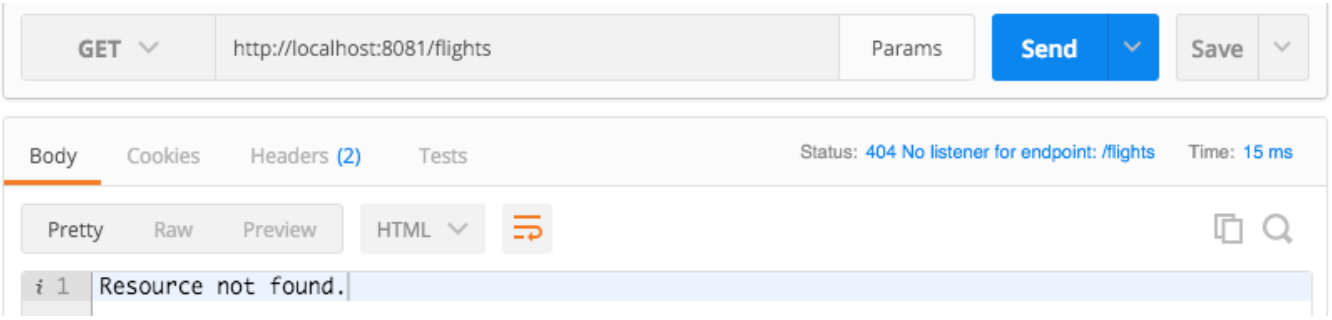
Note: If the API is not displayed in the APIkit Console, change your HTTP Listener Configuration host from 0.0.0.0 to localhost and then run the project again.

23. Click the GET button for /flights.
24. Click the Try It button.
25. Click in the code query parameter field and select one of the enumerated values, like LAX.
26. Click the GET button; you should get a 200 response with the example flight data.



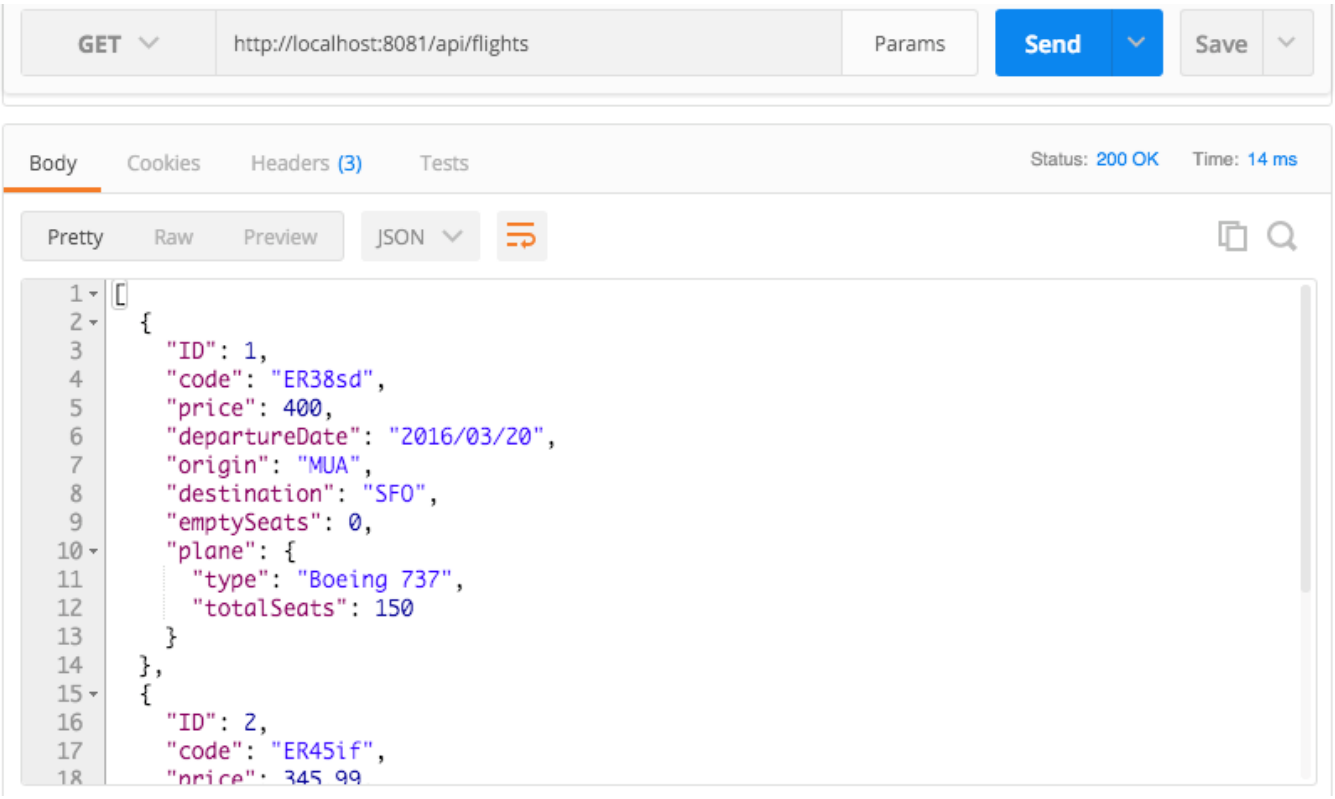
Test the web service in Postman

27. In Postman, make a GET request to http://localhost:8081/flights; you should get a 404 response with a message that the resource was not found because there is no longer a listener for that endpoint.



28. Change the URL to http://localhost:8081/api/flights and send the request; you should see the example data returned.

    Note: Be sure the method is set to GET.

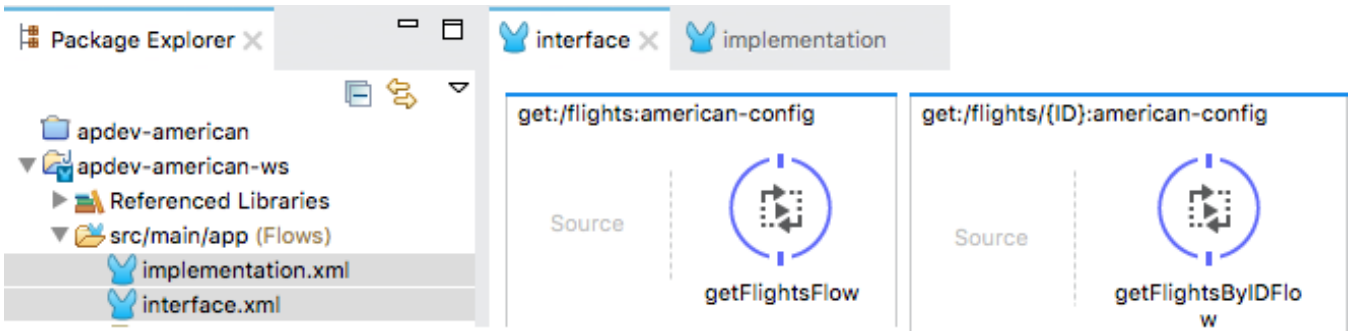


29. Make a request to http://localhost:8081/api/flights/3; you should see the example data returned.
30. Return to Anypoint Studio and stop the project.

Walkthrough 3-6: Implement a RESTful web service

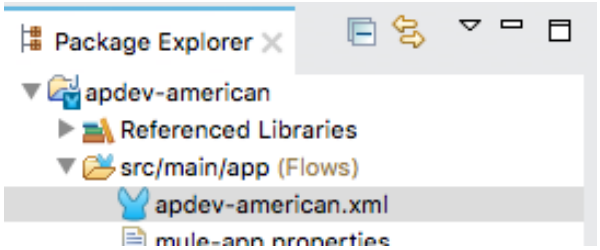
In this walkthrough, you wire the RESTful web service interface up to your back-end logic. You will:

- Pass a message from one flow to another.
- Create new logic for the nested resource call.
- Call the backend flows.
- Test the web service in the APIkit Consoles view and Postman.

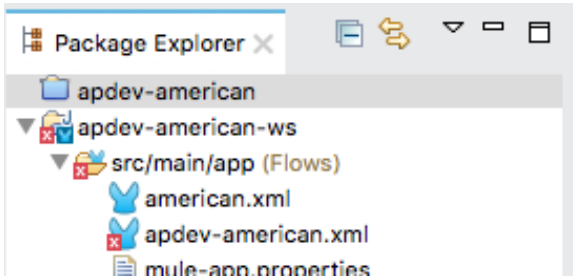


Add implementation code to the new project

1. Return to Anypoint Studio.
2. Expand the apdev-american project.
3. Right-click apdev-american.xml and select Copy.

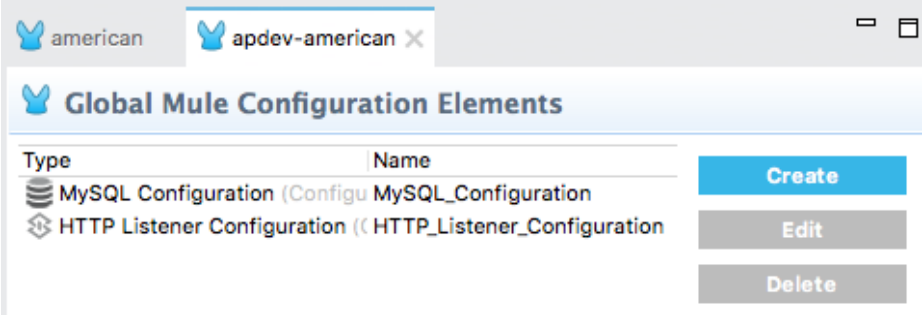


4. In the apdev-american-ws project, right-click the src/main/app folder and select Paste.
5. Right-click the apdev-american project and select Close Project.



6. Open apdev-american.xml in the apdev-american-ws project.
7. Click the Global Elements link at the bottom of the canvas.

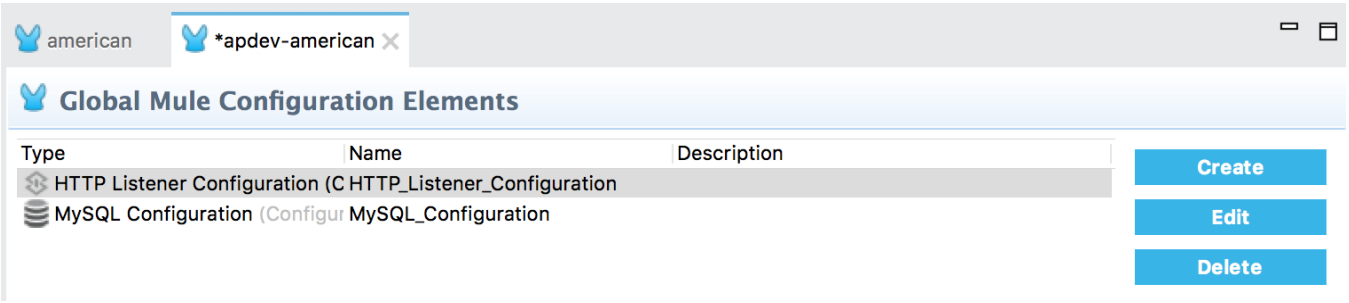
8. Double-click the MySQL Configuration.



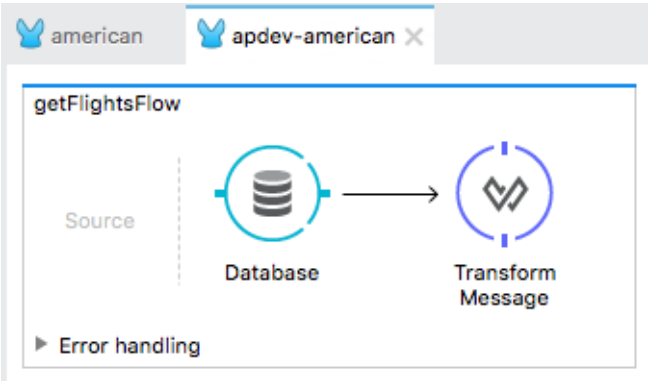
9. In the Global Element Properties dialog box, click the Add File button next to MySQL Driver.
10. Browse to and select the mysql.jar file located in the resources folder in the course student files.
11. Click Open.
12. In the Global Element Properties dialog box, click OK.

Remove the listeners

13. In the Global Elements view, select the HTTP Listener and click Delete.



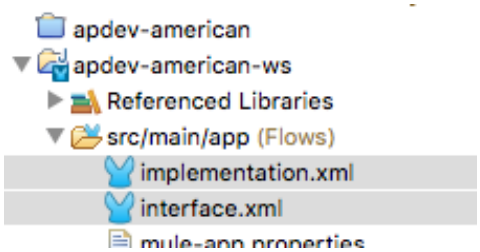
14. Return to the Message Flow view.
15. Right-click the HTTP Listener endpoint in one of the flows and select Delete.



16. Delete the other two HTTP Listener endpoints.

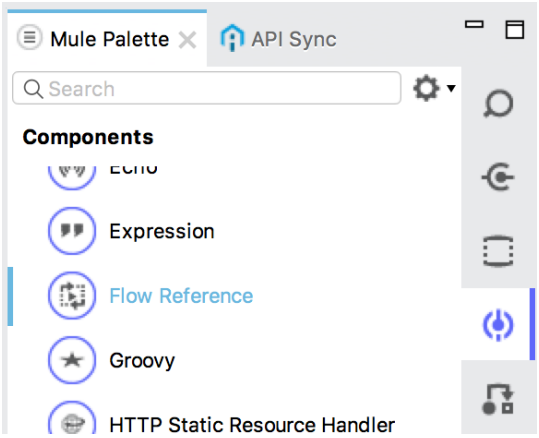
Rename the configuration files

17. Right-click american.xml in the Package Explorer and select Refactor > Rename.
18. In the Rename Resource dialog box, set the new name to interface.xml and click OK.
19. Right-click apdev-american.xml and select Refactor > Rename.
20. In the Rename Resource dialog box, set the new name to implementation.xml and click OK.

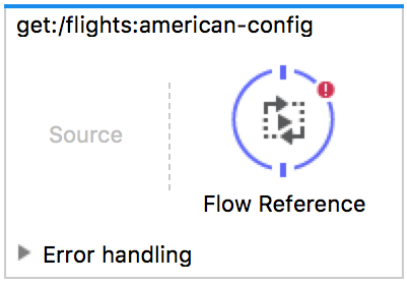


Set logic for the /flights resource

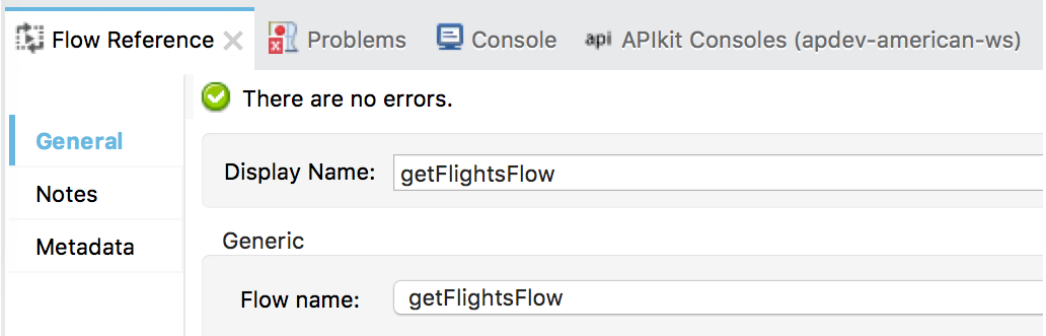
21. Return to interface.xml.
22. Delete the Set Payload transformer in the get:/flights flow.
23. In the Mule Palette, select the Components tab.



24. Drag a Flow Reference component from the Mule Palette and drop it into the process section of the flow.



25. In the Flow Reference properties view, select getFlightsFlow for the flow name.

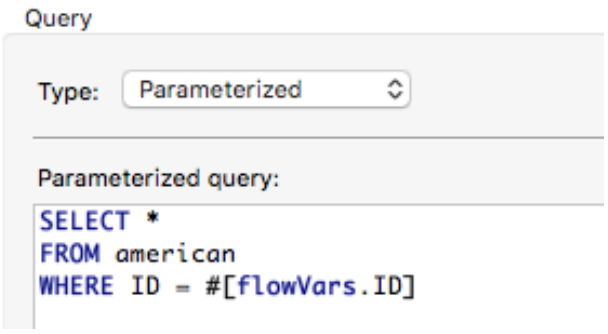


Set logic for the /flights/{ID} resource

26. Delete the Set Payload transformer in the get:/flights/{ID} flow.
27. Drag a Flow Reference component from the Mule Palette and drop it into the flow.
28. In the Flow Reference properties view, select getFlightsByIDFlow for the flow name.

29. Return to implementation.xml.
30. Double-click the Database endpoint in getFlightsByIDFlow.
31. Change the query to use a variable instead of a query parameter.

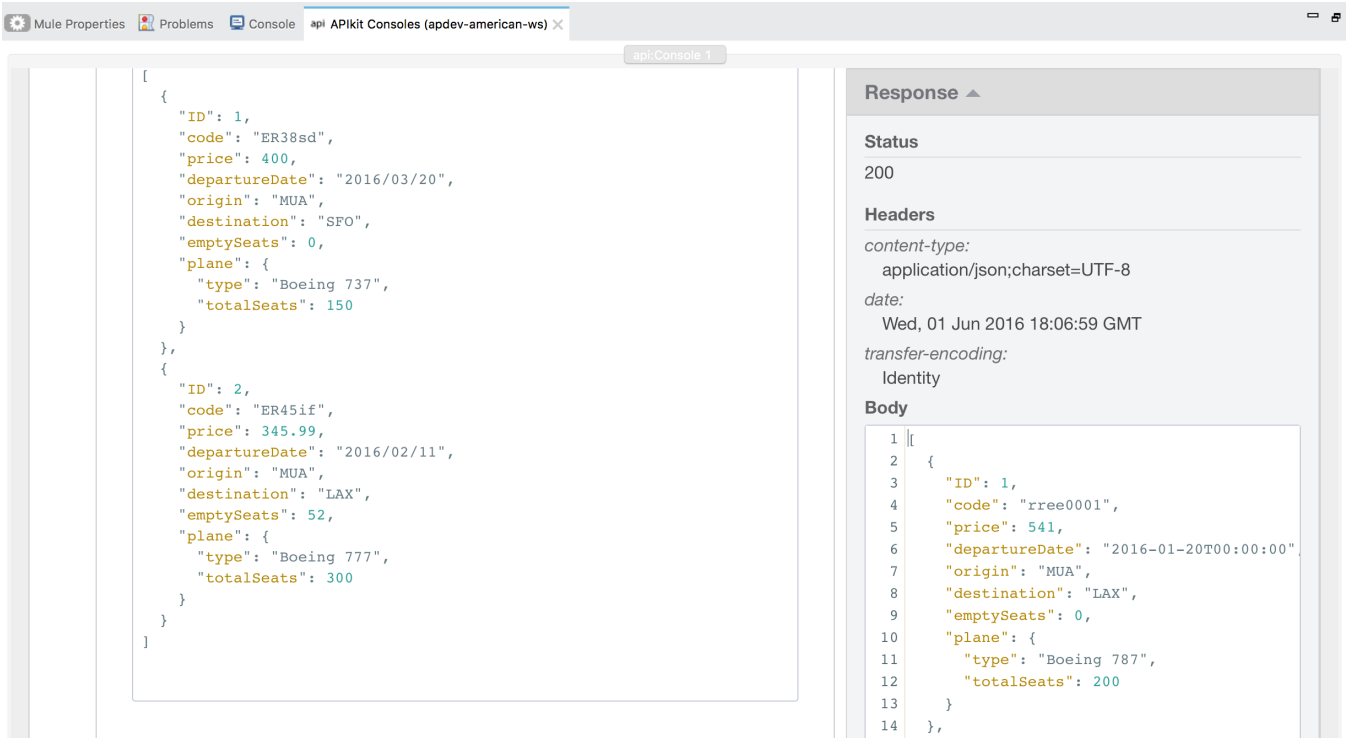
```
WHERE ID = #[flowVars.ID]
```



Note: You learn about the different types of variables in later modules in the developer courses.

Test the web service in the APIkit Consoles view

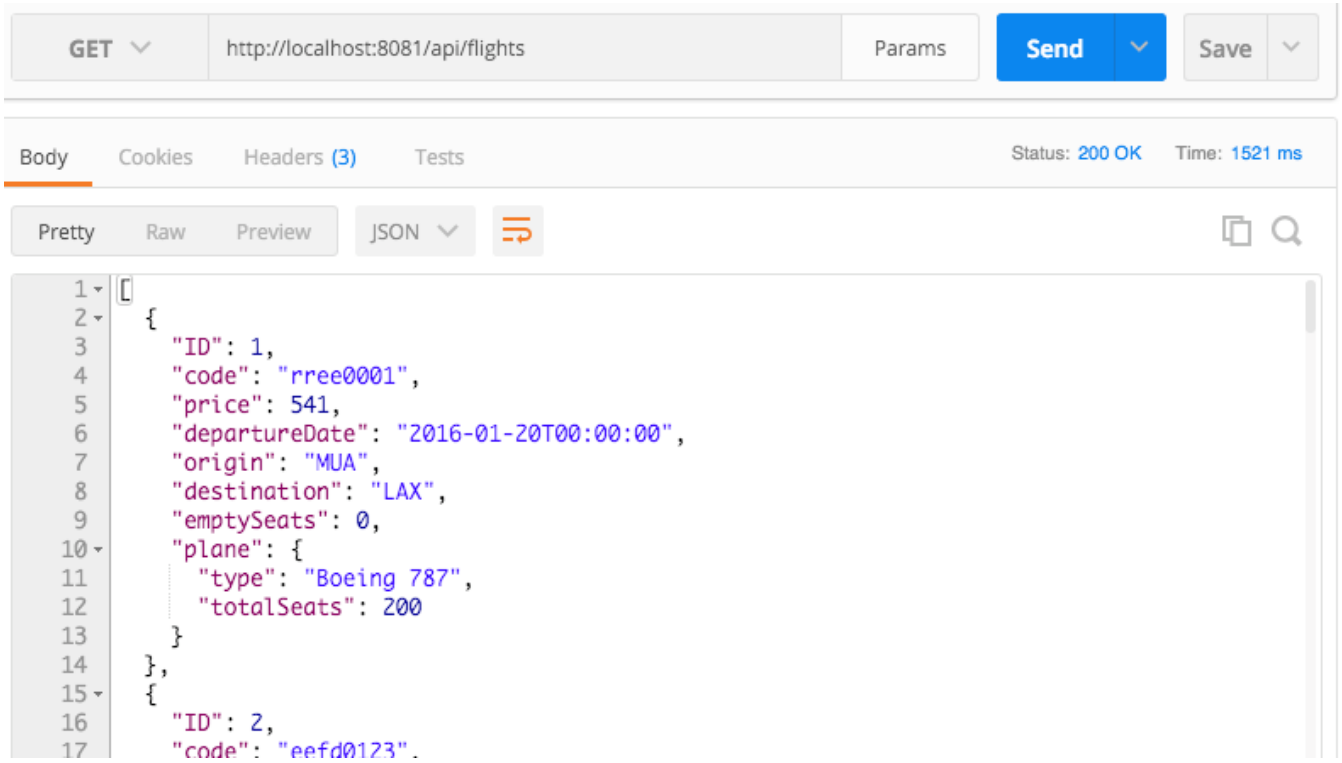
32. Run the project.
33. In the APIkit Consoles view, click the GET button for /flights.
34. Click the Try It button.
35. Click the GET button; you should now see the real data from the database displayed instead of the example data.



36. Scroll down and click the GET button for the /flights/{ID} resource.
37. Click the Try It button.
38. Enter an ID of 2 and click the GET button; you should get the data for that flight.

Test the web service in Postman

39. In Postman, make a request to http://localhost:8081/api/flights; you should now get the data for all the flights from the database instead of the sample data.



40. Make a request to http://localhost:8081/api/flight/3; you should now get the data that flight from the database instead of the sample data.
41. Return to Anypoint Studio and stop the project.