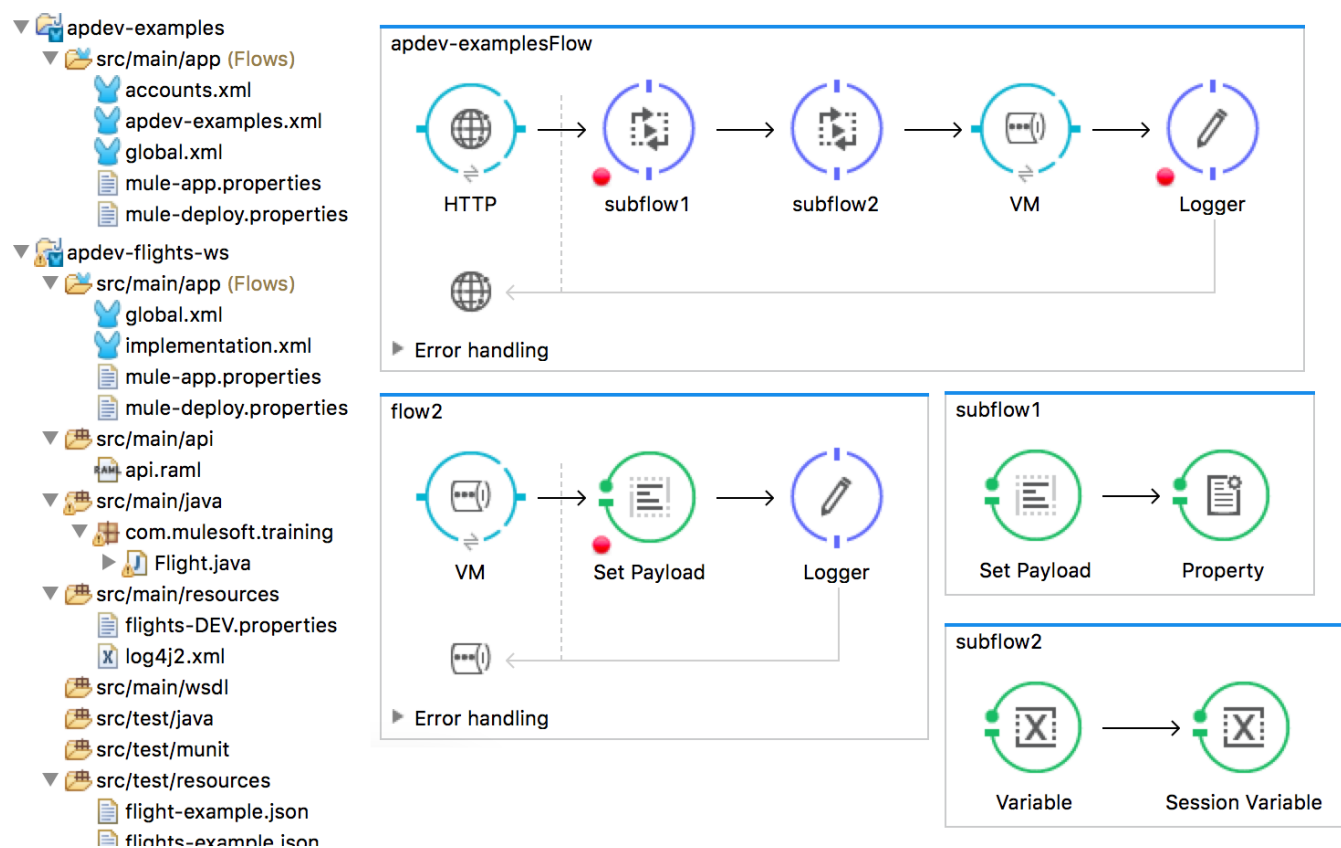


Module 6: Structuring Mule Applications



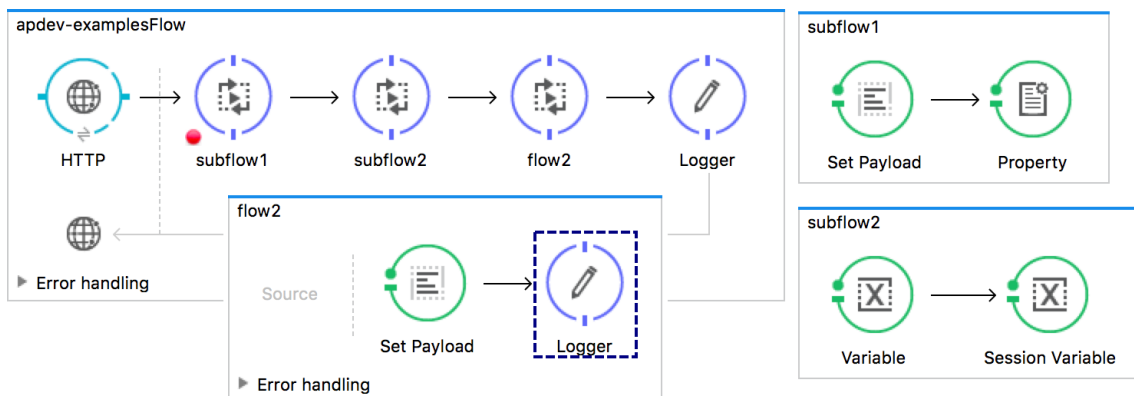
Objectives:

- Create and reference flows and subflows.
- Pass messages between flows using the Java Virtual Machine (VM) transport.
- Investigate variable persistence through subflows and flows and across transport barriers.
- Encapsulate global elements in separate configuration files.
- Explore the files and folder structure of a Mule project.

Walkthrough 6-1: Create and reference flows and subflows

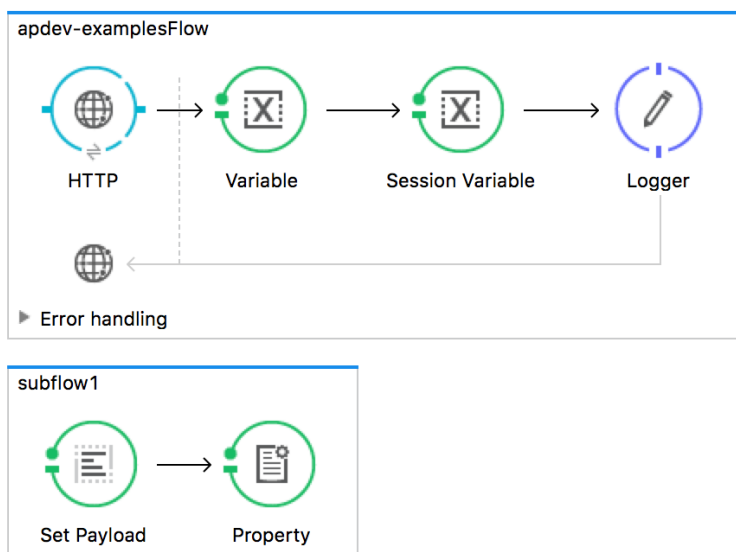
In this walkthrough, you continue to work with `apdev-examples.xml`. You will:

- Extract processors into separate subflows and flows.
- Use the Flow Reference component to reference other flows.
- Explore variable persistence through flows and subflows.



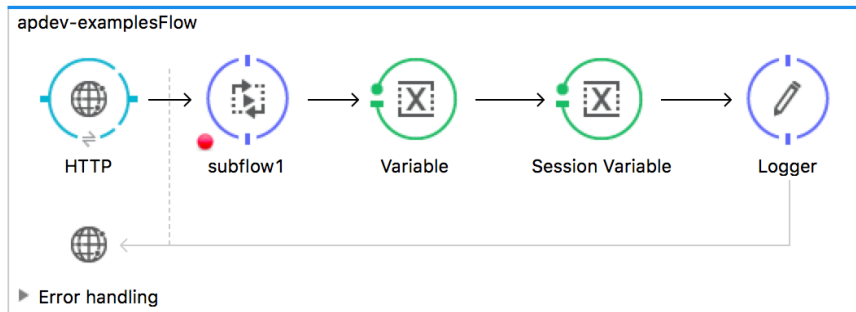
Create a subflow

1. Return to `apdev-examples.xml` in Anypoint Studio.
2. Drag a Sub Flow scope from the Mule Palette and drop it beneath the existing flow in the canvas.
3. Select the **Set Payload** and **Property** transformers in `apdev-examplesFlow` and drag them into the subflow.
4. Change the name of the subflow to `subflow1`.



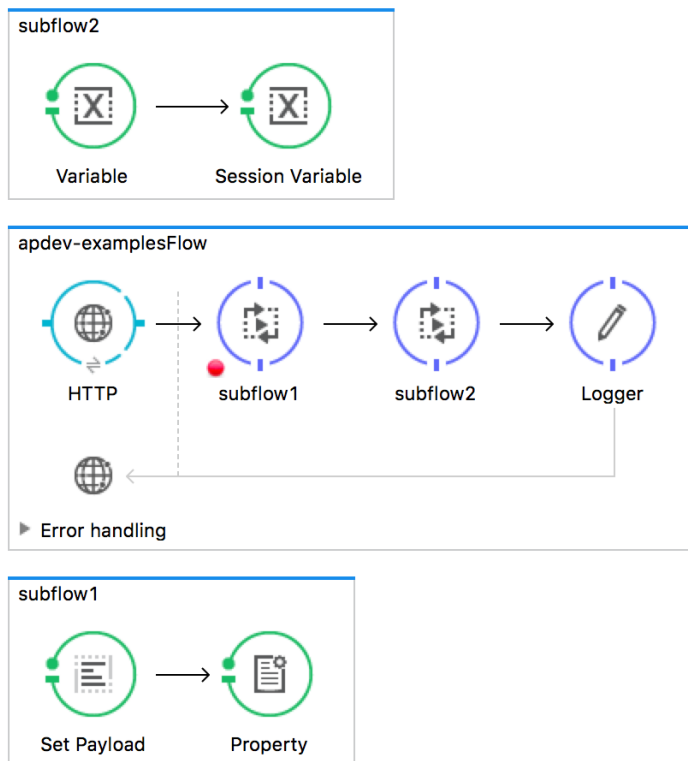
Reference a subflow

5. Drag a Flow Reference component from the Mule Palette and drop it into the apdev-examplesFlow between the HTTP Listener endpoint and the Variable transformer.
6. In the Flow Reference properties view, set the flow name to subflow1.



Extract processors into a subflow

7. Right-click the Variable and Session Variable transformers and select Extract to > Sub Flow.
8. In the Extract Flow dialog box, set the flow name to subflow2.
9. Leave the target Mule configuration set to current and click OK.
10. Look at the new Flow Reference Properties view; the flow name should already be set to subflow2.



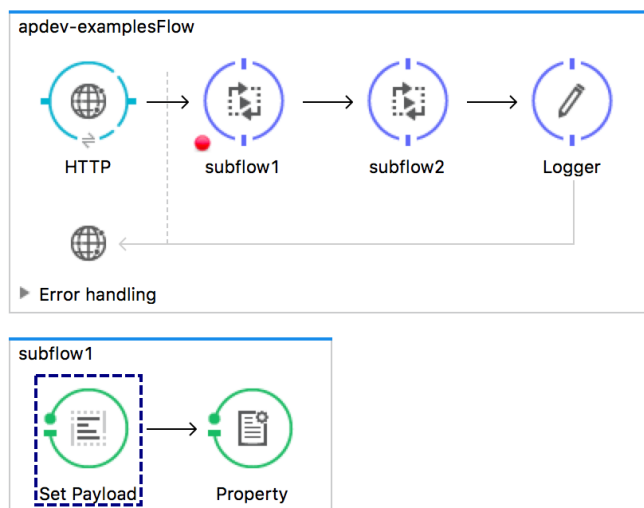
11. Drag subflow2 below subflow1 in the canvas.

Debug the application

12. Debug the project.

13. In Postman, send the same request to <http://localhost:8081/hello?name=max&type=mule>.

14. In the Mule Debugger, step through the application, watching messages move into and out of the subflows.



15. In Postman, send the same request again.

16. In the Mule Debugger, step through the application again, this time watching the values of the inbound properties, variables, outbound properties, and session variables in each of the flows.

Create a second flow

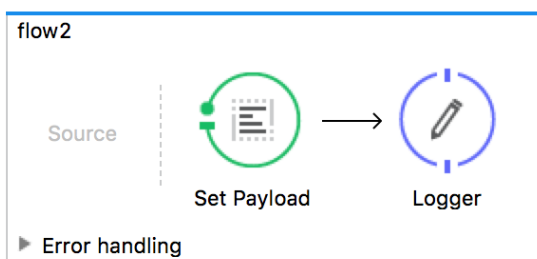
17. Drag a Flow scope element to the canvas and drop it beneath the existing flows.

18. Change the flow name to flow2.

19. Add a Set Payload transformer to the process section of the flow.

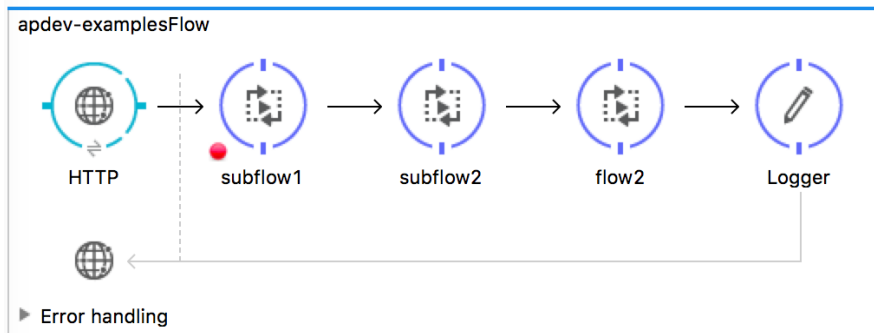
20. In the Set Payload properties view, set the value to Goodbye.

21. Add a Logger after the Set Payload transformer.



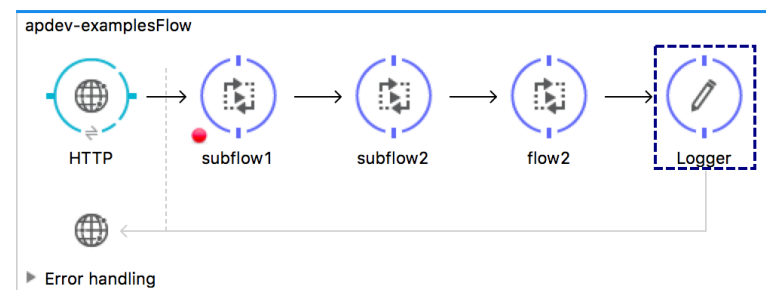
Reference a flow

22. In apdev-examplesFlow, add a Flow Reference component between the subflow2 Flow Reference and the Logger.
23. In the Flow Reference Properties view, set the flow name to flow2.



Debug the application

24. Save the file to redeploy the application in debug mode.
25. In Postman, send the same request.
26. In the Mule Debugger, step through the application, watching the flow move into and out of the second flow; at the end, you should see the payload is the value set in the second flow.



Message Flow Global Elements Configuration XML

Mule Debugger Console Problems Mule Properties

Name	Value	Type
DataType	SimpleDataType{ty...	org.mule.transfor...
Exception	null	
Message		org.mule.DefaultM...
Message Pr...	Logger	org.mule.api.proce...
Payload (mi...	Goodbye	java.lang.String

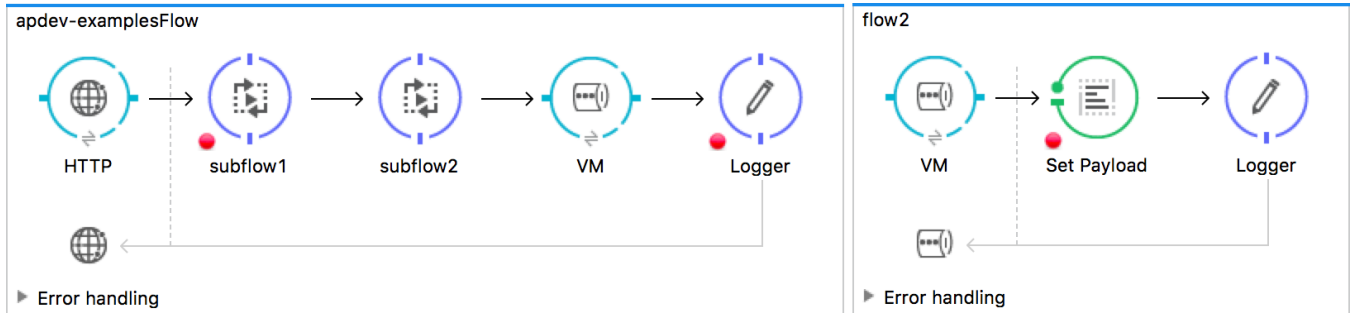
Name	Value	Type
qptype (mime...	mule	java.lang.Stri

27. In Postman, send the same request again.
28. In the Mule Debugger, step through the application again, this time watching the values of the inbound properties, variables, outbound properties, and session variables in each of the flows.
29. Stop the project.

Walkthrough 6-2: Pass messages between flows using the Java Virtual Machine (VM) transport

In this walkthrough, you continue to work with the `apdev-examplesFlow`. You will:

- Pass messages through an HTTP transport barrier.
- Pass messages between flows using the VM transport.
- Explore variable persistence across these transport barriers.



Create an HTTP transport barrier

1. Return to `apdev-examples.xml`.
2. Switch to the Global Elements view.
3. Click Create.
4. In the Choose Global Type dialog box, select Connector Configuration > HTTP Request Configuration and click OK.
5. In the Global Elements dialog box, set the host to `localhost`, the port to `8081`, and click OK.

The screenshot shows the **Global Element Properties** dialog box for **HTTP Request Configuration**. The dialog has tabs for General, TLS/SSL, Proxy, Authentication, Sockets, and Notes. The **General** tab is selected.

Generic section:

- Name: `HTTP_Request_Configuration`

URL Configuration section:

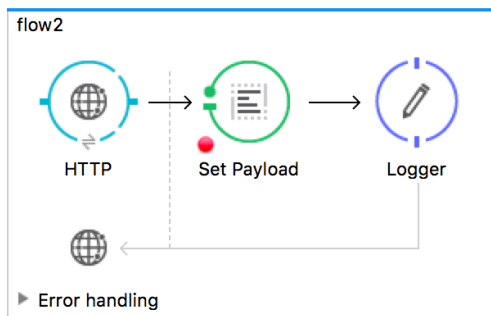
- Protocol: ☒ HTTP ☐ HTTPS
- Host: `localhost`
- Port: `8081`
- Base Path: (empty)

API Configuration section:

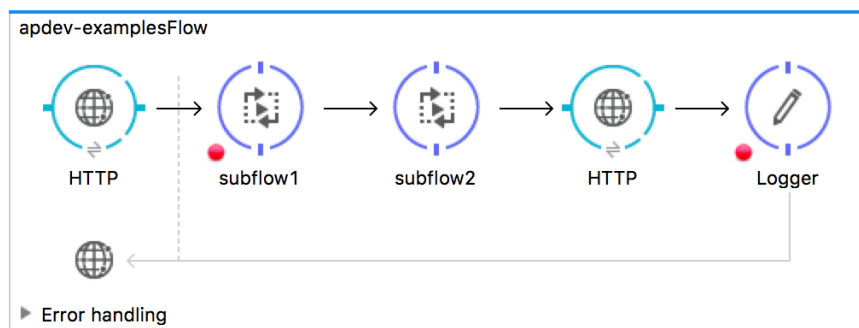
- RAML Location: (empty)

Buttons: Cancel, OK

6. Return to the Message Flow view.
7. Drag an HTTP connector into the Source section of flow2.
8. In the HTTP properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
9. Set the path to /flow2 and the allowed methods to GET.
10. Add a breakpoint to the Set Payload transformer in flow2.



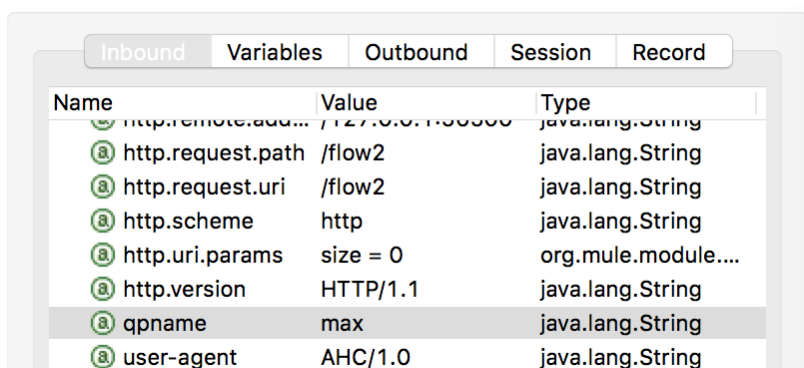
11. Add an HTTP Request endpoint after the flow2 reference in apdev-examplesFlow.
12. Delete the flow2 reference.
13. In the HTTP properties view, set the connector configuration to HTTP_Request_Configuration.
14. Set the path to /flow2 and the method to GET.
15. Add a breakpoint to the Logger.



Debug the application

16. Debug the application.
17. In Postman, send the same request.
18. In the Mule Debugger, step into flow2.

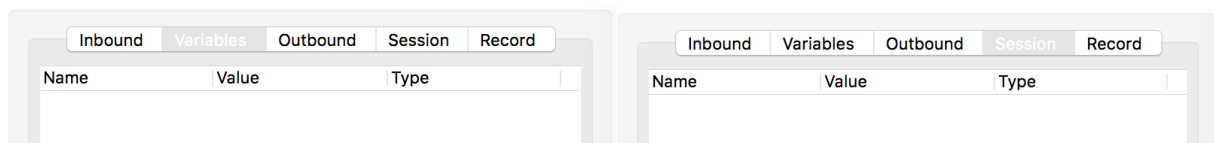
19. Locate the qpname property in the inbound properties.



Name	Value	Type
http.remote.address	/127.0.0.1:8080	java.lang.String
http.request.path	/flow2	java.lang.String
http.request.uri	/flow2	java.lang.String
http.scheme	http	java.lang.String
http.uri.params	size = 0	org.mule.module....
http.version	HTTP/1.1	java.lang.String
qpname	max	java.lang.String
user-agent	AHC/1.0	java.lang.String

20. Look at the outbound properties, the flow variables, and the session variables.

Note: Session variables are persisted across some but not all transport barriers. As you see here, they are not propagated across the HTTP transport barrier.

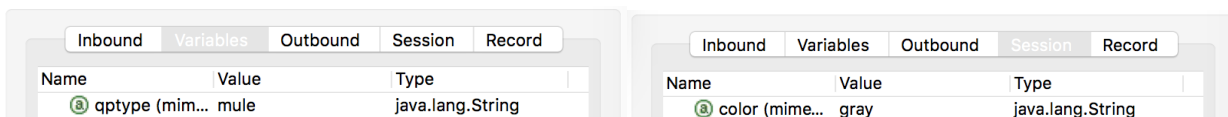


Name	Value	Type
------	-------	------

Name	Value	Type
------	-------	------

21. Step through the flow until the message returns to apdev-examplesFlow.

22. Look at the inbound and outbound properties and the flow and session variables.



Name	Value	Type
qptype (mime...)	mule	java.lang.String

Name	Value	Type
color (mime...)	gray	java.lang.String

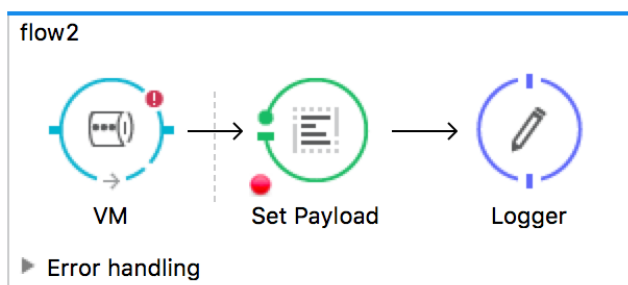
23. Click Resume.

24. Stop the project.

Create a VM transport barrier

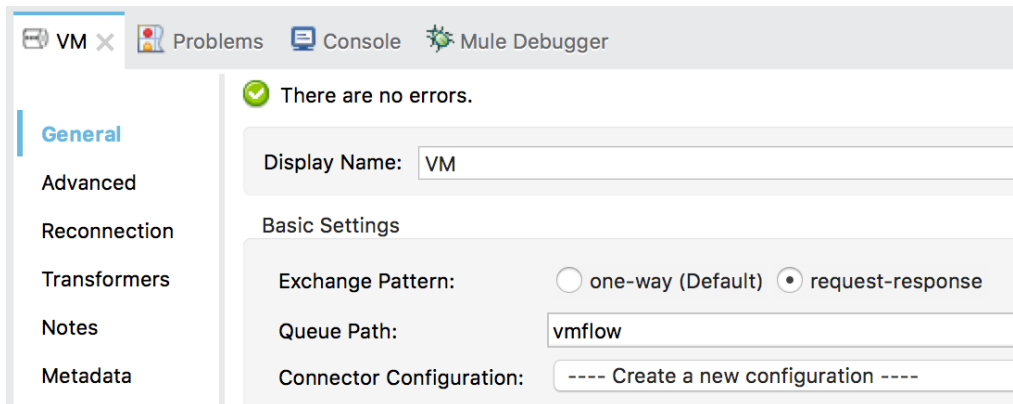
25. In flow2, delete the HTTP Listener endpoint.

26. Drag a VM connector from the Mule Palette into the source section of flow2.



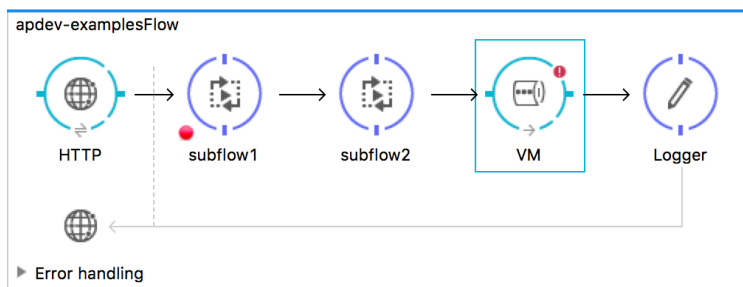
27. In the VM properties view, set the exchange pattern to request-response.

28. Set the queue path to vmflow.



29. In apdev-examplesFlow, add a VM connector endpoint after the HTTP Request endpoint.

30. Delete the HTTP Request endpoint.



31. In the VM properties view, set the exchange pattern to request-response.

32. Set the queue path to vmflow.

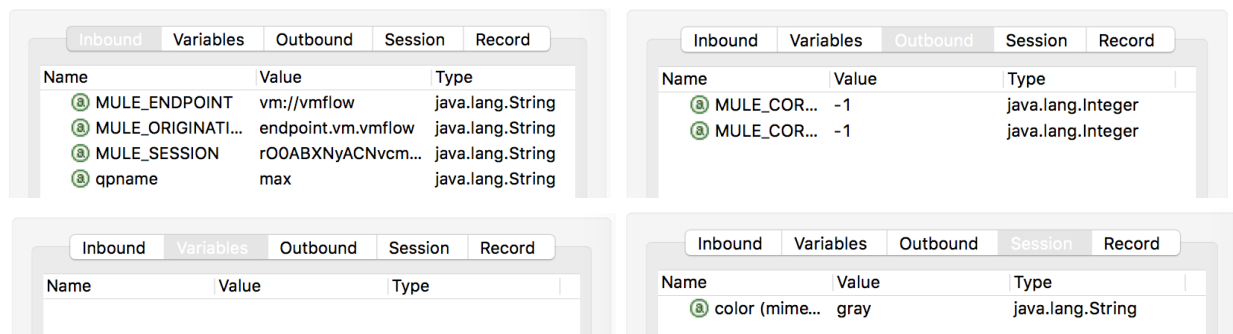
Debug the application

33. Debug the project.

34. In Postman, send the same request.

35. In the Mule Debugger, step through the application into flow2.

36. Look at the inbound and outbound properties and the flow and session variables.



Note: Session variables are persisted across some but not all transport barriers. As you see here, they are propagated across the VM transport barrier.

37. Step through the flow until the message returns to `apdev-examplesFlow`.

38. Look at the inbound and outbound properties and the flow and session variables.

The image displays four screenshots of the MuleSoft IDE, arranged in a 2x2 grid. Each screenshot shows a different view of a message or session variable, with tabs for 'Inbound', 'Variables', 'Outbound', 'Session', and 'Record'. The 'Variables' tab is selected in all four views.

Name	Value	Type
MULE_CORRELATI...	-1	java.lang.Integer
MULE_CORRELATI...	-1	java.lang.Integer
MULE_SESSION	r00ABXNyACNvc...	java.lang.String

Name	Value	Type
MULE_COR...	-1	java.lang.Integer
MULE_COR...	-1	java.lang.Integer
MULE_SESS...	r00ABXNyACNvc...	java.lang.String

Name	Value	Type
qptype (mime...	mule	java.lang.String

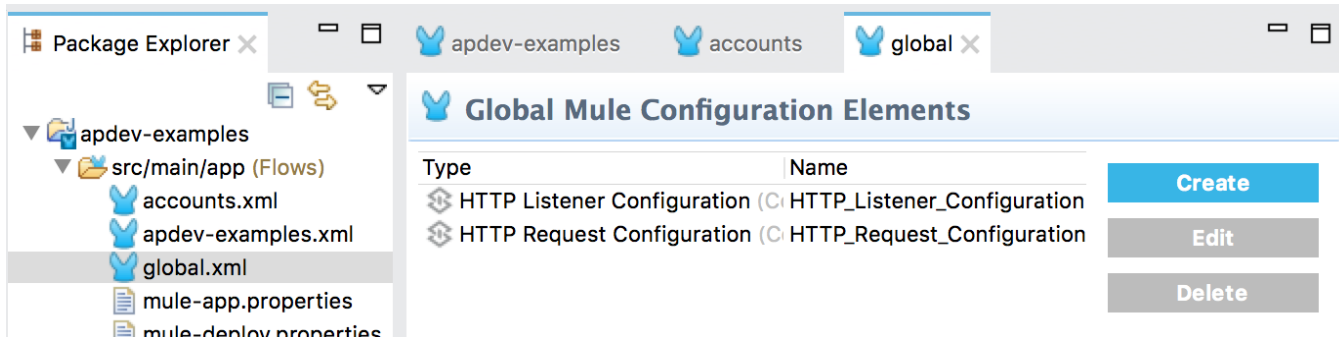
Name	Value	Type
color (mime...	gray	java.lang.String

39. Step through the rest of the application and stop the project.

Walkthrough 6-3: Encapsulate global elements in a separate configuration file

In this walkthrough, you refactor your apdev-examples project. You will:

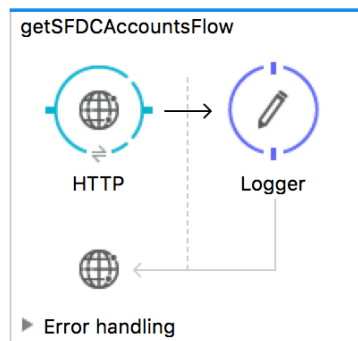
- Create a new configuration file with an endpoint that uses an existing global element.
- Create a configuration file global.xml for just global elements.
- Move the existing global elements to global.xml.



Create a new configuration file

1. Return to the apdev-examples project.
2. Create a new Mule configuration file called accounts.xml.
3. Drag out an HTTP Listener to the canvas.
4. In the HTTP properties view, set the connector configuration to the existing configuration.
5. Set the path to /sfdc and the allowed methods to GET.
6. Add a Logger component to the flow.
7. Change the name of the existing flow to getSFDCAccountsFlow.

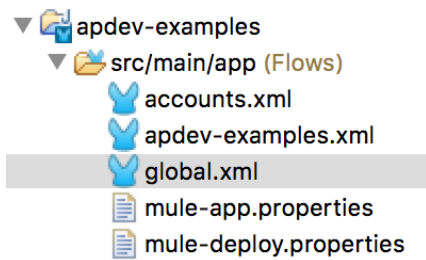
Note: You will use this flow in a later module to retrieve data from Salesforce.



8. Switch to the Global Elements view; you should not see any global elements.

Create a global configuration file

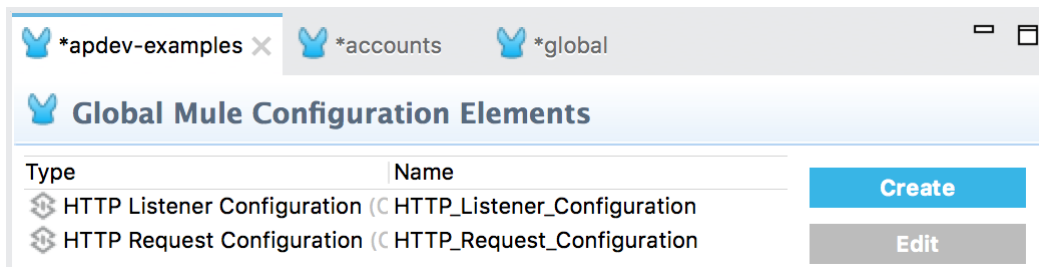
9. Create a new Mule configuration file called global.xml.



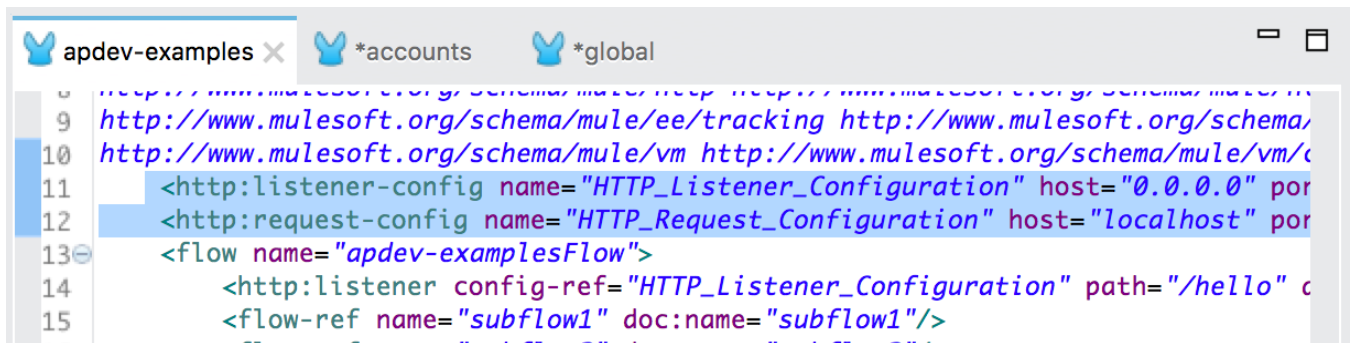
10. In global.xml, switch to the Configuration XML view.
11. Place some empty lines between the start and end mule tags.

Remove the existing global elements

12. Return to apdev-examples.xml.
13. Switch to the Global Elements view and see there are two configurations.



14. Switch to the Configuration XML view.
15. Select and cut the two configuration elements defined before the flows.



Note: If you delete the global elements from the Global Elements view instead, the config-ref values are also removed from the connector endpoints and you need to re-add them.

16. Return to the Message Flow view.

Move the global elements to a new configuration file

17. Return to global.xml.
18. Paste the global elements you cut to the clipboard between the start and end mule tags.



```
8 http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http
9 http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core
10
11 <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0" port="8080">
12 <http:request-config name="HTTP_Request_Configuration" host="localhost" port="8080">
13
14 </mule>
15
```

19. Switch to the Global Elements view; you should see the two configurations.

Test the application

20. Save all the files; any problems should disappear.
21. Return to apdev-examples.xml.
22. Double-click the HTTP Listener connector in apdev-examplesFlow; the connector configuration should still be set to HTTP_Listener_Configuration, which is now defined in global.xml.
23. Run the project.
24. In Postman, send the same request.; you should still get a response of Goodbye.

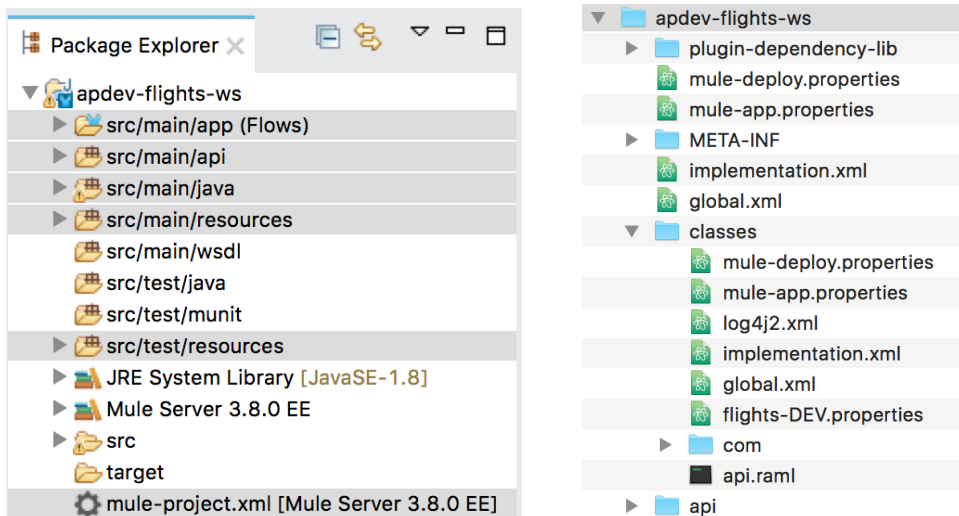
Close the project

25. Return to Anypoint Studio.
26. Stop the project.
27. In the Package Explorer, right-click apdev-examples and select Close Project.

Walkthrough 6-4: Create a well organized Mule project

In this walkthrough, you create a new project for the Mule United Airlines (MUA) flights application that you will build during the course and then review and organize its files and folders. You will:

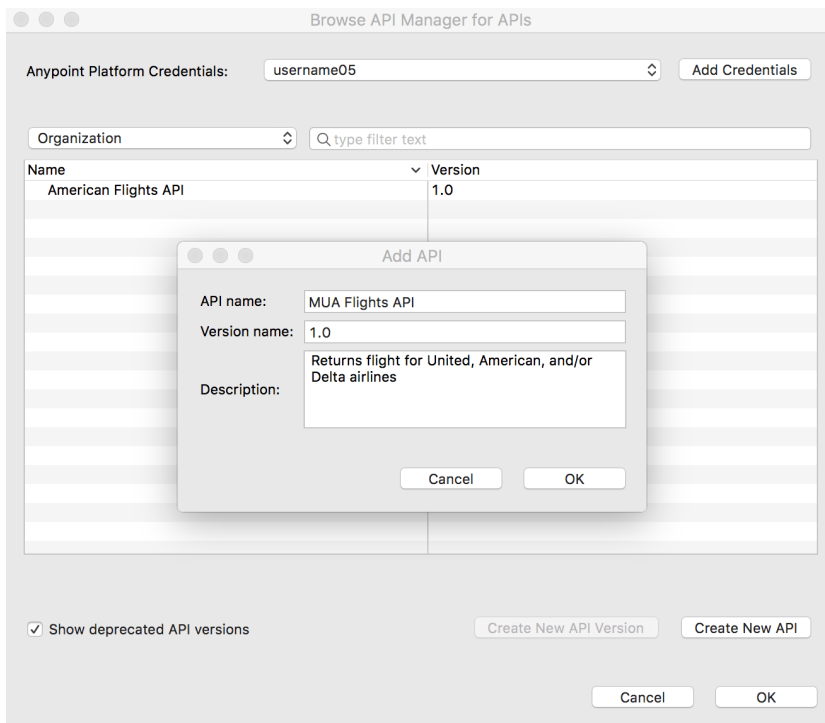
- Create a project with a new RAML file that is added to Anypoint Platform.
- Review the project's configuration and properties files.
- Create an application properties file and a global configuration file for the project.
- Add Java files and test resource files to the project.
- Create and examine the contents of a deployable archive for the project.



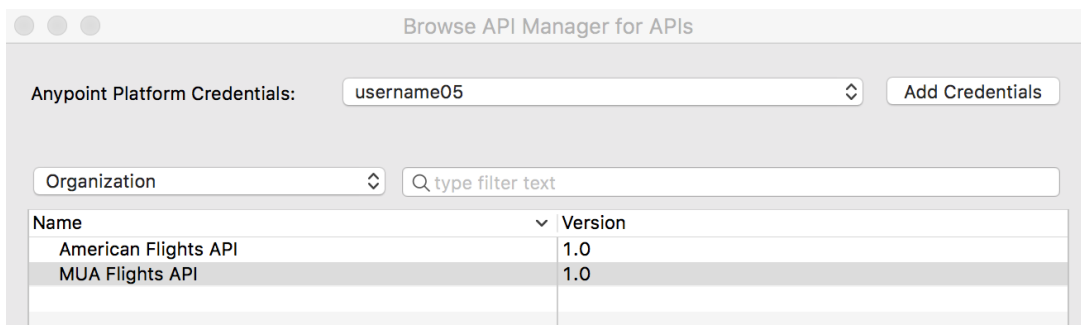
Create a project with a new RAML file that is added to Anypoint Platform

1. Right-click in the Package Explorer and select New > Mule Project.
2. In the New Mule Project dialog box, set the project name to apdev-flights-ws.
3. Check Add APIkit components.
4. Click the browse button next to API Definition and select Anypoint Platform.
5. In the Browse API Manager for APIs dialog box, click the Create New API button.
6. In the Add API dialog box, set the following values:
 - API name: MUA Flights API
 - Version name: 1.0
 - Description: Returns flights for United, American, and/or Delta airlines

7. Click OK.



8. In the Browse API Manager for APIs dialog box, make sure the new MUA Flights API is now listed and selected.

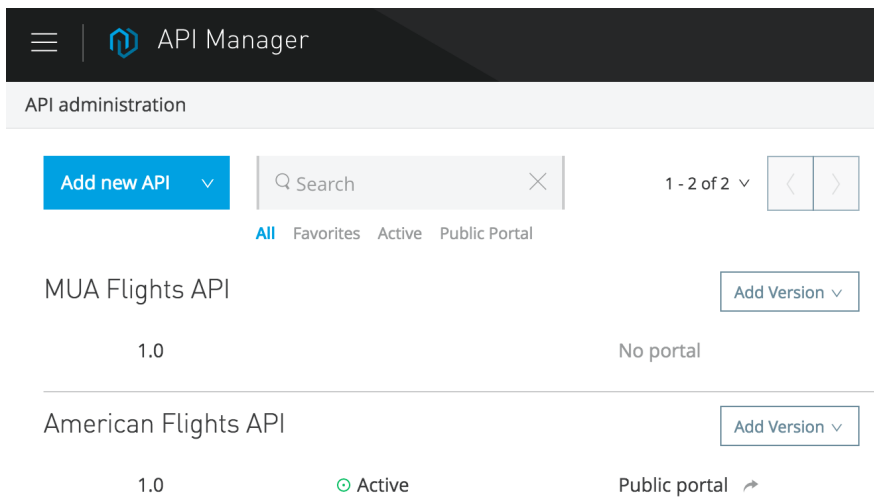


9. Click OK.
10. In the New Mule Project dialog box, click Finish.

Locate the new API on Anypoint Platform

11. Return to Anypoint Platform in a web browser.

12. From the main menu, select API Manager; you should see the new MUA Flights API.



13. Click the 1.0 version of the API.

14. Click the Edit in API designer link and look at the starting RAML code.

15. Click the MUA Flights API – 1.0 link in the menu bar.

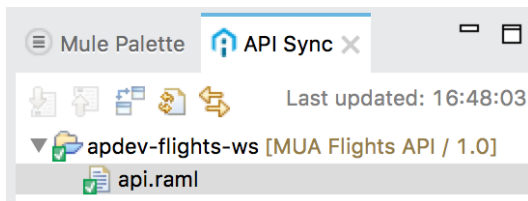
16. Leave this web browser window open.

Locate the new RAML file in Anypoint Studio

17. Return to the apdev-flights-ws project in Anypoint Studio.

18. Make sure you are in the Mule Design perspective.

19. In the API Sync view, locate the new connection to Anypoint Platform.



20. Double-click api.raml; you should see an empty RAML file.

21. In your computer's file browser, locate flights.raml in the resources folder of the student files.

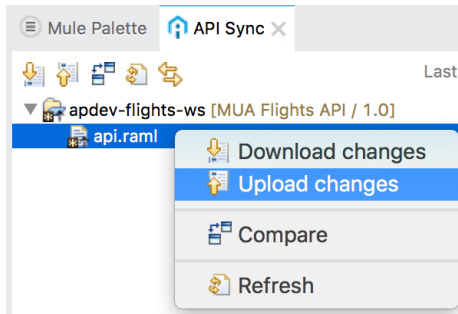
22. Open the file in a text editor and copy the text.

23. Return to api.raml in Anypoint Studio.

24. Delete the existing text and paste the definition you copied.

Note: You are adding the RAML code to the existing api.raml file (instead of adding a new RAML file to the project and deleting the existing one) because an API's main RAML file can't be deleted on Anypoint Platform. You can change its name, but you cannot delete it.

25. Save the file.
26. In the API Sync view, right-click api.raml and select Upload changes.



Review the API on Anypoint Platform

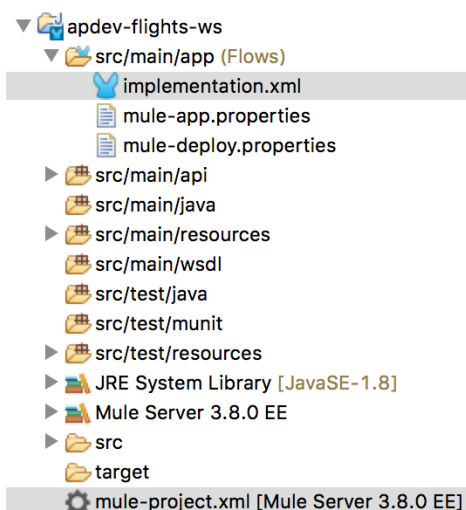
27. Return to Anypoint Platform in a web browser.
28. Click the Edit in API designer link; you should see the modified RAML.

Review project configuration files

29. Return to Anypoint Studio.
30. Look at the apdev-flights-ws.xml file that was created; it should have no contents.

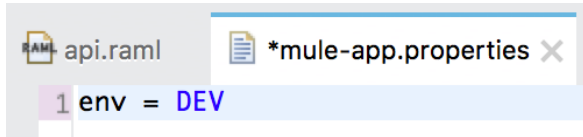
Note: No interface for the API was created because this was a project with a new RAML file with no resources. You can generate the interface later by right-clicking the project and selecting Mule > Generate Flows from RAML.

31. Rename apdev-flight-ws.xml to implementation.xml.
32. Locate mule-project.xml in the project and open it.
33. Review its contents and then close the file.



Review project properties files

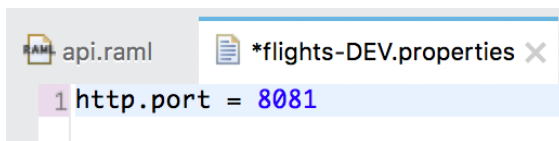
34. Locate mule-app.properties in the project and open it.
35. Add an environment variable called env equal to DEV.



36. Save the file and close it.
37. Locate and open mule-deploy.properties.
38. Review its contents and then close the file.

Create a properties file for application parameters

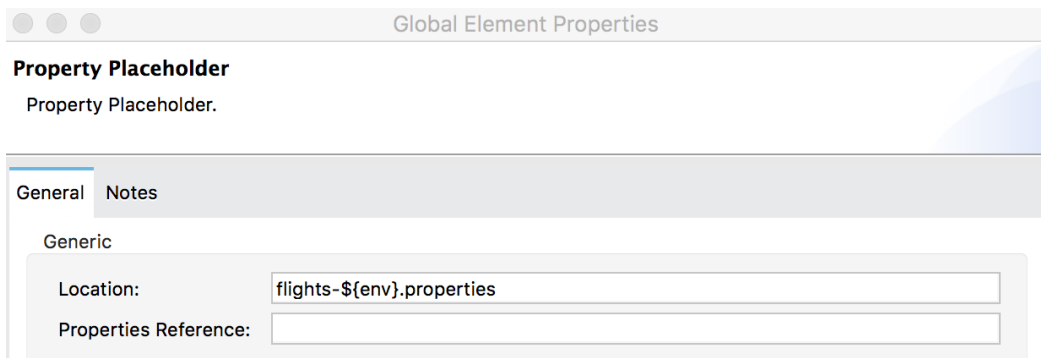
39. Right-click src/main/resources and select New > File.
40. In the New File dialog box, set the file name to flights-DEV.properties and click Finish.
41. Add a property called http.port equal to 8081.



42. Save and close the file.

Create a global configuration file

43. In src/main/app, create a new configuration file called global.xml.
44. Switch to the Global Elements view.
45. Create a new Property Placeholder configuration with a location set to flights-\${env}.properties.



46. Click OK.

47. Create a new HTTP Listener Configuration with a port set to `${http.port}`.

The screenshot shows the 'Global Element Properties' dialog for an 'HTTP Listener Configuration'. The 'General' tab is selected. The 'Name' field is set to 'HTTP_Listener_Configuration'. Under 'URL Configuration', the 'Protocol' is set to 'HTTP (Default)', the 'Host' is 'All Interfaces [0.0.0.0] (Default)', the 'Port' is `${http.port}`, and the 'Base Path' is empty.

48. Confirm you now have two global elements defined in the global.xml file.

The screenshot shows the 'Global Mule Configuration Elements' table. It has two columns: 'Type' and 'Name'. There are two rows: 'Property Placeholder (Config)' with name 'Property Placeholder' and 'HTTP Listener Configuration (HTTP_Listener_Configuration)'. To the right of the table are 'Create' and 'Edit' buttons.

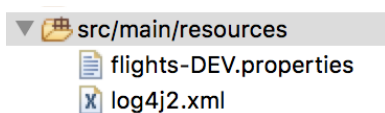
Type	Name	
Property Placeholder (Config)	Property Placeholder	Create
HTTP Listener Configuration (HTTP_Listener_Configuration)	HTTP_Listener_Configuration	Edit

49. Save and close the file.

Review src/main/resources

50. In `src/main/resources`, open `log4j2.xml`.

51. Review and then close the file.



Add Java files to src/main/Java

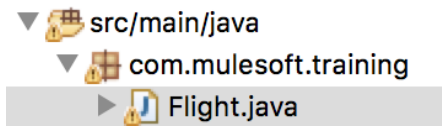
52. In your computer's file browser, locate the `com` folder in the `java` folder of the student files.

53. Drag the `com` folder into the `src/main/java` folder in the Anypoint Studio `apdev-flights-ws` project.

54. Expand the new `com` directory.

55. Open the `Flight.java` file.

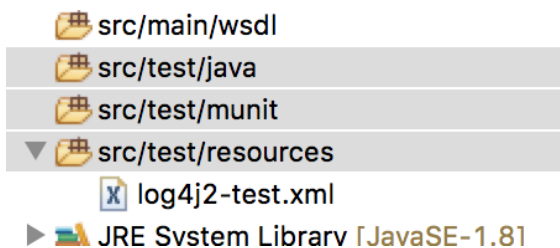
56. Review the code and then close the file.



Review src/test folders

57. In the project, locate the three src/test folders.

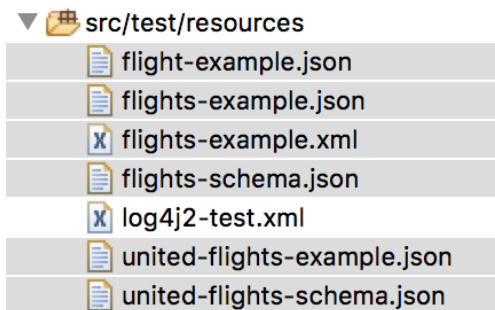
58. Expand the src/test/resources folder.



Add test resources

59. In your computer's file browser, expand the schema-and-examples folder in the student files.

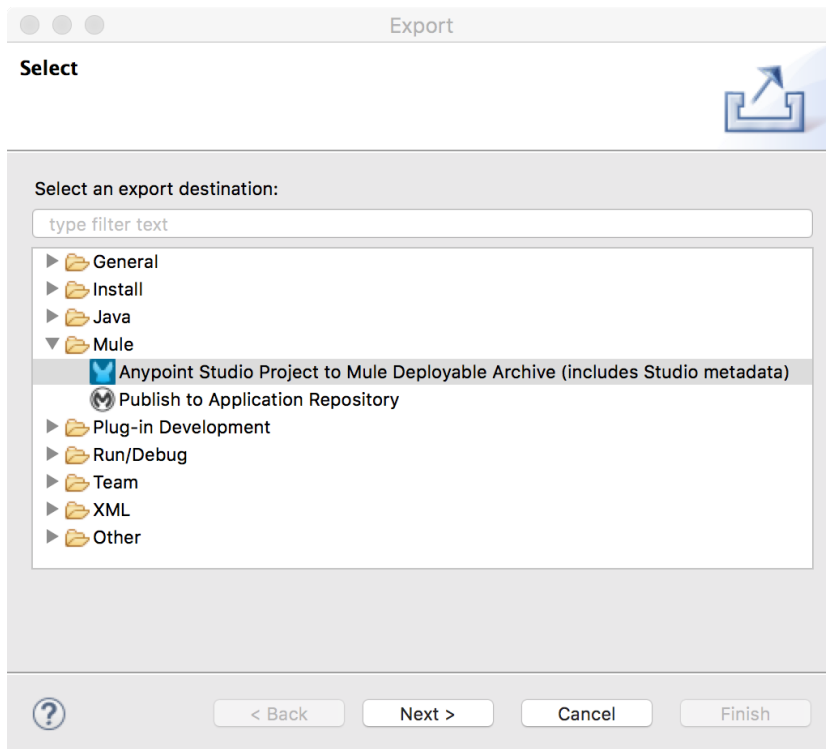
60. Select the four flights and two united-flights JSON files and drag them into the src/test/resources folder in the Anypoint Studio apdev-flights-ws project.



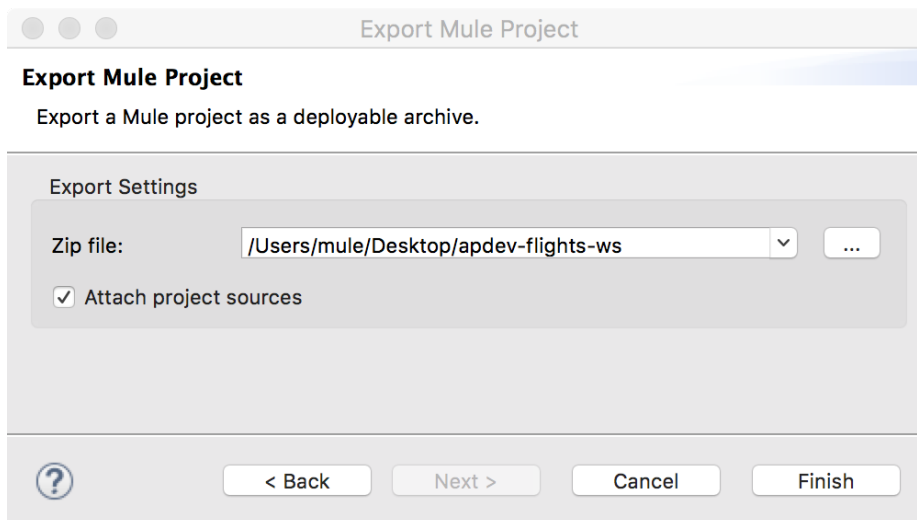
Examine the contents of a deployable archive

61. In Anypoint Studio, right-click the project and select Export.

62. In the Export dialog box, select Mule > Anypoint Studio to Mule Deployable Archive and click Next.



63. Set the Zip file to a location that you can find and click Finish.



64. In your computer's file browser, locate the ZIP file and expand it.

65. Open the resulting apdev-flights-ws folder.

66. Expand the classes folder and examine the contents.

