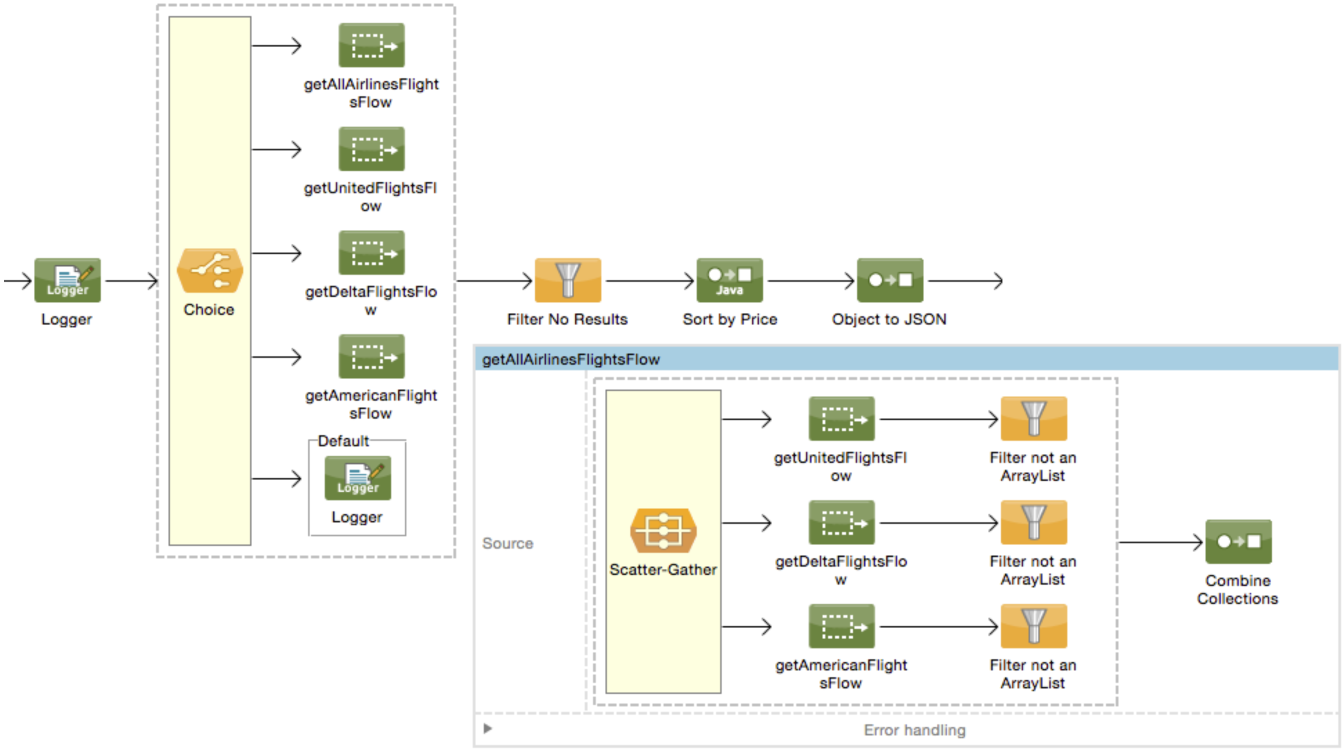
Module 7: Controlling Message Flow



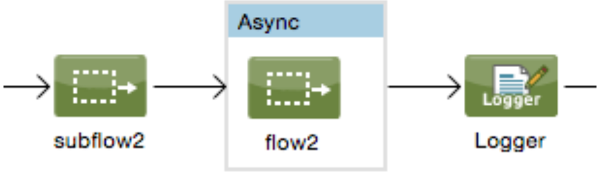
In this module, you will learn:

- About synchronous and asynchronous flows.
- To create an asynchronous flow.
- About flow control and filter elements.
- To multicast a message.
- To route message based on conditions.
- To filter messages.

Walkthrough 7-1: Pass messages to an asynchronous flow

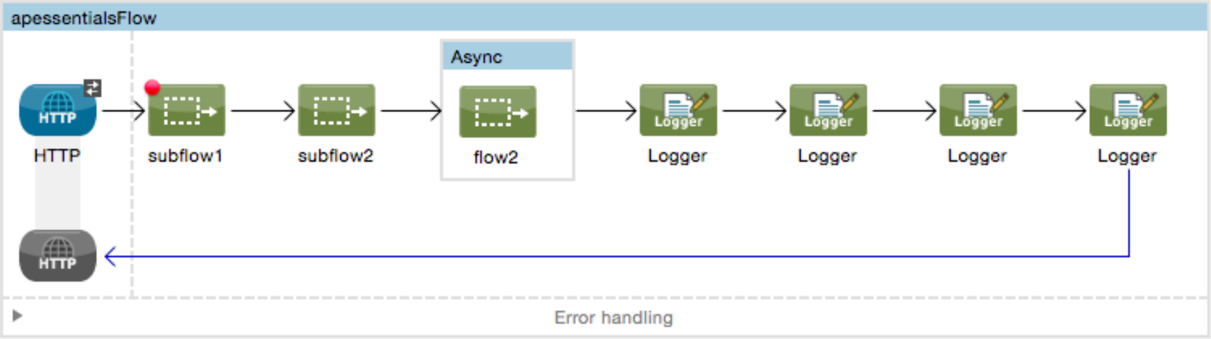
In this walkthrough, you will create and pass messages to an asynchronous flow. You will:

- Use the Async scope element to create an asynchronous flow.
- Use the Mule Debugger to watch messages flow through both flows.

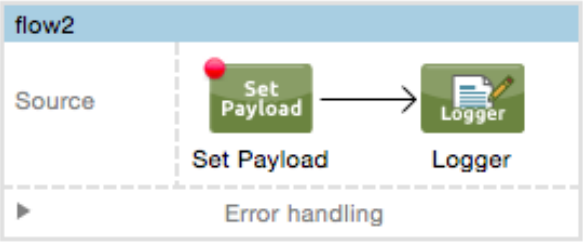


Create an asynchronous flow

1. Return to examples.xml in apessentials2.
2. Drag an Async scope element from the palette and drop it into the apessentialsFlow between the HTTP Request endpoint and the Logger.
3. Delete the HTTP Request endpoint in apessentialsFlow.
4. Drag a Flow Reference into the Async scope.
5. In the Flow Reference Properties view, set the flow name to flow2.
6. Add three more Logger components after the Async scope.



7. Delete the HTTP Listener endpoint in flow2.
8. Add a Logger component after the Set Payload in flow2.
9. Make sure there is a breakpoint on the Set Payload in flow2.



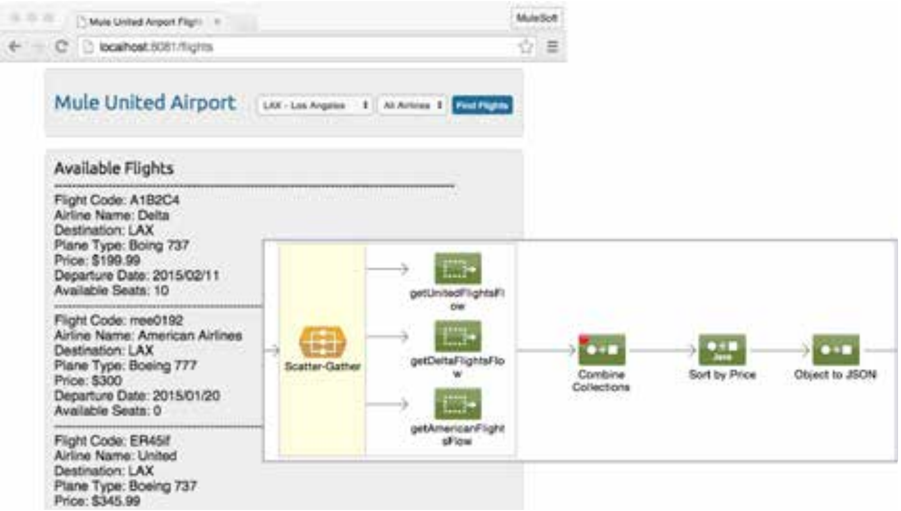
Debug the application

10. Save the file and debug the application.
11. Make another request to http://localhost:8082/?name=max&type=mule using your own query parameter values.
12. Step through the application again; you should see a copy of the message passed to the asynchronous flow2.
13. Watch the values of the payload change in both flows.

Walkthrough 7-2: Multicast a message

In this walkthrough, you will create one flow that calls each of the three airline services and combines and returns the results back to the web form. You will:

- Use a Scatter-Gather router to asynchronously call all three flight services.
- Use a Combine Collections transformer to combine a collection of three ArrayList of Flight objects into one collection.
- Sort the flights and return them as JSON to the form.



Available Flights

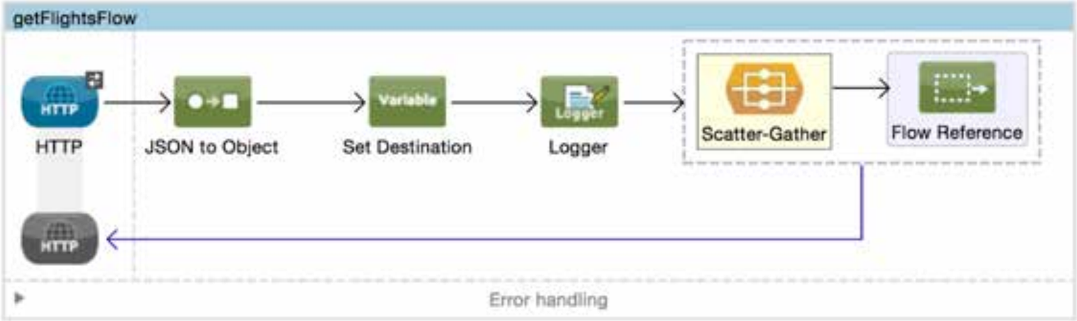
Flight Code: A1B2C4
Airline Name: Delta
Destination: LAX
Plane Type: Boing 737
Price: $199.99
Departure Date: 2015/02/11
Available Seats: 10

Flight Code: rree0192
Airline Name: American Airlines
Destination: LAX
Plane Type: Boeing 777
Price: $300
Departure Date: 2015/01/20
Available Seats: 0

Flight Code: ER45if
Airline Name: United
Destination: LAX
Plane Type: Boeing 737
Price: $345.99

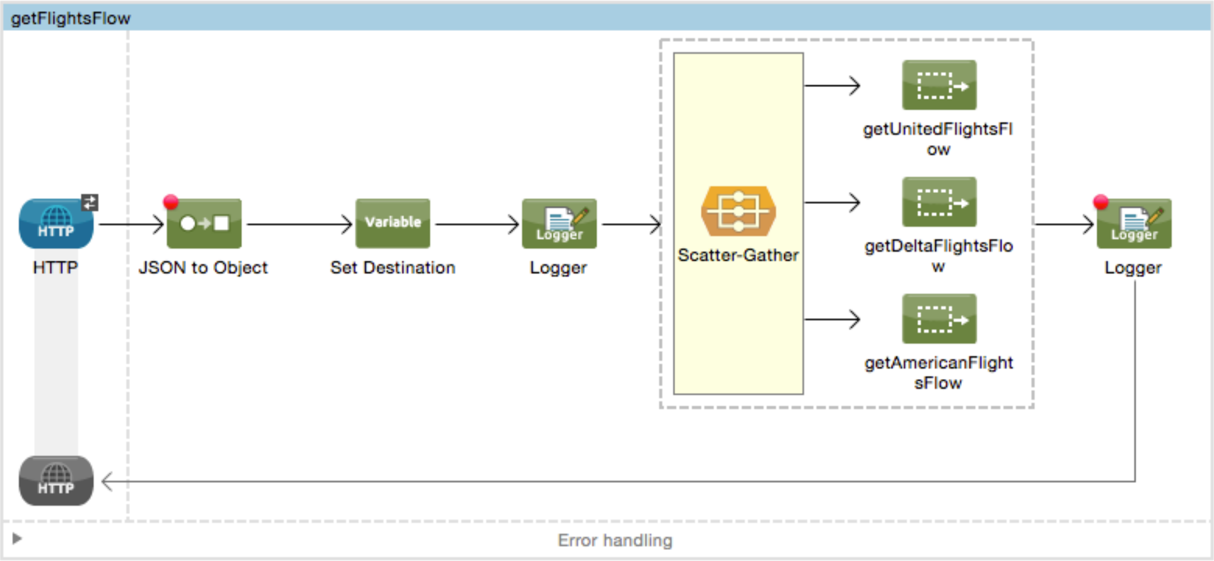
Add a router to call all three airline services

1. Return to getFlights.xml in apessentials.
2. Add a Scatter-Gather flow control element after the Logger in getFlightsFlow.
3. Add a Flow Reference component to the Scatter-Gather router.



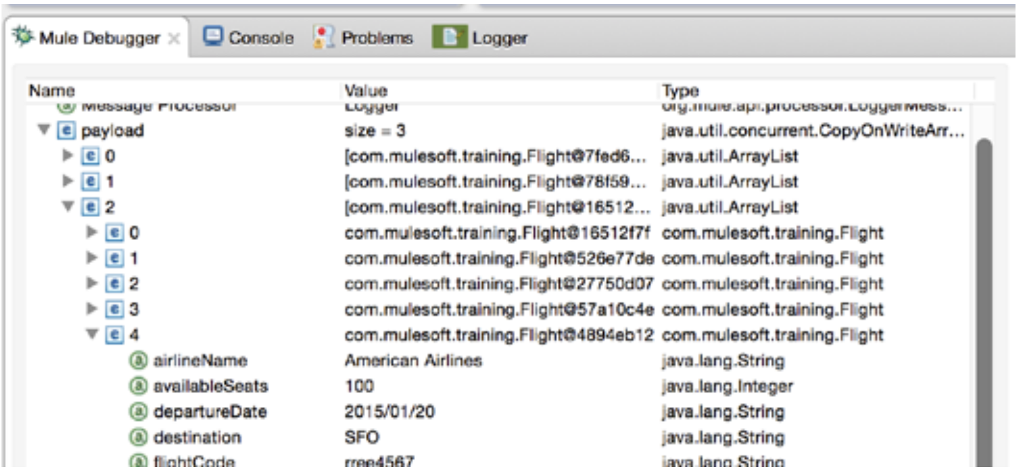
4. In the Properties view for the Flow Reference, set the flow name to getUnitedFlightsFlow.
5. Add a second Flow Reference component to the Scatter-Gather router.
6. In the Properties view for the Flow Reference, set the flow name to getDeltaFlightsFlow.
7. Add a third Flow Reference component to the Scatter-Gather router.

8. In the Properties view for the Flow Reference, set the flow name to getAmericanFlightsFlow.
9. Add a Logger after the router.
10. Add a breakpoint to the Logger.
11. Save the file.



Test the application

12. Click the arrow next to the Debug button and select apessentials.
13. Make a request to http://localhost:8081/flights and submit the form.
14. Step through the application to the Logger after the router; you should step through each of the airline flows.
15. Look the Mule Debugger view and see what the data type of the payload is after the router.

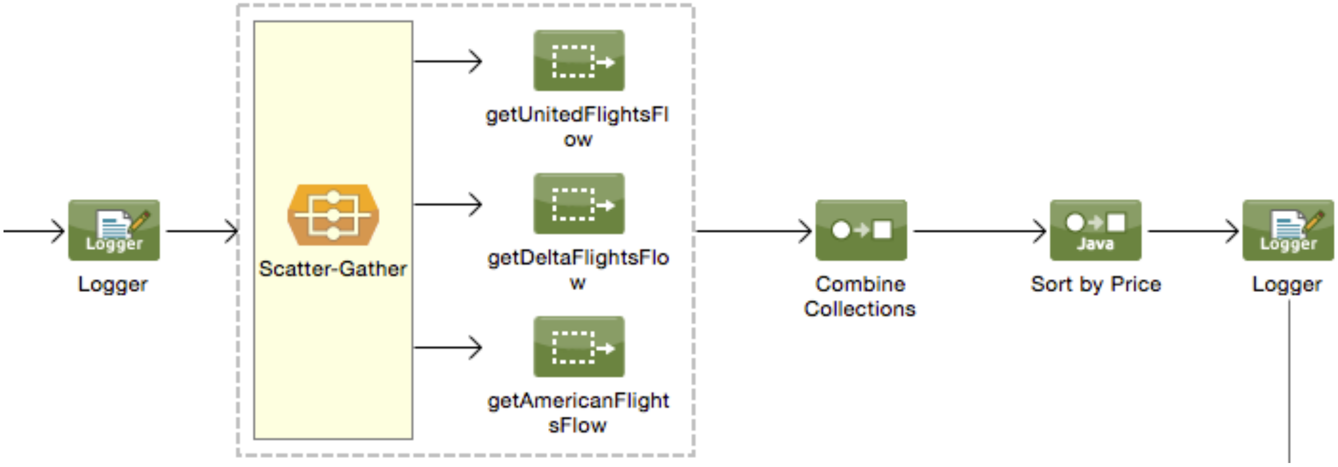


Name	Value	Type
Message Processor	Logger	org.mule.api.processor.LoggerMess...
payload	size = 3	java.util.concurrent.CopyOnWriteArr...
0	[com.mulesoft.training.Flight@7fed6...	java.util.ArrayList
1	[com.mulesoft.training.Flight@78f59...	java.util.ArrayList
2	[com.mulesoft.training.Flight@16512...	java.util.ArrayList
0	com.mulesoft.training.Flight@16512f7f	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@526e77de	com.mulesoft.training.Flight
2	com.mulesoft.training.Flight@27750d07	com.mulesoft.training.Flight
3	com.mulesoft.training.Flight@57a10c4e	com.mulesoft.training.Flight
4	com.mulesoft.training.Flight@4894eb12	com.mulesoft.training.Flight
airlineName	American Airlines	java.lang.String
availableSeats	100	java.lang.Integer
departureDate	2015/01/20	java.lang.String
destination	SFO	java.lang.String
flightCode	rree4567	java.lang.String

Note: If you step through to the end of the application, ignore any errors you get.

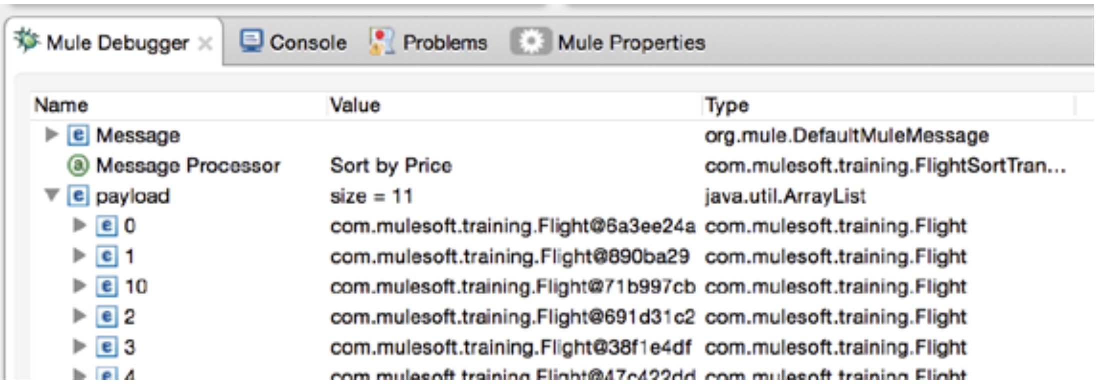
Transform and sort the results

16. Add a Combine Collections transformer before the Logger.
17. Move the Sort by Price Java transformer from the getDeltaFlightsFlow to after the Combine Collections transformer.



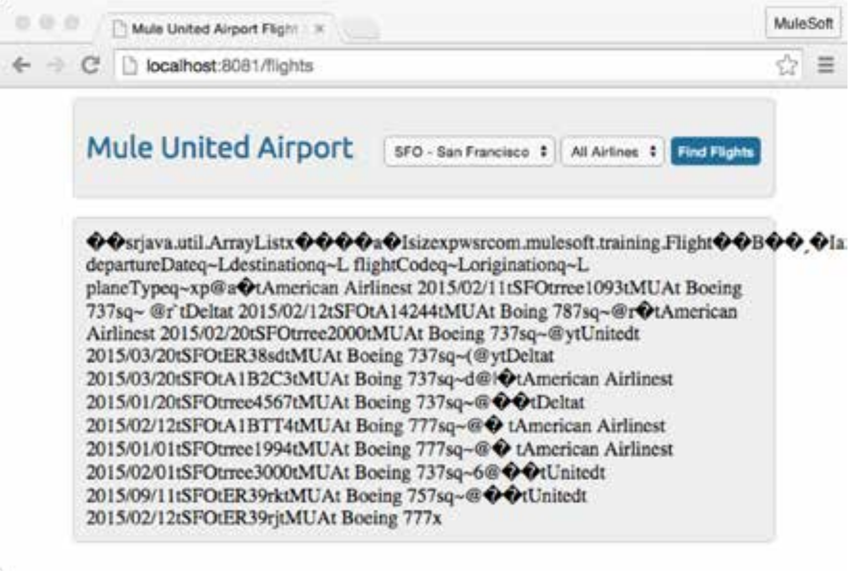
Test the application

18. Save the file to redeploy the application.
19. Make a request to http://localhost:8081/flights and submit the form.
20. Step through the application to the Logger after the router; you should see the payload is now one ArrayList of Flight objects.



Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Sort by Price	com.mulesoft.training.FlightSortTran...
payload	size = 11	java.util.ArrayList
0	com.mulesoft.training.Flight@6a3ee24a	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@890ba29	com.mulesoft.training.Flight
10	com.mulesoft.training.Flight@71b997cb	com.mulesoft.training.Flight
2	com.mulesoft.training.Flight@691d31c2	com.mulesoft.training.Flight
3	com.mulesoft.training.Flight@38f1e4df	com.mulesoft.training.Flight
4	com.mulesoft.training.Flight@47c422dd	com.mulesoft.training.Flight

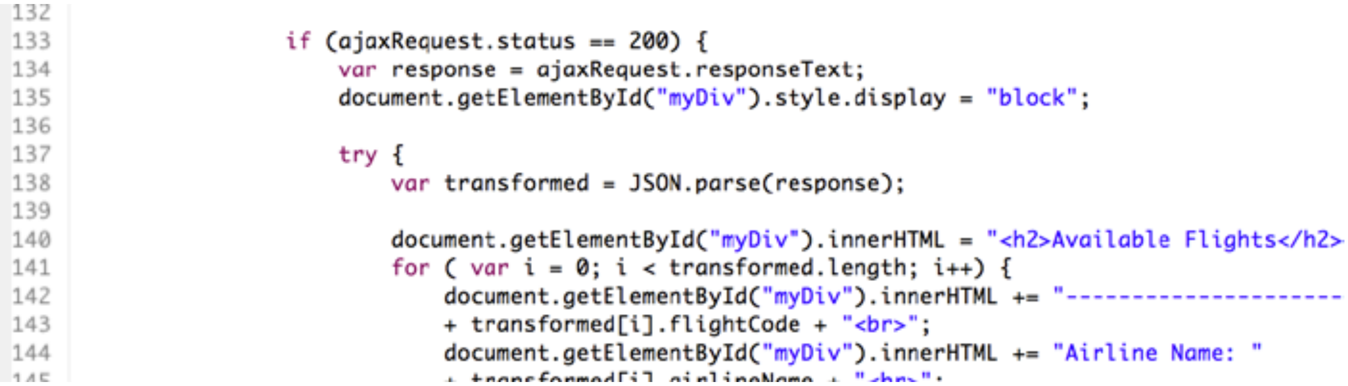
21. Step through to the end of the application; you should see a jumble of data returned.



```
��srjava.util.ArrayListx����a�Isizexpwsrcom.mulesoft.training.Flight��B��¸�Ia
departureDateq~Ldestinationq~L flightCodeq~Loriginationq~L
planeTypeq~xp@a�tAmerican Airlinest 2015/02/11tSFOtrree1093tMUAt Boeing
737sq~ @r`tDeltat 2015/02/12tSFOtA14244tMUAt Boing 787sq~@r�tAmerican
Airlinest 2015/02/20tSFOtrree2000tMUAt Boeing 737sq~@ytUnitedt
2015/03/20tSFOtER38sdtMUAt Boeing 737sq~(@ytDeltat
2015/03/20tSFOtA1B2C3tMUAt Boing 737sq~d@l�tAmerican Airlinest
2015/01/20tSFOtrree4567tMUAt Boeing 737sq~@��tDeltat
2015/02/12tSFOtA1BTT4tMUAt Boing 777sq~@� tAmerican Airlinest
2015/01/01tSFOtrree1994tMUAt Boeing 777sq~@� tAmerican Airlinest
2015/02/01tSFOtrree3000tMUAt Boeing 737sq~6@��tUnitedt
2015/09/11tSFOtER39rktMUAt Boeing 757sq~@��tUnitedt
2015/02/12tSFOtER39rjtMUAt Boeing 777x
```

Review HTML form code to see what data format it expects

22. Open FlightFinder.html in src/main/resources and locate in what format it expects the data to be sent back.



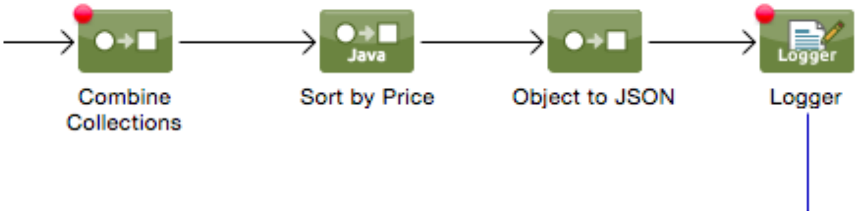
```
                if (ajaxRequest.status == 200) {
                    var response = ajaxRequest.responseText;
                    document.getElementById("myDiv").style.display = "block";

                    try {
                        var transformed = JSON.parse(response);

                        document.getElementById("myDiv").innerHTML = "<h2>Available Flights</h2>
                        for ( var i = 0; i < transformed.length; i++) {
                            document.getElementById("myDiv").innerHTML += "---------------------
                            + transformed[i].flightCode + "<br>";
                            document.getElementById("myDiv").innerHTML += "Airline Name: "
                            + transformed[i].airlineName + "<br>";
```

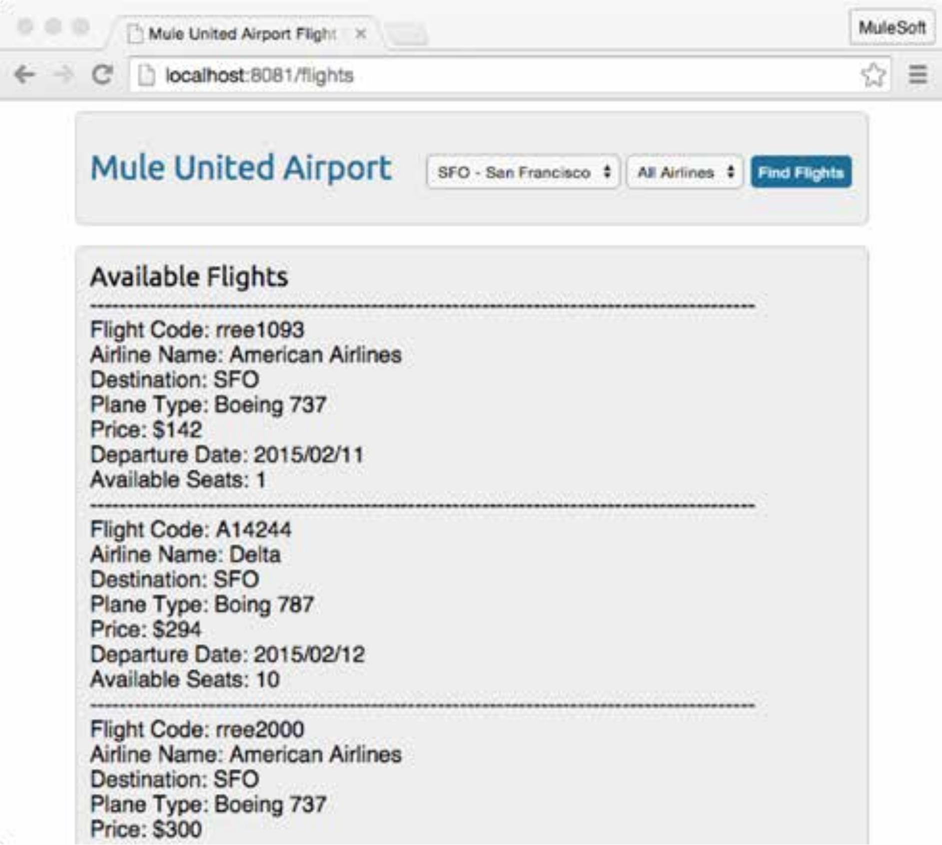
Return the data as JSON

23. In getFlights.xml, add an Object to JSON transformer after the Java Sort by Price transformer.



Test the application

24. Save the file and run the application.
25. Make a request to http://localhost:8081/flights and submit the form; you should see SFO flights for all three airlines returned and the flights should be sorted by price.



Available Flights

Flight Code: rree1093
Airline Name: American Airlines
Destination: SFO
Plane Type: Boeing 737
Price: $142
Departure Date: 2015/02/11
Available Seats: 1

Flight Code: A14244
Airline Name: Delta
Destination: SFO
Plane Type: Boing 787
Price: $294
Departure Date: 2015/02/12
Available Seats: 10

Flight Code: rree2000
Airline Name: American Airlines
Destination: SFO
Plane Type: Boeing 737
Price: $300

Note: If the sort is not working, the visual view and the XML may be out of sync. Check the order of the elements in the XML and fix them if necessary.

26. Submit the form for different destinations; flights for SFO are returned every time.

Use the destination in the airline flows

27. Locate the Set Destination transformer in the getAmericanFlightsFlow.

Note: There are two options at this point: 1) to remove the HTTP endpoints and Set Destination transformers from each of the airline flows or 2) to use a more complicated expression to check and see if a destination flow variable already exists so that the individual airline flows can still be called and tested individually.

28. Modify the value so it only uses the expression to set the destination flowVars variable if it does not already exist.

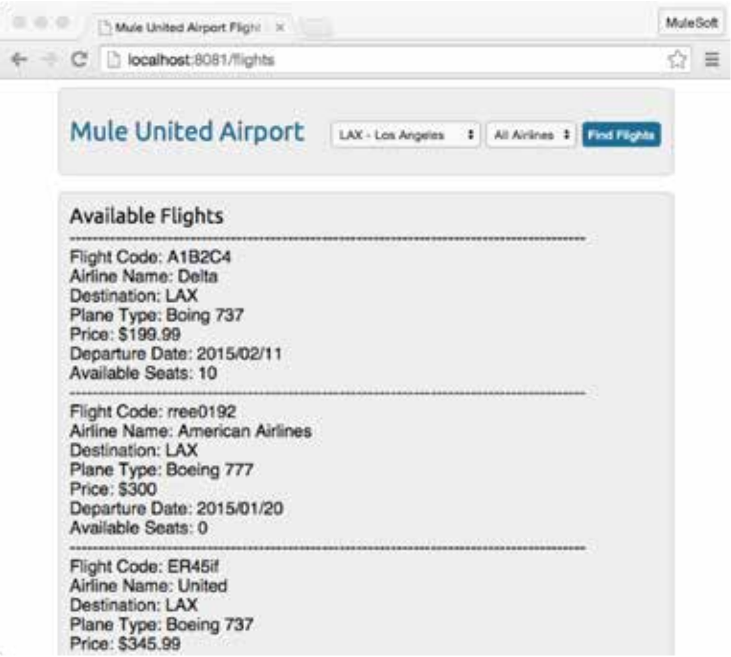
```
#[(flowVars.destination ==empty &&
message.inboundProperties.'http.query.params'.code == empty) ? 'SFO' :
(flowVars.destination != empty ? flowVars.destination :
message.inboundProperties.'http.query.params'.code)]
```

Note: This expression is also located in the course snippets.txt file and can be copied from there.

29. Use the same expression in the Set Destination in getDeltaFlightsFlow and getUnitedFlightsFlow.

Test the application

30. Save the file to redeploy the application.
31. Make a request to http://localhost:8081/flights and submit the form; you should still see SFO flights.
32. Submit the form for a different destination including LAX, CLE, or PDX; you should see the correct flights returned.



Available Flights

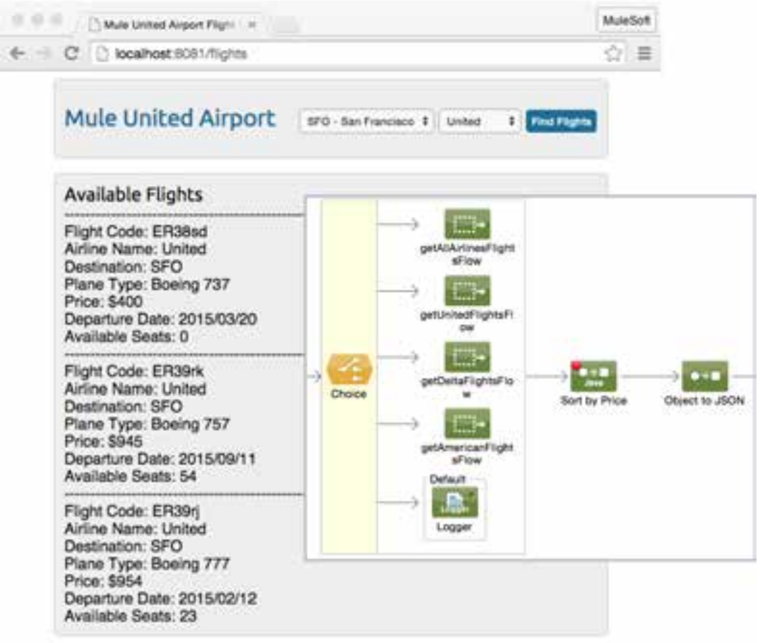
```
--------------------------------------------------------------
Flight Code: A1B2C4
Airline Name: Delta
Destination: LAX
Plane Type: Boing 737
Price: $199.99
Departure Date: 2015/02/11
Available Seats: 10
--------------------------------------------------------------
Flight Code: rree0192
Airline Name: American Airlines
Destination: LAX
Plane Type: Boeing 777
Price: $300
Departure Date: 2015/01/20
Available Seats: 0
--------------------------------------------------------------
Flight Code: ER45if
Airline Name: United
Destination: LAX
Plane Type: Boeing 737
Price: $345.99
```

33. Submit the form for a destination of PDF; you should get an error in the console.
34. Submit the form for a destination of FOO; you should get an error in the console.
35. Submit the form for SFO, LAX, CLE, or PDX and a different airline; you should still see flights for all three airlines returned.

Walkthrough 7-3: Route messages based on conditions

In this walkthrough, you will create a flow that routes messages based on conditions. You will:

- Use a Choice router to get flight results for all three airlines or only a specific airline.
- Set the router paths based on the airline value sent from the flight form.



Available Flights

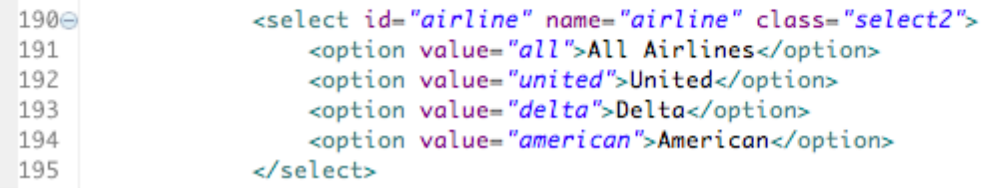
Flight Code: ER38sd
Airline Name: United
Destination: SFO
Plane Type: Boeing 737
Price: $400
Departure Date: 2015/03/20
Available Seats: 0

Flight Code: ER39rk
Airline Name: United
Destination: SFO
Plane Type: Boeing 757
Price: $945
Departure Date: 2015/09/11
Available Seats: 54

Flight Code: ER39rj
Airline Name: United
Destination: SFO
Plane Type: Boeing 777
Price: $954
Departure Date: 2015/02/12
Available Seats: 23

Look at selected airline values in HTML form

1. In Anypoint Studio, return to FlightFinder.html.
2. Locate the select box that sets the airline and see what values are set and returned.

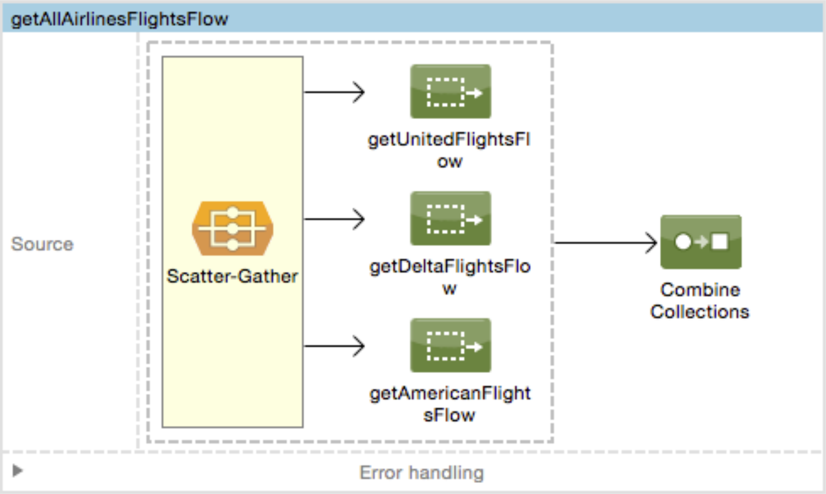


```
<select id="airline" name="airline" class="select2">
    <option value="all">All Airlines</option>
    <option value="united">United</option>
    <option value="delta">Delta</option>
    <option value="american">American</option>
</select>
```

Move the Scatter-Gather to a separate flow

3. Return to getFlights.xml.
4. Shift+click to select the Scatter-Gather router and the Combine Collections transformer.
5. Right-click and select Extract to > Flow.
6. In the Extract Flow dialog box, set the flow name to getAllAirlinesFlightsFlow.

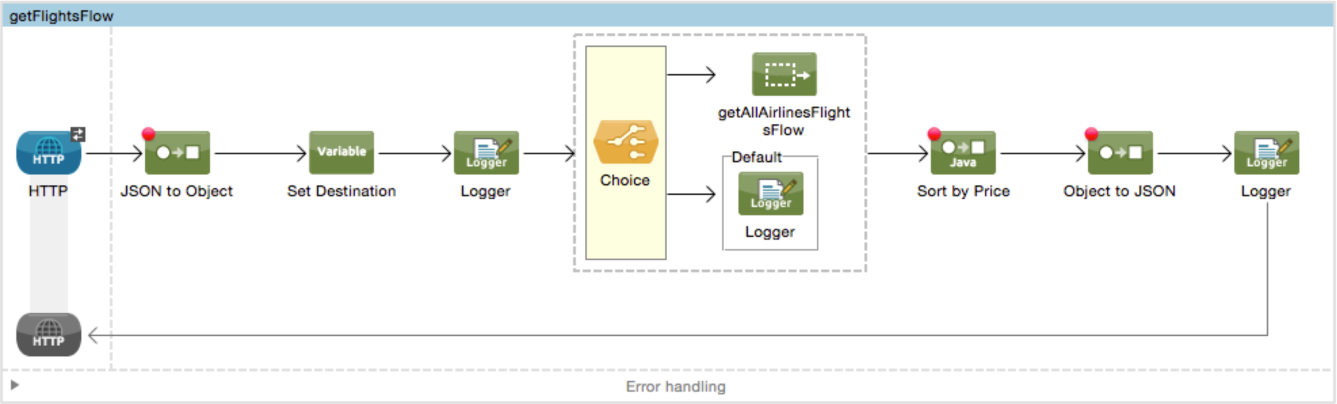
7. Leave the target Mule configuration set to current and click OK.



8. In getFlightsFlow, double-click the new Flow Reference.
9. Change its display name to getAllAirlinesFlightsFlow.
10. Move getAllAirlinesFlightsFlow beneath the getFlightFormFlow.
11. Add a breakpoint to the Combine Collections transformer.

Add a Choice router

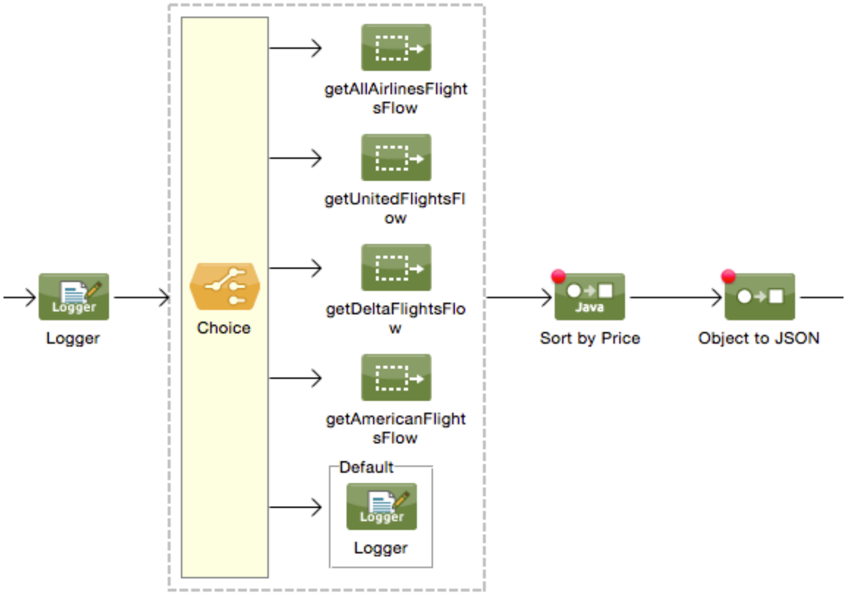
12. In getFlightsFlow, add a Choice flow control element between the Logger and the getAllAirlinesFlightsFlow flow reference.
13. Drag the getAllAirlinesFlightsFlow flow reference into the router.
14. Add a new Logger component to the default branch.



15. Add three additional Flow Reference components to the Choice router to create a total of five branches.
16. In the Properties view for the first new flow reference, set the flow name to getUnitedFlightsFlow.

17. In the Properties view for the second flow reference, set the flow name to getDeltaFlightsFlow.

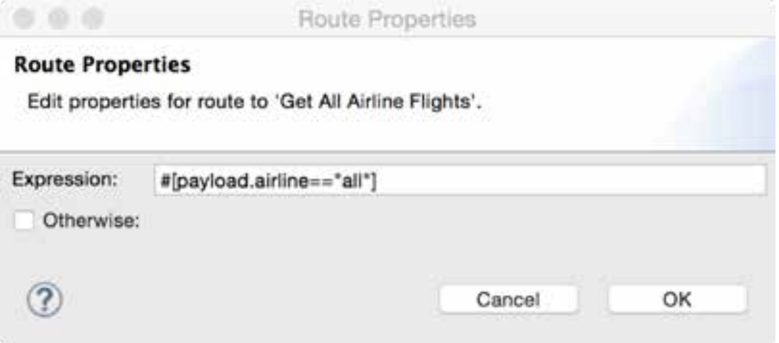
18. In the Properties view for the third flow reference, set the flow name to getAmericanFlightsFlow.



Configure the Choice router

19. In the Properties view for the Choice router, double-click the getAllAirlinesFlightsFlow route.

20. In the Route Properties dialog box, set the expression to #[payload.airline== "all"] and click OK.



Route Properties

Edit properties for route to 'Get All Airline Flights'.

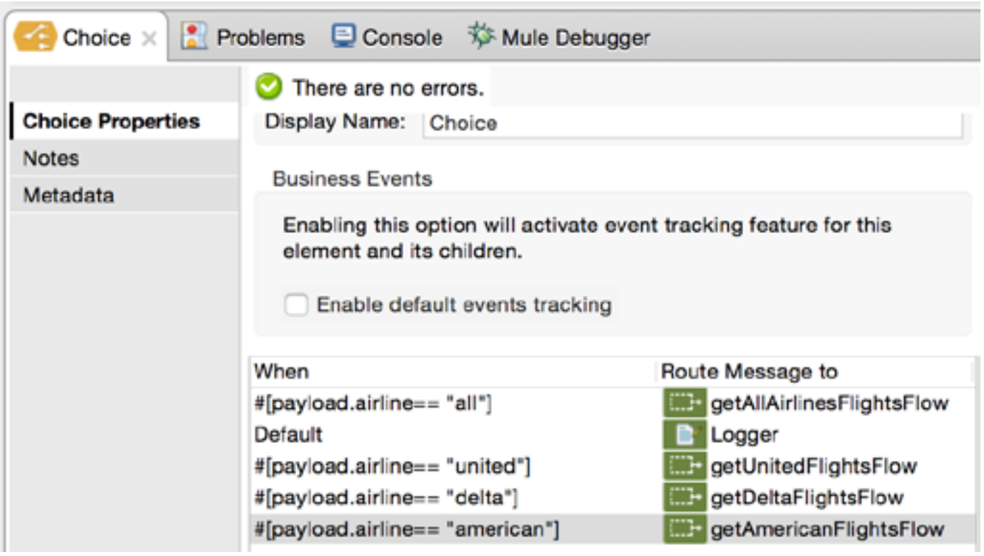
Expression: #[payload.airline=="all"]

□ Otherwise:

21. Set a similar expression for the United route, routing to it when payload.airline is equal to united.

22. Set a similar expression for the Delta route, routing to it when payload.airline is equal to delta.

23. Set a similar expression for the American route, routing to it when payload.airline is equal to american.



There are no errors.

Display Name: Choice

Business Events

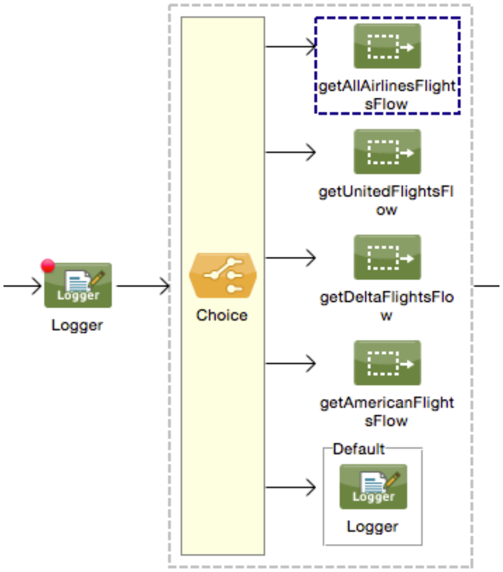
Enabling this option will activate event tracking feature for this element and its children.

☐ Enable default events tracking

When	Route Message to
#[payload.airline== "all"]	getAllAirlinesFlightsFlow
Default	Logger
#[payload.airline== "united"]	getUnitedFlightsFlow
#[payload.airline== "delta"]	getDeltaFlightsFlow
#[payload.airline== "american"]	getAmericanFlightsFlow

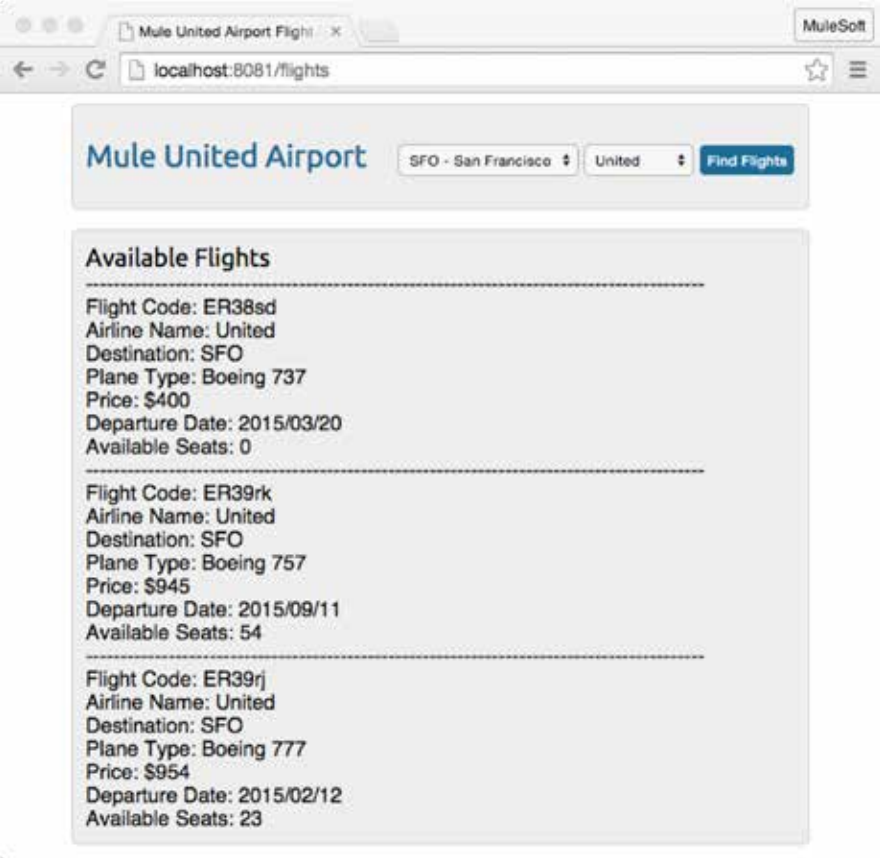
Debug the application

24. Add a breakpoint to the Logger before the Choice router.
25. Save the file and debug the application.
26. Make a request to http://localhost:8081/flights.
27. On the form page, leave SFO and All Airlines selected and click Find Flights.
28. Return to the Mule Debugger view and step through the application; you should see the Choice router pass the message to the getAllAirlinesFlightsFlow branch.



29. Click the Resume button until flight data for all airlines is returned back to the form page.

30. On the form page, select SFO and United and click Find Flights.
31. Return to the Mule Debugger view and step through the application; you should see the Choice router pass the message to the United branch.
32. Click the Resume button until flight data is returned back to the form page; you should see only United flights.



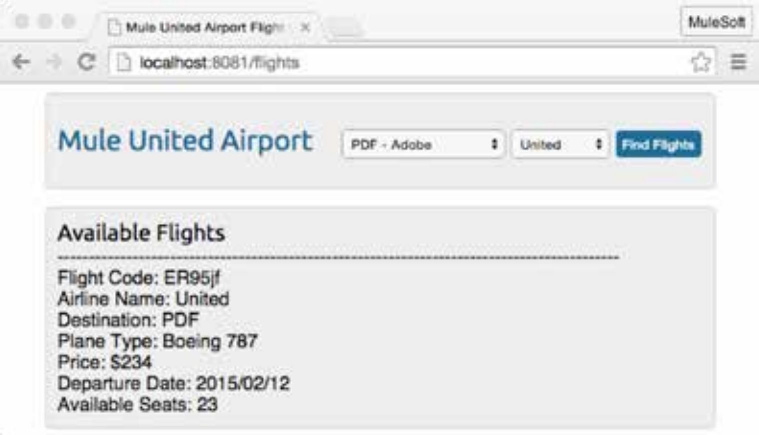
Available Flights

```
----------------------------------------------------------------
Flight Code: ER38sd
Airline Name: United
Destination: SFO
Plane Type: Boeing 737
Price: $400
Departure Date: 2015/03/20
Available Seats: 0
----------------------------------------------------------------
Flight Code: ER39rk
Airline Name: United
Destination: SFO
Plane Type: Boeing 757
Price: $945
Departure Date: 2015/09/11
Available Seats: 54
----------------------------------------------------------------
Flight Code: ER39rj
Airline Name: United
Destination: SFO
Plane Type: Boeing 777
Price: $954
Departure Date: 2015/02/12
Available Seats: 23
```

Test the application

33. Stop the debugger and run the application.
34. Make a request to http://localhost:8081/flights.
35. Test the Delta and American choices with SFO, LAX, or CLE; you should see the appropriate flights returned.

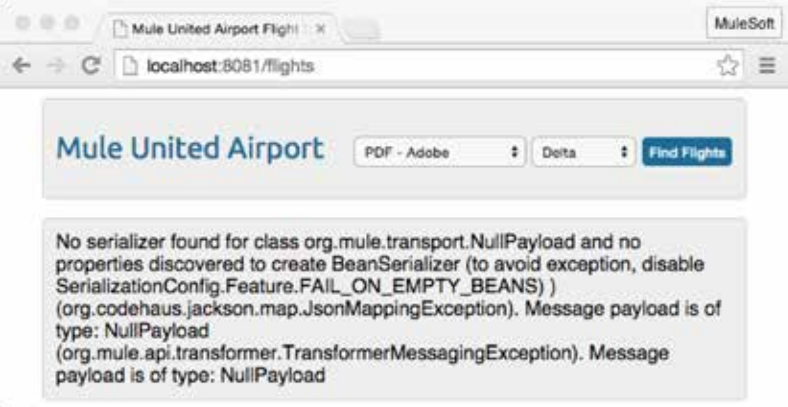
36. Test the PDF location with United; you should see one flight.



Available Flights

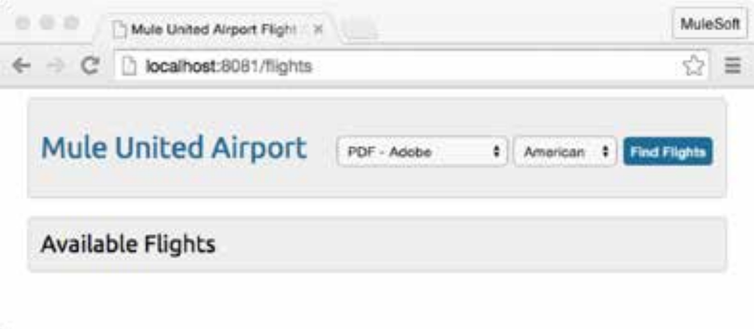
Flight Code: ER95jf
Airline Name: United
Destination: PDF
Plane Type: Boeing 787
Price: $234
Departure Date: 2015/02/12
Available Seats: 23

37. Test the PDF location with Delta; you should get an error.



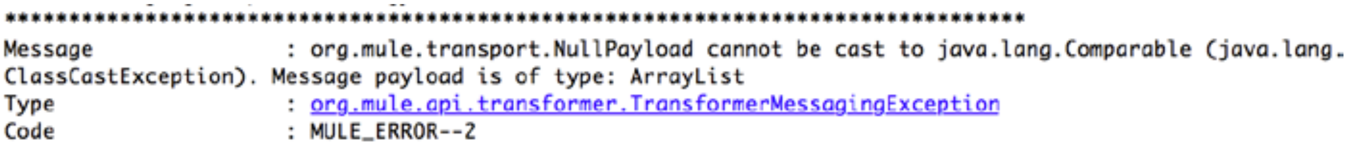
No serializer found for class org.mule.transport.NullPayload and no properties discovered to create BeanSerializer (to avoid exception, disable SerializationConfig.Feature.FAIL_ON_EMPTY_BEANS)) (org.codehaus.jackson.map.JsonMappingException). Message payload is of type: NullPayload (org.mule.api.transformer.TransformerMessagingException). Message payload is of type: NullPayload

38. Test the PDF location with American; you should get no results.



Available Flights

39. Test the PDF location with All Airlines; you should get no results and an exception in the console.

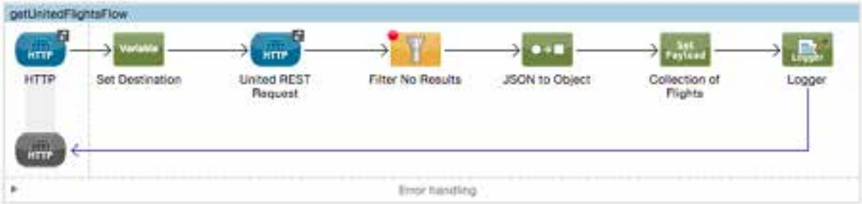


```
********************************************************************************
Message               : org.mule.transport.NullPayload cannot be cast to java.lang.Comparable (java.lang.
ClassCastException). Message payload is of type: ArrayList
Type                  : org.mule.api.transformer.TransformerMessagingException
Code                  : MULE_ERROR--2
```

Walkthrough 7-4: Filter messages

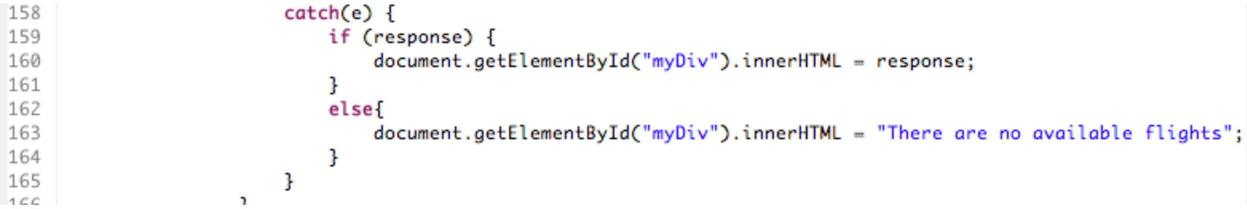
In this walkthrough, you will continue to work with the airline flight results. You will:

- Add filters to the airline flows to keep them from returning messages when there are no flights in the payload.
- Filter the results in the multicast to ensure they are ArrayLists (which will be needed when the payload is set to an error message when an exception is thrown in the next module).
- Use the Expression and Payload filters.
- Create and use a global filter.



Look at the HTML form and find the display if no flights are returned

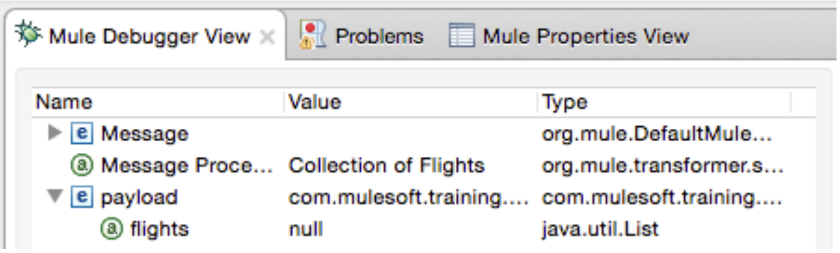
1. In Anypoint Studio, return to FlightFinder.html.
2. Locate the code that sets the text if no flights are returned – i.e., the response cannot be parsed as JSON.



```
                    catch(e) {
                        if (response) {
                            document.getElementById("myDiv").innerHTML = response;
                        }
                        else{
                            document.getElementById("myDiv").innerHTML = "There are no available flights";
                        }
                    }
                }
```

Debug the application for when Delta returns no results

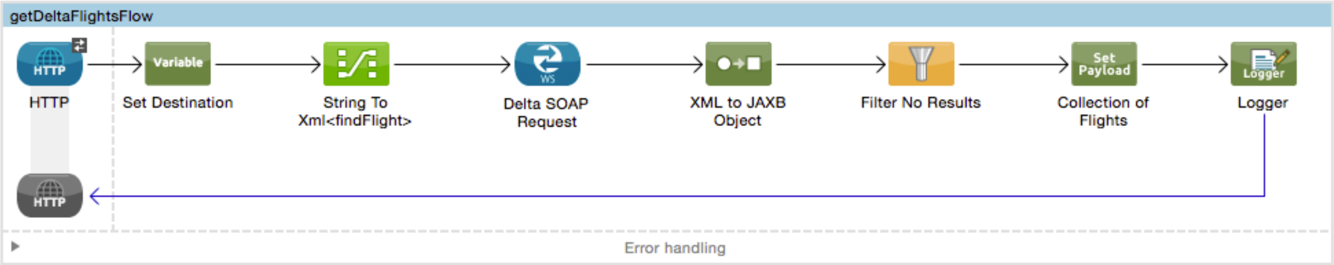
3. Return to getFlights.xml and debug the application.
4. Make a request to http://localhost:8081/flights and submit the form for PDF and Delta.
5. Step through the application; you should see that payload.flights is null after the XML to JAXB Object transformer.



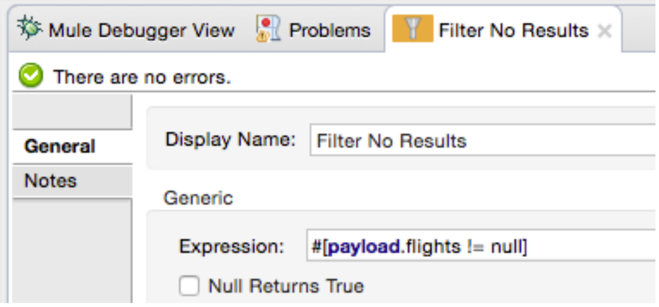
Name	Value	Type
▶ e Message		org.mule.DefaultMule...
a Message Proce...	Collection of Flights	org.mule.transformer.s...
▼ e payload	com.mulesoft.training....	com.mulesoft.training....
a flights	null	java.util.List

Add an Expression filter to the Delta flow

6. In getDeltaFlightsFlow, add an Expression filter after the XML to JAXB Object transformer.
7. In the Expression Properties view, set the display name to Filter No Results.



8. Set the expression to #[payload.flights != null].



There are no errors.

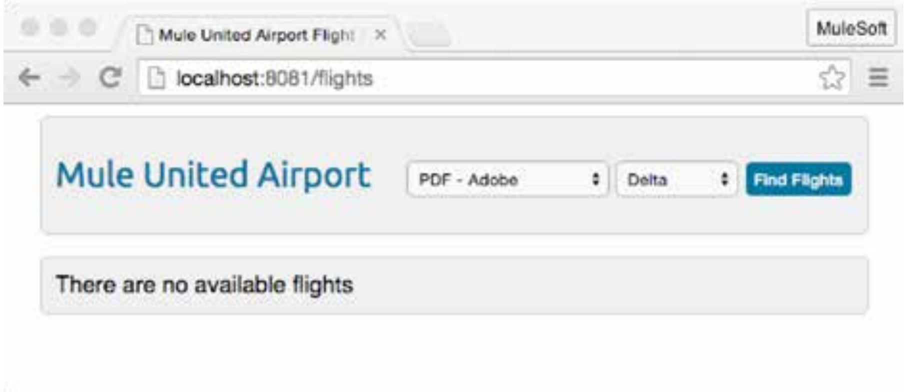
Display Name: Filter No Results

Expression: #[payload.flights != null]

□ Null Returns True

Test the application

9. Save the file to redeploy the application.
10. Refresh the browser window and submit the form for PDF and Delta.
11. Step through the application; you should see the message filtered and then a message that there are no available flights in the browser window.

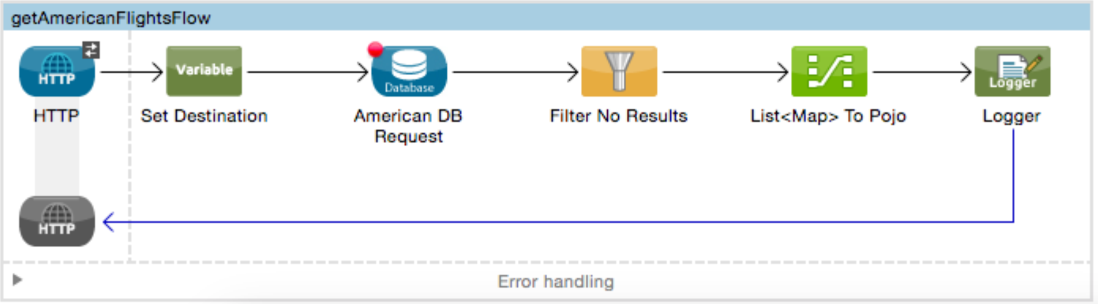


There are no available flights

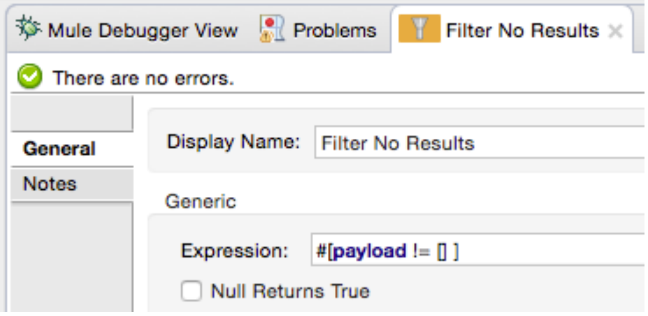
12. Submit the form for PDF and American.
13. Step through the application; you should see that payload is an empty LInkedList after the American DB Request endpoint.

Add an Expression filter to the American flow

14. In getAmericanFlightsFlow, add an Expression filter after the American DB Request endpoint.
15. In the Expression Properties view, set the display name to Filter No Results.



16. Set the expression to #[payload != []].



There are no errors.

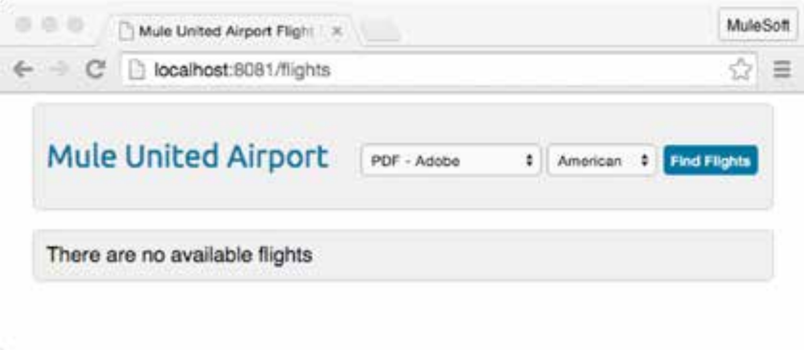
Display Name: Filter No Results

Expression: #[payload != []]

□ Null Returns True

Test the application

17. Save the file to redeploy the application.
18. Refresh the browser window and submit the form for PDF and American.
19. Step through the application; you should see the message filtered and then a message that there are no available flights in the browser window.



localhost:8081/flights

PDF - Adobe

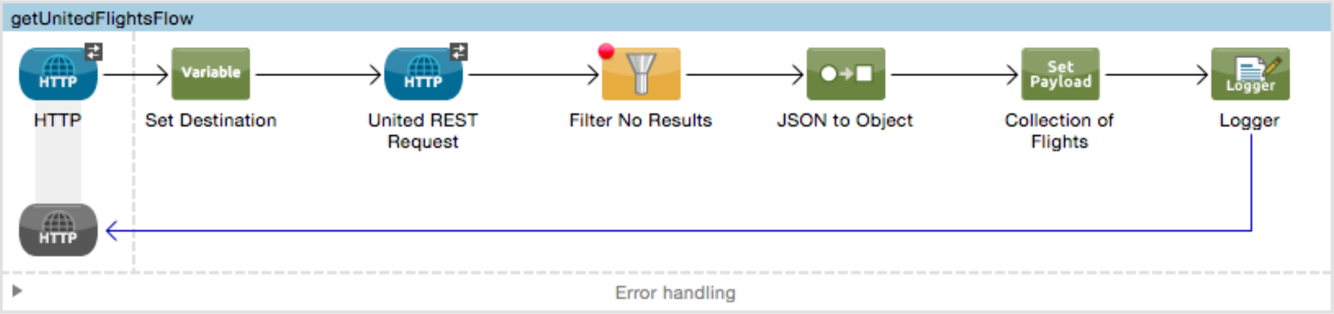
American

There are no available flights

20. Submit the form for FOO and United.
21. Step through the application; you should see that payload is a BufferInputStream after the United REST Request endpoint.

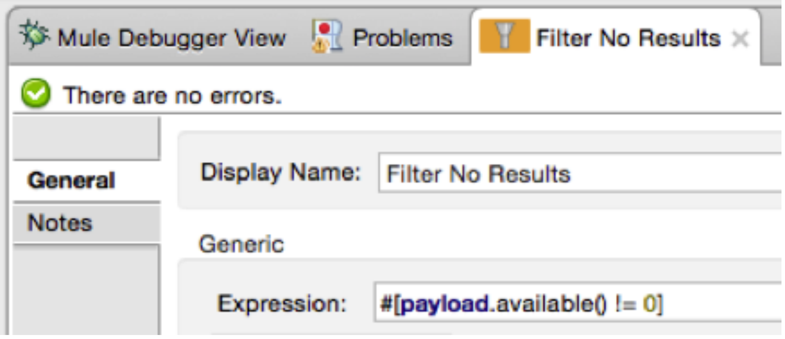
Add an Expression filter to the United flow

22. In getUnitedFlightsFlow, add an Expression filter after the United REST Request endpoint.
23. In the Expression Properties view, set the display name to Filter No Results.



24. Set the expression to #[payload.available() != 0].

Note: This expression checks to see if there is any data in the stream to be read. You could instead use an expression that checks to see if the inbound http.status property is equal to 200: #[message.inboundProperties.'http.status' == 200].



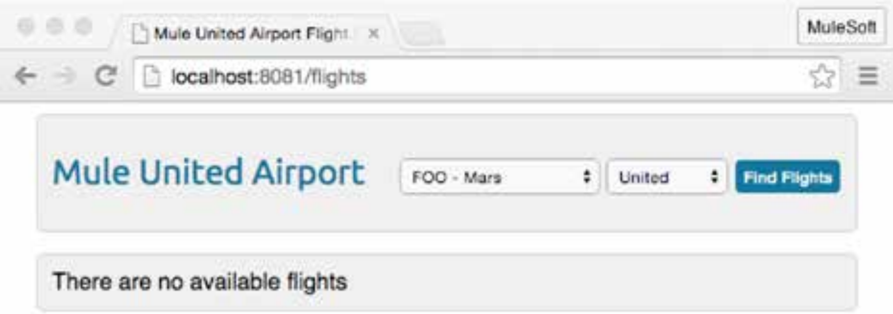
There are no errors.

Display Name: Filter No Results

Expression: #[payload.available() != 0]

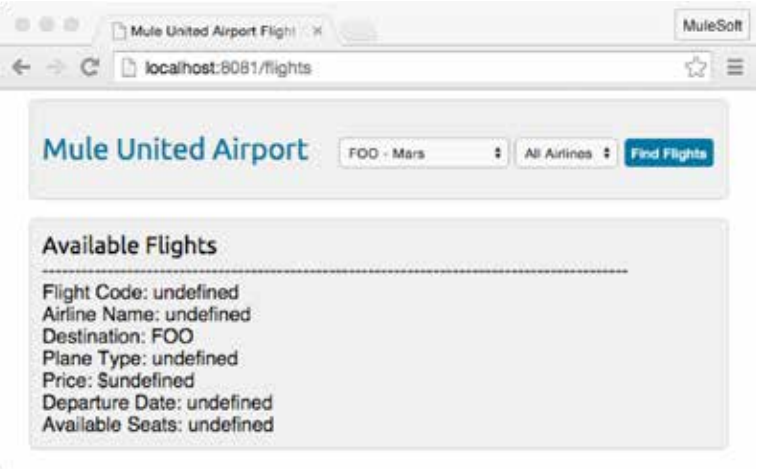
Test the application

25. Save the file to redeploy the application.
26. Refresh the browser window and submit the form for FOO and United.
27. Step through the application; you should see the message filtered and then a message that there are no available flights in the browser window.



There are no available flights

28. Add a breakpoint to the Combine Collections transformer in getAllAirlinesFlightsFlow (if there is not one there already).
29. Submit the form for FOO and All Airlines and step through the application; the FlightRequest object is incorrectly returned back.

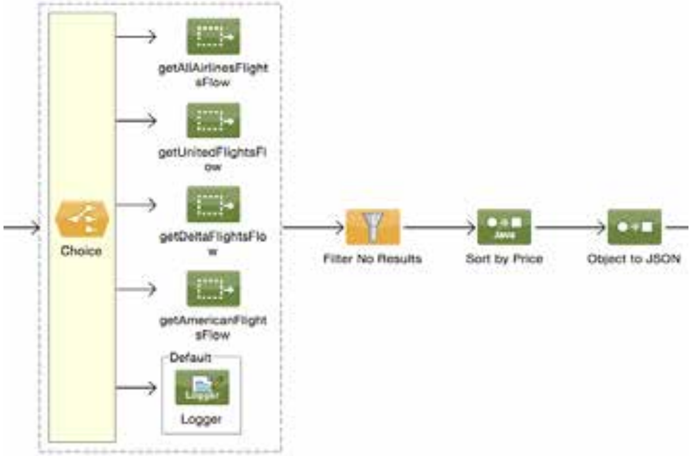


Available Flights

Flight Code: undefined
Airline Name: undefined
Destination: FOO
Plane Type: undefined
Price: $undefined
Departure Date: undefined
Available Seats: undefined

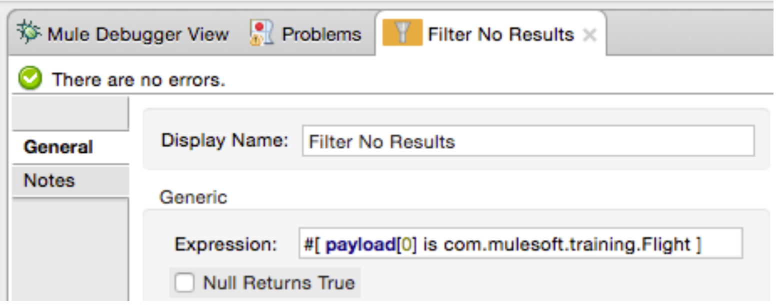
Add an Expression filter to the All Airlines flow

30. In getFlightsFlow, add an Expression filter after the Choice router.
31. In the Expression Properties view, set the display name to Filter No Results.



32. Set the expression to see if the items in the payload are Flight objects.

```
#[payload[0] is com.mulesoft.training.Flight]
```



There are no errors.

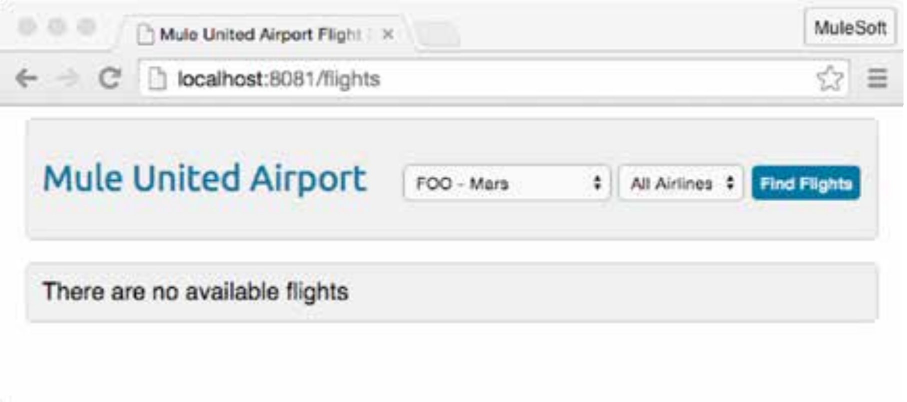
Display Name: Filter No Results

Expression: #[payload[0] is com.mulesoft.training.Flight]

□ Null Returns True

Test the application

33. Save the file to redeploy the application.
34. Submit the form for FOO and All Airlines.
35. Step through the application; you should see the message filtered after the Choice router and then a message that there are no available flights in the browser window.

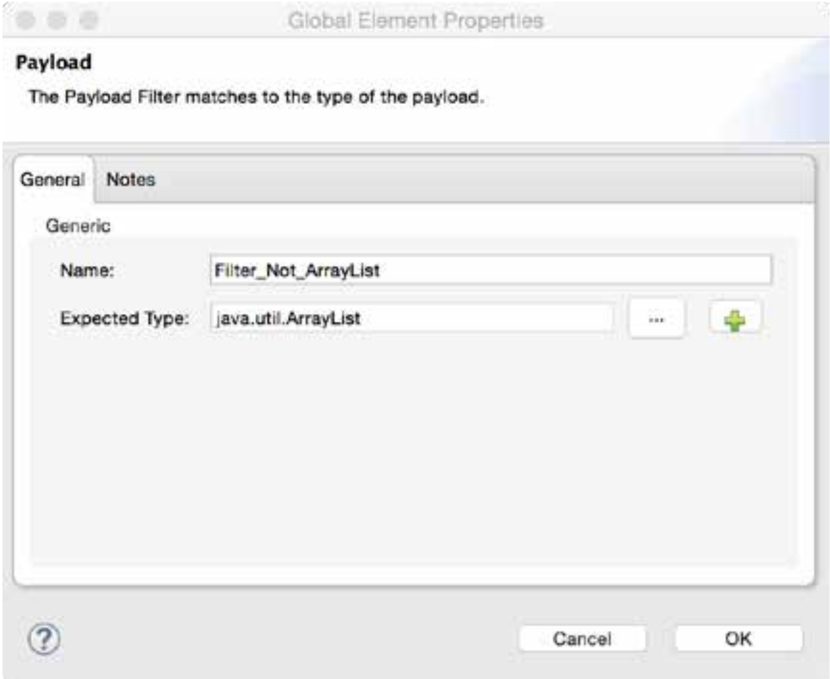


There are no available flights

Create a global filter

36. Open global.xml in apessentials.
37. Switch to the Global Elements view and click Create.
38. In the Choose Global Type dialog box, select Filters > Payload and click OK.

39. In the Global Element Properties dialog box, set the name to Filter_Not_ArrayList.
40. Set the expected type to java.util.ArrayList and click OK.



Payload

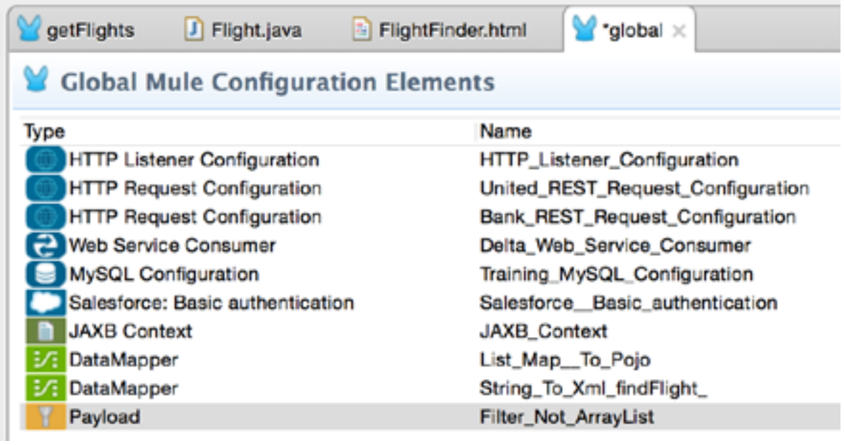
The Payload Filter matches to the type of the payload.

Generic

Name: Filter_Not_ArrayList

Expected Type: java.util.ArrayList

41. See your new global filter in the list.



Global Mule Configuration Elements

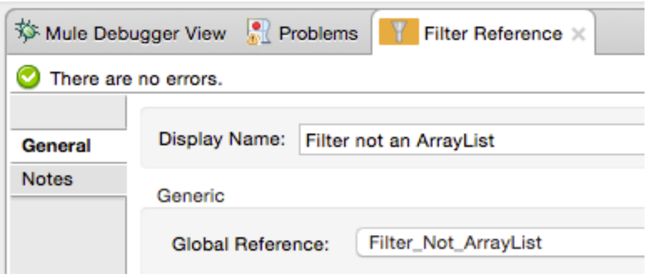
Type	Name
HTTP Listener Configuration	HTTP_Listener_Configuration
HTTP Request Configuration	United_REST_Request_Configuration
HTTP Request Configuration	Bank_REST_Request_Configuration
Web Service Consumer	Delta_Web_Service_Consumer
MySQL Configuration	Training_MySQL_Configuration
Salesforce: Basic authentication	Salesforce__Basic_authentication
JAXB Context	JAXB_Context
DataMapper	List_Map__To_Pojo
DataMapper	String_To_Xml_findFlight_
Payload	Filter_Not_ArrayList

42. Save the file.

Use the global filter

43. Return to getFlights.xml.
44. Drag a Filter Reference from the palette and drop it after the getUnitedFlightsFlow flow reference in getAllAirlinesFlightsFlow.

45. In the Filter Reference Properties view, set the display name to Filter not an ArrayList and set the global reference to Filter_Not_ArrayList.

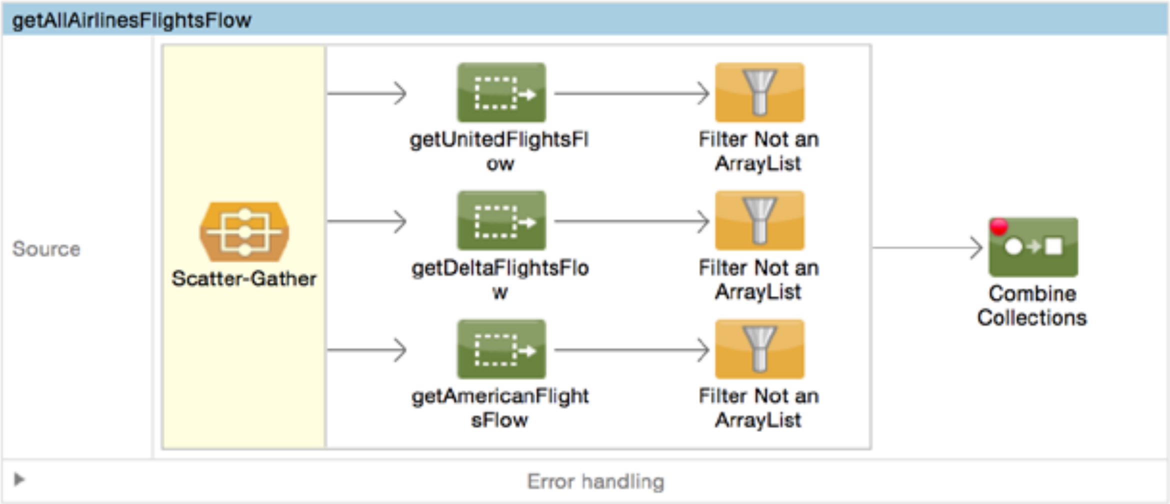


There are no errors.

Display Name: Filter not an ArrayList

Global Reference: Filter_Not_ArrayList

46. Add a Filter Reference after the getDeltaFlightsFlow flow reference.
47. In the Filter Reference Properties view, set the display name to Filter not an ArrayList and set the global reference to Filter_Not_ArrayList.
48. Add a Filter Reference after the getAmericanFlightsFlow flow reference.
49. In the Filter Reference Properties view, set the display name to Filter not an ArrayList and set the global reference to Filter_Not_ArrayList.
50. Save the file.



Note: These filters will be needed in the next module when the flows set the payload to an error message string when an exception is thrown.