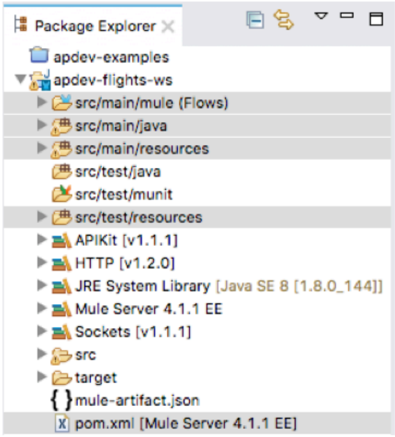
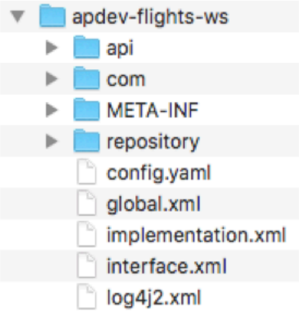
Walkthrough 7-5: Create a well-organized Mule project

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.



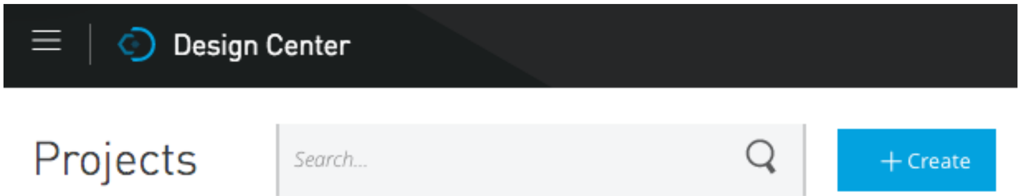
```
Package Explorer
  apdev-examples
▼ apdev-flights-ws
   ▶ src/main/mule (Flows)
   ▶ src/main/java
   ▶ src/main/resources
     src/test/java
     src/test/munit
   ▶ src/test/resources
   ▶ APIKit [v1.1.1]
   ▶ HTTP [v1.2.0]
   ▶ JRE System Library [Java SE 8 [1.8.0_144]]
   ▶ Mule Server 4.1.1 EE
   ▶ Sockets [v1.1.1]
   ▶ src
   ▶ target
     mule-artifact.json
     pom.xml [Mule Server 4.1.1 EE]
```



```
▼ apdev-flights-ws
   ▶ api
   ▶ com
   ▶ META-INF
   ▶ repository
     config.yaml
     global.xml
     implementation.xml
     interface.xml
     log4j2.xml
```

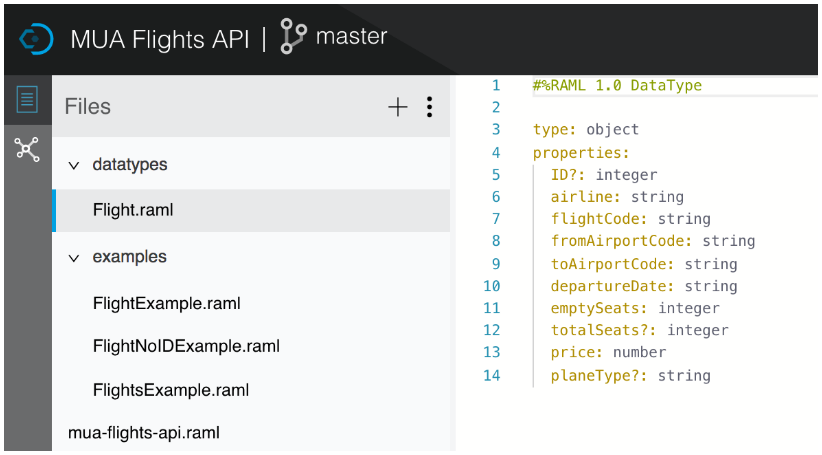
Create a new API in Design Center

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, the name of the query parameter, and the structure of the Flight data type.

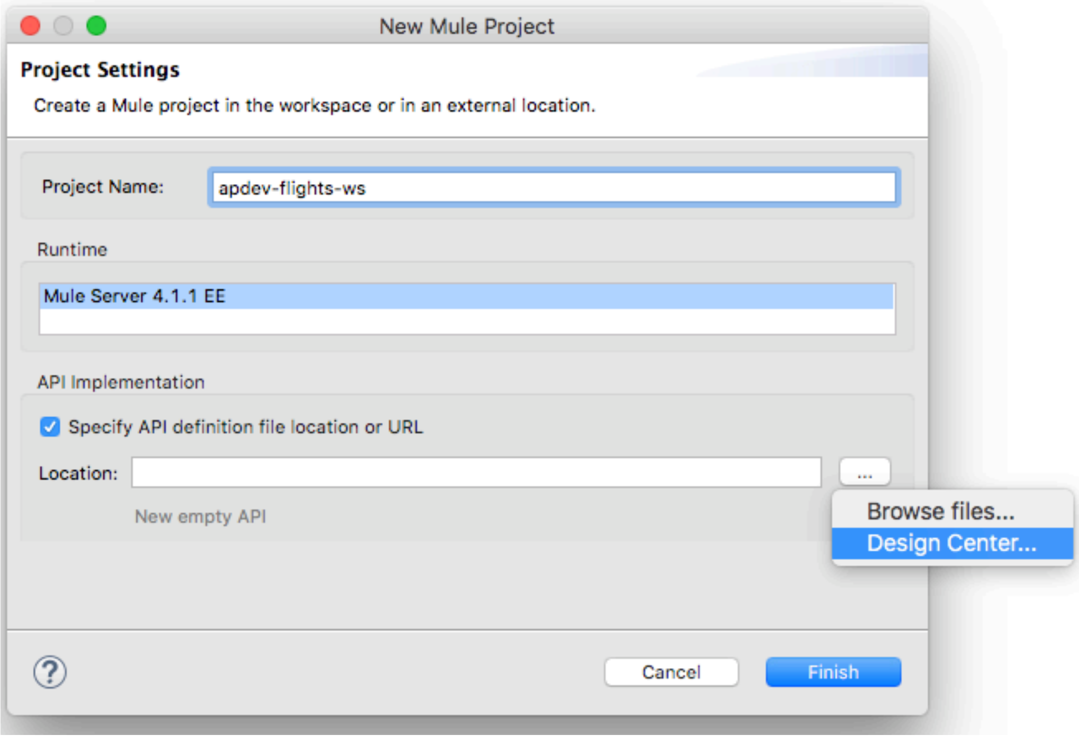


```
#%RAML 1.0 DataType

type: object
properties:
  ID?: integer
  airline: string
  flightCode: string
  fromAirportCode: string
  toAirportCode: string
  departureDate: string
  emptySeats: integer
  totalSeats?: integer
  price: number
  planeType?: string
```

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 Specify API definition file location or URL.
12. Click the browse button next to location and select Design Center.



Project Settings

Create a Mule project in the workspace or in an external location.

Project Name: apdev-flights-ws

Runtime

Mule Server 4.1.1 EE

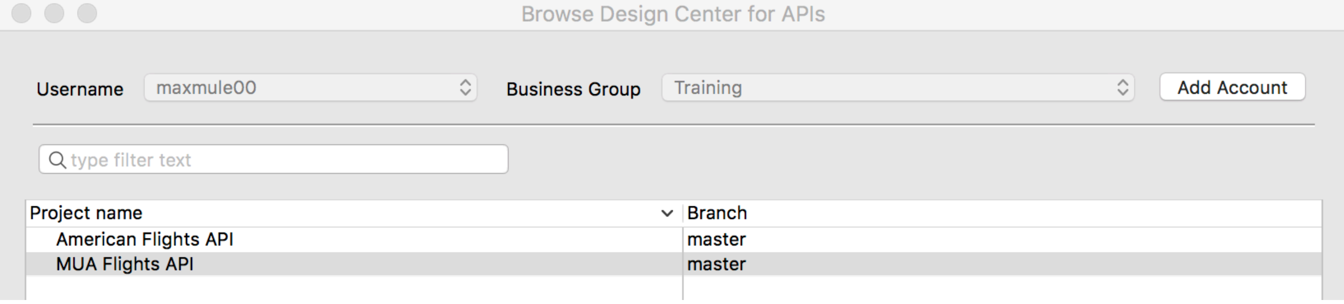
API Implementation

☑ Specify API definition file location or URL

Location:

New empty API

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



Username: maxmule00

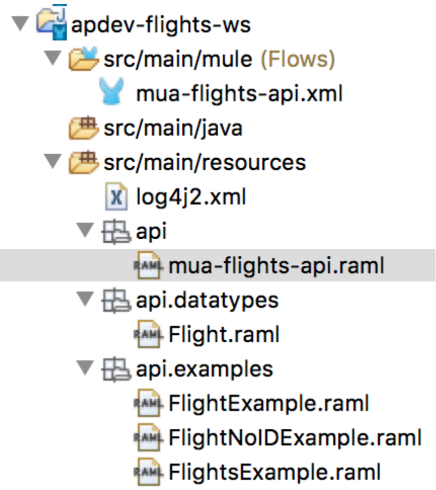
Business Group: Training

Project name	Branch
American Flights API	master
MUA Flights API	master

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 folders in src/main/resources folder; you should see the MUA Flights API files.



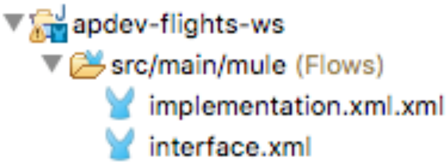
```
apdev-flights-ws
  src/main/mule (Flows)
    mua-flights-api.xml
  src/main/java
  src/main/resources
    log4j2.xml
    api
      mua-flights-api.raml
    api.datatypes
      Flight.raml
    api.examples
      FlightExample.raml
      FlightNoIDExample.raml
      FlightsExample.raml
```

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

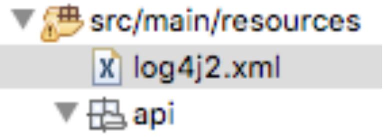
Review project configuration files

19. Look at the mua-flights-api.xml file that was created; it should have a get:\flights flow and a post:\flights flow.
20. Rename mua-flights-api.xml to interface.xml.

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

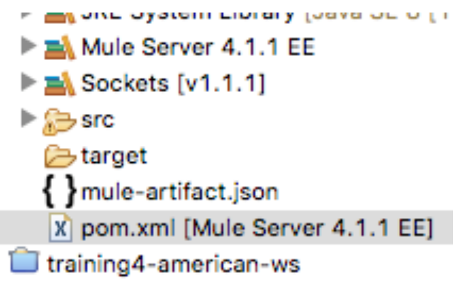


22. Locate log4j2.xml in src/main/resources and open it.



23. Review its contents and then close the file.

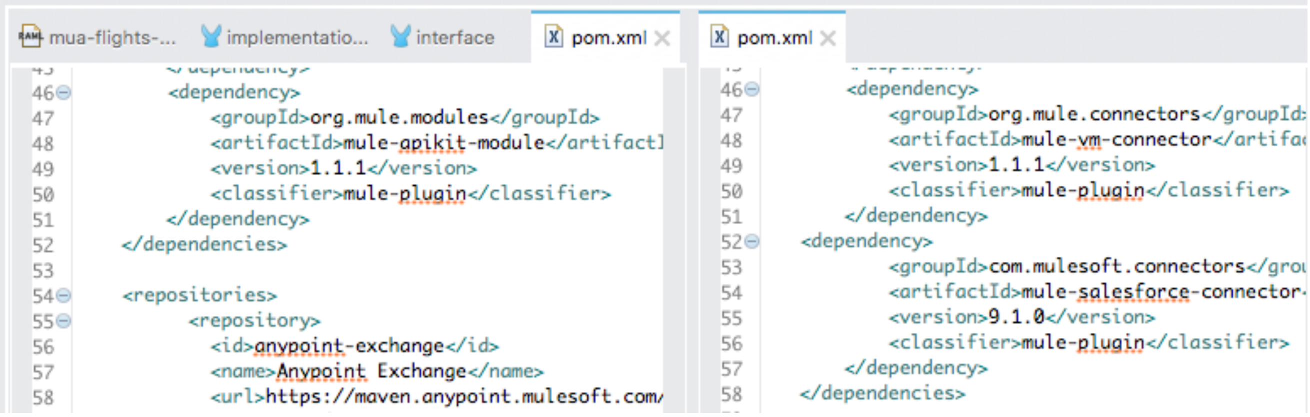
24. Locate pom.xml in the project and open it.



25. Review its contents.

26. Return to the apdev-examples project and open its pom.xml file.

27. Compare the two pom.xml files.



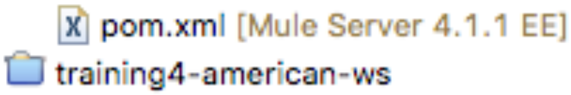
```
        <dependency>
            <groupId>org.mule.modules</groupId>
            <artifactId>mule-apikit-module</artifactI
            <version>1.1.1</version>
            <classifier>mule-plugin</classifier>
        </dependency>
    </dependencies>

    <repositories>
        <repository>
            <id>anypoint-exchange</id>
            <name>Anypoint Exchange</name>
            <url>https://maven.anypoint.mulesoft.com/
```

```
        <dependency>
            <groupId>org.mule.connectors</groupId
            <artifactId>mule-vm-connector</artifa
            <version>1.1.1</version>
            <classifier>mule-plugin</classifier>
        </dependency>
    <dependency>
            <groupId>com.mulesoft.connectors</grou
            <artifactId>mule-salesforce-connector
            <version>9.1.0</version>
            <classifier>mule-plugin</classifier>
        </dependency>
    </dependencies>
```

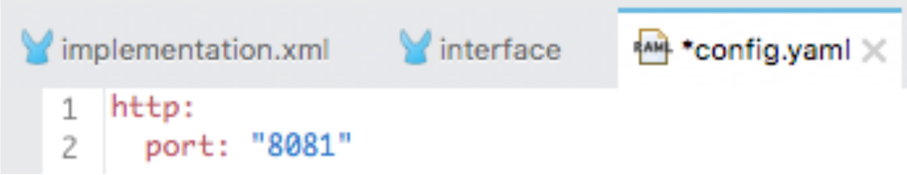
28. Close both pom.xml files.

29. In the Package Explorer, right-click apdev-examples and select Close Project.



Create a properties file for application properties

30. In the apdev-flights-ws project, right-click src/main/resources and select New > File.
31. In the New File dialog box, set the file name to config.yaml and click Finish.
32. In config.yaml, define a property called http.port equal to “8081”.

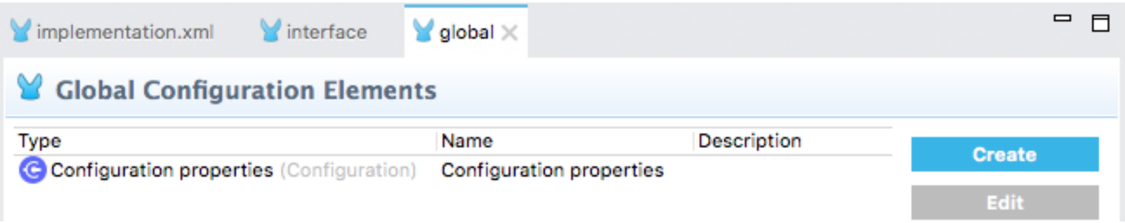


```
http:
  port: "8081"
```

33. Save and close the file.

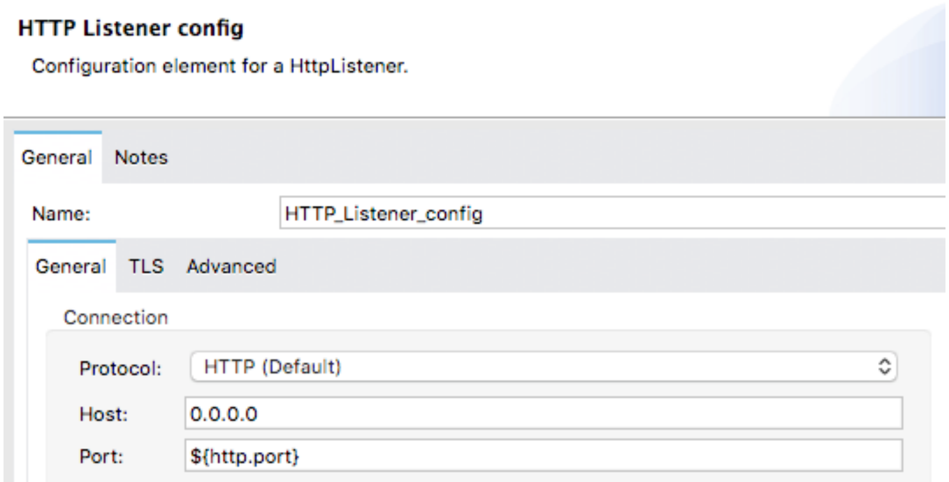
Create a global configuration file

34. In src/main/mule, create a new Mule configuration file called global.
35. In global.xml, switch to the Global Elements view.
36. Create a new Configuration properties element with the file set to config.yaml.



Type	Name	Description
Configuration properties (Configuration)	Configuration properties	

37. Create a new HTTP Listener config element with the host set to 0.0.0.0 and the port set to the http.port property.



HTTP Listener config

Configuration element for a HttpListener.

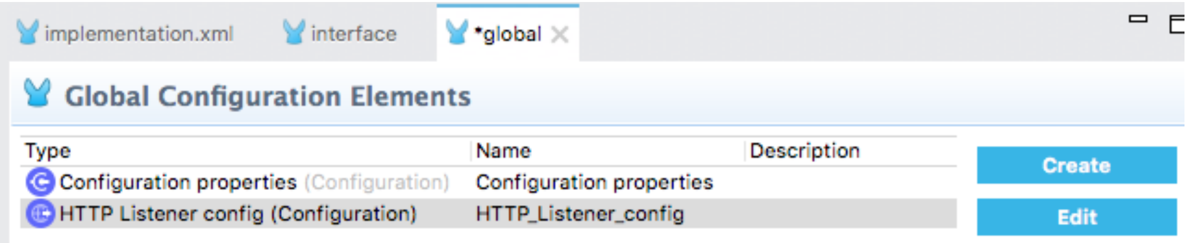
Name: HTTP_Listener_config

Protocol: HTTP (Default)

Host: 0.0.0.0

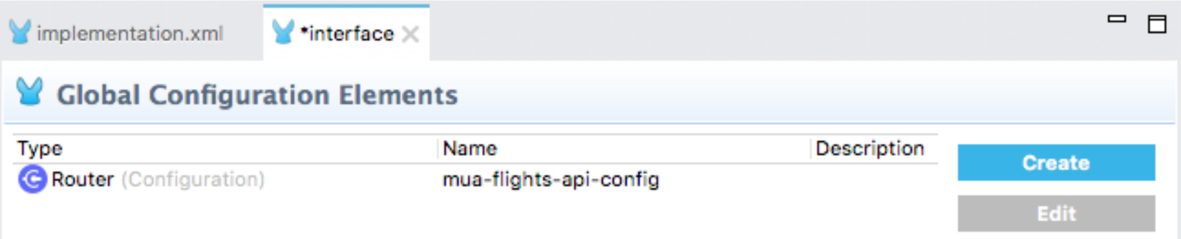
Port: ${http.port}

38. Confirm you now have two global elements defined in global.xml.



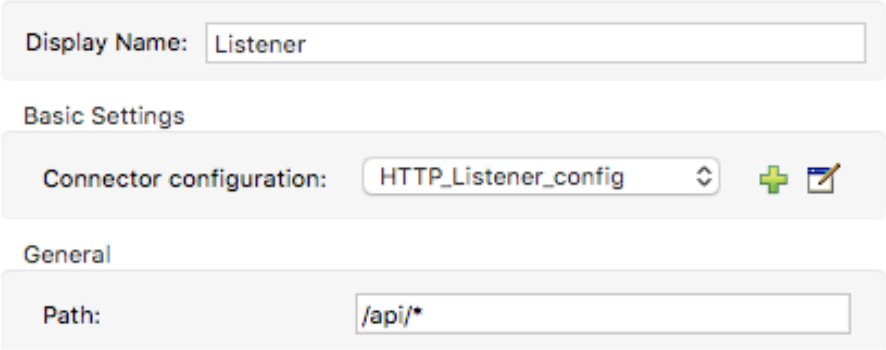
Type	Name	Description
Configuration properties (Configuration)	Configuration properties	
HTTP Listener config (Configuration)	HTTP_Listener_config	

39. Save and close the file.
40. Go to the Global Elements view in interface.xml.
41. Delete the mua-flights-api-httpListener HTTP Listener Configuration.



Type	Name	Description
Router (Configuration)	mua-flights-api-config	

42. Return to the Message Flow view.
43. Select the HTTP Listener in mua-flights-api-main, and in the Listener properties view set the Connector configuration to HTTP_Listener_config.



Display Name: Listener

Basic Settings

Connector configuration: HTTP_Listener_config

General

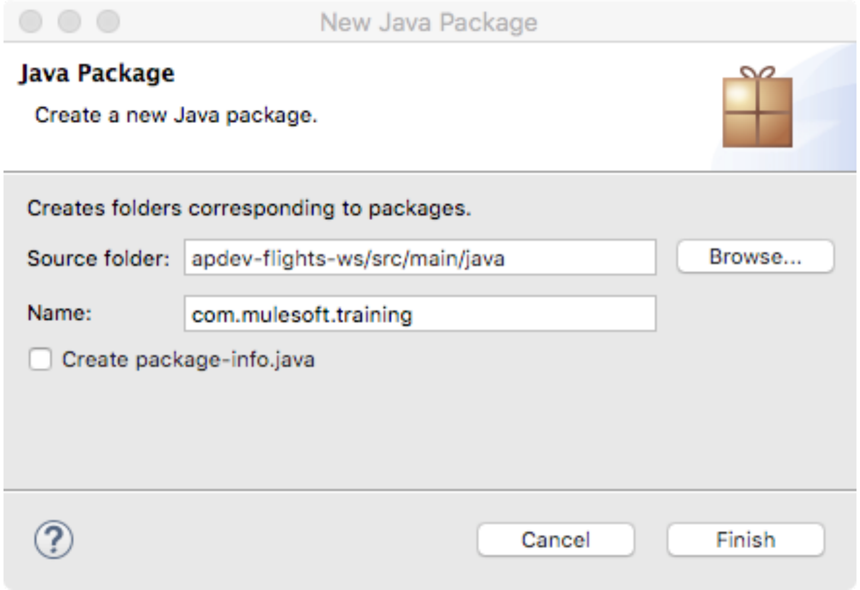
Path: /api/*

44. Select the HTTP Listener in mua-flights-api-console, and in the Listener properties view set the connector configuration to HTTP_Listener_config.

Add Java files to src/main/java

45. In the Package Explorer, right-click the src/main/java folder and select New > Package.

46. In the New Java Package dialog box, set the name to com.mulesoft.training and click Finish.



New Java Package

Java Package

Create a new Java package.

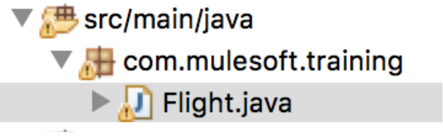
Creates folders corresponding to packages.

Source folder: apdev-flights-ws/src/main/java

Name: com.mulesoft.training

□ Create package-info.java

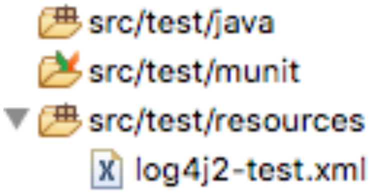
47. In your computer's file browser, locate the Flight.java file in the resources folder.
48. Drag the Flight.java file into the new com.mulesoft.training package in the src/main/java folder in the Anypoint Studio apdev-flights-ws project.
49. In the File Operation dialog box, select Copy files and click OK.
50. Open the Flight.java file.
51. Review the code.



```
▼ src/main/java
  ▼ com.mulesoft.training
    ▶ Flight.java
```

Review src/test folders

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

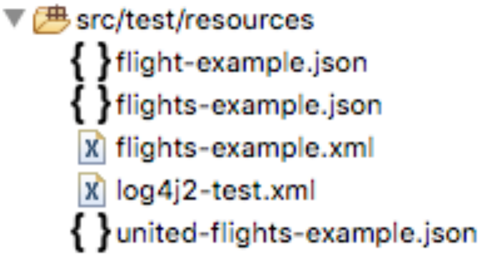


```
  src/test/java
  src/test/munit
▼ src/test/resources
    log4j2-test.xml
```

Add test resources

54. In your computer's file browser, expand the examples folder in the student files.

55. Select the three 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.

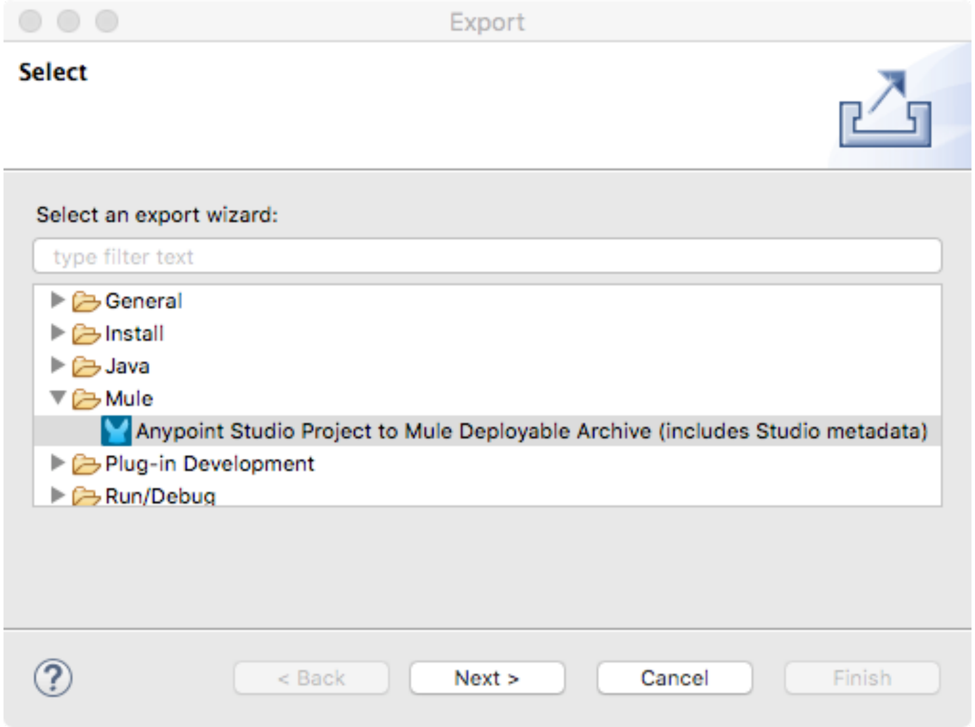


```
▼ src/test/resources
    { } flight-example.json
    { } flights-example.json
    flights-example.xml
    log4j2-test.xml
    { } united-flights-example.json
```

Examine the contents of a deployable archive

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

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



Export

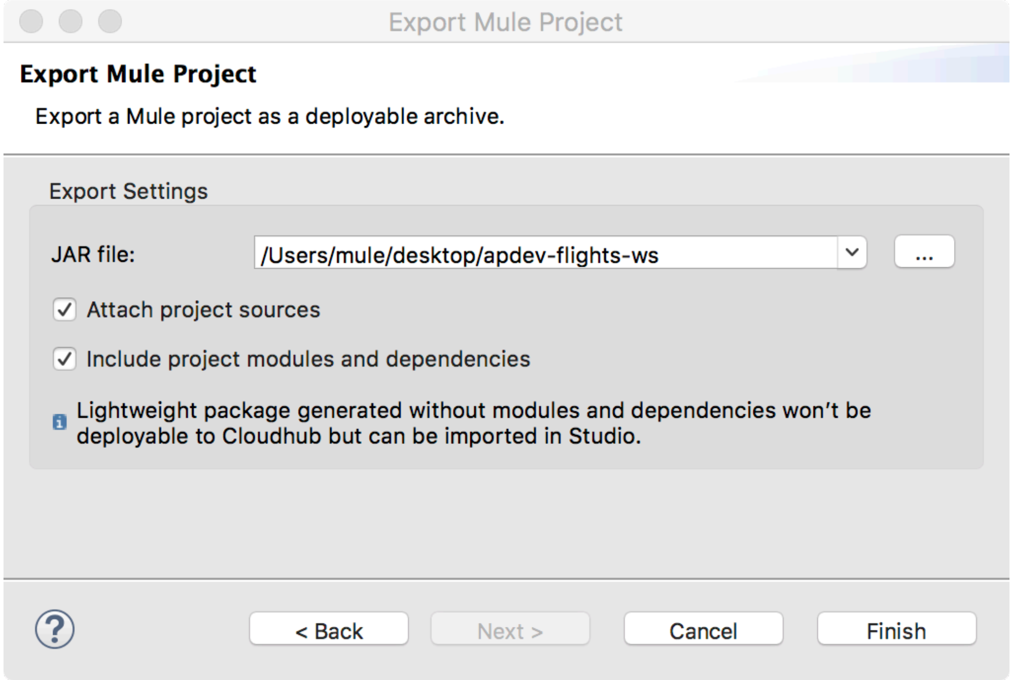
Select

Select an export wizard:

type filter text

```
▶ General
▶ Install
▶ Java
▼ Mule
    Anypoint Studio Project to Mule Deployable Archive (includes Studio metadata)
▶ Plug-in Development
▶ Run/Debug
```

58. In the Export Mule Project dialog box, set the JAR file to a location that you can find and click Finish.



Export Mule Project

Export a Mule project as a deployable archive.

Export Settings

JAR file: /Users/mule/desktop/apdev-flights-ws

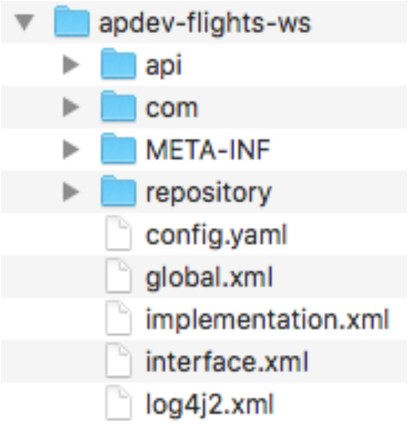
☑ Attach project sources

☑ Include project modules and dependencies

Lightweight package generated without modules and dependencies won't be deployable to Cloudhub but can be imported in Studio.

59. In your computer's file browser, locate the JAR file and unzip it.

60. Open the resulting apdev-flights-ws folder and examine the contents.



```
▼ apdev-flights-ws
    ▶ api
    ▶ com
    ▶ META-INF
    ▶ repository
      config.yaml
      global.xml
      implementation.xml
      interface.xml
      log4j2.xml
```