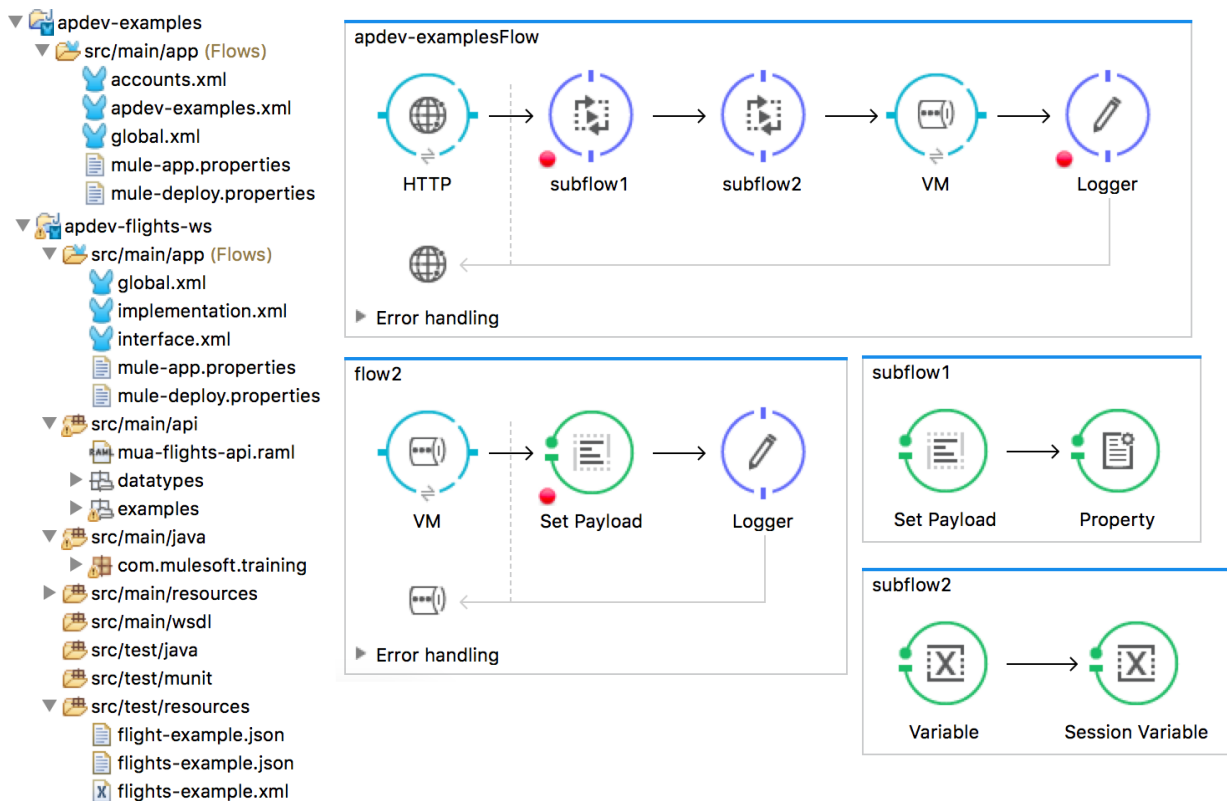


Module 7: Structuring Mule Applications



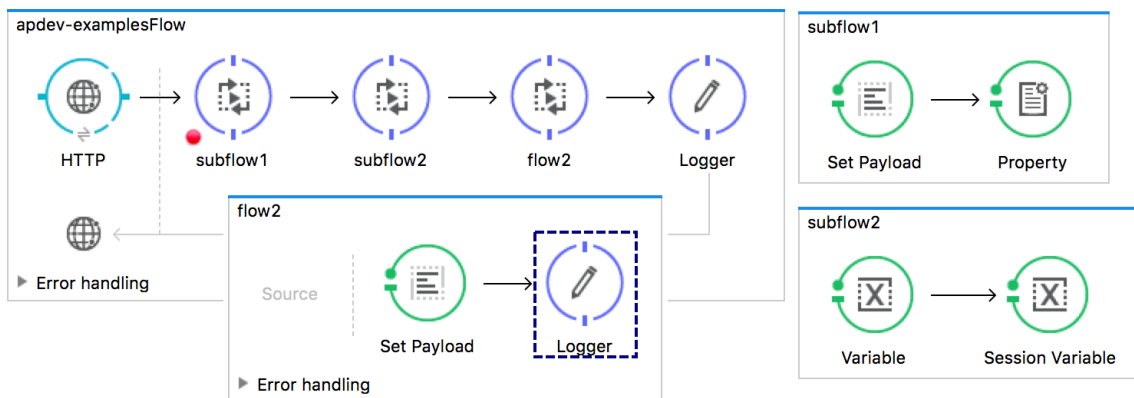
At the end of this module, you should be able to:

- 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 Mule projects.

Walkthrough 7-1: Create and reference flows and subflows

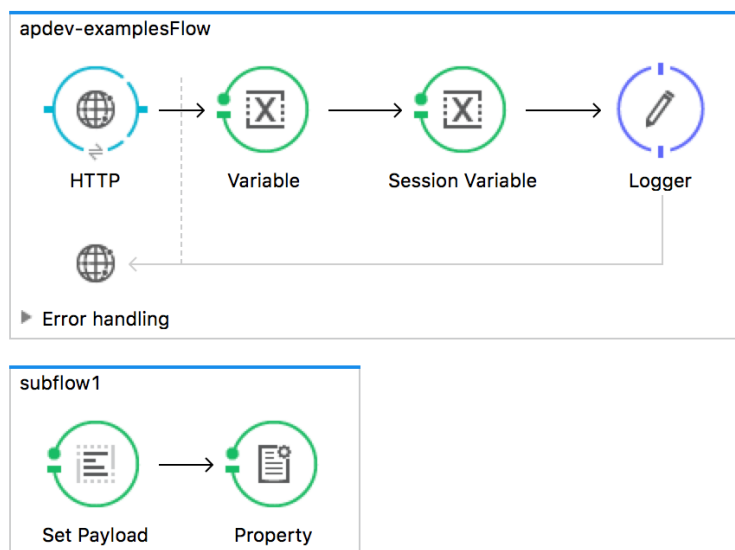
In this walkthrough, you continue to work with `apdev-examples.xml`. You will:

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



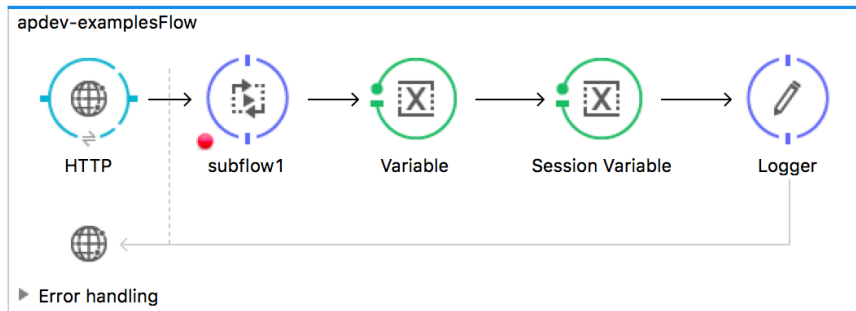
Create a subflow

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`.



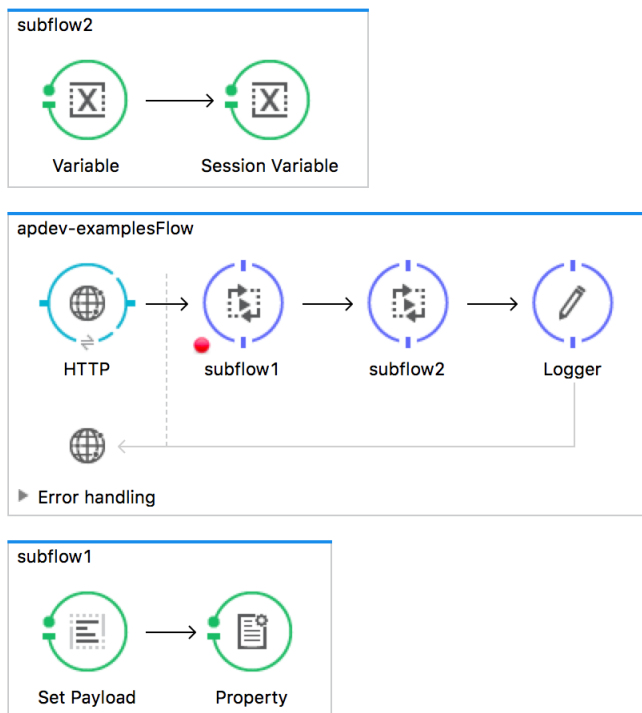
Reference a subflow

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.
7. Add a breakpoint to the subflow1 Flow Reference.



Extract processors into a subflow

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



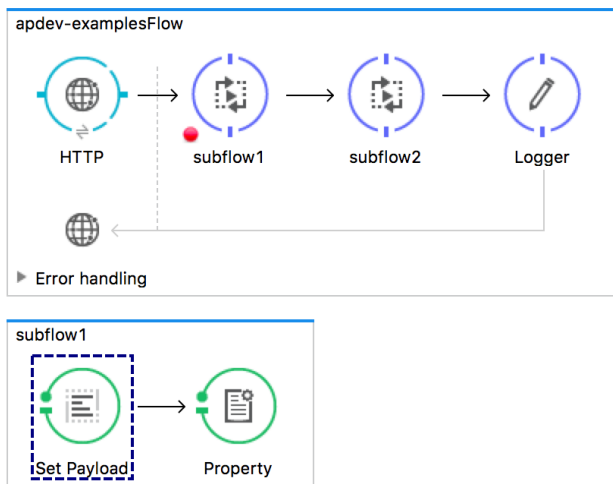
12. Drag subflow2 below subflow1 in the canvas.

Debug the application

13. Debug the project.

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

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



16. In Postman, send the same request again.

17. 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

18. Switch perspectives.

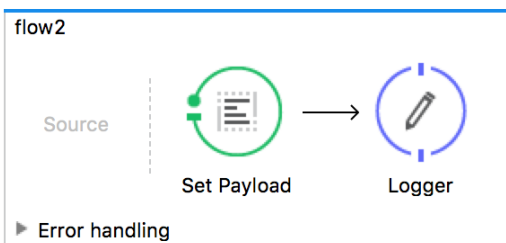
19. Drag a Flow scope element to the canvas and drop it beneath the existing flows.

20. Change the flow name to flow2.

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

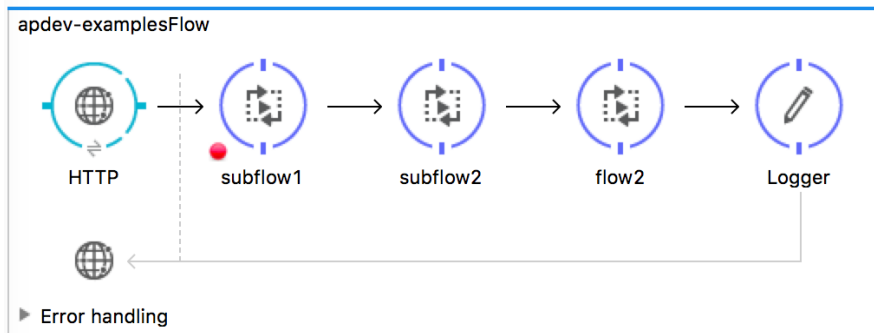
22. In the Set Payload properties view, set the value to Goodbye.

23. Add a Logger after the Set Payload transformer.



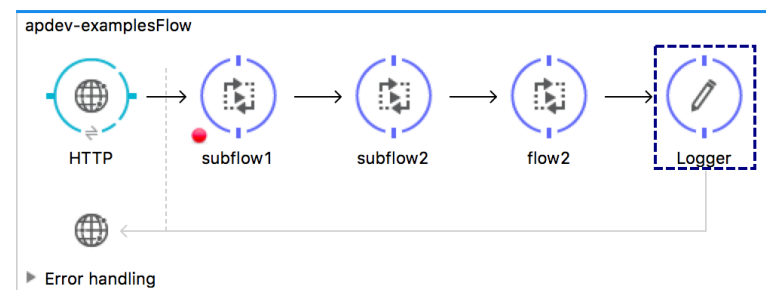
Reference a flow

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



Debug the application

26. Save the file to redeploy the application in debug mode.
27. In Postman, send the same request.
28. 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

The screenshot shows the Mule Debugger interface. At the top, there are tabs for 'Mule Debugger', 'Console', 'Problems', and 'Mule Properties'. Below these, there are two main panels. The left panel shows a tree view of the message flow components. The right panel shows the 'Inbound' tab, which displays the message 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

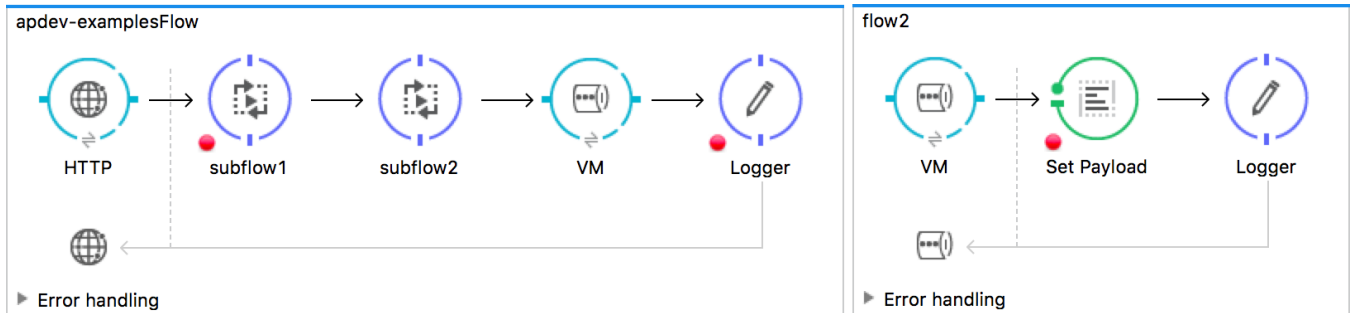
Name	Value	Type
qptype (mime...	mule	java.lang.Stri

29. In Postman, send the same request again.
30. 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.
31. Stop the project and switch perspectives.

Walkthrough 7-2: 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.

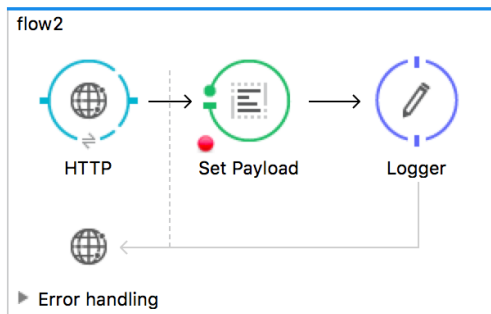
HTTP Request Configuration

Create reusable HTTP request manually or by adding your REST API definition

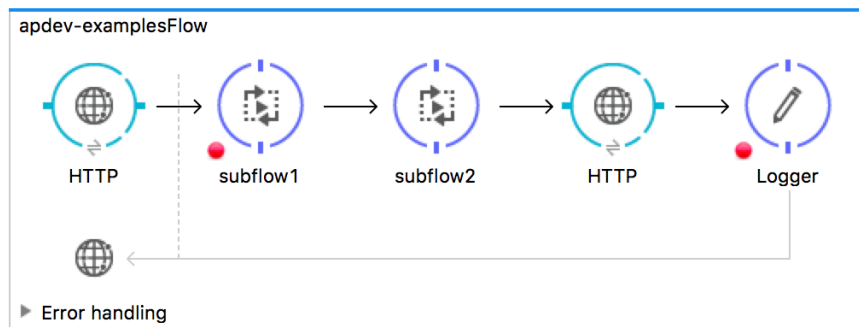
The screenshot shows the 'HTTP Request Configuration' dialog box in MuleSoft. The 'General' tab is selected. The 'Generic' section has a 'Name' field with the value 'HTTP_Request_Configuration'. The 'URL Configuration' section has a 'Protocol' dropdown set to 'HTTP', a 'Host' field with the value 'localhost', a 'Port' field with the value '8081', and an empty 'Base Path' field.

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.



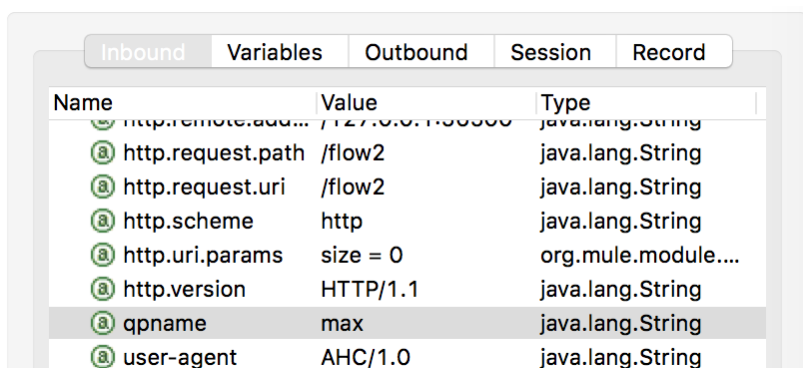
11. Add an HTTP Request endpoint after the flow2 reference in apdev-examplesFlow.
12. Delete the flow2 reference.
13. In the HTTP properties view, set the connector configuration to HTTP_Request_Configuration.
14. Set the path to /flow2 and the method to GET.
15. Add a breakpoint to the Logger.



Debug the application

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

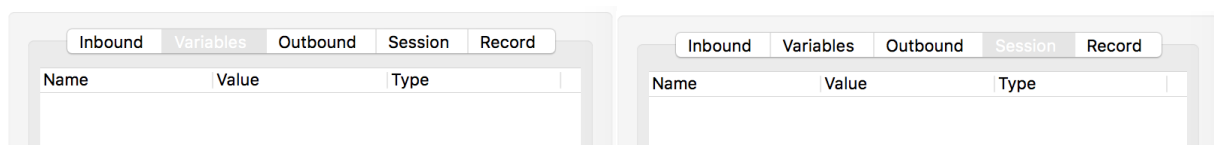
19. Locate the qpname property in the inbound properties.



Name	Value	Type
http.remote.add...	/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.



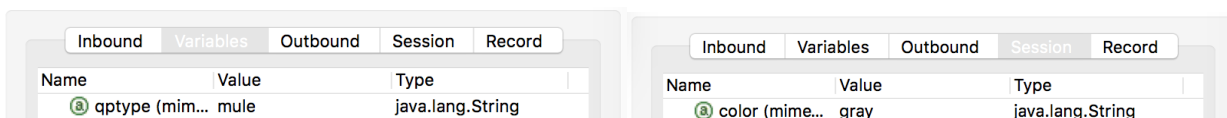
Name	Value	Type
------	-------	------

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

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

Note: If the application times out, debug it again and this time use the Resume button to quickly step to the breakpoint in flow2 and then step through 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

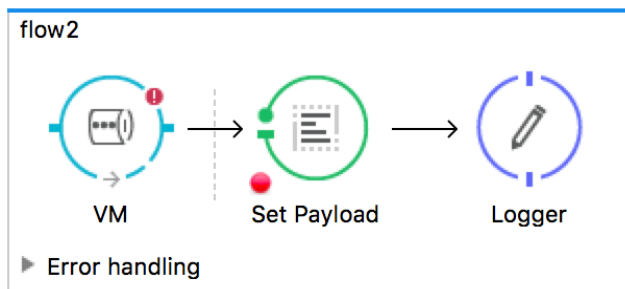
23. Click Resume.

24. Stop the project and switch perspectives.

Create a VM transport barrier

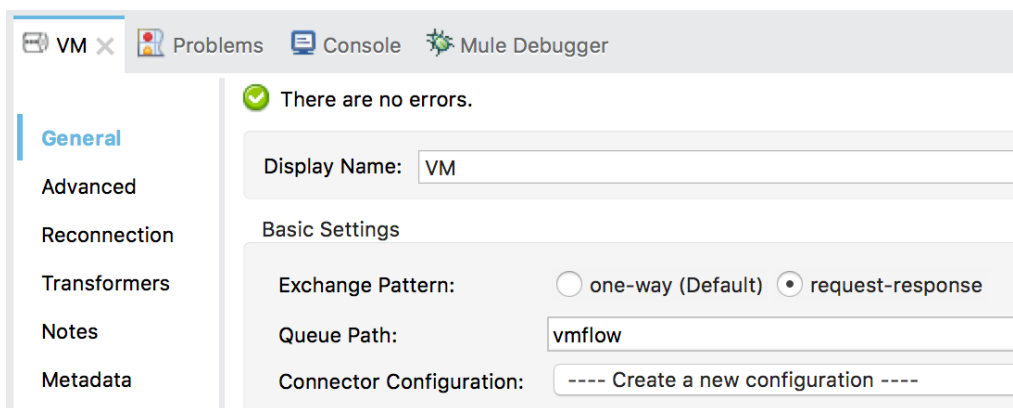
25. In flow2, delete the HTTP Listener endpoint.

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



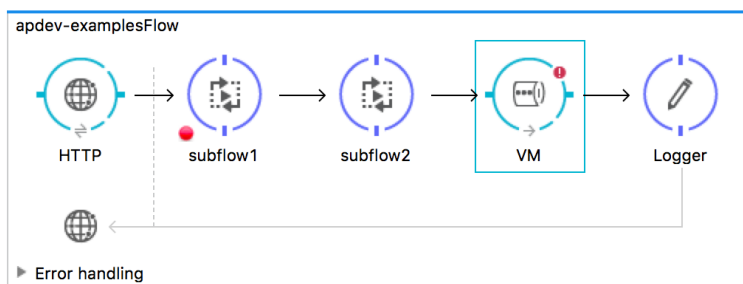
27. In the VM properties view, set the exchange pattern to request-response.

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.

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
MULE_ENDPOINT	vm://vmflow	java.lang.String		
MULE_ORIGINATI...	endpoint.vm.vmflow	java.lang.String		
MULE_SESSION	r00ABXNyACNvc...	java.lang.String		
qpname	max	java.lang.String		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
MULE_COR...	-1	java.lang.Integer		
MULE_COR...	-1	java.lang.Integer		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
color (mime...	gray	java.lang.String		

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.

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
MULE_CORRELATI...	-1	java.lang.Integer		
MULE_CORRELATI...	-1	java.lang.Integer		
MULE_SESSION	r00ABXNyACNvc...	java.lang.String		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
MULE_COR...	-1	java.lang.Integer		
MULE_COR...	-1	java.lang.Integer		
MULE_SESS...	r00ABXNyACNvc...	java.lang.String		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
qptype (mim...	mule	java.lang.String		

Inbound	Variables	Outbound	Session	Record
Name	Value	Type		
color (mime...	gray	java.lang.String		

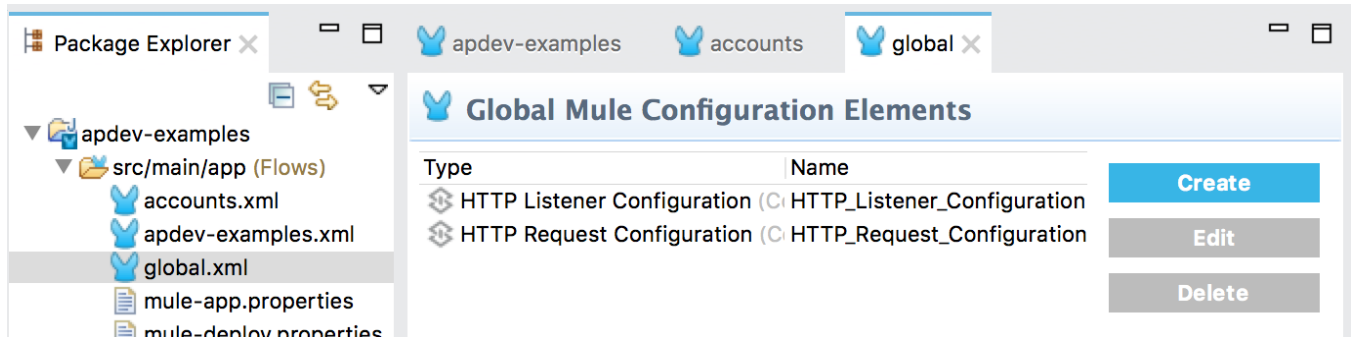
39. Step through the rest of the application.

40. Stop the project and switch perspectives.

Walkthrough 7-3: Encapsulate global elements in a separate configuration file

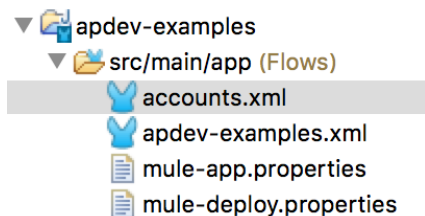
In this walkthrough, you refactor your apdev-examples project. You will:

- 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

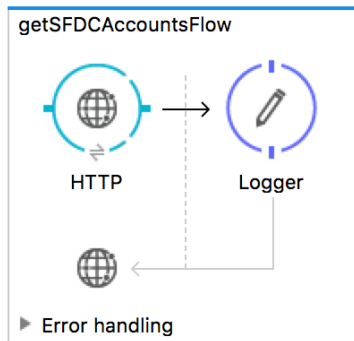
1. Return to the apdev-examples project.
2. In the Package Explorer, right-click the project and select New > Mule Configuration File.
3. In the New Mule Configuration File dialog box, set the name to accounts.xml and click Finish.



4. Drag out an HTTP Listener to the canvas.
5. In the HTTP properties view, set the connector configuration to the existing configuration.
6. Set the path to /sfdc and the allowed methods to GET.
7. Add a Logger component to the flow.

8. Change the name of the existing flow to getSFDCAccountsFlow.

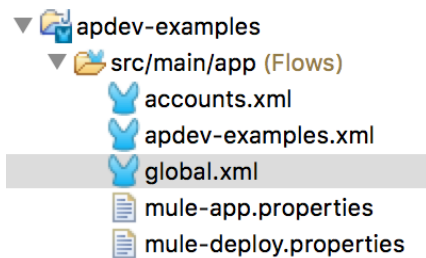
Note: You will use this flow in a later module to retrieve data from Salesforce.



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

Create a global configuration file

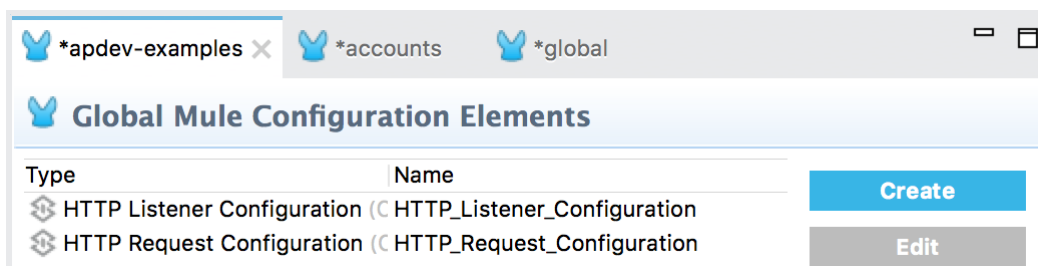
10. Create a new Mule configuration file called global.xml.



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

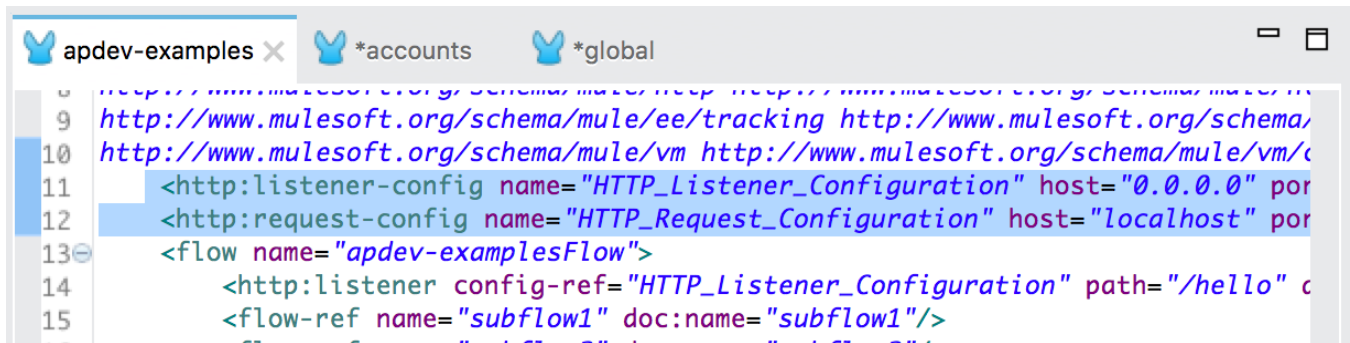
Remove the existing global elements

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



15. Switch to the Configuration XML view.

16. Select and cut the two configuration elements defined before the flows.



```
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 http://www.mulesoft.org/schema/mule/vm http://www.mulesoft.org/schema/mule/vm/core
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 <flow name="apdev-examplesFlow">
14 <http:listener config-ref="HTTP_Listener_Configuration" path="/hello" response="Goodbye">
15 <flow-ref name="subflow1" doc:name="subflow1"/>
16 </flow>
17 </http:listener-config>
18 </http:request-config>
19 </mule>
```

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.

17. Return to the Message Flow view.

Move the global elements to a new configuration file

18. Return to global.xml.
19. 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
```

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

Test the application

21. Save all the files; any problems should disappear.
22. Return to apdev-examples.xml.
23. 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.
24. Run the project.
25. In Postman, send the same request; you should still get a response of Goodbye.

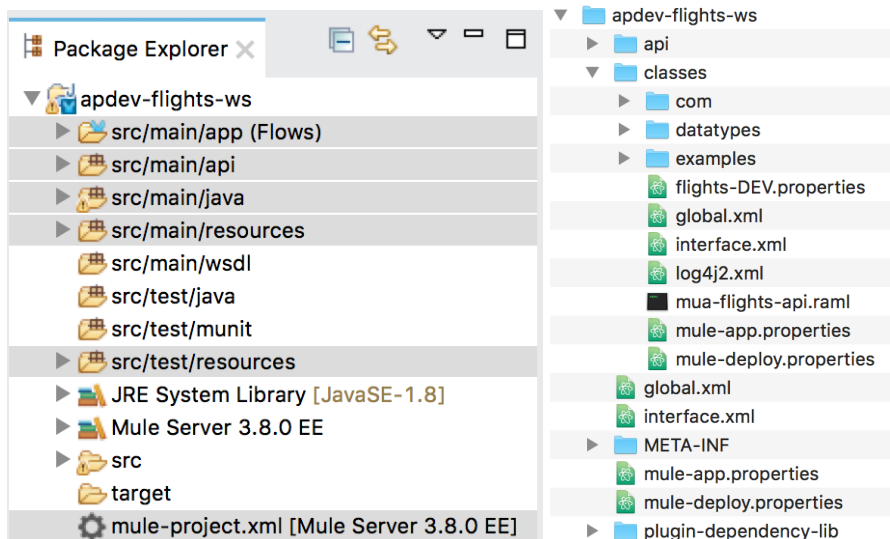
Close the project

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

Walkthrough 7-4: 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:

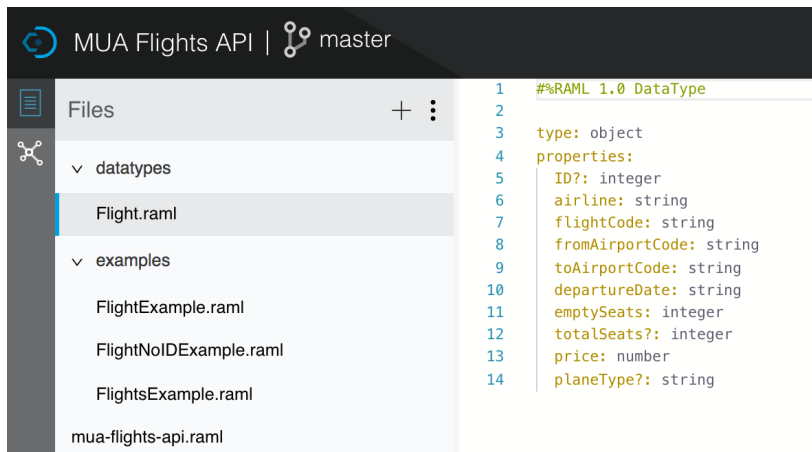
- Create a project based on a new API in Anypoint Platform Design Center.
- Review the project's configuration and properties files.
- Create an application properties file and a global configuration file for the project.
- Add Java files and test resource files to the project.
- Create and examine the contents of a deployable archive for the project.



Create a new API in Design Center

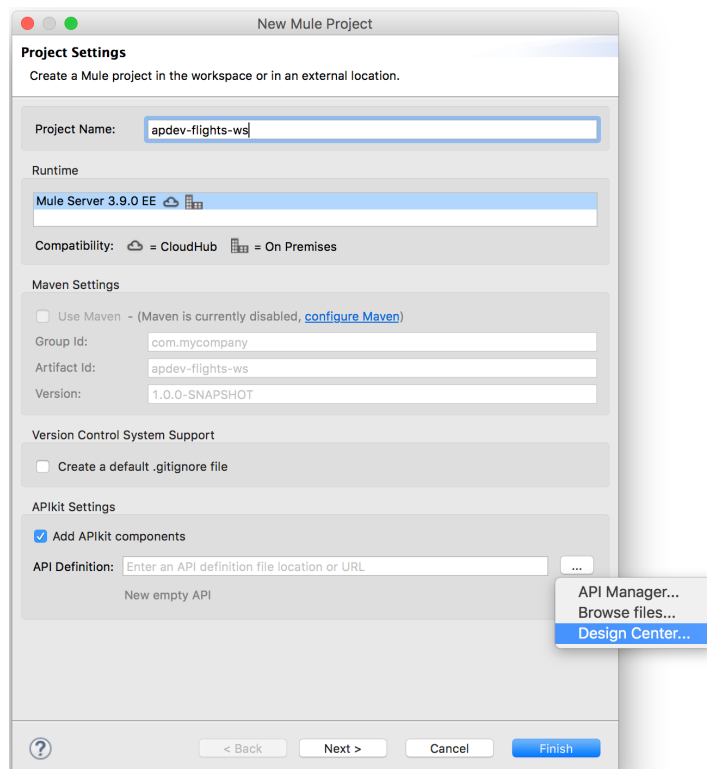
1. Return to Anypoint Platform in a web browser.
2. In Design Center, create a new API Specification project called MUA Flights API.
3. In API designer, click the options menu in the file browser and select Import.
4. In the Import dialog box, browse to your student files and select the MUA-Flights-API.zip located in the resources folder.
5. Click Import.
6. In the Replace dialog box, click Replace file.

7. Explore the API; be sure to look at what resources are defined and the structure of the Flight data type.

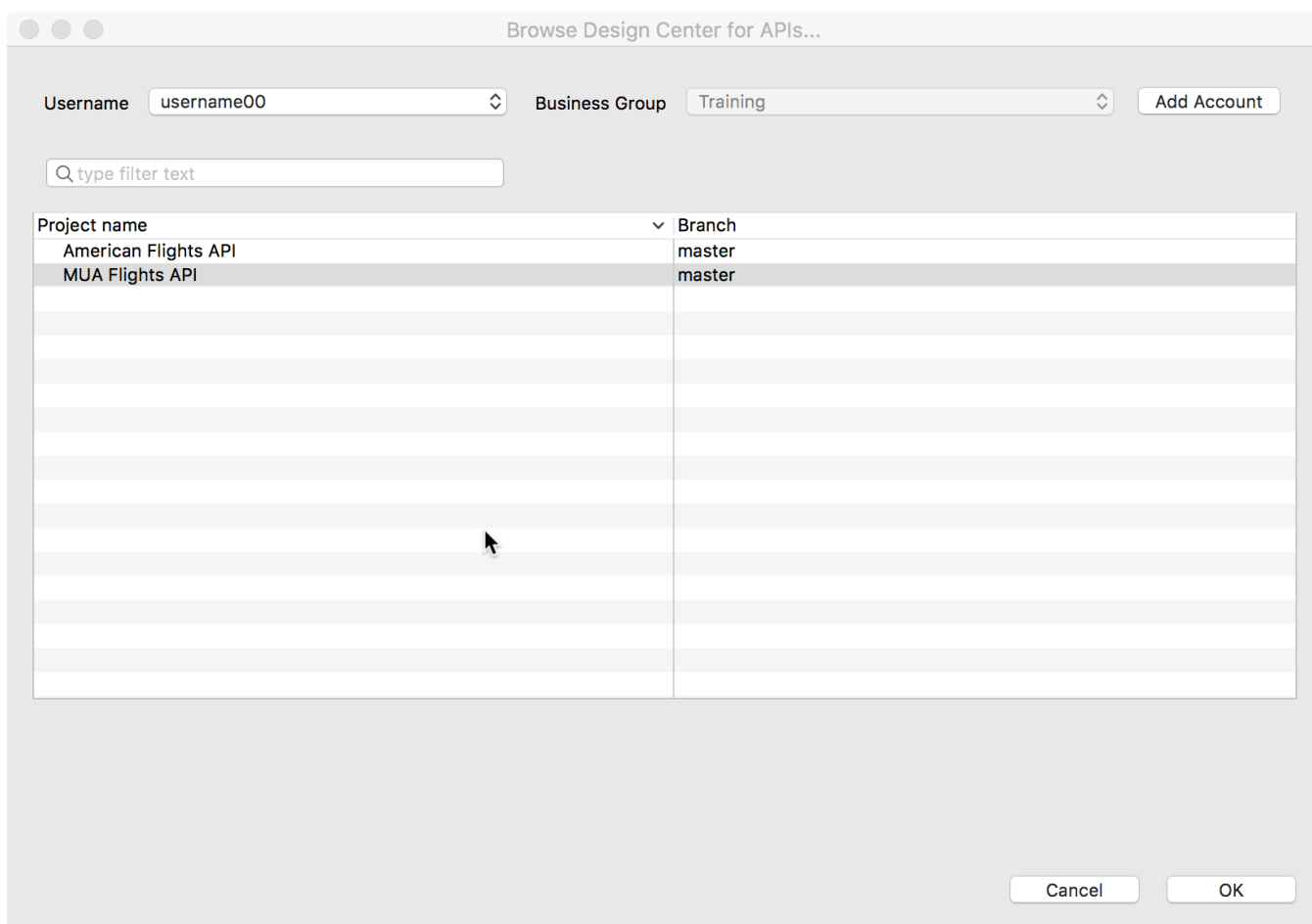


Create a new project to implement this API in Anypoint Studio

8. Return to Anypoint Studio.
9. Right-click in the Package Explorer and select New > Mule Project.
10. In the New Mule Project dialog box, set the project name to apdev-flights-ws.
11. Check Add APIkit components.
12. Click the browse button next to API Definition and select Design Center.



13. In the Browse API Design Center for APIs dialog box, select MUA Flights API and click OK.

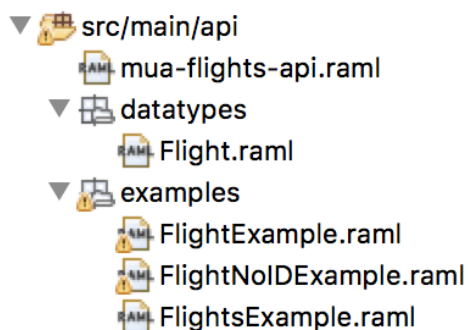


14. In the New Mule Project dialog box, click Finish.

Locate the new RAML files in Anypoint Studio

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

16. In the Package Explorer, expand the `src/main/api` folder; you should see the MUA Flights API files.



17. Open mua-flights-api.raml and review the file.

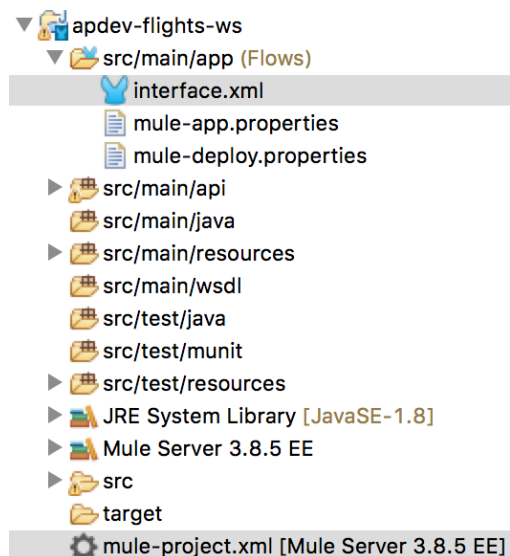
Review project configuration files

18. Look at the mua-flights-api.xml file that was created; it should have a get:/flights flow.

19. Rename mua-flights-api.xml to interface.xml.

20. Locate mule-project.xml in the project and open it.

21. Review its contents and then close the file.

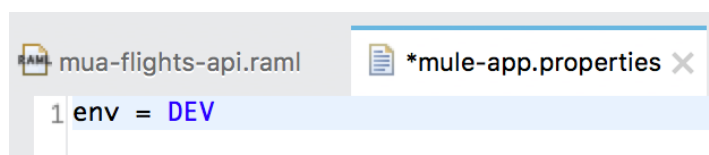


22. Create a new Mule configuration file called implementation.xml.

Review project properties files

23. Locate mule-app.properties in the project and open it.

24. Add an environment variable called env equal to DEV.



25. Save the file and close it.

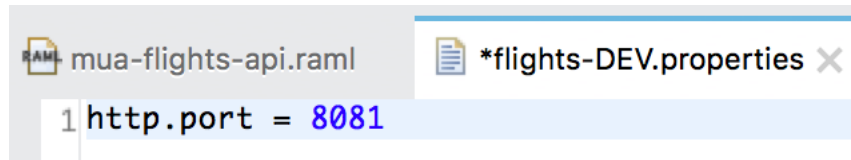
26. Locate and open mule-deploy.properties.

27. Review its contents and then close the file.

Create a properties file for application parameters

28. Right-click src/main/resources and select New > File.

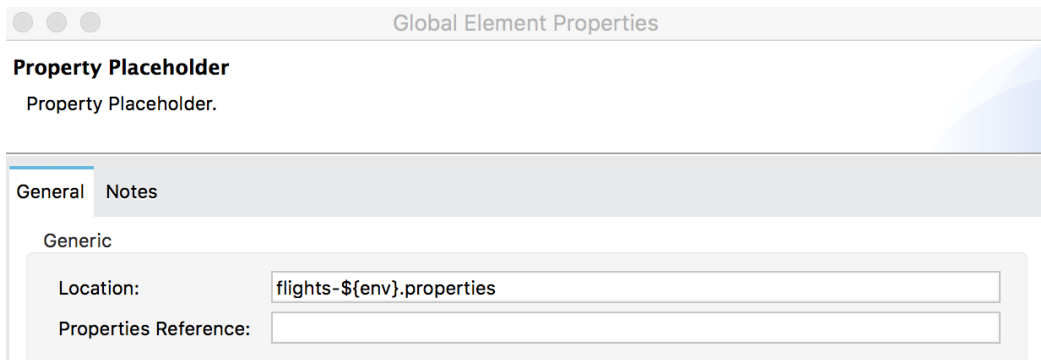
29. In the New File dialog box, set the file name to flights-DEV.properties and click Finish.
30. Add a property called http.port equal to 8081.



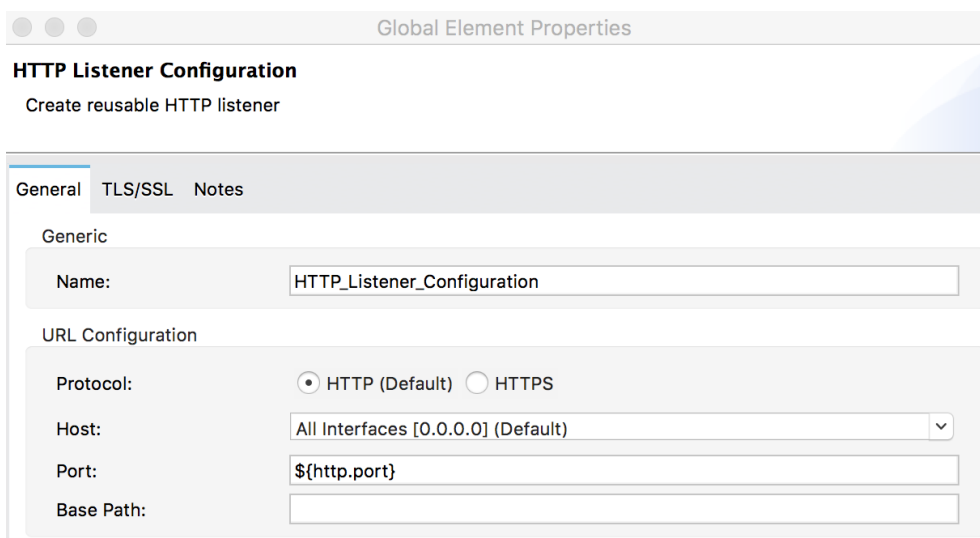
31. Save and close the file.

Create a global configuration file

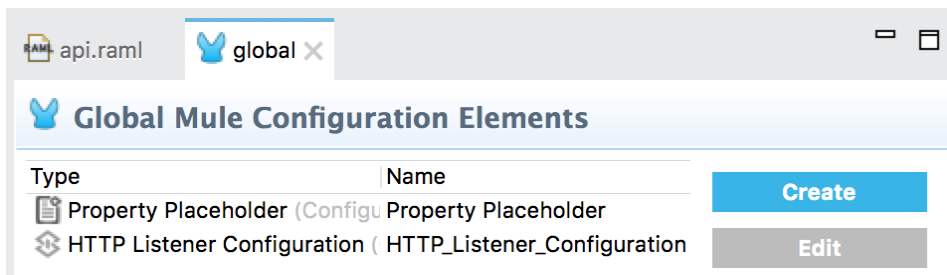
32. In src/main/app, create a new Mule configuration file called global.xml.
33. Switch to the Global Elements view.
34. Create a new Property Placeholder configuration with a location set to flights-\${env}.properties.



35. Click OK.
36. Create a new HTTP Listener Configuration with a port set to \${http.port}.



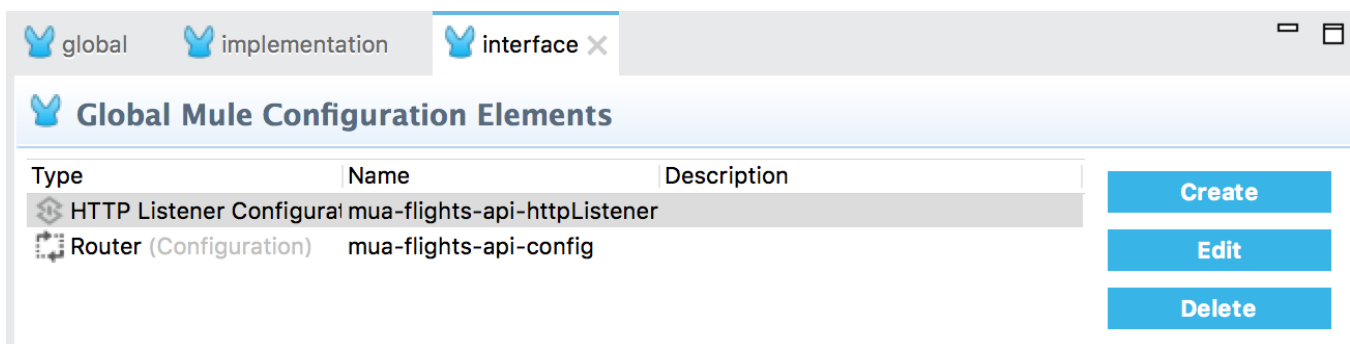
37. Confirm you now have two global elements defined in the global.xml file.



38. Save and close the file.

39. Go to the Global Elements view in interface.xml.

40. Delete the mua-flights-api-httpListener HTTP Listener Configuration.



41. Return to the Message Flow view.

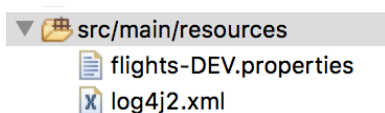
42. In the Properties view for the HTTP Listener in mua-flights-api-main, set the connector configuration to HTTP_Listener_Configuration.

43. In the Properties view for the HTTP Listener in mua-flights-api-console, set the connector configuration to HTTP_Listener_Configuration.

Review src/main/resources

44. In src/main/resources, open log4j2.xml.

45. Review and then close the file.



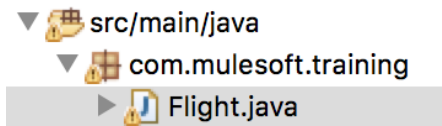
Add Java files to src/main/Java

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

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

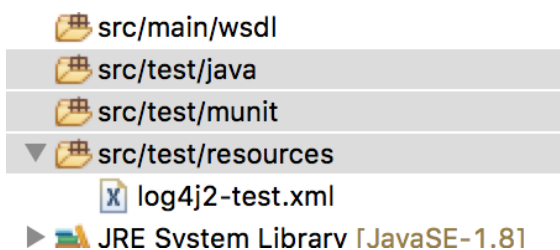
48. Expand the new com directory.

49. Open the Flight.java file.
50. Review the code and then close the file.



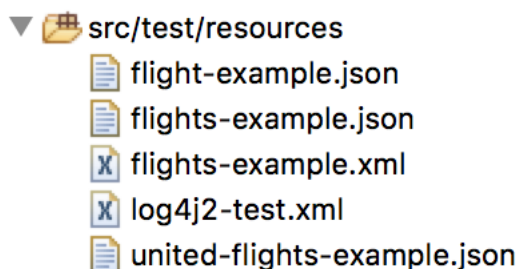
Review src/test folders

51. In the project, locate the three src/test folders.
52. Expand the src/test/resources folder.



Add test resources

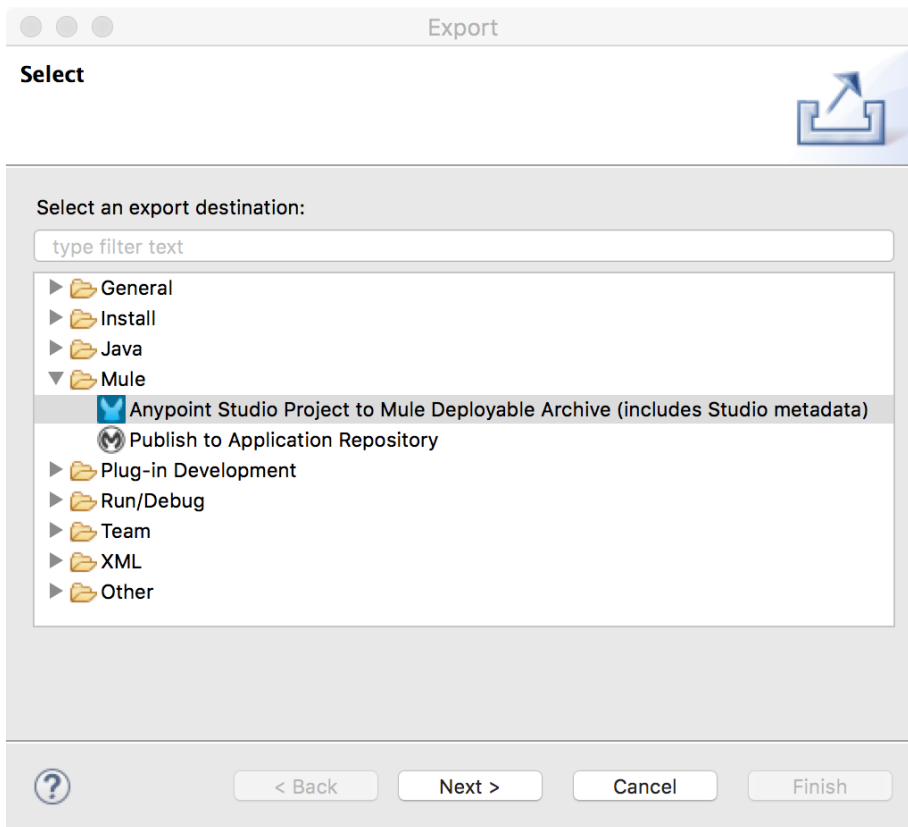
53. In your computer's file browser, expand the examples folder in the student files.
54. Select the four flights and one united-flights files (JSON AND XML) and drag them into the src/test/resources folder in the Anypoint Studio apdev-flights-ws project.



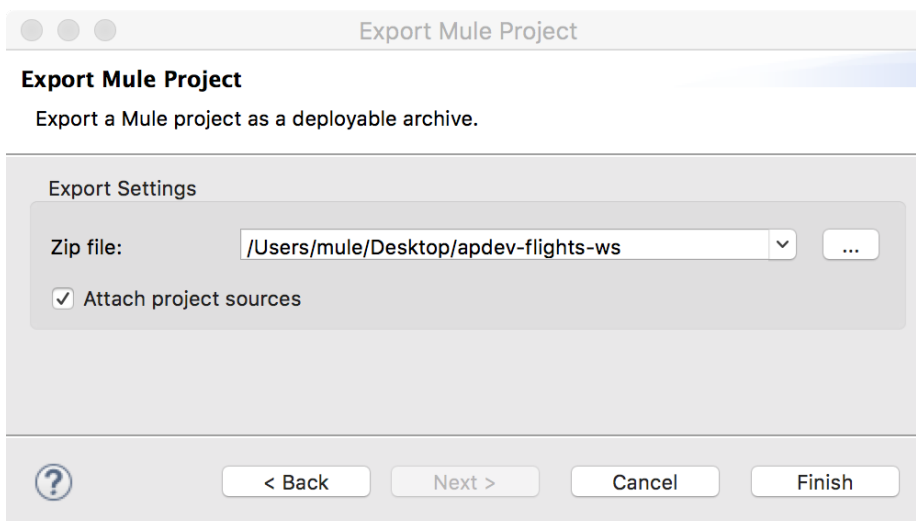
Examine the contents of a deployable archive

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

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



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



58. In your computer's file browser, locate the ZIP file and expand it.
59. Open the resulting apdev-flights-ws folder.

60. Expand the classes folder and examine the contents.

