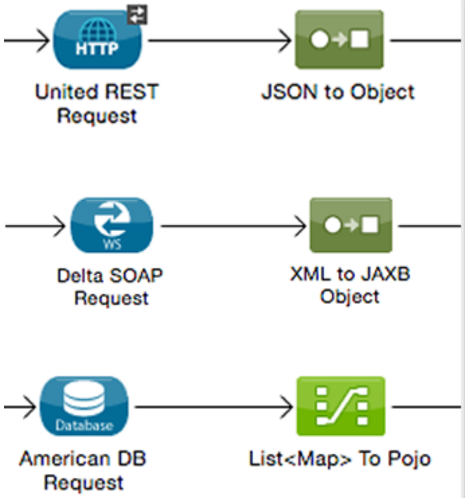
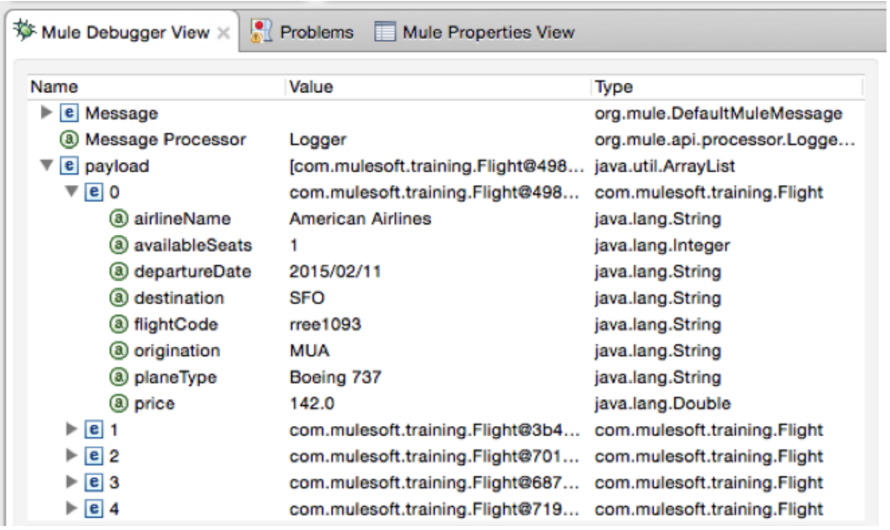
Module 5: Transforming Data





Name	Value	Type
▶ e Message		org.mule.DefaultMuleMessage
a Message Processor	Logger	org.mule.api.processor.Logge...
▼ e payload	[com.mulesoft.training.Flight@498...	java.util.ArrayList
▼ e 0	com.mulesoft.training.Flight@498...	com.mulesoft.training.Flight
a airlineName	American Airlines	java.lang.String
a availableSeats	1	java.lang.Integer
a departureDate	2015/02/11	java.lang.String
a destination	SFO	java.lang.String
a flightCode	rree1093	java.lang.String
a origination	MUA	java.lang.String
a planeType	Boeing 737	java.lang.String
a price	142.0	java.lang.Double
▶ e 1	com.mulesoft.training.Flight@3b4...	com.mulesoft.training.Flight
▶ e 2	com.mulesoft.training.Flight@701...	com.mulesoft.training.Flight
▶ e 3	com.mulesoft.training.Flight@687...	com.mulesoft.training.Flight
▶ e 4	com.mulesoft.training.Flight@719...	com.mulesoft.training.Flight

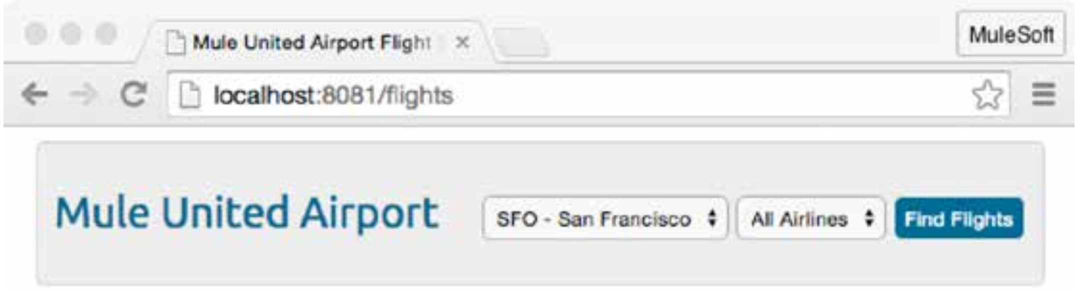
In this module, you will learn:

- About the different types of transformers.
- To transform objects to and/from XML and JSON.
- To do more complicated JSON to object mappings with Jackson annotations.
- To do more complicated XML to object mappings with JAXB annotations.
- To streamline complex data transformations with DataSense and the DataMapper.
- To create a custom transformer with Java.

Walkthrough 5-1: Load external content into a message

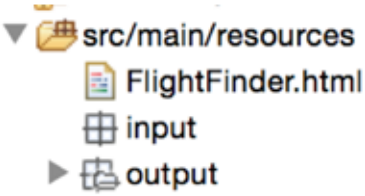
In this walkthrough, you will add an HTML form to your application that will eventually serve as a front end for the MUA integration application. You will:

- Create a new flow that receives GET requests at http://localhost:8081/flights.
- Use the Parse Template transformer to load the content of an HTML file into a flow.



Add an HTML file to the project

1. On your computer, navigate to the student files folder for the course, MUEssentials3.7_studentFiles_{date}.
2. Copy and paste (or drag) the resources/FlightFinder.html file into your project's src/main/resources folder.



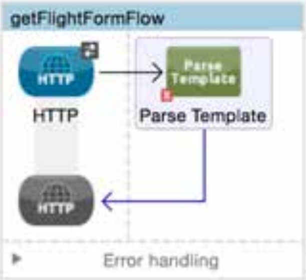
Note: You will examine the HTML code soon – after you load the form and see what it looks like.

Create a flow

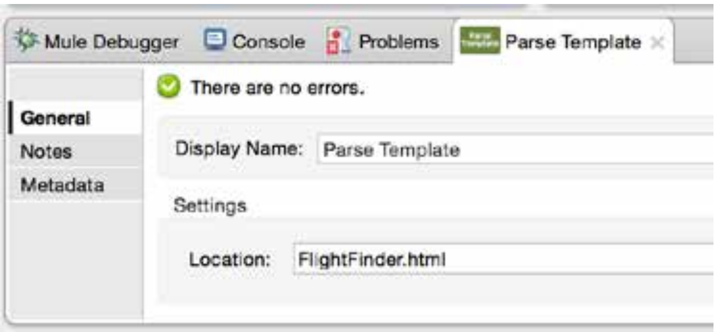
3. Return to apessentials.xml.
4. Drag out another HTTP connector to create a new flow at the top of the canvas and name it getFlightFormFlow.
5. In the HTTP Properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
6. Set the path to /flights and set the allowed methods to GET.

Set the payload to HTML and JavaScript

7. Add a Parse Template transformer to the process section of the flow.



8. In the Properties view for the transformer, set the location to FlightFinder.html.



There are no errors.

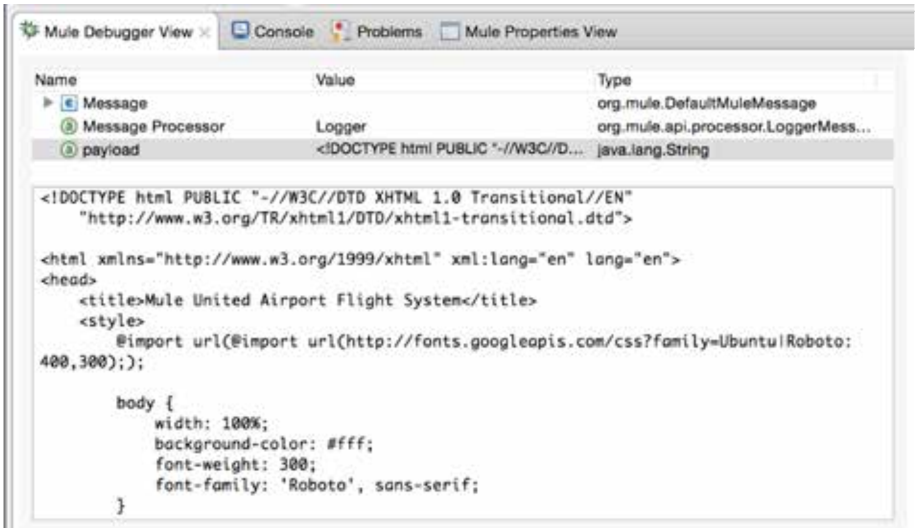
Display Name: Parse Template

Settings

Location: FlightFinder.html

Debug the application

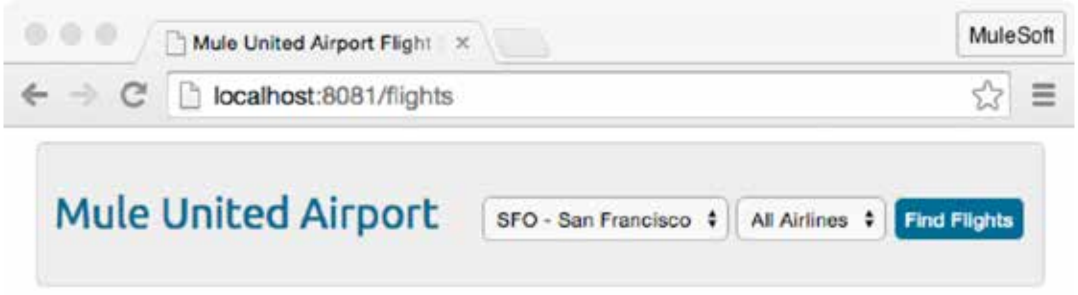
9. Add a breakpoint to the Parse Template transformer.
10. Add a Logger component after the transformer.
11. Save the file and debug the application.
12. In a browser, make a request to http://localhost:8081/flights.
13. Step through the flow and watch the value of the payload change.



Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMess...
payload	<!DOCTYPE html PUBLIC "-//W3C//D...	java.lang.String

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Mule United Airport Flight System</title>
    <style>
        @import url(@import url(http://fonts.googleapis.com/css?family=Ubuntu|Roboto:
400,300););

        body {
            width: 100%;
            background-color: #fff;
            font-weight: 300;
            font-family: 'Roboto', sans-serif;
        }
```

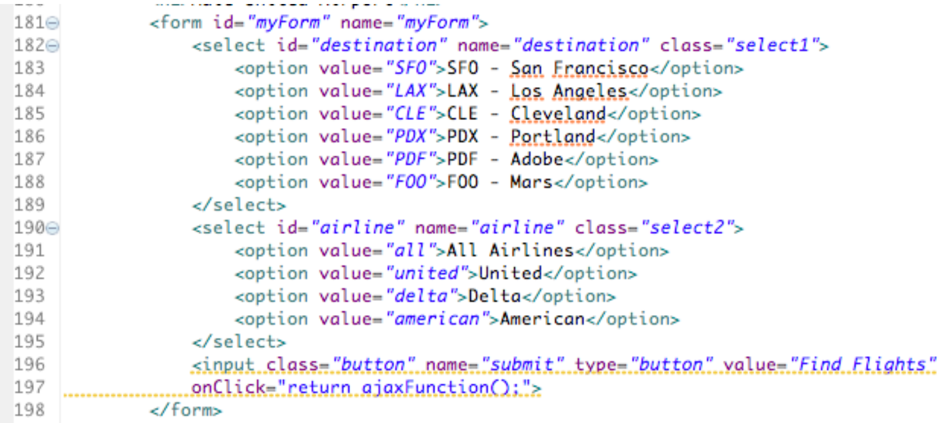
14. Step through the rest of the flow and then return to the browser window; you should see the HTML form.



15. Click the Find Flights button; what happens?

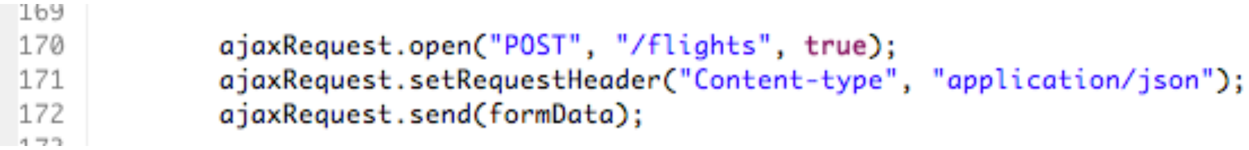
Review the HTML form code

16. Open FlightFinder.html in Anypoint Studio.
17. Locate the form at the bottom and find the names of the select boxes and their option values.



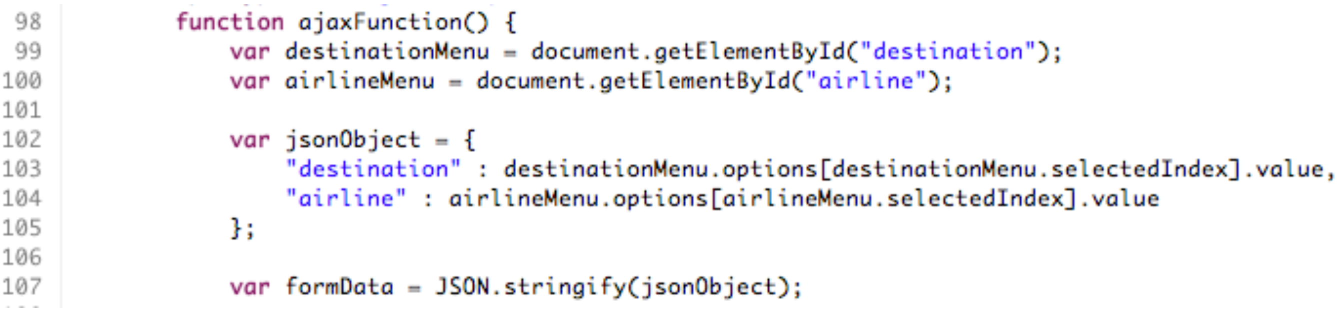
```
<form id="myForm" name="myForm">
    <select id="destination" name="destination" class="select1">
        <option value="SFO">SFO - San Francisco</option>
        <option value="LAX">LAX - Los Angeles</option>
        <option value="CLE">CLE - Cleveland</option>
        <option value="PDX">PDX - Portland</option>
        <option value="PDF">PDF - Adobe</option>
        <option value="FOO">FOO - Mars</option>
    </select>
    <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>
    <input class="button" name="submit" type="button" value="Find Flights"
    onClick="return ajaxFunction();">
</form>
```

18. Locate to where the form data is posted.



```
        ajaxRequest.open("POST", "/flights", true);
        ajaxRequest.setRequestHeader("Content-type", "application/json");
        ajaxRequest.send(formData);
```

19. Locate in what format the data is that is posted.

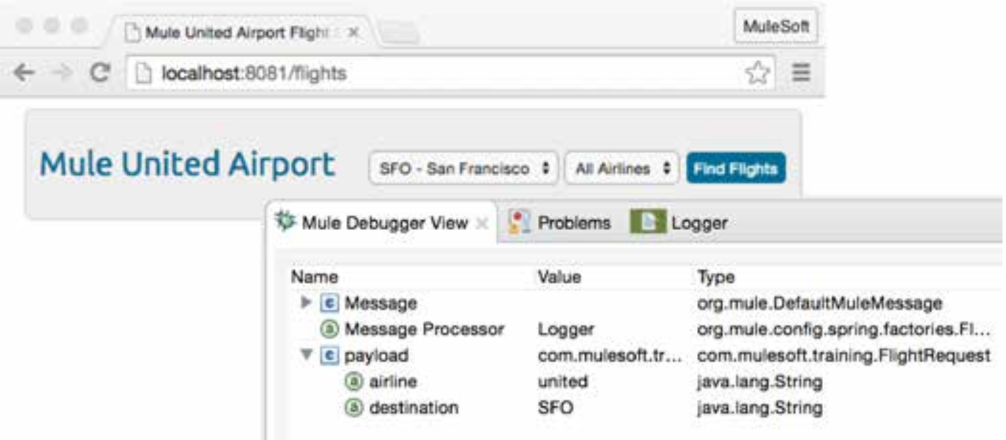


```
    function ajaxFunction() {
        var destinationMenu = document.getElementById("destination");
        var airlineMenu = document.getElementById("airline");

        var jsonObject = {
            "destination" : destinationMenu.options[destinationMenu.selectedIndex].value,
            "airline" : airlineMenu.options[airlineMenu.selectedIndex].value
        };

        var formData = JSON.stringify(jsonObject);
```

Walkthrough 5-2: Transform JSON to an object

In this walkthrough, you will work with the data posted from the HTML form. You will:

- Create a new flow that receives POST requests at http://localhost:8081/flights.
- Transform the form data from JSON to a FlightRequest object and save the destination in a flow variable.
- Use the JSON to Object transformer.



Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.config.spring.factories.Fl...
payload	com.mulesoft.tr...	com.mulesoft.training.FlightRequest
airline	united	java.lang.String
destination	SFO	java.lang.String

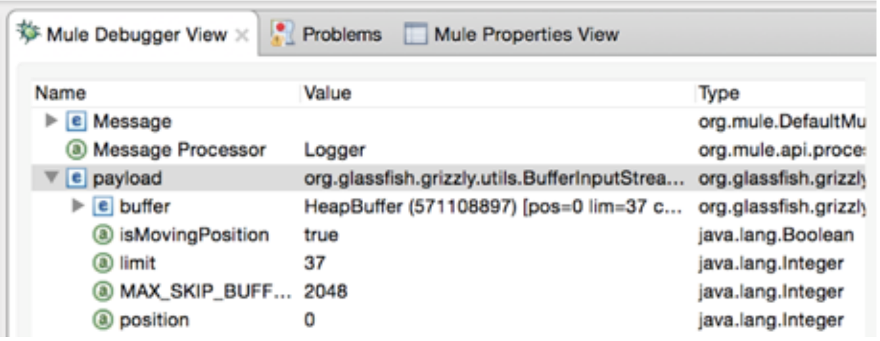
Create a new flow

1. Return to apessentials.xml.
2. Drag out another HTTP connector to create a new flow and name it getFlightsFlow
3. In the HTTP Properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
4. Set the path to /flights (the same as the form flow) and set the allowed methods to POST.

Look at the data posted from the form

5. Add a Logger to the flow and add a breakpoint to it.
6. Save the file to redeploy the application in debug mode.
7. Make another request to http://localhost:8081/flights.
8. Step through or remove the breakpoints in the getFlightsFormFlow to get to the form in the browser window.
9. Click the Find Flights button.

10. Look at the Mule Debugger view; you should see the return data is a BufferInputStream.



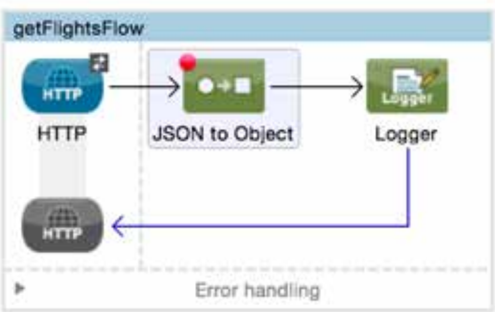
Name	Value	Type
▶ Message		org.mule.DefaultMu
Message Processor	Logger	org.mule.api.proce:
▼ payload	org.glassfish.grizzly.utils.BufferInputStrea...	org.glassfish.grizzly
▶ buffer	HeapBuffer (571108897) [pos=0 lim=37 c...	org.glassfish.grizzly
isMovingPosition	true	java.lang.Boolean
limit	37	java.lang.Integer
MAX_SKIP_BUFF...	2048	java.lang.Integer
position	0	java.lang.Integer

11. Click the Resume button.

Add a JSON to Object transformer

12. Add a JSON to Object transformer before the Logger.

13. Add a breakpoint to the JSON to Object transformer if it does not have one.

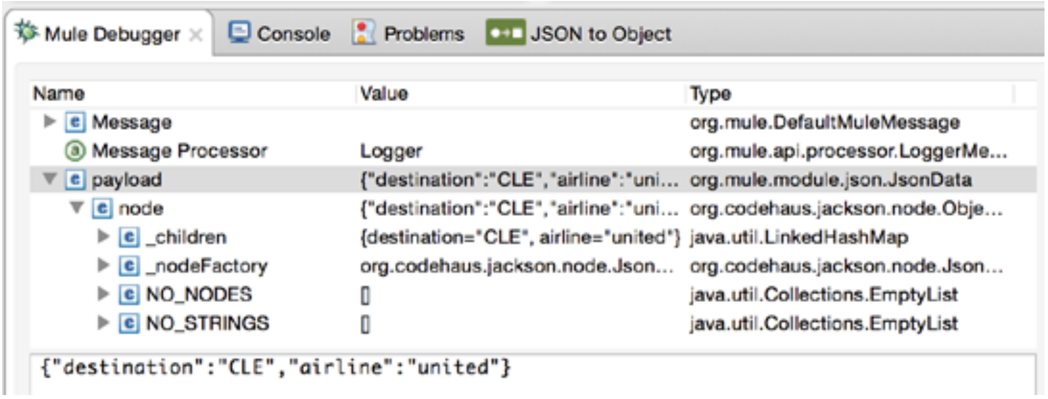


Debug the application

14. Save the file to redeploy the application in debug mode.

15. Make a request to http://localhost:8081/flights and submit the form again.

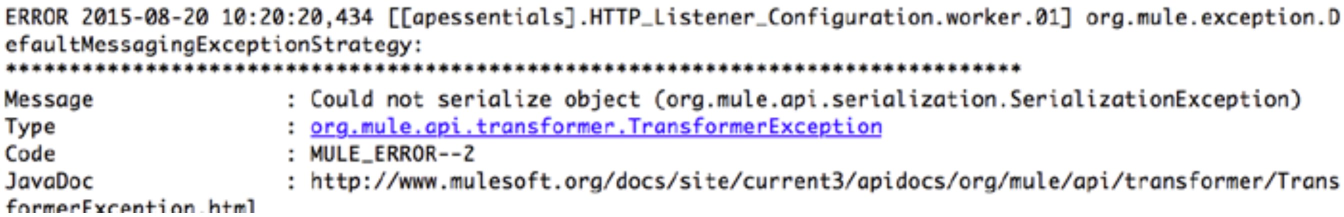
16. Step to the Logger in the getFlightsFlow; you should see the form data is a JsonData object.



Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMe...
▼ payload	{"destination":"CLE","airline":"uni...	org.mule.module.json.JsonData
▼ node	{"destination":"CLE","airline":"uni...	org.codehaus.jackson.node.Obje...
▶ _children	{destination="CLE", airline="united"}	java.util.LinkedHashMap
▶ _nodeFactory	org.codehaus.jackson.node.Json...	org.codehaus.jackson.node.Json...
▶ NO_NODES	[]	java.util.Collections.EmptyList
▶ NO_STRINGS	[]	java.util.Collections.EmptyList

```
{"destination":"CLE","airline":"united"}
```

17. Resume code execution and look at the console; you should get a TransformerException.

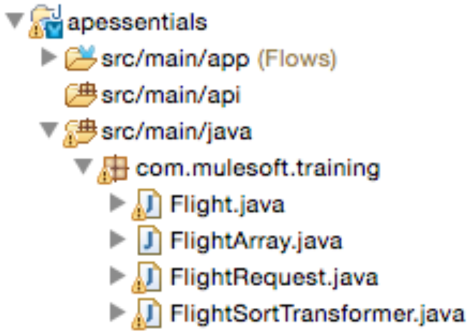


```
ERROR 2015-08-20 10:20:20,434 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.exception.D
efaultMessagingExceptionStrategy:
********************************************************************************
Message               : Could not serialize object (org.mule.api.serialization.SerializationException)
Type                  : org.mule.api.transformer.TransformerException
Code                  : MULE_ERROR--2
JavaDoc               : http://www.mulesoft.org/docs/site/current3/apidocs/org/mule/api/transformer/Trans
formerException.html
```

Add Java classes file to the project

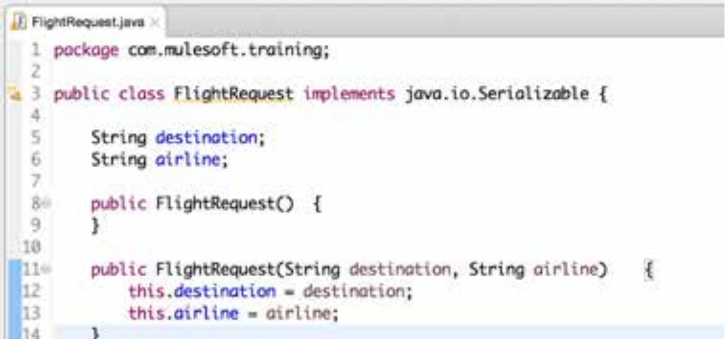
18. On your computer, navigate to the java folder in the student files folder for the course: MUEssentials3.7_studentFiles_{date}/java.

19. Copy and paste (or drag) the com.mulesoft.training folder into your Anypoint Studio project's src/main/java folder.



```
▼ apessentials
    ▶ src/main/app (Flows)
      src/main/api
    ▼ src/main/java
        ▼ com.mulesoft.training
            ▶ Flight.java
            ▶ FlightArray.java
            ▶ FlightRequest.java
            ▶ FlightSortTransformer.java
```

20. Open the FlightRequest.java class and examine the code.

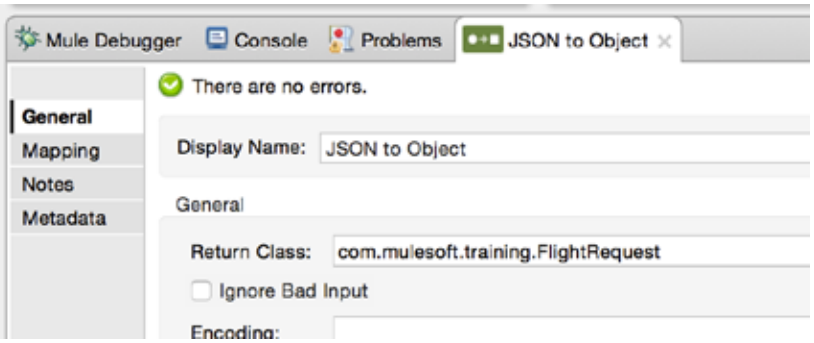


```
package com.mulesoft.training;

public class FlightRequest implements java.io.Serializable {

    String destination;
    String airline;

    public FlightRequest()  {
    }

    public FlightRequest(String destination, String airline)    {
        this.destination = destination;
        this.airline = airline;
    }
```

Set the transformer return type

21. Return to apessentials.xml.

22. In the JSON to Object Properties view, set the return class to com.mulesoft.training.FlightRequest.



There are no errors.

Display Name: JSON to Object

General

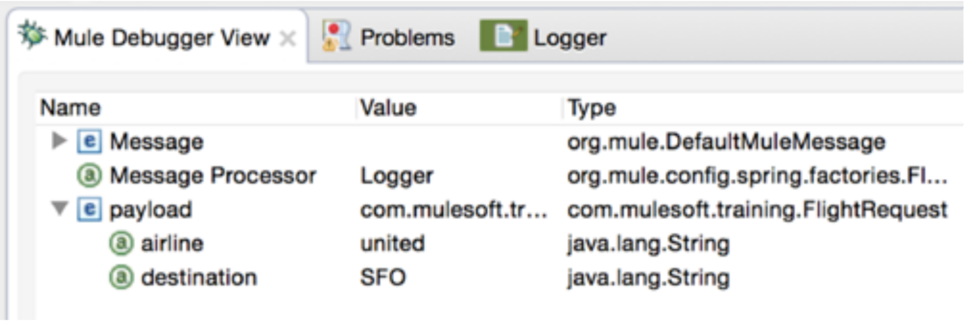
Return Class: com.mulesoft.training.FlightRequest

□ Ignore Bad Input

Encoding:

Debug the application

23. Save the file to redeploy the application in debug mode.
24. Make a request to http://localhost:8081/flights and submit the form again.
25. Step to the Logger in the getFlightsFlow; you should now see the see the form data as a FlightRequest object.

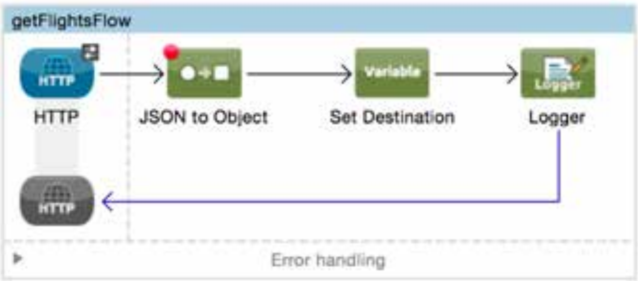


Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.config.spring.factories.Fl...
▼ payload	com.mulesoft.tr...	com.mulesoft.training.FlightRequest
airline	united	java.lang.String
destination	SFO	java.lang.String

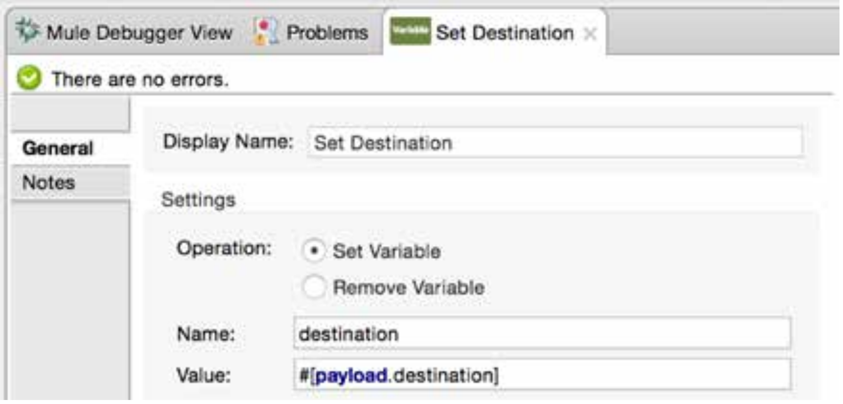
26. Click the Resume button.

Store form data in a flow variable

27. Add a Variable transformer after the JSON to Object transformer.
28. In the Properties view, set the display name to Set Destination.



29. Select Set Variable and set the name to destination and the value to #[payload.destination].



There are no errors.

Display Name: Set Destination

Settings

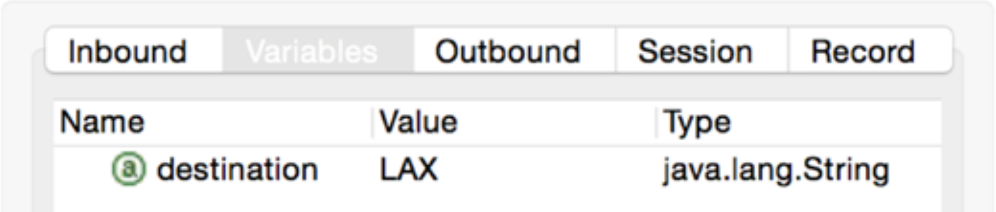
Operation: ● Set Variable ○ Remove Variable

Name: destination

Value: #[payload.destination]

Test the application

30. Save the file to redeploy the application in debug mode.
31. Make a request to http://localhost:8081/flights and submit the form again.
32. Step to the Logger and click the Variables tab; you should see your flow variable.



Name	Value	Type
ⓐ destination	LAX	java.lang.String

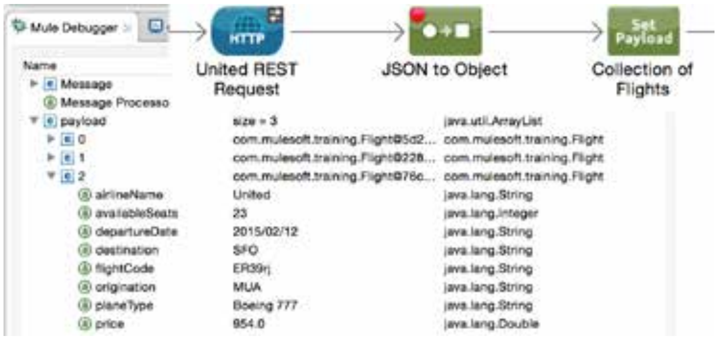
33. Stop the Mule Debugger.

Note: You will continue working with this flow in a later walkthrough. You will add functionality to get the requested flight data from the other flows and return it back to the form.

Walkthrough 5-3: Transform JSON to a collection of objects

In this walkthrough, you will work with the United flight data. You will:

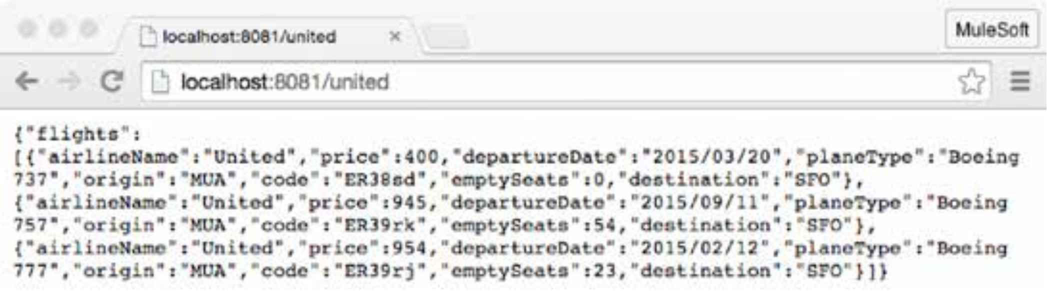
- Transform the United flight data from JSON to a collection of Flight objects.
- Use the JSON to Object transformer with a return class using Jackson annotations.



```
Name
▶ Message
  Message Processo
▼ payload              size = 3                          java.util.ArrayList
  ▶ 0                  com.mulesoft.training.Flight@5d2...  com.mulesoft.training.Flight
  ▶ 1                  com.mulesoft.training.Flight@228...  com.mulesoft.training.Flight
  ▼ 2                  com.mulesoft.training.Flight@76c...  com.mulesoft.training.Flight
      airlineName      United                            java.lang.String
      availableSeats   23                                java.lang.Integer
      departureDate    2015/02/12                        java.lang.String
      destination      SFO                               java.lang.String
      flightCode       ER39rj                            java.lang.String
      origination      MUA                               java.lang.String
      planeType        Boeing 777                        java.lang.String
      price            954.0                             java.lang.Double
```

Review the United flight data

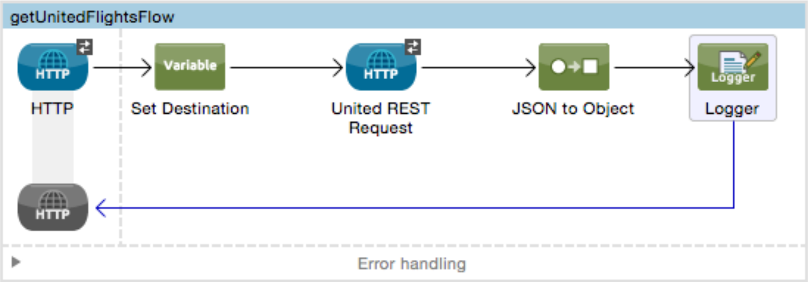
1. Return to apessentials.xml.
2. Run the application and make a request to http://localhost:8081/united.
3. Look at the names of the object keys.



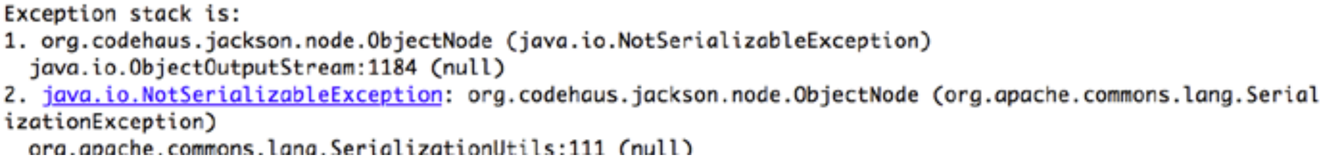
```
{"flights":
[{"airlineName":"United","price":400,"departureDate":"2015/03/20","planeType":"Boeing
737","origin":"MUA","code":"ER38sd","emptySeats":0,"destination":"SFO"},
{"airlineName":"United","price":945,"departureDate":"2015/09/11","planeType":"Boeing
757","origin":"MUA","code":"ER39rk","emptySeats":54,"destination":"SFO"},
{"airlineName":"United","price":954,"departureDate":"2015/02/12","planeType":"Boeing
777","origin":"MUA","code":"ER39rj","emptySeats":23,"destination":"SFO"}]}
```

Transform JSON to a general collection of objects

4. Locate getUnitedFlightsFlow.
5. Add a JSON to Object transformer after the United REST Request.
6. Add a Logger after the transformer.

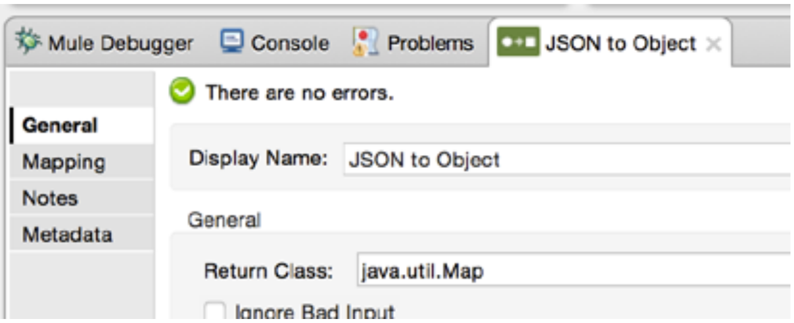


7. Save the file to redeploy the application.
8. Make a request to http://localhost:8081/united; you should get a NotSerializableException in the console.



```
Exception stack is:
1. org.codehaus.jackson.node.ObjectNode (java.io.NotSerializableException)
  java.io.ObjectOutputStream:1184 (null)
2. java.io.NotSerializableException: org.codehaus.jackson.node.ObjectNode (org.apache.commons.lang.Serial
izationException)
  org.apache.commons.lang.SerializationUtils:111 (null)
```

9. Navigate to the Properties view for the JSON to Object transformer.
10. Set the return class to java.util.Map.



There are no errors.

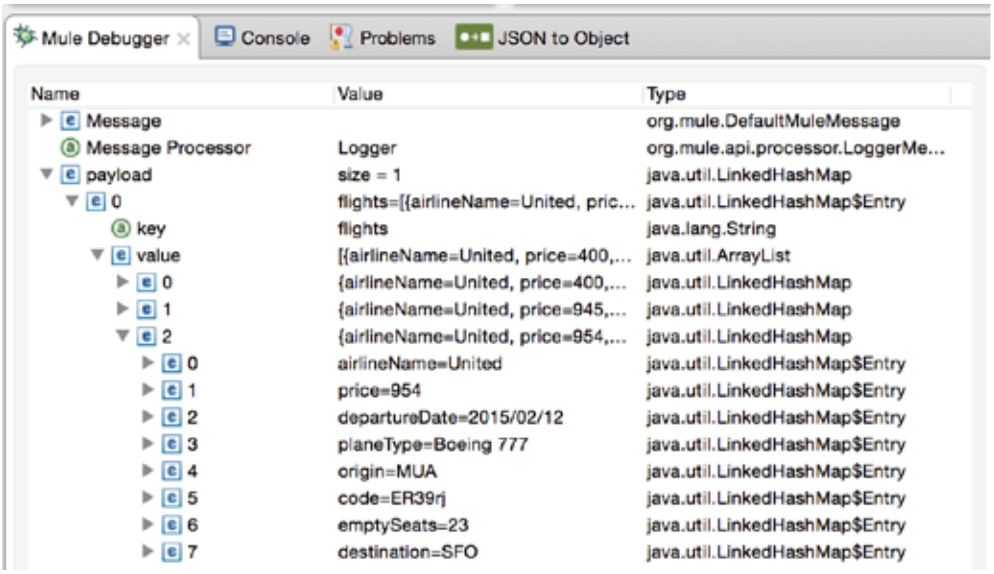
Display Name: JSON to Object

Return Class: java.util.Map

□ Ignore Bad Input

Debug the application

11. Add a breakpoint to the JSON to Object transformer.
12. Save the file and debug the application.
13. Make a request to http://localhost:8081/united.
14. Watch the payload value and step through the flow; you should see the payload transformed to a collection of objects.

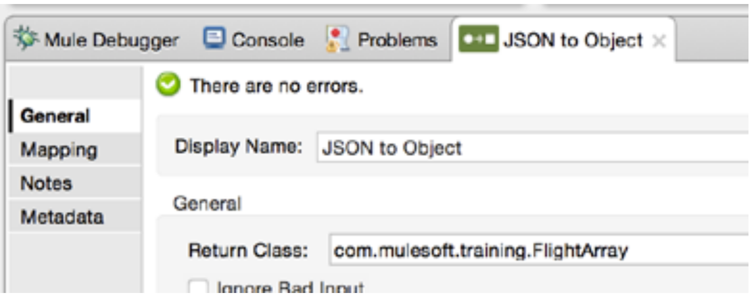


Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMe...
▼ payload	size = 1	java.util.LinkedHashMap
▼ 0	flights=[{airlineName=United, pric...	java.util.LinkedHashMap$Entry
key	flights	java.lang.String
▼ value	[{airlineName=United, price=400,...	java.util.ArrayList
▶ 0	{airlineName=United, price=400,...	java.util.LinkedHashMap
▶ 1	{airlineName=United, price=945,...	java.util.LinkedHashMap
▼ 2	{airlineName=United, price=954,...	java.util.LinkedHashMap
▶ 0	airlineName=United	java.util.LinkedHashMap$Entry
▶ 1	price=954	java.util.LinkedHashMap$Entry
▶ 2	departureDate=2015/02/12	java.util.LinkedHashMap$Entry
▶ 3	planeType=Boeing 777	java.util.LinkedHashMap$Entry
▶ 4	origin=MUA	java.util.LinkedHashMap$Entry
▶ 5	code=ER39rj	java.util.LinkedHashMap$Entry
▶ 6	emptySeats=23	java.util.LinkedHashMap$Entry
▶ 7	destination=SFO	java.util.LinkedHashMap$Entry

15. Step through the rest of the flow and then return to the request window; you should get some garbled data or a download file – your request tool's interpretation of the complex object.

Transform JSON to a collection of custom objects

16. Open the FlightArray and Flight classes and examine the code.
17. Return to the Properties view for the JSON to Object transformer in apessentials.xml.
18. Change the return class to com.mulesoft.training.FlightArray.



There are no errors.

Display Name: JSON to Object

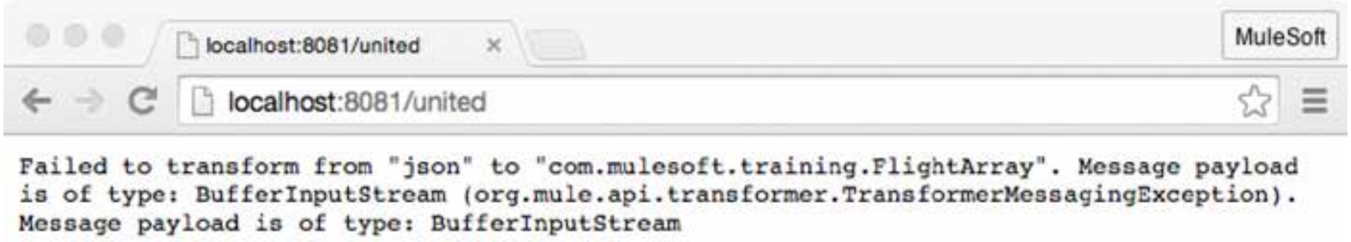
General

Return Class: com.mulesoft.training.FlightArray

□ Ignore Bad Input

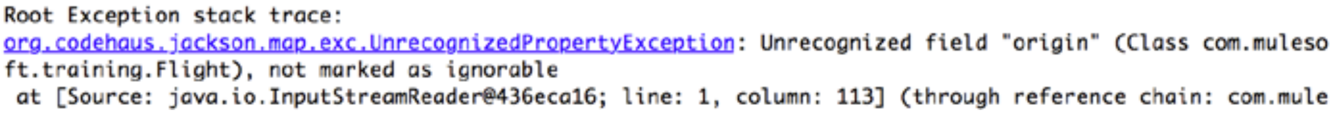
Test the application

19. Save the file to redeploy the application.
20. Make a request to http://localhost:8081/united and step through the application; you should get a Failed to transform error.



```
Failed to transform from "json" to "com.mulesoft.training.FlightArray". Message payload
is of type: BufferInputStream (org.mule.api.transformer.TransformerMessagingException).
Message payload is of type: BufferInputStream
```

21. Locate the error details in the console; you should see a reference to unrecognized origin.



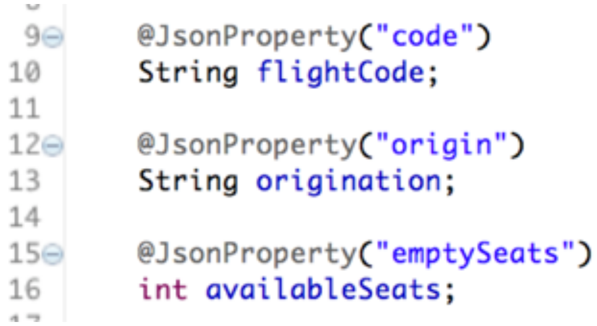
```
Root Exception stack trace:
org.codehaus.jackson.map.exc.UnrecognizedPropertyException: Unrecognized field "origin" (Class com.muleso
ft.training.Flight), not marked as ignorable
 at [Source: java.io.InputStreamReader@436eca16; line: 1, column: 113] (through reference chain: com.mule
```

Add Jackson annotations to the Flight class

22. Return to Flight.java.

23. Add a @JsonProperty annotation to map the following JSON keys to Flight properties.

- code to flightCode
- origin to origination
- emptySeats to availableSeats

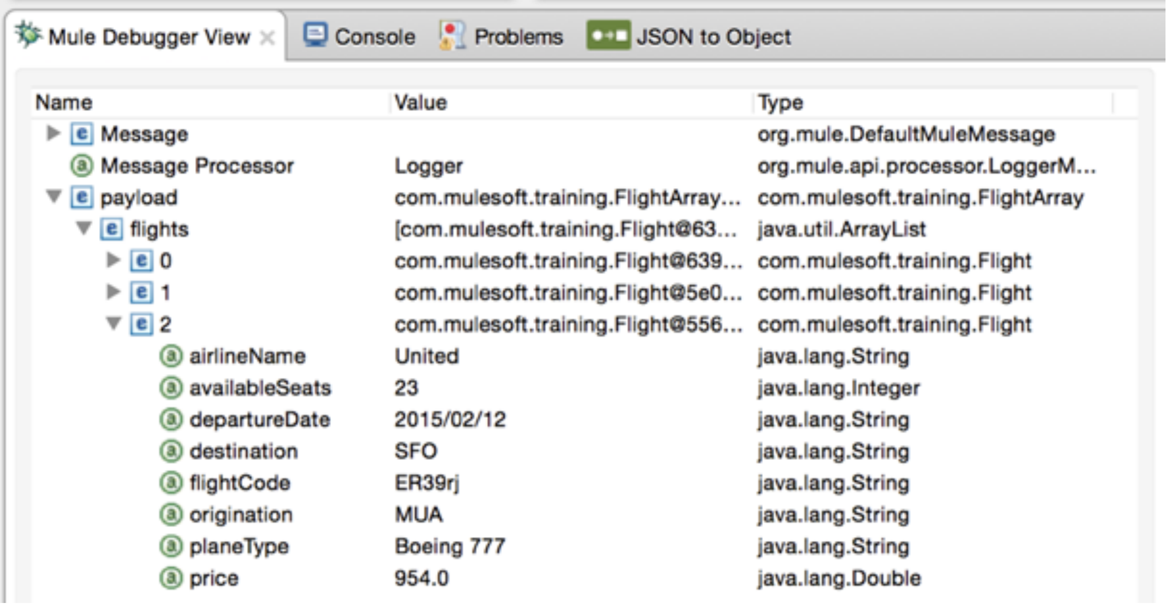


```
    @JsonProperty("code")
    String flightCode;

    @JsonProperty("origin")
    String origination;

    @JsonProperty("emptySeats")
    int availableSeats;
```

24. Click one of the Error icons next to the line numbers and double-click Import JsonProperty (code. org.codehaus.jackson.annotate.JsonProperty).
25. Save the file.

Debug the application

26. Return to apessentials.xml.
27. Debug the application and make another request to http://localhost:8081/united.
28. Watch the payload value and step through the flow; you should see the payload transformed to a FlightArray of Flight objects.

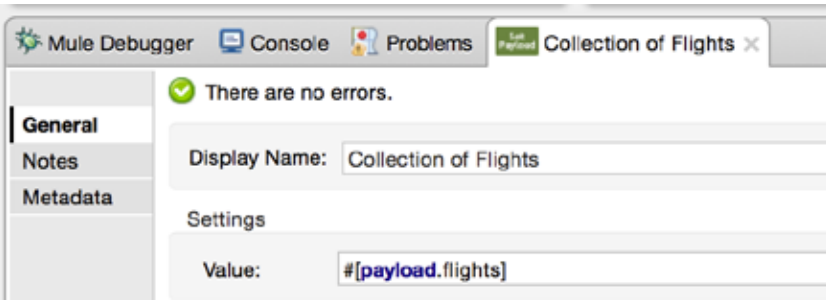


Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerM...
▼ payload	com.mulesoft.training.FlightArray...	com.mulesoft.training.FlightArray
▼ flights	[com.mulesoft.training.Flight@63...	java.util.ArrayList
▶ 0	com.mulesoft.training.Flight@639...	com.mulesoft.training.Flight
▶ 1	com.mulesoft.training.Flight@5e0...	com.mulesoft.training.Flight
▼ 2	com.mulesoft.training.Flight@556...	com.mulesoft.training.Flight
airlineName	United	java.lang.String
availableSeats	23	java.lang.Integer
departureDate	2015/02/12	java.lang.String
destination	SFO	java.lang.String
flightCode	ER39rj	java.lang.String
origination	MUA	java.lang.String
planeType	Boeing 777	java.lang.String
price	954.0	java.lang.Double

29. Click the Resume button.

Return only the collection of flights

30. Add a Set Payload transformer after the JSON to Object transformer.
31. Set it's display name to Collection of Flights and its value to #[payload.flights].



There are no errors.

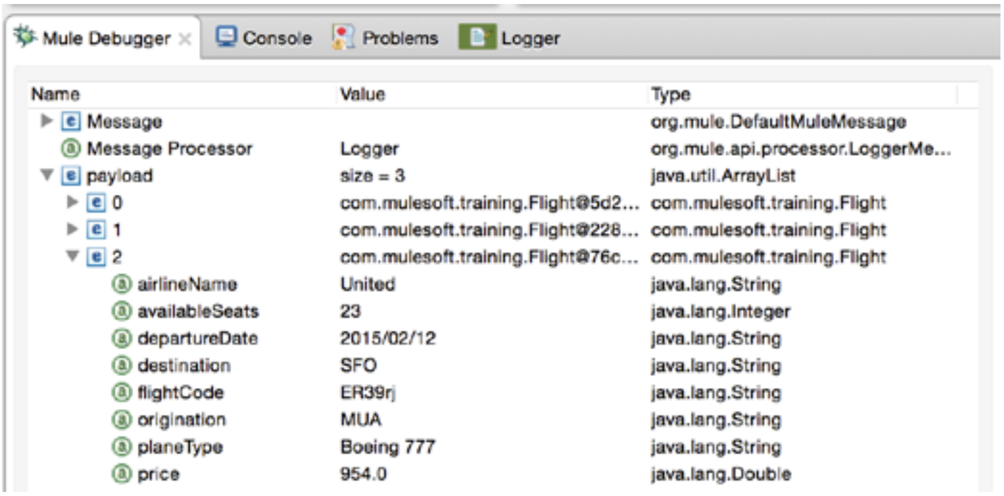
Display Name: Collection of Flights

Value: #[payload.flights]

32. Set the Logger message to #[payload].

Test the application

33. Save the file to redeploy the application.
34. Make another request to http://localhost:8081/united,
35. Step through to the Logger; you should now see just the collection of objects returned.



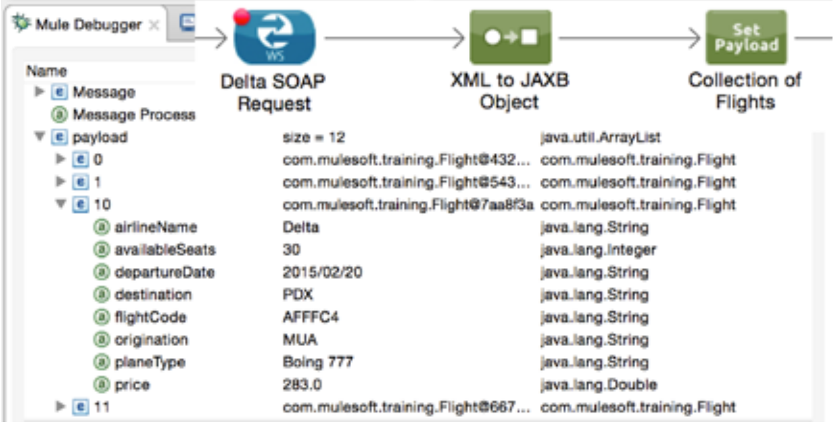
Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMe...
payload	size = 3	java.util.ArrayList
0	com.mulesoft.training.Flight@5d2...	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@228...	com.mulesoft.training.Flight
2	com.mulesoft.training.Flight@76c...	com.mulesoft.training.Flight
airlineName	United	java.lang.String
availableSeats	23	java.lang.Integer
departureDate	2015/02/12	java.lang.String
destination	SFO	java.lang.String
flightCode	ER39rj	java.lang.String
origination	MUA	java.lang.String
planeType	Boeing 777	java.lang.String
price	954.0	java.lang.Double

36. Stop the debugger.

Walkthrough 5-4: Transform XML to a collection of objects

In this walkthrough, you will work with the Delta flight data. You will:

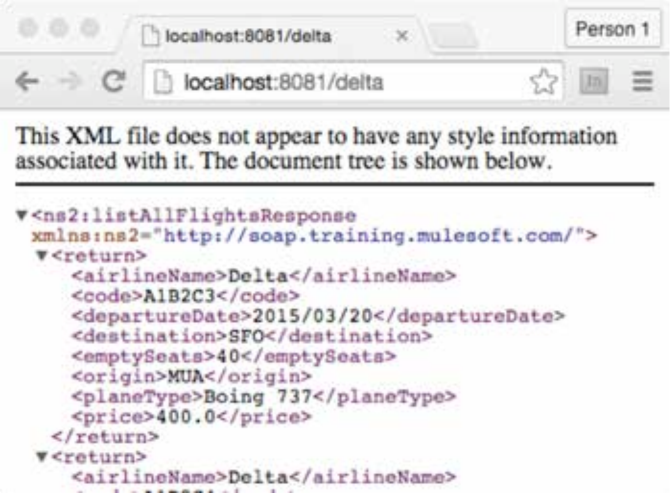
- Transform the Delta flight data from XML to a collection of Flight objects.
- Use the XML to JAXB Object transformer with a return class using JAXB annotations.



```
Name
▶ e Message
  a Message Process
▼ e payload                    size = 12                             java.util.ArrayList
  ▶ e 0                        com.mulesoft.training.Flight@432...   com.mulesoft.training.Flight
  ▶ e 1                        com.mulesoft.training.Flight@543...   com.mulesoft.training.Flight
  ▼ e 10                       com.mulesoft.training.Flight@7aa8f3a  com.mulesoft.training.Flight
      a airlineName            Delta                                 java.lang.String
      a availableSeats         30                                    java.lang.Integer
      a departureDate          2015/02/20                            java.lang.String
      a destination            PDX                                   java.lang.String
      a flightCode             AFFFC4                                java.lang.String
      a origination            MUA                                   java.lang.String
      a planeType              Boing 777                             java.lang.String
      a price                  283.0                                 java.lang.Double
  ▶ e 11                       com.mulesoft.training.Flight@667...   com.mulesoft.training.Flight
```

Review the Delta flight data

1. Return to apessentials.xml and locate getDeltaFlightsFlow.
2. Run the application and make a request to http://localhost:8081/delta.
3. Look at the names of the XML elements.



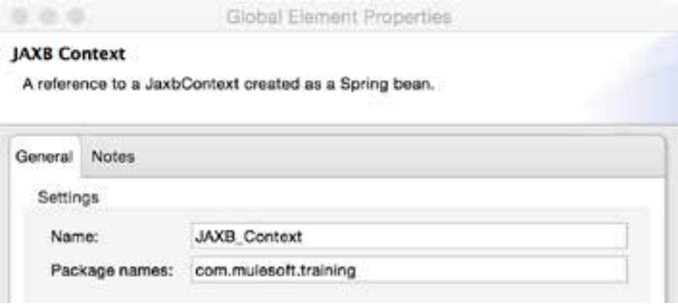
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ns2:listAllFlightsResponse
  xmlns:ns2="http://soap.training.mulesoft.com/">
  ▼<return>
     <airlineName>Delta</airlineName>
     <code>A1B2C3</code>
     <departureDate>2015/03/20</departureDate>
     <destination>SFO</destination>
     <emptySeats>40</emptySeats>
     <origin>MUA</origin>
     <planeType>Boing 737</planeType>
     <price>400.0</price>
   </return>
  ▼<return>
     <airlineName>Delta</airlineName>
```

Add and configure an XML to JAXB Object transformer

4. Delete the DOM to XML transformer.
5. Add an XML to JAXB Object transformer after the Delta SOAP Request.

6. Navigate to the XML to JAXB Object Properties view; you should see a message that a jaxbContext-ref is required.
7. Scroll down in the Properties view and locate the Settings > Context section.
8. Click the Add button next to context.
9. In the Global Element Properties dialog box, set the package name to com.mulesoft.training and click OK.



JAXB Context

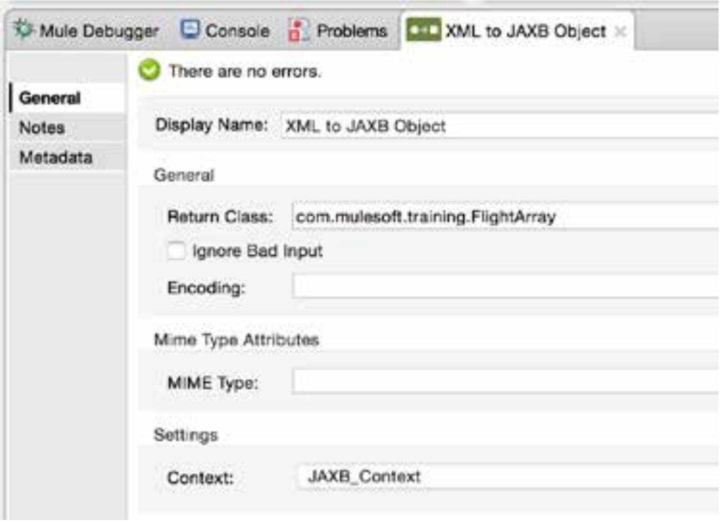
A reference to a JaxbContext created as a Spring bean.

Settings

Name: JAXB_Context

Package names: com.mulesoft.training

10. Back in the Properties view, set the return class to com.mulesoft.training.FlightArray.



There are no errors.

Display Name: XML to JAXB Object

General

Return Class: com.mulesoft.training.FlightArray

□ Ignore Bad Input

Encoding:

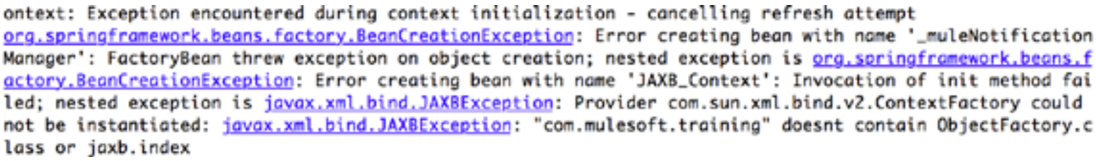
Mime Type Attributes

MIME Type:

Settings

Context: JAXB_Context

11. Save the file to redeploy the application; you should get an error in the console that com.mulesoft.training does not have a jaxb.ini file.

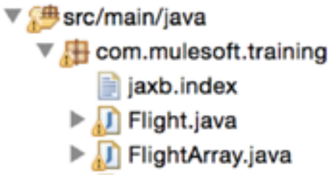


```
ontext: Exception encountered during context initialization - cancelling refresh attempt
org.springframework.beans.factory.BeanCreationException: Error creating bean with name '_muleNotification
Manager': FactoryBean threw exception on object creation; nested exception is org.springframework.beans.f
actory.BeanCreationException: Error creating bean with name 'JAXB_Context': Invocation of init method fai
led; nested exception is javax.xml.bind.JAXBException: Provider com.sun.xml.bind.v2.ContextFactory could
not be instantiated: javax.xml.bind.JAXBException: "com.mulesoft.training" doesnt contain ObjectFactory.c
lass or jaxb.index
```

Create a jaxb.index file

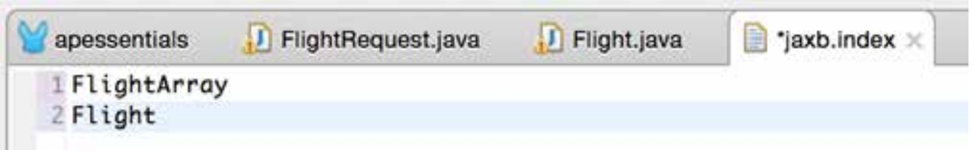
12. Right-click the com.mulesoft.training package in the Package Explorer and select New > File.

13. In the New file dialog box, set the file name to jaxb.index and click Finish.



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

14. Open jaxb.index and list the two Java classes to be used in the mapping.

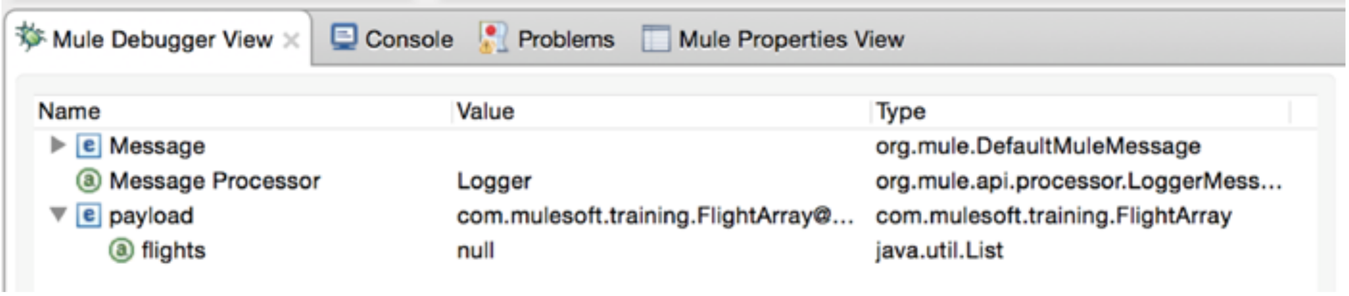


```
FlightArray
Flight
```

15. Save the file.

Debug the application

16. Return to apessentials.xml.
17. Make sure there is a breakpoint on the Delta SOAP Request and debug the application.
18. Make a request to http://localhost:8081/delta.
19. Watch the payload value and step to the Logger; you should see the payload transformed to a FlightArray but the flights property is null – there wasn't enough data to map the XML to the objects correctly.

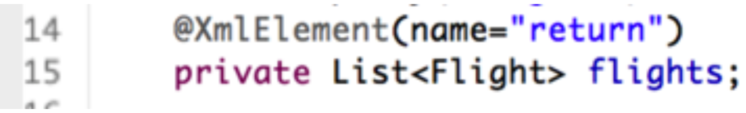


Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMess...
▼ payload	com.mulesoft.training.FlightArray@...	com.mulesoft.training.FlightArray
flights	null	java.util.List

20. Stop the debugger.

Add JAXB annotations to the FlightArray class

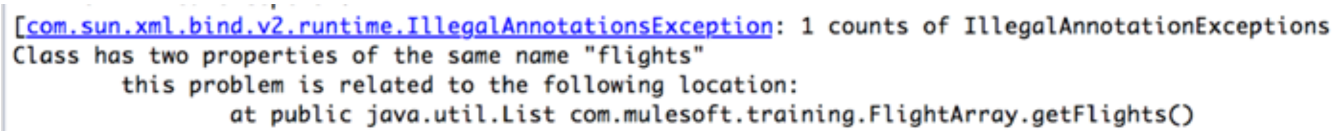
21. Open FlightArray.java.
22. Add an @XmlElement annotation to map the XML return node to the flights property; be sure to import the appropriate class (javax.xml.bind.annotation.XmlElement).



```
    @XmlElement(name="return")
    private List<Flight> flights;
```

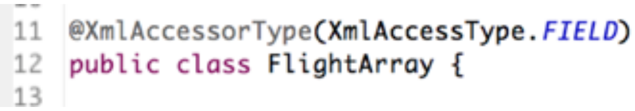
23. Save the file.

24. If you want, run the application again and make a request to http://localhost:8081/delta; you will get an IllegalAnnotationsException.



```
[com.sun.xml.bind.v2.runtime.IllegalAnnotationsException: 1 counts of IllegalAnnotationExceptions
Class has two properties of the same name "flights"
        this problem is related to the following location:
                at public java.util.List com.mulesoft.training.FlightArray.getFlights()
```

25. Add an @XmlAccessorType annotation for the class set to the static value, XmlAccessType.Field; be sure to import the appropriate classes.

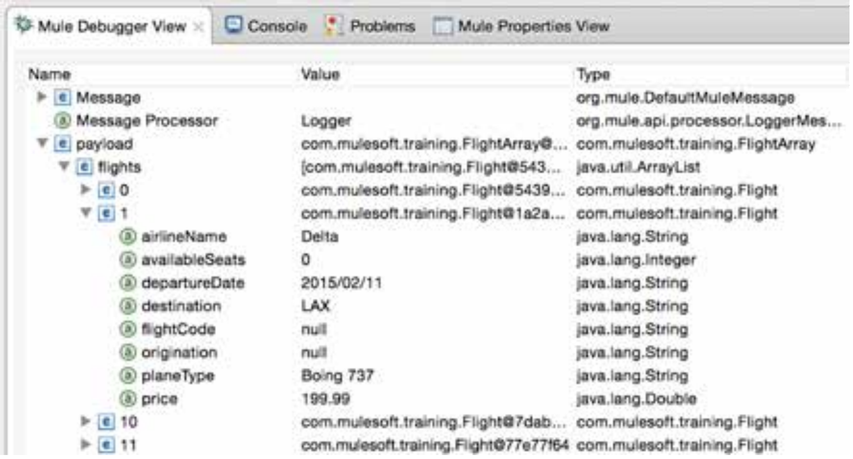


```
@XmlAccessorType(XmlAccessType.FIELD)
public class FlightArray {

```

26. Save the file.

Debug the application

27. Debug the application and make a request to http://localhost:8081/delta.
28. Watch the payload value and step to the Logger; you should see the payload transformed to a FlightArray of Flight objects.

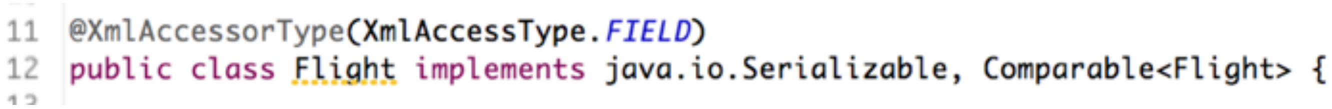


Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMes...
payload	com.mulesoft.training.FlightArray@...	com.mulesoft.training.FlightArray
flights	[com.mulesoft.training.Flight@543...	java.util.ArrayList
0	com.mulesoft.training.Flight@5439...	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@1a2a...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	0	java.lang.Integer
departureDate	2015/02/11	java.lang.String
destination	LAX	java.lang.String
flightCode	null	java.lang.String
origination	null	java.lang.String
planeType	Boing 737	java.lang.String
price	199.99	java.lang.Double
10	com.mulesoft.training.Flight@7dab...	com.mulesoft.training.Flight
11	com.mulesoft.training.Flight@77e77f64	com.mulesoft.training.Flight

29. Explore the data and notice that availableSeats is always zero and the flightCode and origination null.
30. Stop the debugger.

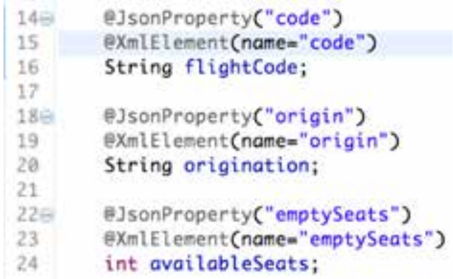
Add JAXB annotations to the Flight class

31. Return to Flight.java.
32. Add an @XmlAccessorType annotation for the class set to the static value, XmlAccessType.Field; be sure to import the appropriate classes.



```
@XmlAccessorType(XmlAccessType.FIELD)
public class Flight implements java.io.Serializable, Comparable<Flight> {
```

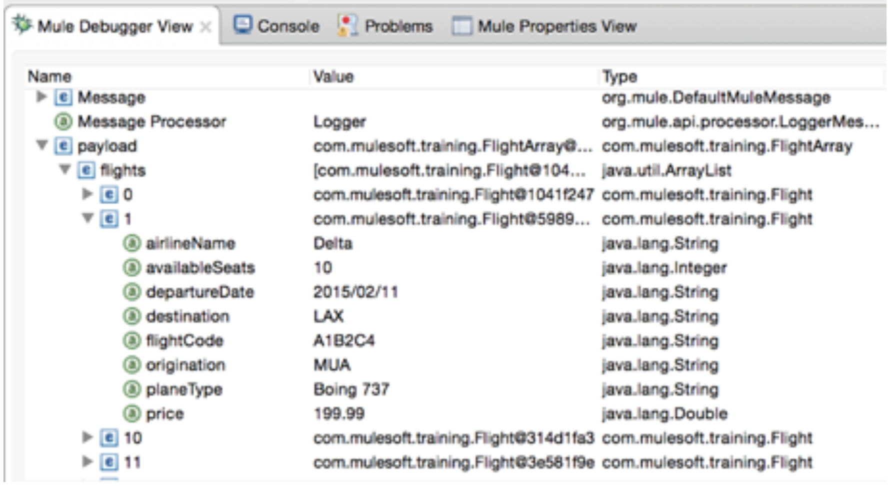
33. Add an @XmlElement annotation to map the XML flightCode node to the code property; be sure to import the appropriate class.
34. Add an @XmlElement annotation to map the XML origin node to the origination property.
35. Add an @XmlElement annotation to map the XML emptySeats node to the availableSeats property.



```
    @JsonProperty("code")
    @XmlElement(name="code")
    String flightCode;

    @JsonProperty("origin")
    @XmlElement(name="origin")
    String origination;

    @JsonProperty("emptySeats")
    @XmlElement(name="emptySeats")
    int availableSeats;
```

Debug the application

36. Debug the application again and make a request to http://localhost:8081/delta.
37. Watch the payload value and step to the Logger; you should see the payload successfully transformed to a FlightArray of Flight objects with all fields properly mapped.



Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMes...
▼ payload	com.mulesoft.training.FlightArray@...	com.mulesoft.training.FlightArray
▼ flights	[com.mulesoft.training.Flight@104...	java.util.ArrayList
▶ 0	com.mulesoft.training.Flight@1041f247	com.mulesoft.training.Flight
▼ 1	com.mulesoft.training.Flight@5989...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	10	java.lang.Integer
departureDate	2015/02/11	java.lang.String
destination	LAX	java.lang.String
flightCode	A1B2C4	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 737	java.lang.String
price	199.99	java.lang.Double
▶ 10	com.mulesoft.training.Flight@314d1fa3	com.mulesoft.training.Flight
▶ 11	com.mulesoft.training.Flight@3e581f9e	com.mulesoft.training.Flight

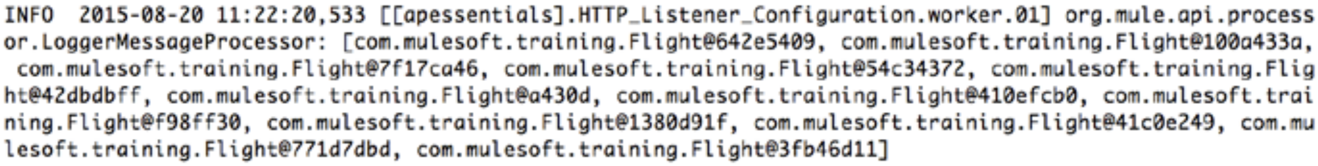
38. Resume the application; you should get a transformer exception.

Return only the collection of flights

39. Return to apessentials.xml.
40. Add a Set Payload transformer after the XML to JAXB Object transformer.
41. Set it's display name to Collection of Flights and its value to #[payload.flights].

Test the application

42. Save the file to redeploy the application.
43. Make a request to http://localhost:8081/united.
44. Step though or resume the application; you should now see just the collection of objects returned.

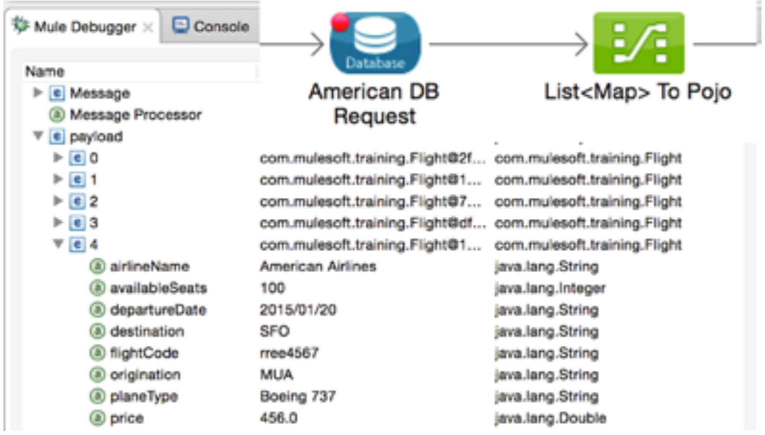


```
INFO  2015-08-20 11:22:20,533 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.api.process
or.LoggerMessageProcessor: [com.mulesoft.training.Flight@642e5409, com.mulesoft.training.Flight@100a433a,
 com.mulesoft.training.Flight@7f17ca46, com.mulesoft.training.Flight@54c34372, com.mulesoft.training.Flig
ht@42dbdbff, com.mulesoft.training.Flight@a430d, com.mulesoft.training.Flight@410efcb0, com.mulesoft.trai
ning.Flight@f98ff30, com.mulesoft.training.Flight@1380d91f, com.mulesoft.training.Flight@41c0e249, com.mu
lesoft.training.Flight@771d7dbd, com.mulesoft.training.Flight@3fb46d11]
```

Walkthrough 5-5: Use the DataMapper to map complex objects

In this walkthrough, you will work with the American flight data. You will:

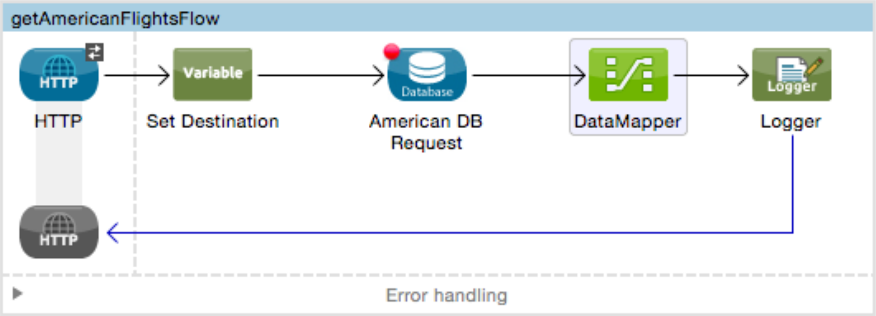
- Transform the American flight data from a List of Map objects to a collection of Flight objects.
- Use the DataMapper transformer.



Name		
▶ Message		
Message Processor		
▼ payload		
▶ 0	com.mulesoft.training.Flight@2f...	com.mulesoft.training.Flight
▶ 1	com.mulesoft.training.Flight@1...	com.mulesoft.training.Flight
▶ 2	com.mulesoft.training.Flight@7...	com.mulesoft.training.Flight
▶ 3	com.mulesoft.training.Flight@df...	com.mulesoft.training.Flight
▼ 4	com.mulesoft.training.Flight@1...	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
origination	MUA	java.lang.String
planeType	Boeing 737	java.lang.String
price	456.0	java.lang.Double

Add a DataMapper

1. Locate getAmericanFlightsFlow in apessentials.xml.
2. Delete the Object to String transformer.
3. Replace it with a DataMapper transformer.

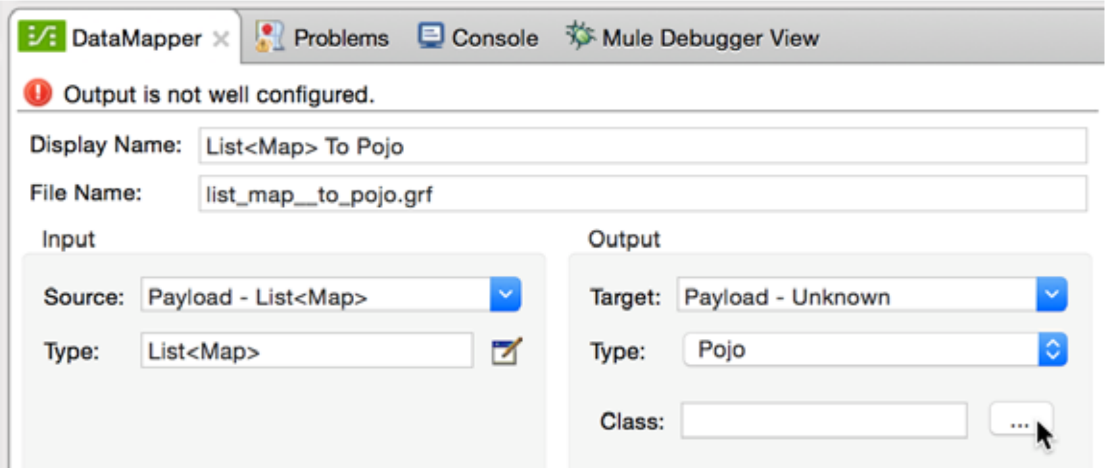


4. Look at the DataMapper Properties view; the input should be automatically populated from DataSense as a List<Map> because the Database connector configuration is DataSense enabled by default.

Configure the DataMapper output

5. In the Output section, set the type to Pojo.

6. Beneath the type, click the button next to Class.



Output is not well configured.

Display Name: List<Map> To Pojo

File Name: list_map__to_pojo.grf

Input

Source: Payload - List<Map>

Type: List<Map>

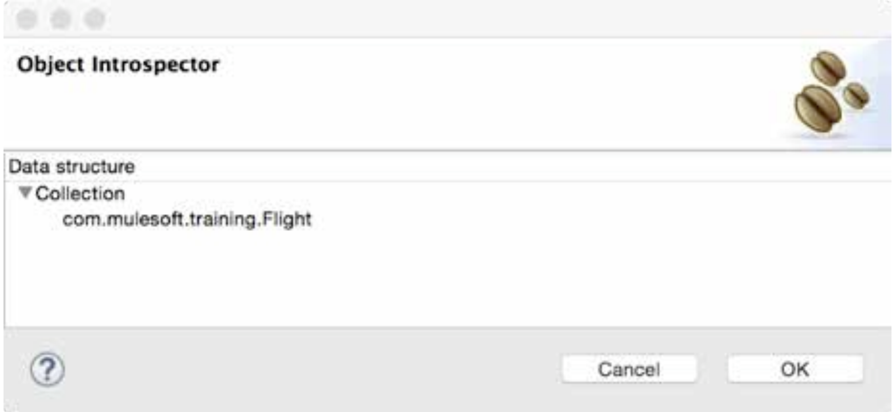
Output

Target: Payload - Unknown

Type: Pojo

Class:

7. In the Object Introspector window that opens, click the Click here to select an option link and select Collection.
8. Click the Click here to select an option link again and select Java Object.
9. In the Select entries window that opens, enter Fli and then press the Enter key to add an entry of com.mulesoft.training.Flight.

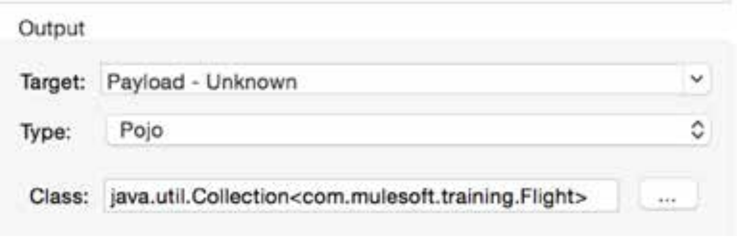


Object Introspector

Data structure

- Collection
  - com.mulesoft.training.Flight

10. Click OK.
11. Look at the output section; the class should now be set to a collection of Flight objects.



Output

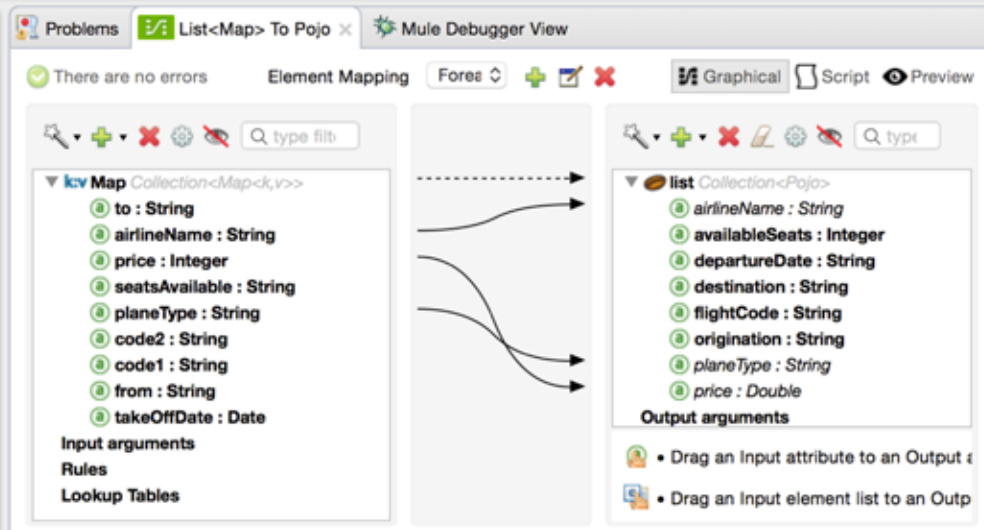
Target: Payload - Unknown

Type: Pojo

Class: java.util.Collection<com.mulesoft.training.Flight>

Create the mapping

12. Click the Create mapping button at the bottom of the Properties view; some of the fields in the objects had the same names so were automatically mapped.



There are no errors

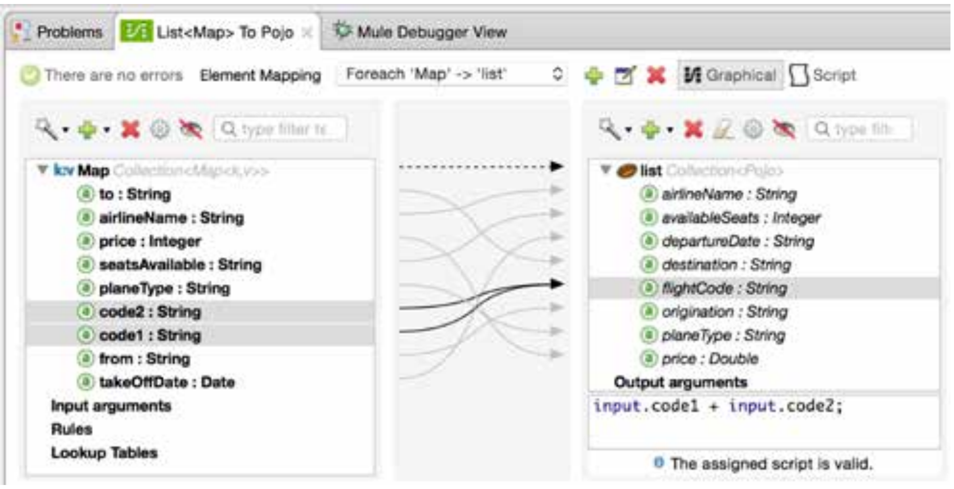
• Drag an Input attribute to an Output a

• Drag an Input element list to an Outp

13. Drag and drop fields from the left pane to the right pane to make the following mappings.

- toAirport > destination
- seatsAvailable > availableSeats
- fromAirport > origination
- takeOffDate > departureDate
- code1 > flightCode
- code2 > flightCode

14. Select flightCode in the right pane and see the concatenation expression below the right pane.



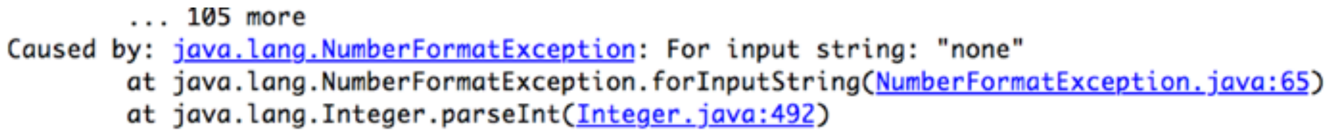
There are no errors

```
input.code1 + input.code2;
```

The assigned script is valid.

Debug the application

15. Save the file.
16. Add a breakpoint to the American DB Request endpoint and debug the application.
17. Make a request to http://localhost:8081/american.
18. Watch the payload value and step through the flow; you should get a NumberFormatException for the input string “none”.



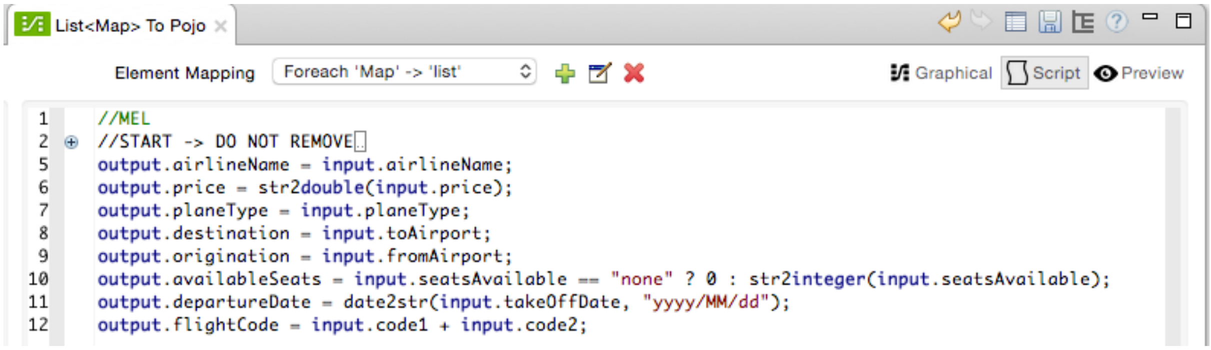
```
        ... 105 more
Caused by: java.lang.NumberFormatException: For input string: "none"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:492)
```

Modify the DataMapper

19. Return to the DataMapper Properties view.
20. Click the Script button in the upper-right corner of the Properties view.
21. In the Script window, modify the expression setting availableSeats to use a ternary expression to set it to 0 or to the input value.

```
output.availableSeats = input.seatsAvailable == "none" ? 0 :
str2integer(input.seatsAvailable);
```

Note: Be sure to end the line with a semi-colon.



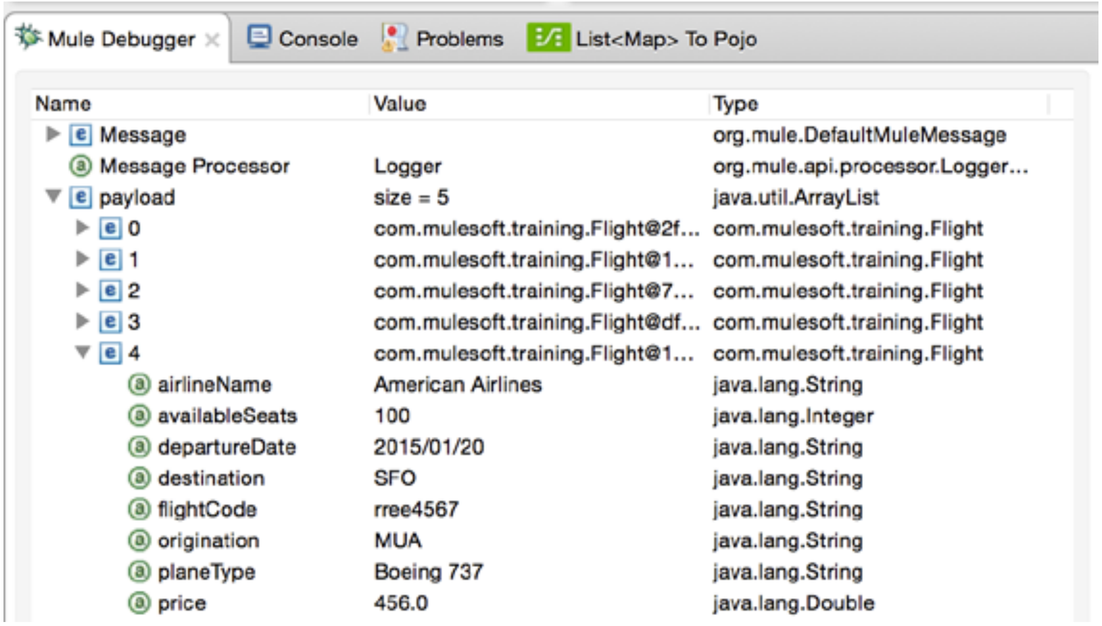
Element Mapping Foreach 'Map' -> 'list'

```
//MEL
//START -> DO NOT REMOVE
output.airlineName = input.airlineName;
output.price = str2double(input.price);
output.planeType = input.planeType;
output.destination = input.toAirport;
output.origination = input.fromAirport;
output.availableSeats = input.seatsAvailable == "none" ? 0 : str2integer(input.seatsAvailable);
output.departureDate = date2str(input.takeOffDate, "yyyy/MM/dd");
output.flightCode = input.code1 + input.code2;
```

Debug the application

22. Save the file to redeploy the application.
23. Make a request to http://localhost:8081/american.

24. Watch the payload value and step through the flow; you should now see the payload transformed into an ArrayList of Flight objects.



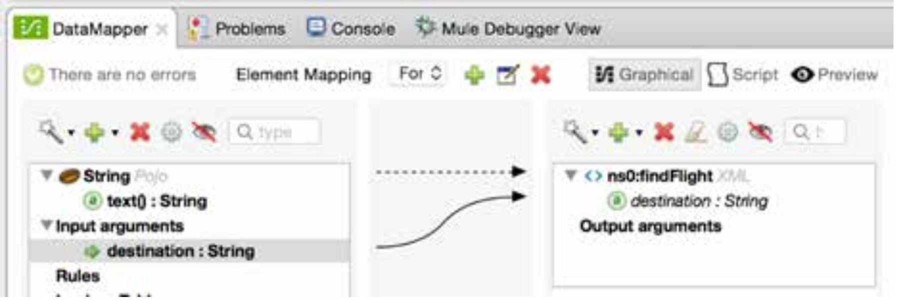
Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.Logger...
payload	size = 5	java.util.ArrayList
0	com.mulesoft.training.Flight@2f...	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@1...	com.mulesoft.training.Flight
2	com.mulesoft.training.Flight@7...	com.mulesoft.training.Flight
3	com.mulesoft.training.Flight@df...	com.mulesoft.training.Flight
4	com.mulesoft.training.Flight@1...	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
origination	MUA	java.lang.String
planeType	Boeing 737	java.lang.String
price	456.0	java.lang.Double

25. Stop the Mule Debugger.

Walkthrough 5-6: Use the DataMapper to pass arguments to a SOAP web service

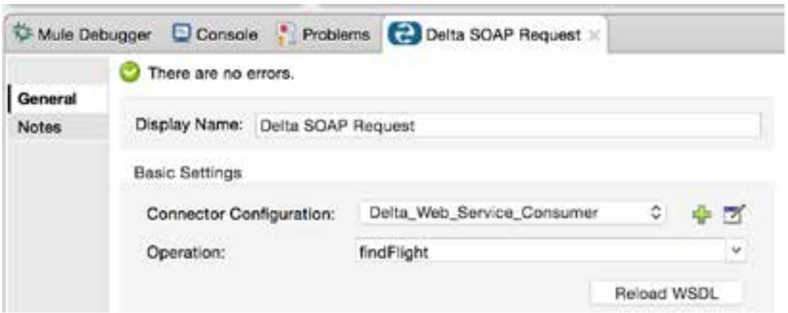
In this walkthrough, you will work with the Delta flight data. You will:

- Return the flights for a specific destination instead of all the flights.
- Change the web service operation invoked to one that requires a destination as an input argument.
- Use the DataMapper to pass an argument to a web service operation.
- Create a variable to set the destination to a dynamic query parameter value.



Call a different web service operation

1. Return to apessentials.com and locate getDeltaFlightsFlow.
2. In the Properties view for the Delta SOAP Request endpoint, change the operation to findFlight.

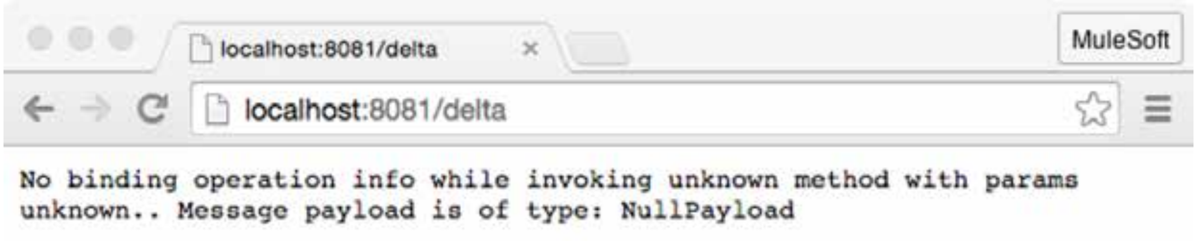


Display Name: Delta SOAP Request

Connector Configuration: Delta_Web_Service_Consumer

Operation: findFlight

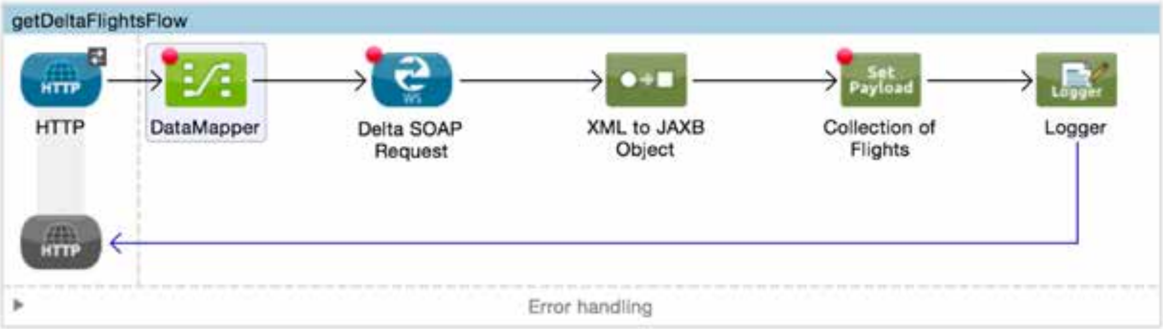
3. Apply the changes and run the application.
4. Make a request to http://localhost:8081/delta; you should get an error because you are calling an operation that requires parameters but you have not passed it any.



```
No binding operation info while invoking unknown method with params
unknown.. Message payload is of type: NullPayload
```

Add a DataMapper

5. Add a DataMapper transformer to the left of the Delta SOAP Request endpoint.



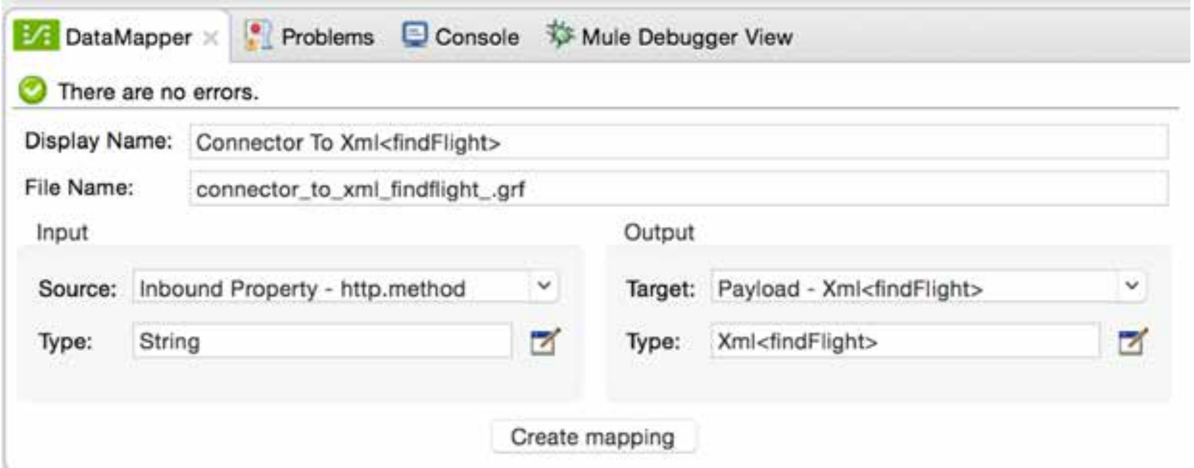
6. In the DataMapper Properties view, verify that the output target and type are set to Xml <findFlight>; the Web Service Consumer global element is set to use DataSense by default.

Use the DataMapper to pass an argument to the web service

7. Set the Input Source to any one of the inbound properties that is of type String, like http.method.

   Note: It does not matter which value you select; it will not be used, as you will see in the next steps.

8. Click the Create mapping button.



There are no errors.

Display Name: Connector To Xml<findFlight>

File Name: connector_to_xml_findflight_.grf

Input

Source: Inbound Property - http.method

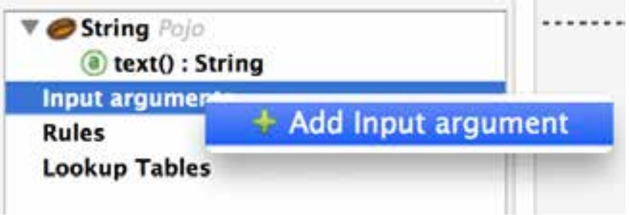
Type: String

Output

Target: Payload - Xml<findFlight>

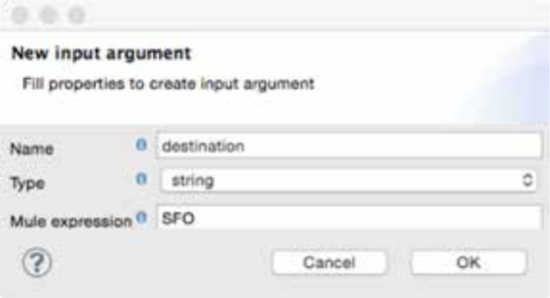
Type: Xml<findFlight>

9. In the graphical mapping editor, right-click Input arguments and select Add Input argument.



10. In the New input argument dialog box, set the following values and click OK.

- Name: destination
- Type: string
- Mule expression: SFO



New input argument

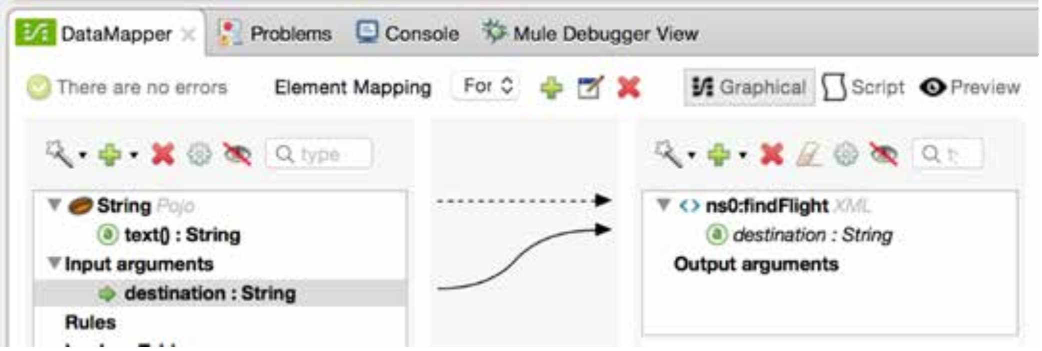
Fill properties to create input argument

Name: destination

Type: string

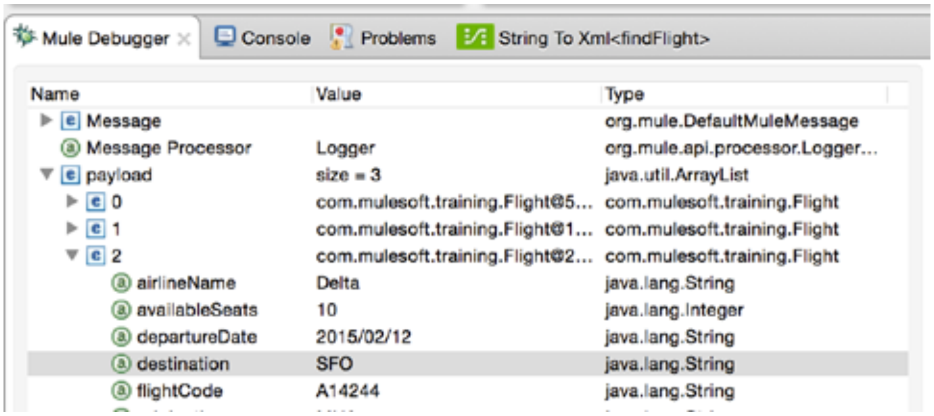
Mule expression: SFO

11. Drag and drop the destination input argument to the destination argument of findFlight.



Test the application

12. Save the file and debug the application.
13. Make a request to http://localhost:8081/delta and step to the Logger; you should now see only the SFO flights.

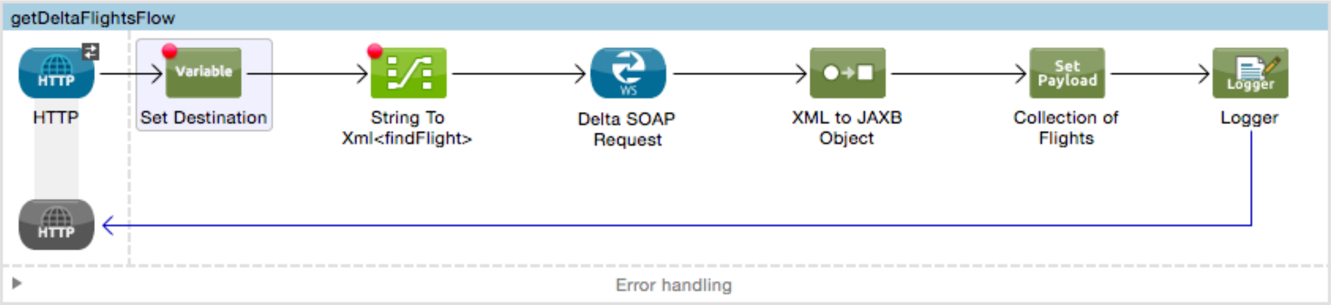


Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.Logger...
payload	size = 3	java.util.ArrayList
0	com.mulesoft.training.Flight@5...	com.mulesoft.training.Flight
1	com.mulesoft.training.Flight@1...	com.mulesoft.training.Flight
2	com.mulesoft.training.Flight@2...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	10	java.lang.Integer
departureDate	2015/02/12	java.lang.String
destination	SFO	java.lang.String
flightCode	A14244	java.lang.String

14. Step through the rest of the application.

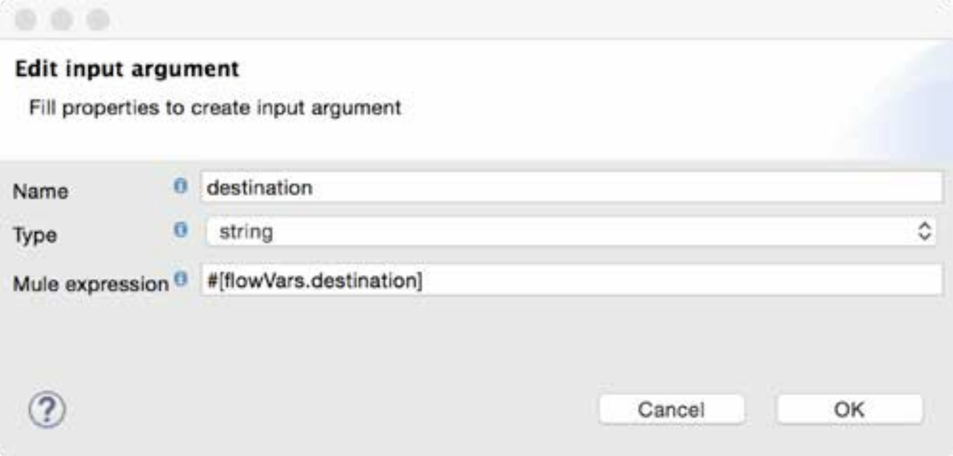
Set a flow variable to hold the value of a query parameter

15. Select the Set Destination transformer in getUnitedFlightsFlow and from the main menu, select Edit > Copy (or press Cmd+C/Ctrl+C).
16. Click in the process section of getDeltaFlightsFlow and from the main menu, select Edit > paste (or press Cmd+V/Ctrl+V); you should see a copy of the transformer added to the flow.
17. Move the transformer before the DataMapper.
18. In the Variable Properties view, change the display name to Set Destination and review the expression.



Modify the input argument to use a dynamic destination from a query parameter

19. In the Properties view for the DataMapper, double-click the Input Argument to modify it.
20. Change the value from SFO to use the flow variable, #[flowVars.destination].



Edit input argument

Fill properties to create input argument

Name: destination

Type: string

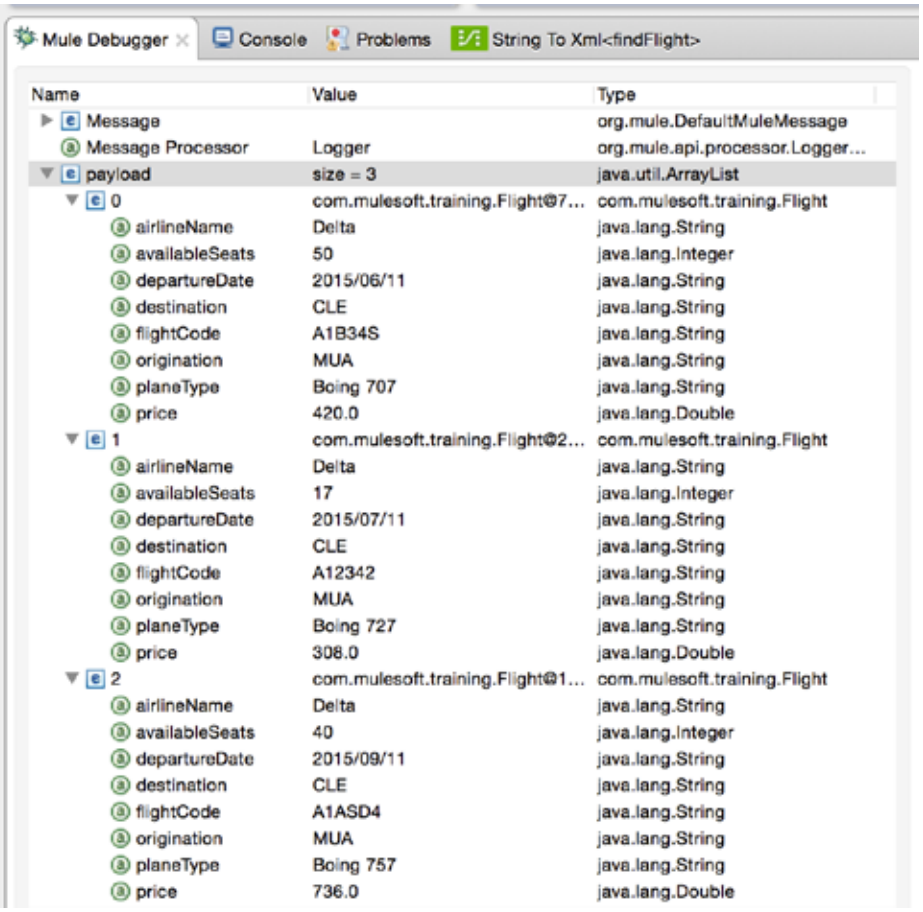
Mule expression: #[flowVars.destination]

21. Click OK.

Test the application

22. Save the file and redeploy the application.
23. Make a request to http://localhost:8081/delta and step to the Logger; you should only see flights to SFO.

24. Click Resume.
25. Make another request to http://localhost:8081/delta?code=CLE and step to the Logger; you should now see flights to CLE.
26. Look at the price of each flight; the flights are not ordered by price.



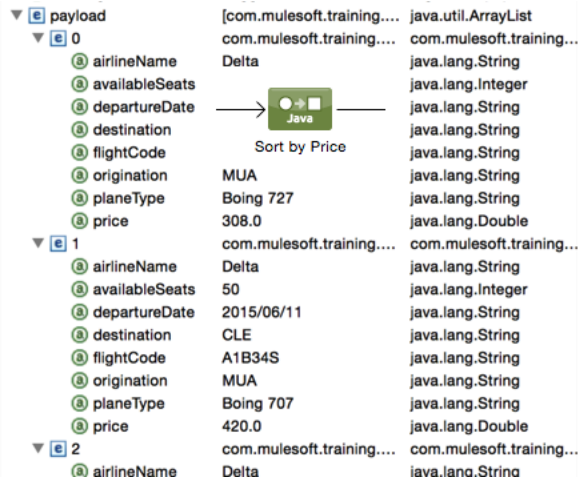
Name	Value	Type
▶ Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.Logger...
▼ payload	size = 3	java.util.ArrayList
▼ 0	com.mulesoft.training.Flight@7...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	50	java.lang.Integer
departureDate	2015/06/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A1B34S	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 707	java.lang.String
price	420.0	java.lang.Double
▼ 1	com.mulesoft.training.Flight@2...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	17	java.lang.Integer
departureDate	2015/07/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A12342	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 727	java.lang.String
price	308.0	java.lang.Double
▼ 2	com.mulesoft.training.Flight@1...	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	40	java.lang.Integer
departureDate	2015/09/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A1ASD4	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 757	java.lang.String
price	736.0	java.lang.Double

27. Click the Resume button.

Walkthrough 5-7: Use a custom Java transformer

In this walkthrough, you will use a custom transformer to sort the airline flight results from lowest to highest price. You will:

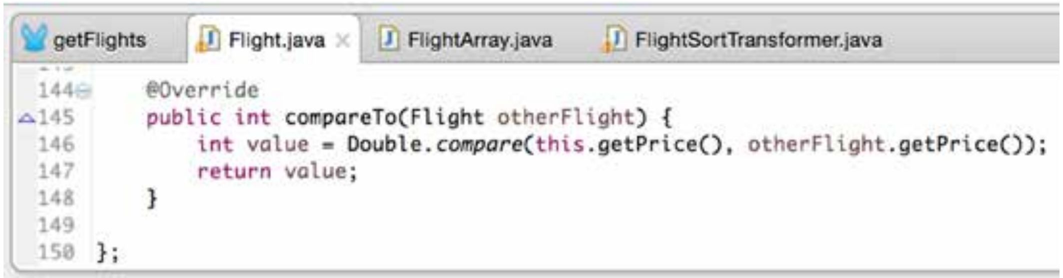
- Review a pre-written Java class to be used as the transformer class.
- Use a custom Java transformer.
- Sort the United flight data by price.



```
▼ payload                  [com.mulesoft.training....  java.util.ArrayList
  ▼ 0                      com.mulesoft.training....   com.mulesoft.training...
      airlineName          Delta                       java.lang.String
      availableSeats                                   java.lang.Integer
      departureDate                                    java.lang.String
      destination                                      java.lang.String
      flightCode                                       java.lang.String
      origination          MUA                         java.lang.String
      planeType            Boing 727                   java.lang.String
      price                308.0                       java.lang.Double
  ▼ 1                      com.mulesoft.training....   com.mulesoft.training...
      airlineName          Delta                       java.lang.String
      availableSeats       50                          java.lang.Integer
      departureDate        2015/06/11                  java.lang.String
      destination          CLE                         java.lang.String
      flightCode           A1B34S                      java.lang.String
      origination          MUA                         java.lang.String
      planeType            Boing 707                   java.lang.String
      price                420.0                       java.lang.Double
  ▼ 2                      com.mulesoft.training....   com.mulesoft.training...
      airlineName          Delta                       java.lang.String
```

Review the Java class

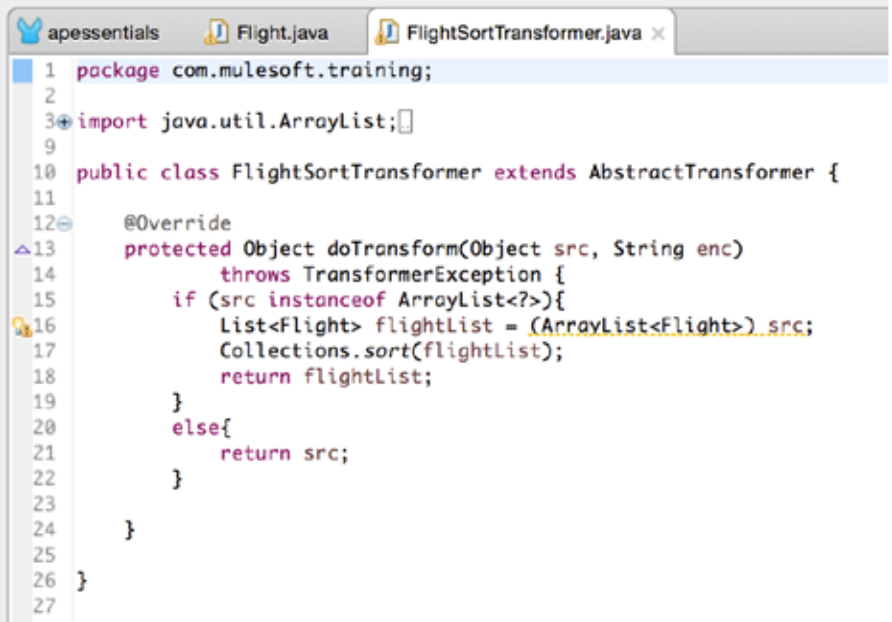
1. Open Flight.java.
2. Review the code; notice that it implements Comparable and has a compareTo() function.



```
    @Override
    public int compareTo(Flight otherFlight) {
        int value = Double.compare(this.getPrice(), otherFlight.getPrice());
        return value;
    }

};
```

3. Open the FlightSortTransformer class, which is also located in the com.mulesoft.training package of your project.

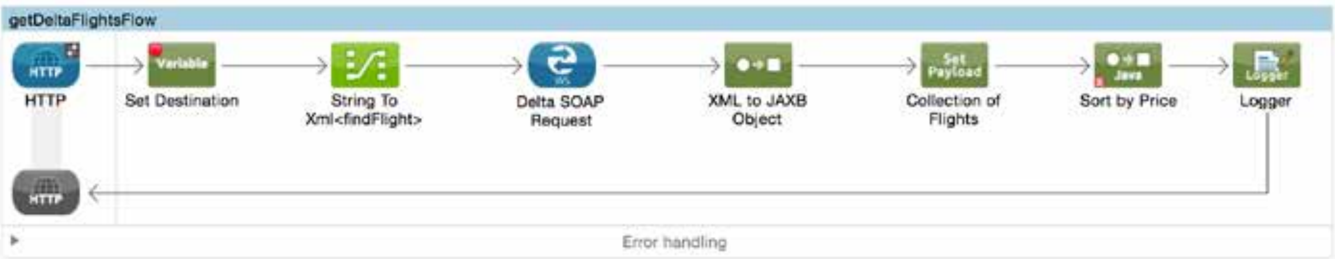
4. Review the code; because the class only needs to operate on the payload (to sort it) and not the rest of the message, it extends the org.mule.transformer.AbstractTransformer class.



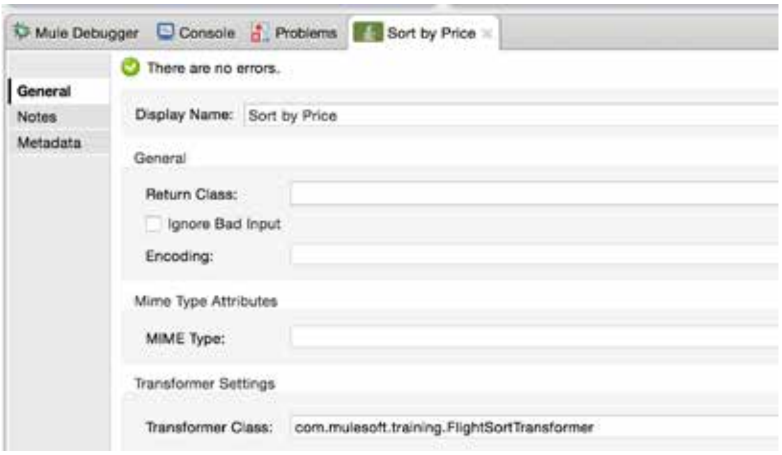
```
package com.mulesoft.training;

import java.util.ArrayList;

public class FlightSortTransformer extends AbstractTransformer {

    @Override
    protected Object doTransform(Object src, String enc)
            throws TransformerException {
        if (src instanceof ArrayList<?>){
            List<Flight> flightList = (ArrayList<Flight>) src;
            Collections.sort(flightList);
            return flightList;
        }
        else{
            return src;
        }

    }

}
```

Add a Java transformer

5. Navigate to getDeltaFlightsFlow in apessentials.xml.
6. Add a Java transformer after the Collection of Flights transformer.
7. In the Java Properties view, set the display name to Sort by Price.



8. Set the transformer class to com.mulesoft.training.FlightSortTransformer.



There are no errors.

Display Name: Sort by Price

General

Return Class:

□ Ignore Bad Input

Encoding:

Mime Type Attributes

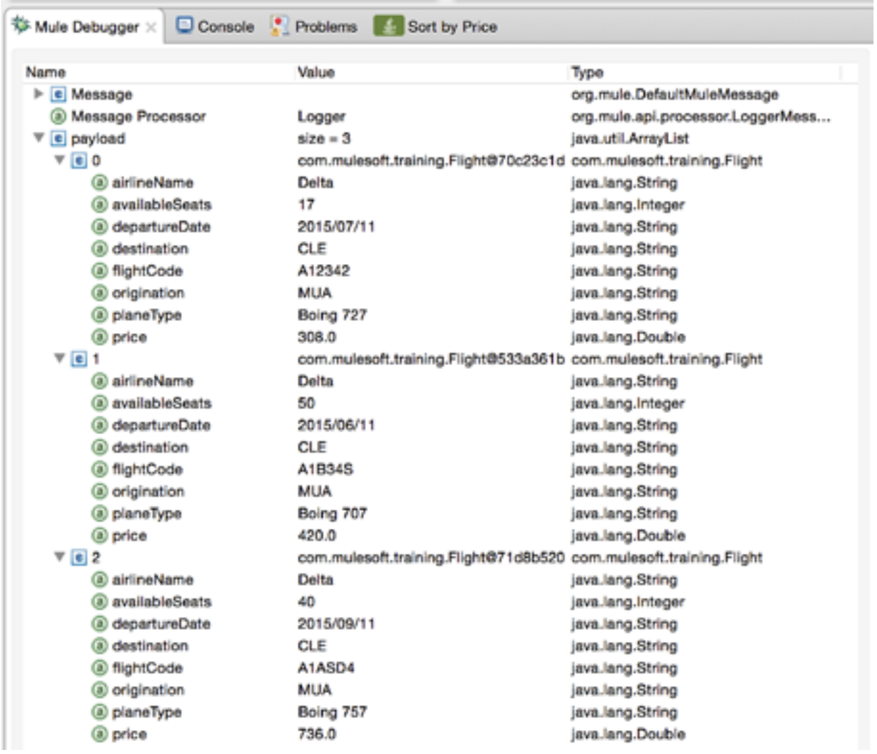
MIME Type:

Transformer Settings

Transformer Class: com.mulesoft.training.FlightSortTransformer

Debug the application

9. Save the file to redeploy the application in debug mode.
10. Make a request to http://localhost:8081/delta?code=CLE.
11. Step through to the Logger; you should see the collection has been reordered by price.



Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMess...
payload	size = 3	java.util.ArrayList
0	com.mulesoft.training.Flight@70c23c1d	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	17	java.lang.Integer
departureDate	2015/07/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A12342	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 727	java.lang.String
price	308.0	java.lang.Double
1	com.mulesoft.training.Flight@533a361b	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	50	java.lang.Integer
departureDate	2015/06/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A1B34S	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 707	java.lang.String
price	420.0	java.lang.Double
2	com.mulesoft.training.Flight@71d8b520	com.mulesoft.training.Flight
airlineName	Delta	java.lang.String
availableSeats	40	java.lang.Integer
departureDate	2015/09/11	java.lang.String
destination	CLE	java.lang.String
flightCode	A1ASD4	java.lang.String
origination	MUA	java.lang.String
planeType	Boing 757	java.lang.String
price	736.0	java.lang.Double

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.

12. Stop the Mule Debugger.