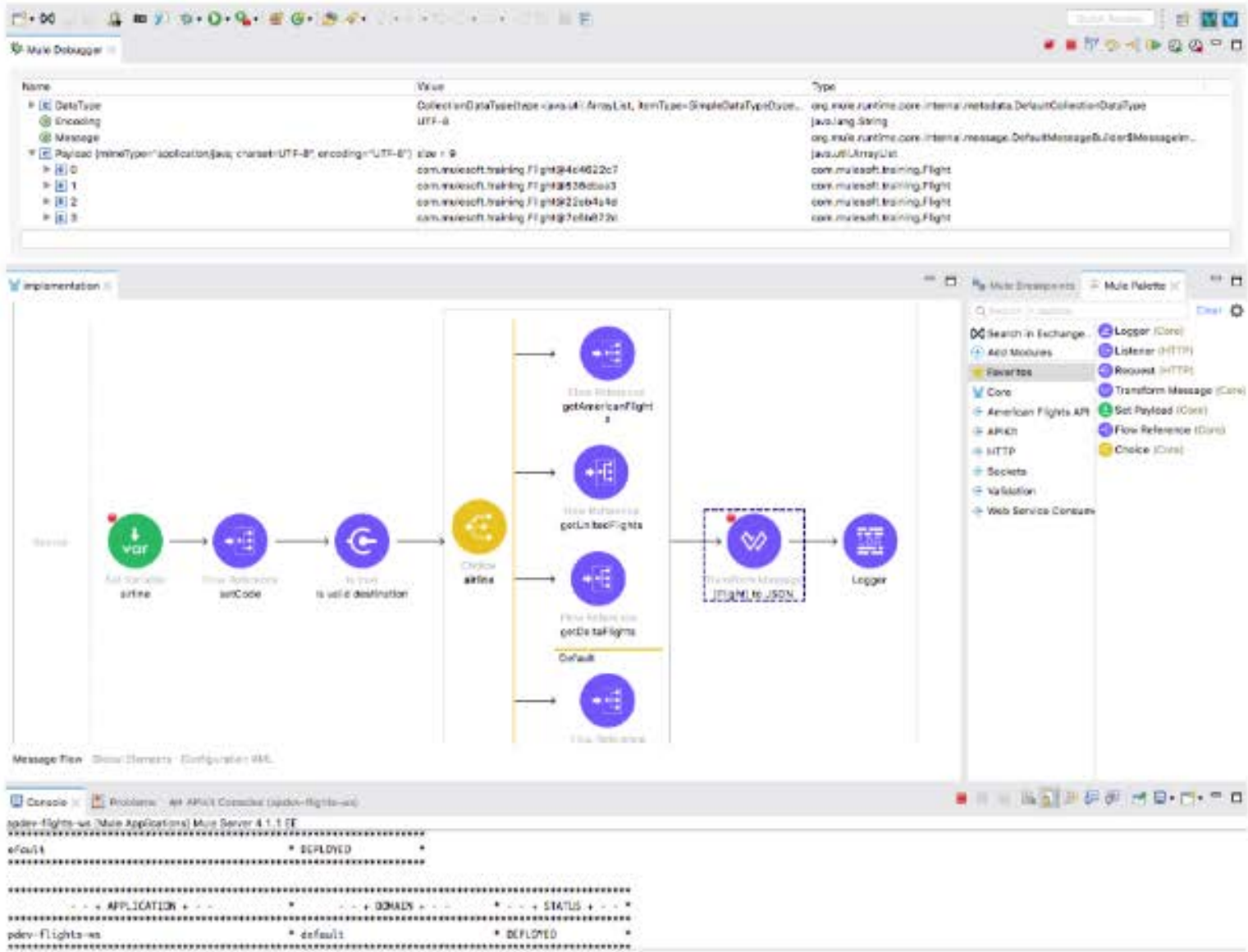
PART 2: Building Applications with Anypoint Studio

Goal

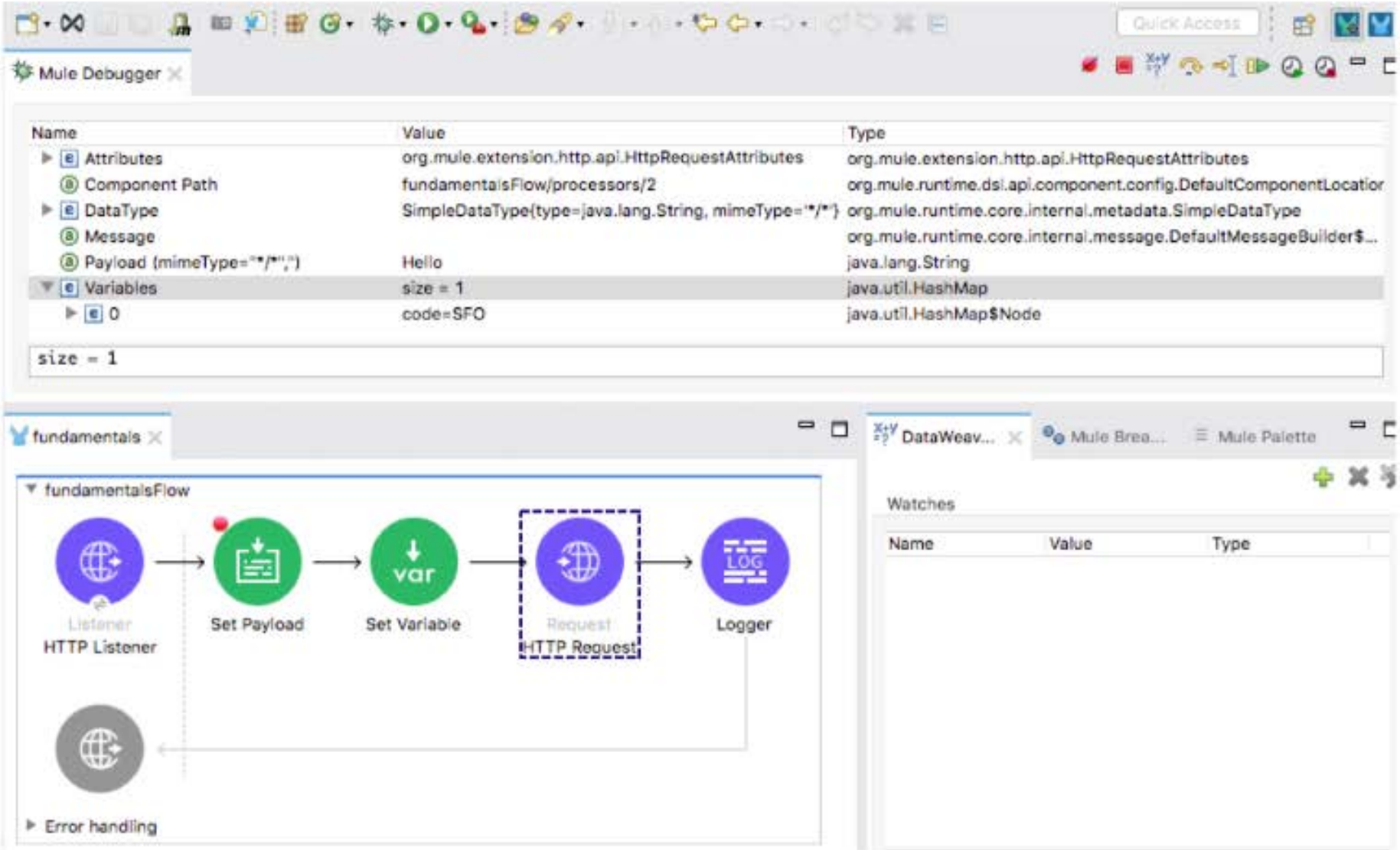


At the end of this part, you should be able to

- Debug Mule applications
- Read and write event payloads, attributes, and variables using the DataWeave Expression Language
- Structure Mule applications using flows, subflows, asynchronous queues, properties files, and configuration files
- Call RESTful and SOAP web services
- Route and validate events and handle messaging errors
- Write DataWeave scripts for transformations


MuleSoft
Module 6: Accessing and Modifying Mule Events

Goal

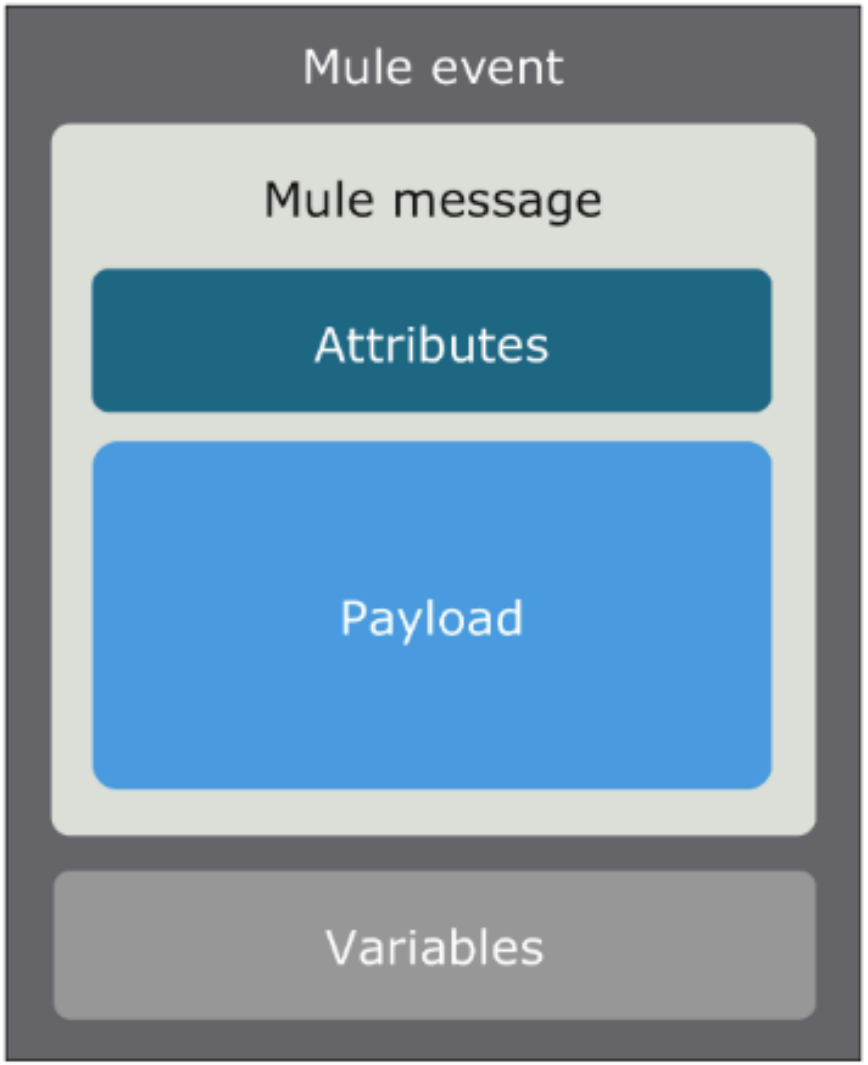


At the end of this module, you should be able to

- Log event data
- Debug Mule applications
- Read and write event properties
- Write expressions with the DataWeave expression language
- Create variables

Viewing information about Mule 4 events

Reviewing the structure of Mule 4 events



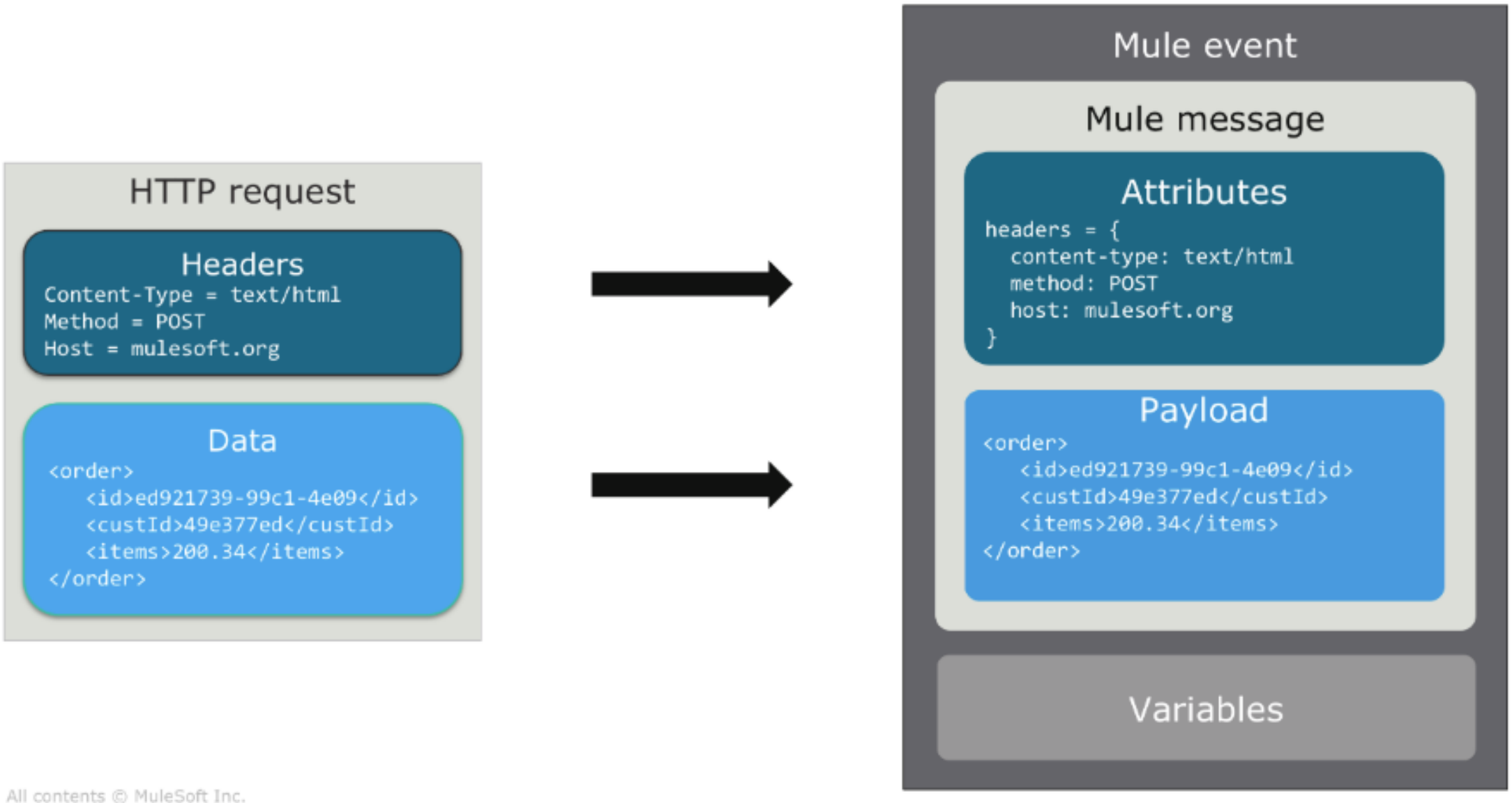
The data that passes through flows in the app

Metadata contained in the message header

The core info of the message - the data the app processes

Metadata for the Mule event - can be defined and referenced in the app processing the event

Event data populated from an HTTP request



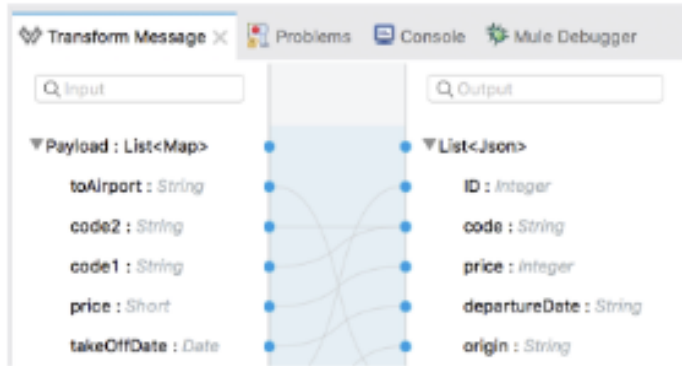
Viewing event data

- At **design time** in Anypoint Studio
  - What did we see so far in the course and where?

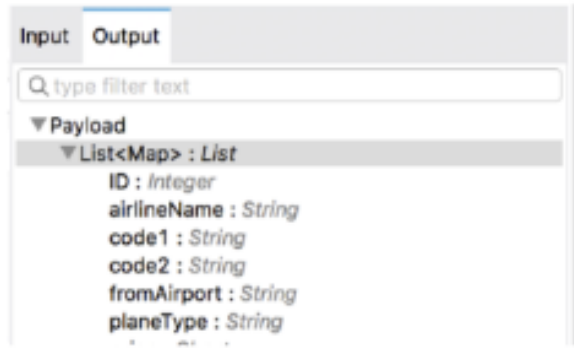
- At **run time**
  - What did we see so far in the course and where?

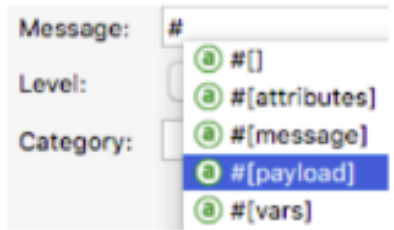
Viewing event data at design time

- At **design time** in Anypoint Studio using DataSense
  - In the Transform Message component
  - In the DataSense Explorer
  - When writing expressions using auto-completion



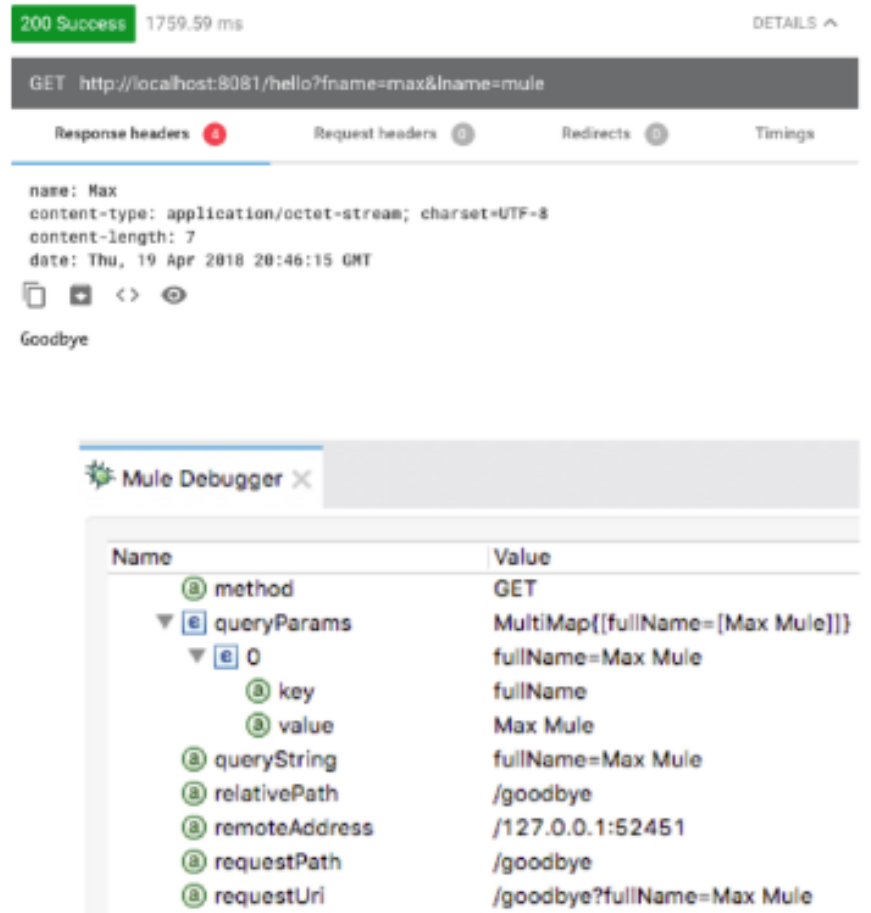
- **DataSense** is the ability to proactively discover metadata from internal and external resources
  - Keeps you from having to manually discover info about the data
  - Facilitates transformations by providing DataWeave expected input or output





Viewing event data at run time

- At **run time**
  - In the client when making a request
  - For deployed applications, in the log files
  - In the Anypoint Studio console by using a Logger
  - In Anypoint Studio using the Visual Debugger
    - Most comprehensive way



View event data by logging it

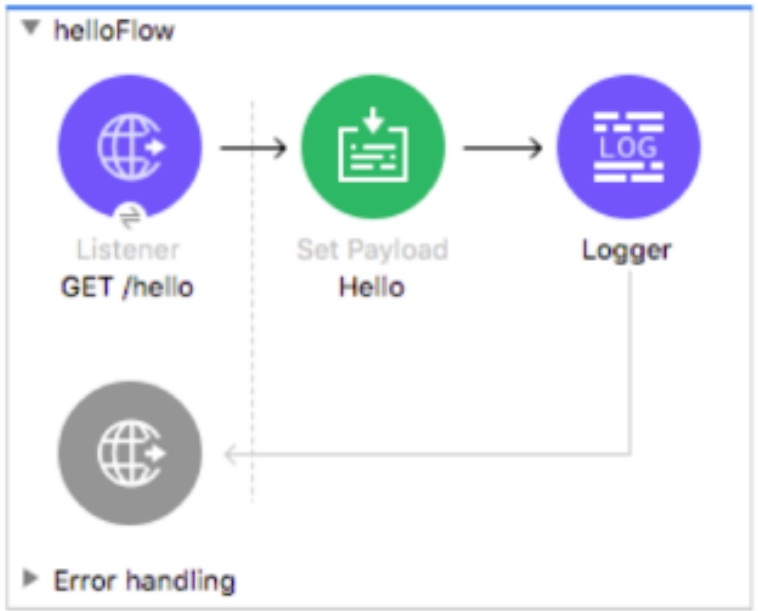
- Add a **Logger** component to a flow and view its output

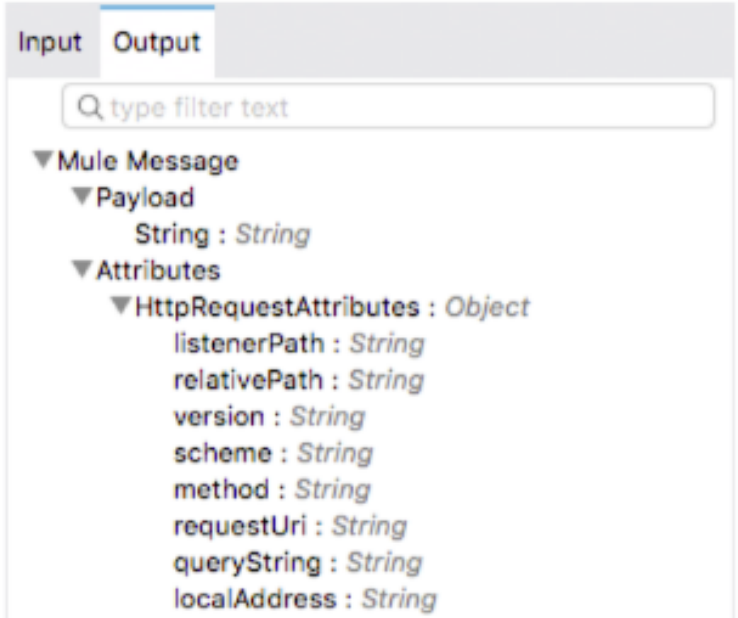
Logger

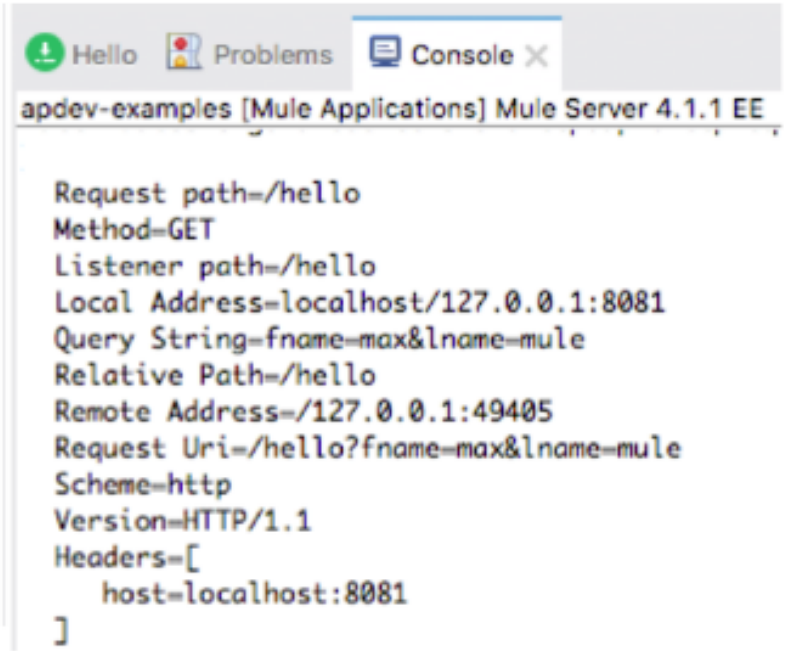
- Logged valued are displayed
  - For an application run from Anypoint Studio with embedded runtime, in the **Console view**
  - For applications deployed to CloudHub or customer-hosted runtimes, in the **log files**

Walkthrough 6-1: View event data

- Create a new Mule project with an HTTP Listener and set the payload
- View event data in the DataSense Explorer
- Use a Logger to view event data in the Anypoint Studio console



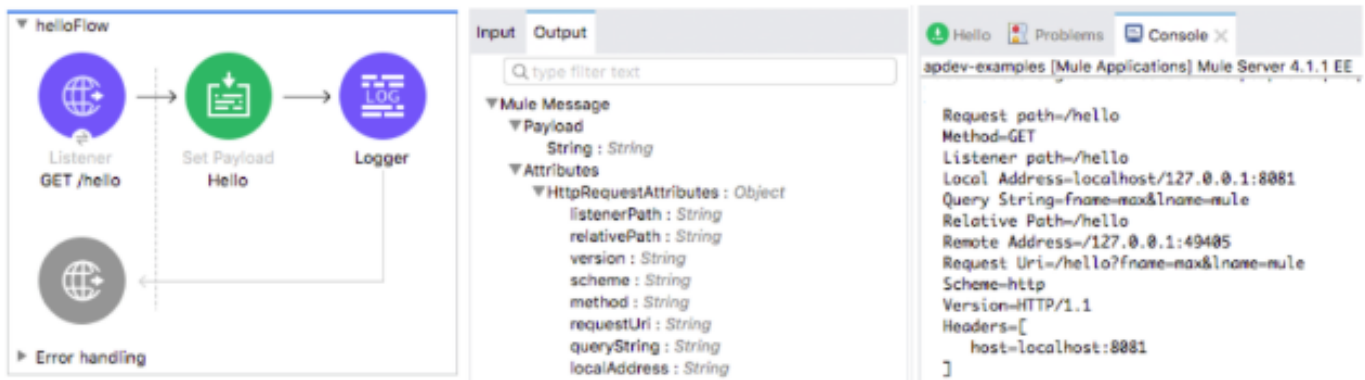




Walkthrough 6-1: View event data

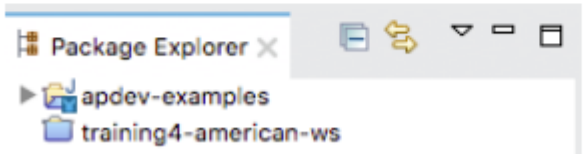
In this walkthrough, you create a new project to use in the next two modules to learn about Mule events and Mule applications. You will:

- Create a new Mule project with an HTTP Listener and set the message payload.
- View event data in the DataSense Explorer.
- Use a Logger to view event data in the Anypoint Studio console.



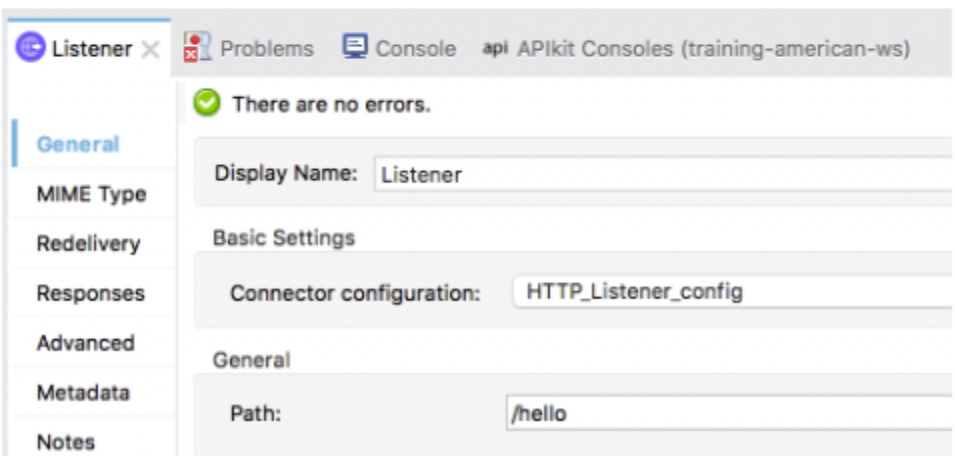
Create a new Mule project

1. Return to Anypoint Studio.
2. Right-click training4-american-ws and select Close Project.
3. Select File > New > Mule Project.
4. Set the project name to apdev-examples and click Finish.

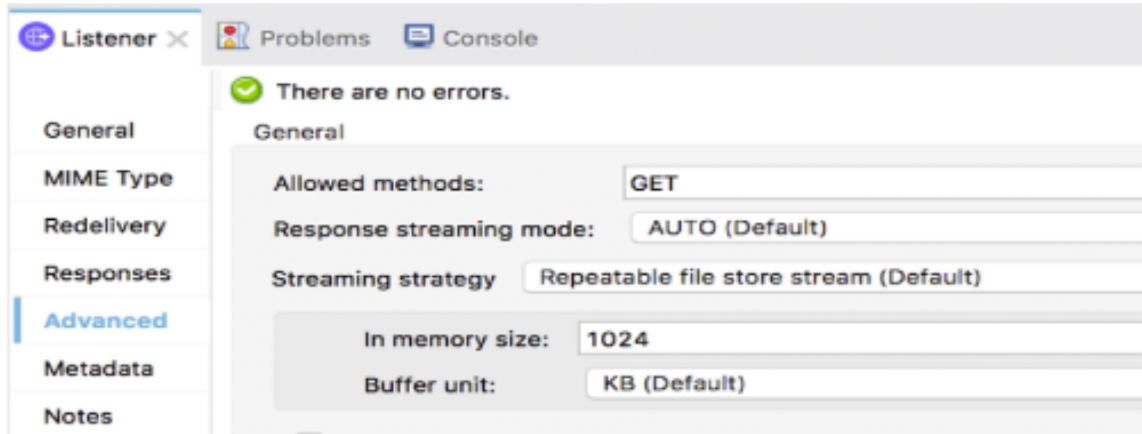


Create an HTTP Listener to receive requests

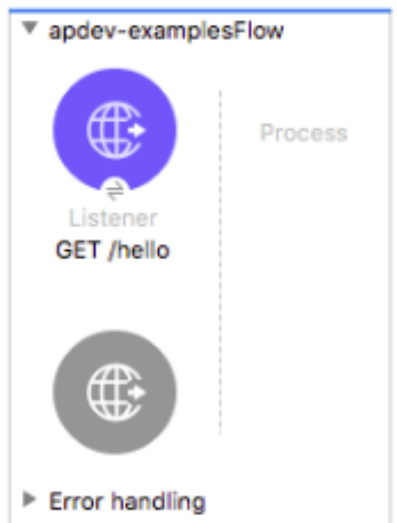
5. In the Mule Palette, select Favorites.
6. Drag an HTTP Listener from the Mule Palette to the canvas.
7. In the Listener properties view, click the Add button next to Connector configuration.
8. In the Global Element Properties dialog box, verify the host value is set to 0.0.0.0 and the port value to 8081.
9. Click OK.
10. In the Listener properties view, set the path to /hello.



11. Click the Advanced tab and set the allowed methods to GET.



12. Click the General tab and set the display name to GET /hello.

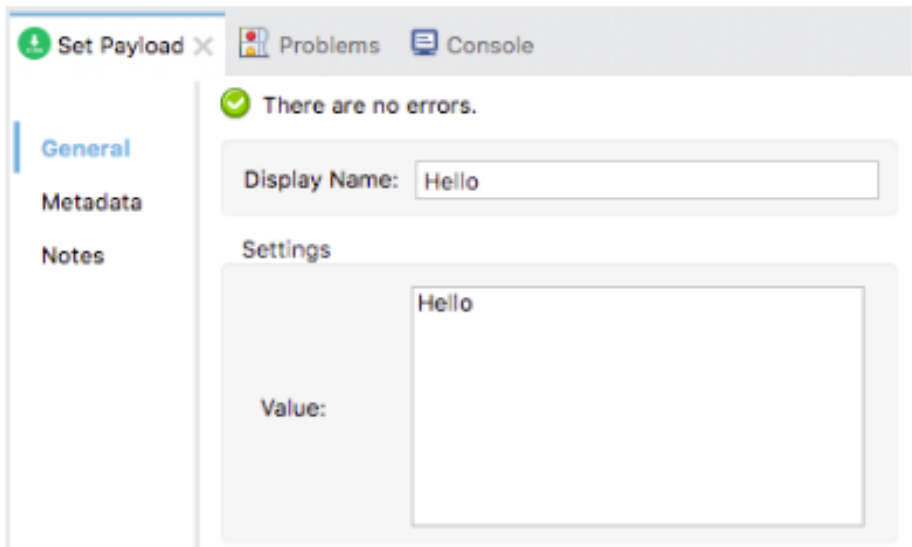


Change the flow name

13. Select the flow.
14. In the apdev-examplesFlow properties view, change the name to helloFlow.

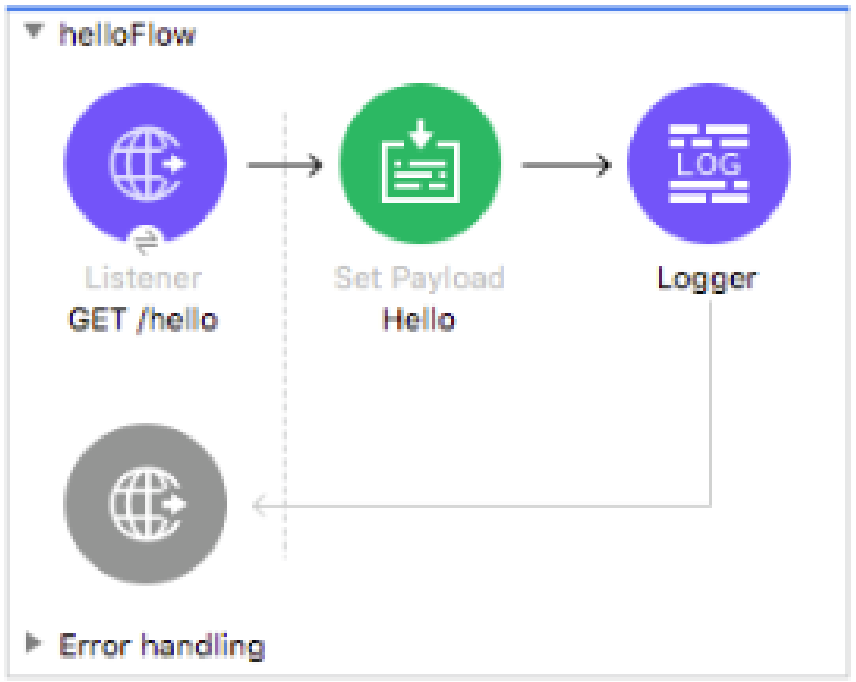
Set the message payload

15. Drag a Set Payload transformer from the Favorites section of the Mule Palette into the process section of the flow.
16. In the Set Payload properties view, set the display name to Hello.
17. Set the value to Hello.



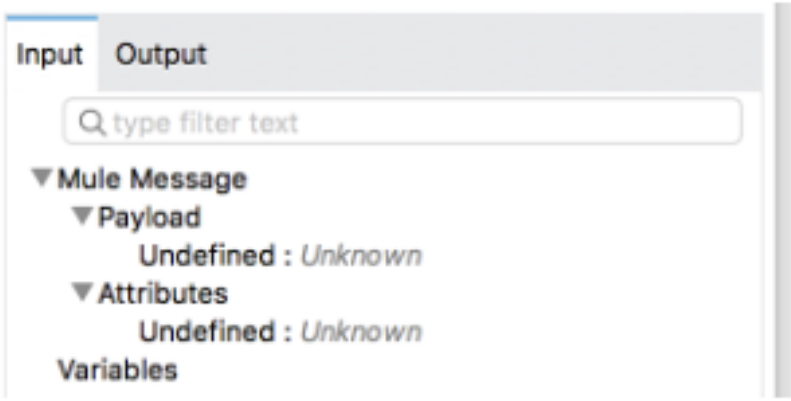
Add a Logger

18. Drag a Logger component from the Mule Palette and drop it at the end of the flow.

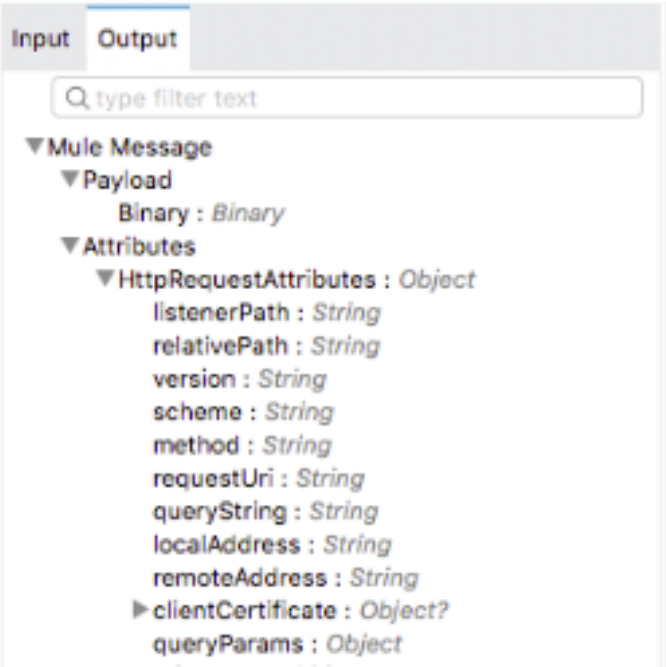


View event structure and metadata in the DataSense Explorer

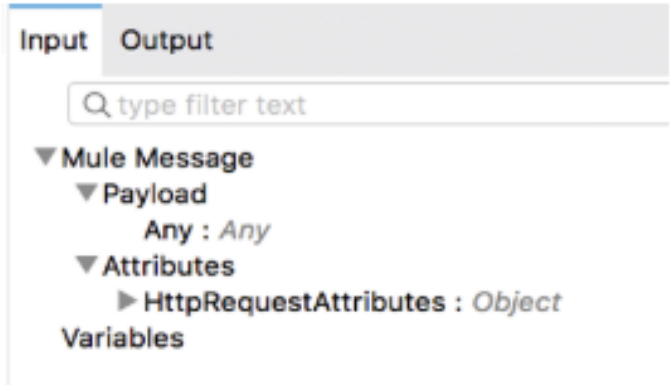
19. Select the GET /hello HTTP Listener and locate the DataSense Explorer in the right-side of its properties view.
20. Select the Input tab and expand Payload and Attributes.



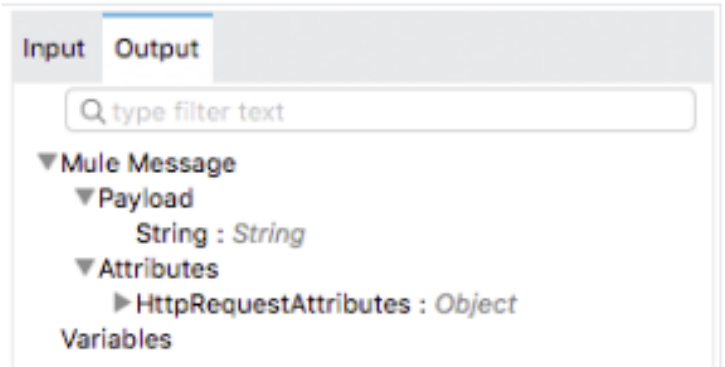
21. Select the Output tab and expand Payload and Attributes.



23. In the DataSense Explorer, select the Input tab and expand Payload and Attributes.



24. Select the Output tab and expand Payload and Attributes.



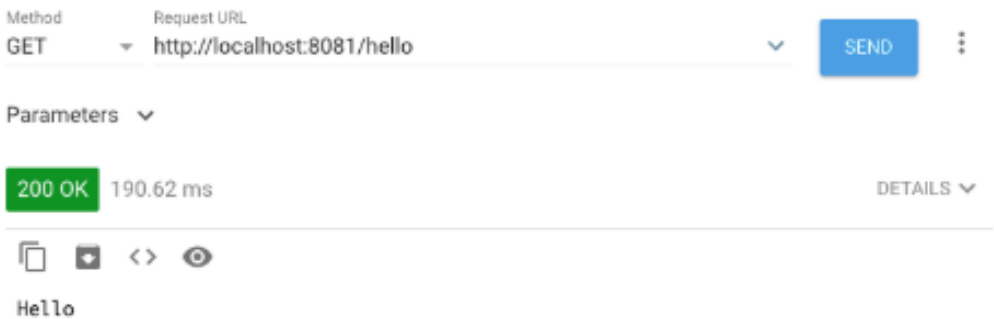
25. Select the Logger component in helloFlow.
26. In the DataSense Explorer, select the Input tab and expand Payload and Attributes.
27. Select the Output tab and expand the Payload and Attributes.

Run the application and review response data

28. Save the file and run the project.
29. Return to Advanced REST Client and click the button to create a new tab.

    Note: You are adding a new tab so that you can keep the request to your American API saved in another tab for later use.

30. In the new tab, make a GET request to http://localhost:8081/hello; you should see Hello displayed.



View event data in the Anypoint Studio console

31. Return to Anypoint Studio and look at the console.
32. Locate the data displayed by using the Logger.
33. Find where the data type of the payload is specified.
34. Review the event attributes.

Send query parameters with a request

35. Return to Advanced REST Client and add a query parameter with a key of fname and a value of max.
36. Add a second key/value pair of lname and mule.
37. Click Send.

38. Return to Anypoint Studio and look at the console.
39. Locate the query parameters in the logged event data.

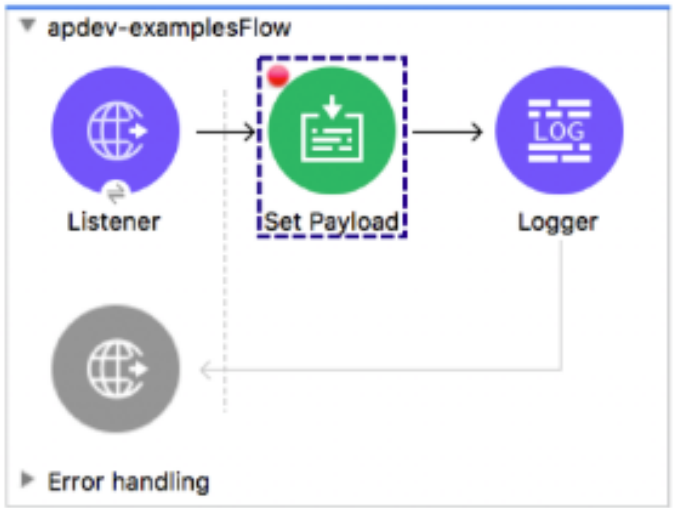
```
{
  payload=java.lang.String
  mediaType=*/*
  attributes=org.mule.extension.http.api.Htt
{
   Request path=/hello
   Method=GET
   Listener path=/hello
   Local Address=localhost/127.0.0.1:8081
   Query String=fname=max&lname=mule
   Relative Path=/hello
   Remote Address=/127.0.0.1:49405
   Request Uri=/hello?fname=max&lname=mule
   Scheme=http
   Version=HTTP/1.1
   Headers=[
      host=localhost:8081
   ]
   Query Parameters=[
      fname=max
      lname=mule
   ]
   URI Parameters=[]
}
  attributesMediaType=*/*
  exceptionPayload=<not set>
}
```

40. Stop the project.

Debugging Mule applications

Debugging applications with the Mule Debugger

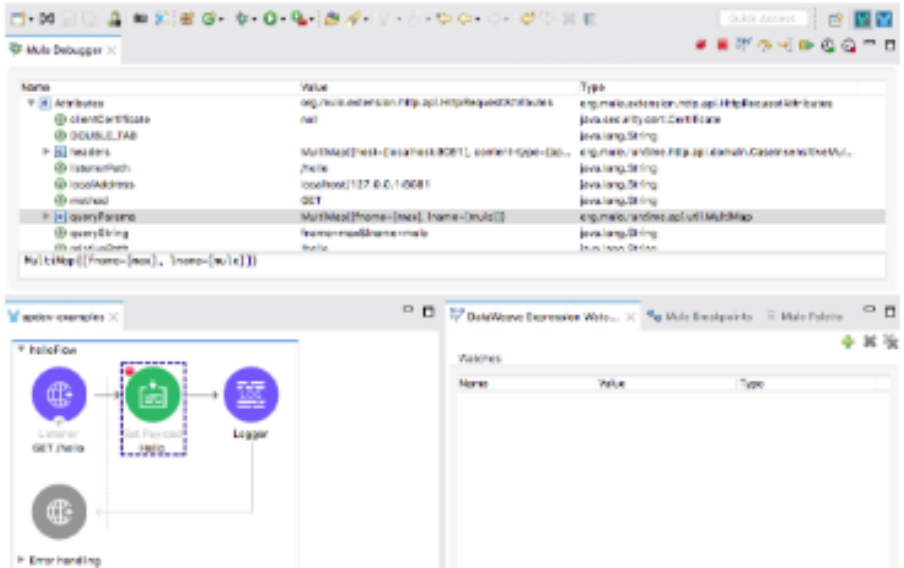
- Can add breakpoints to processors and step through the application
  - Watch event properties and values
  - Watch and evaluate DataWeave expressions



- By default, Debugger listens for incoming TCP connections on localhost port 6666
  - Can change this in a project's run configuration

Walkthrough 6-2: Debug a Mule application

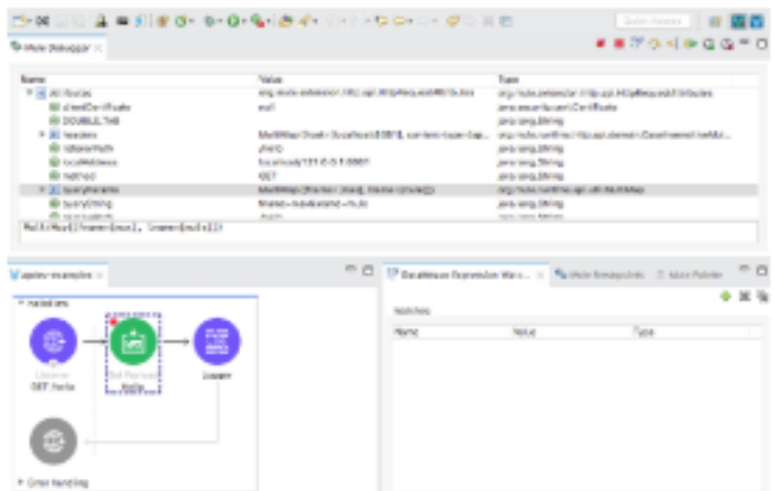
- Locate the port used by the Mule Debugger
- Add a breakpoint, debug an application, and step through the code
- Use the Mule Debugger to view event properties
- Pass query parameters to a request and locate them in the Debugger
- Increase the request timeout for Advanced REST Client



Walkthrough 6-2: Debug a Mule application

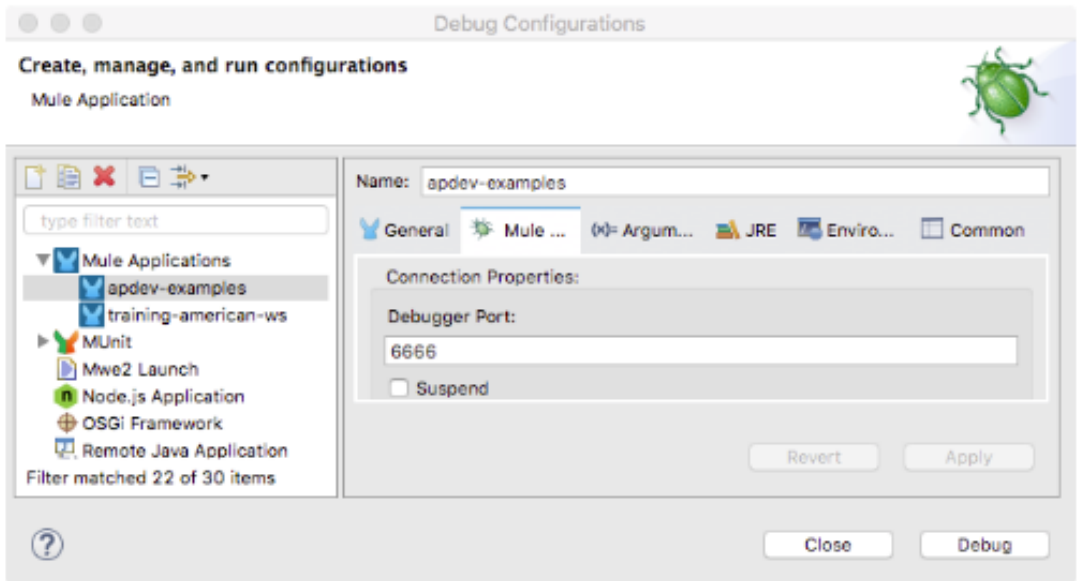
In this walkthrough, you debug and step through the code in a Mule application. You will:

- Locate the port used by the Mule Debugger.
- Add a breakpoint, debug an application, and step through the code.
- Use the Mule Debugger to view event properties.
- Pass query parameters to a request and locate them in the Mule Debugger.
- Increase the request timeout for Advanced REST Client.



Locate the port used by the Mule Debugger

1. Return to Anypoint Studio.
2. In the main menu bar, select Run > Debug Configurations.
3. Select apdev-examples in the left menu bar under Mule Applications; you should see that the apdev-examples project is selected to launch.
4. Select the Mule Debug tab; you should see the debugger port is set to 6666 for the project.

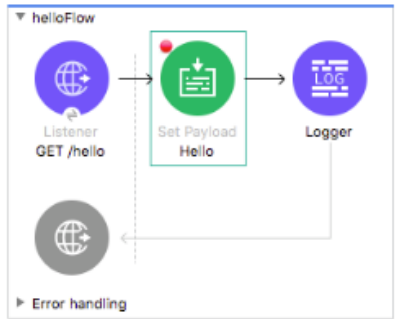


Note: If you know you have another program using port 6666 like McAfee on Windows, change this to a different value. Otherwise, you can test the Debugger first and come back and change the value here later if there is a conflict.

5. Click Close.

Add a breakpoint

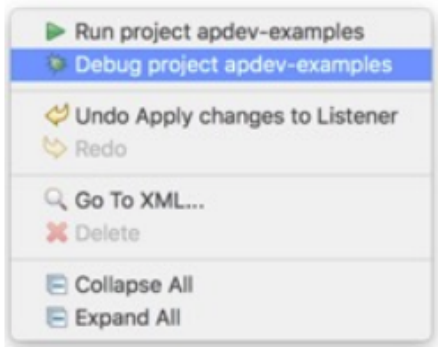
6. Right-click the Set Payload component and select Toggle breakpoint.



Debug the application

7. Right-click in the canvas and select Debug project apdev-examples.

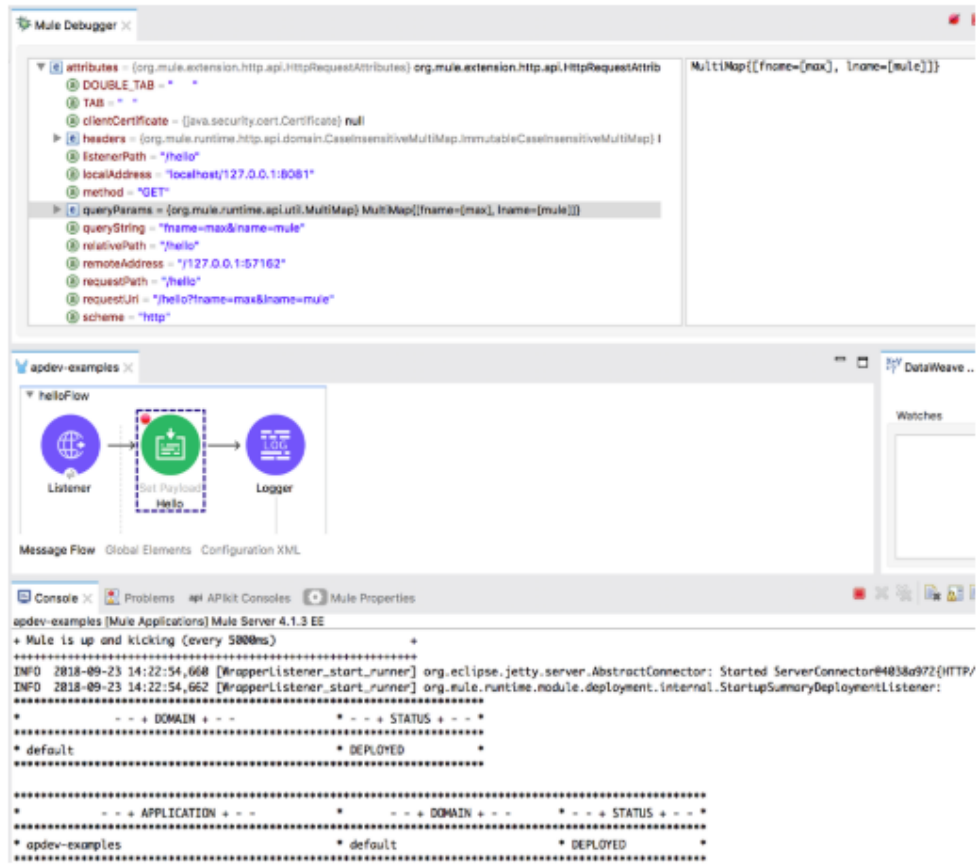
   Note: You can also select Run > Debug or click the Debug button in the main menu bar.



8. If you get a Confirm Perspective Switch dialog box, select Remember my decision and click Yes.
9. In the Debug perspective, close the MUnit view.
10. Look at the console and wait until the application starts.
11. In Advanced REST Client, make another request to http://localhost:8081/hello?fname=max&lname=mule.

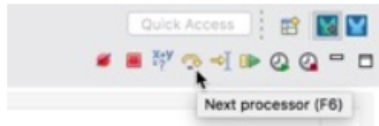
View event data in the Mule Debugger

12. Return to Anypoint Studio and locate the Mule Debugger view.
13. Look at the value of the payload.
14. Expand Attributes and review the values.
15. Locate the queryParams object.

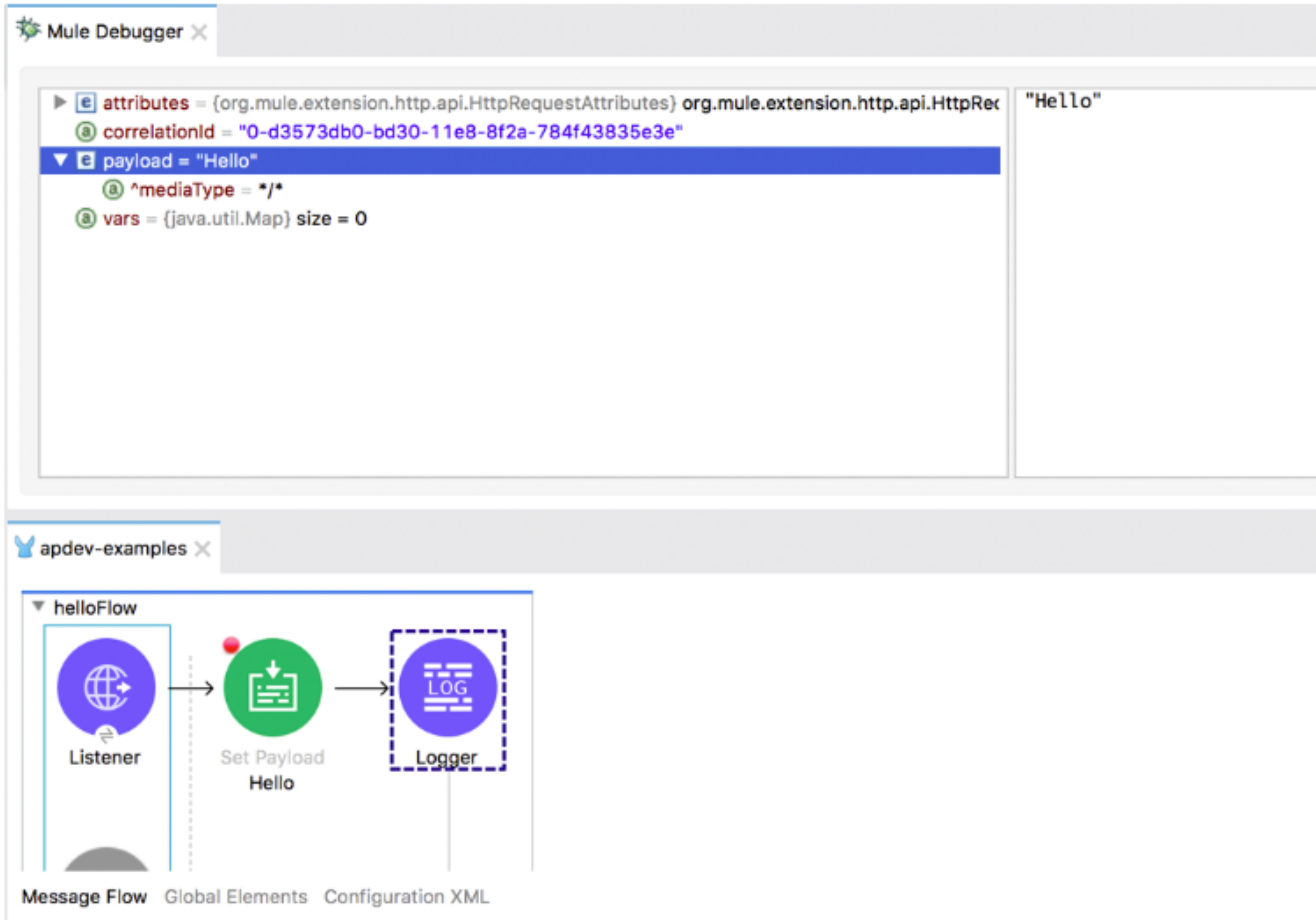


Step through the application

16. Click the Next processor button.



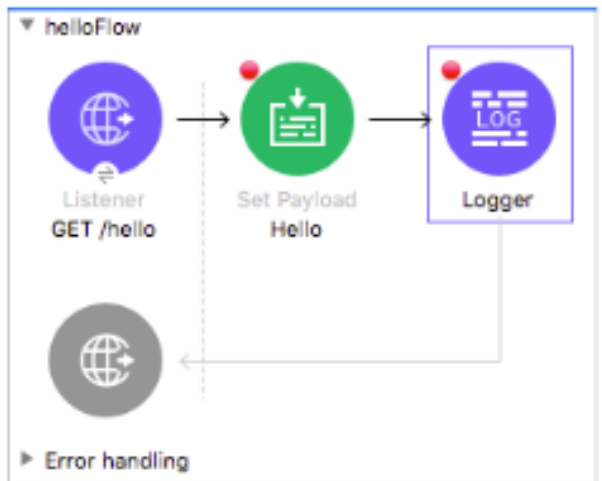
17. Look at the new value of the payload; it should be Hello.



18. Expand Attributes and review the values.
19. Click the Next processor button again to finish stepping through the application.

Use Resume to step to the next breakpoint and then the end of the application

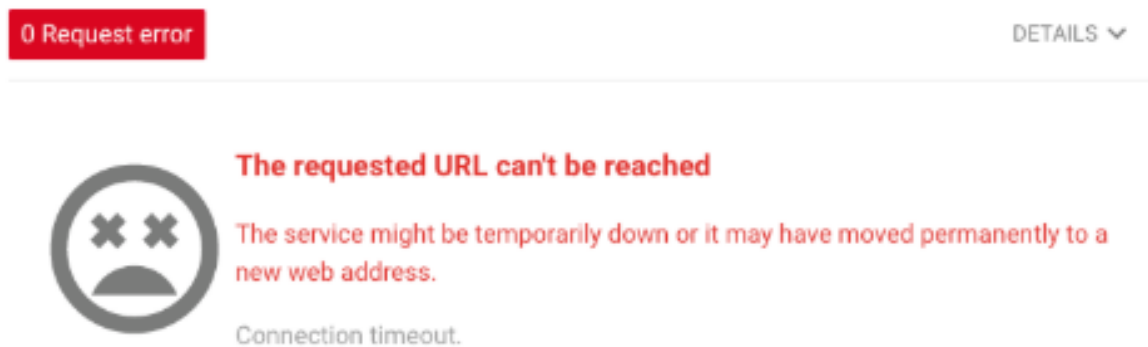
20. Add a breakpoint to the Logger.



21. In Advanced REST Client, make another request to http://localhost:8081/hello?fname=max&lname=mule.
22. In the Mule Debugger, click the Resume button; you should step to the Logger.
23. Click the Resume button again; you should step through the rest of the application.

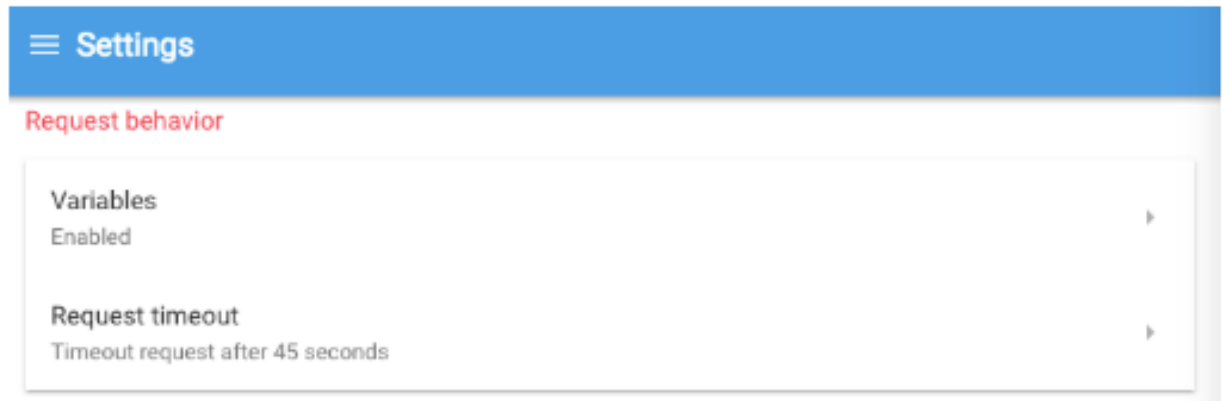
Cause the HTTP request to timeout

24. In Advanced REST Client, make another request to http://localhost:8081/hello?fname=max&lname=mule.
25. Do not step through the application and wait (45 seconds) for the request to timeout.

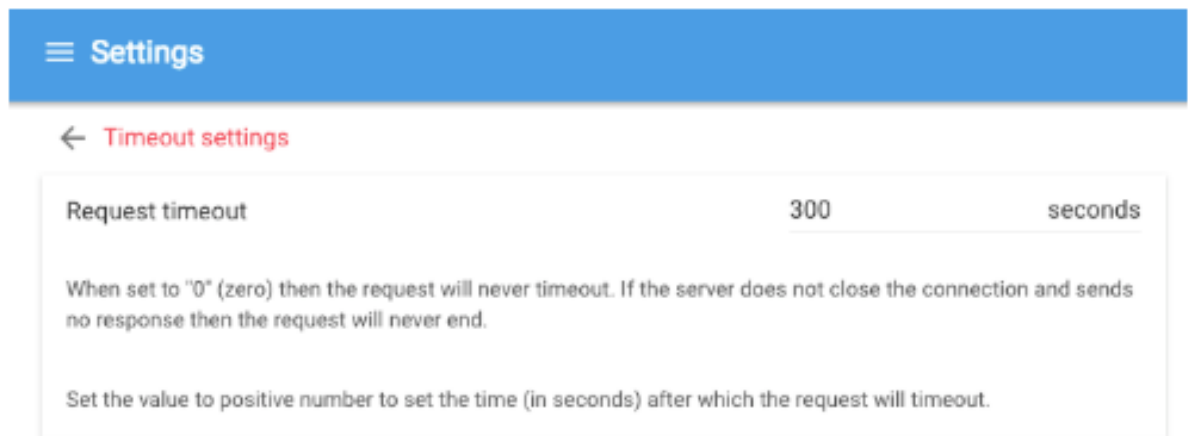


Increase the request timeout for Advanced REST Client

26. In the Advanced REST Client main menu, select Preferences.
27. In the Settings, locate the Request timeout setting and click the arrow next to it.

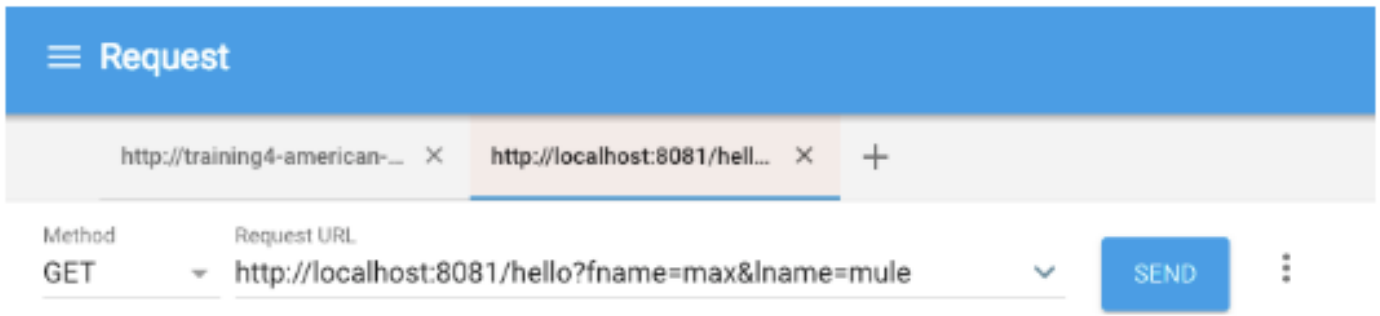


28. Change the request timeout to 300 seconds.



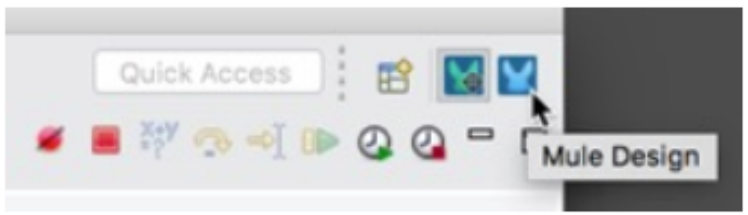
29. Click the Toggle Drawer button in the upper-left corner next to Settings and click HTTP Request.

30. Click the Toggle Drawer button again to hide the drawer; you should now see the Request screen again.



Switch perspectives

31. Return to Anypoint Studio.
32. Click the Resume button.
33. Click the Mule Design button in the upper-right corner of Anypoint Studio to switch perspectives.



Note: In Eclipse, a perspective is a specific arrangement of views in specific locations. You can rearrange the perspective by dragging and dropping tabs and views to different locations. Use the Window menu in the main menu bar to save and reset perspectives.

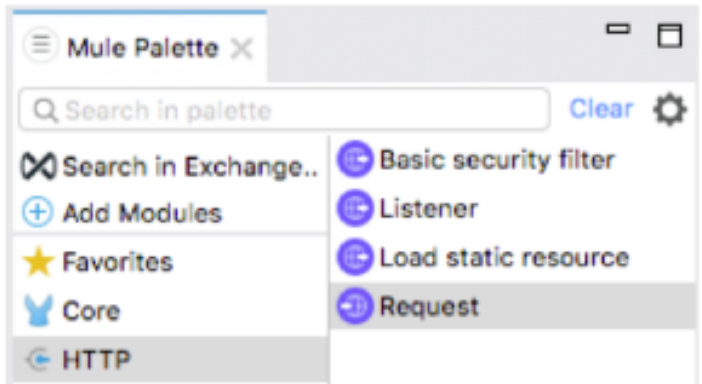
34. Leave the project running.

Tracking event data as it moves in and out of Mule applications

Changes to the event object

- We examined changes to an event as it moved through a flow and we set the payload

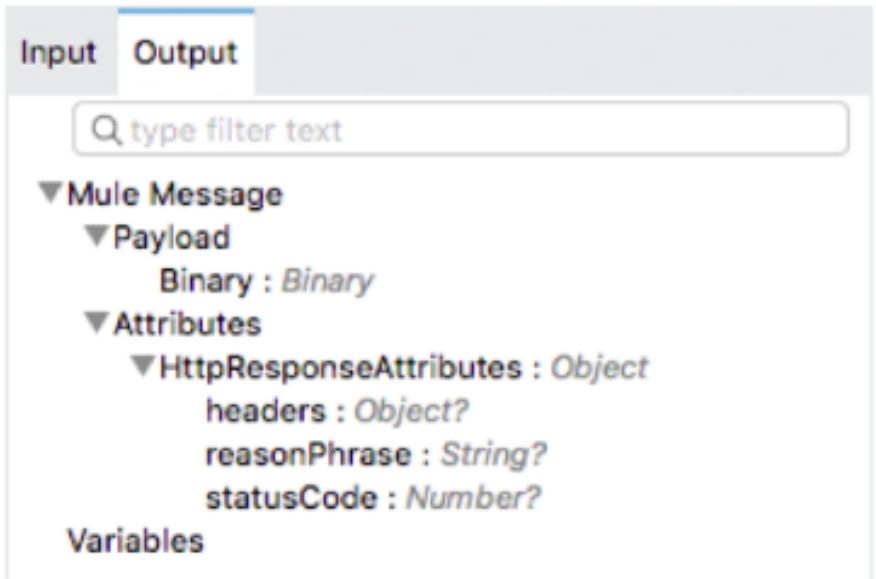
- What happens to the event object when calls are made to an external resource from a flow?
  - For example, you call a web service, and get return data?



Walkthrough 6-3: Track event data as it moves in and out of a Mule application

- Create a second flow with an HTTP Listener
- Make an HTTP request from the first flow to the new HTTP Listener
- View the event data as it moves through both flows

*Note: You are making an HTTP request from one flow to another in this exercise **only** so you can watch the value of event data as it moves in and out of a Mule application. You will learn how to pass events between flows within and between Mule applications in the next module.*

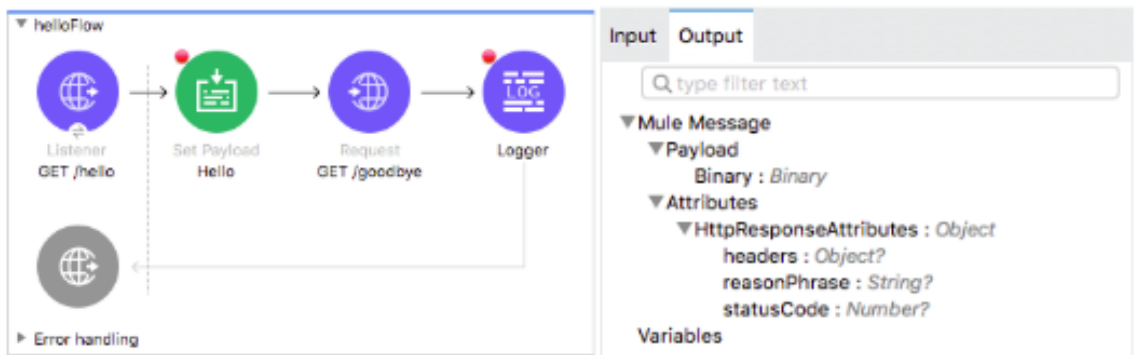


Walkthrough 6-3: Track event data as it moves in and out of a Mule application

In this walkthrough, you call an external resource, which for simplicity is another HTTP Listener in the same application, so that you so you can watch event data as it moves in an out of a Mule application. You will:

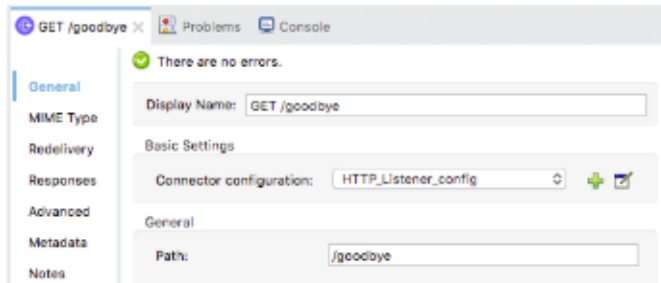
- Create a second flow with an HTTP Listener.
- Make an HTTP request from the first flow to the new HTTP Listener.
- View the event data as it moves through both flows.

  Note: You are making an HTTP request from one flow to another in this exercise ***only*** *so you can watch the value of event data as it moves in and out of a Mule application. You will learn how to pass events between flows within and between Mule applications in the next module.*



Create a second flow with an HTTP Listener

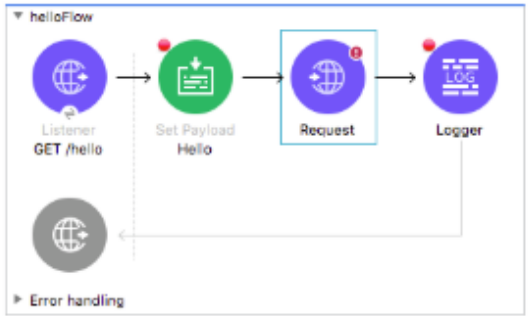
1. Return to apdev-examples.xml.
2. Drag an HTTP Listener from the Mule Palette and drop it in the canvas beneath the first flow.
3. Change the name of the flow to goodbyeFlow.
4. In the Listener view, set the connector configuration to the existing HTTP_Listener_config.
5. Set the path to /goodbye and the allowed methods to GET.
6. Set the display name to GET /goodbye.



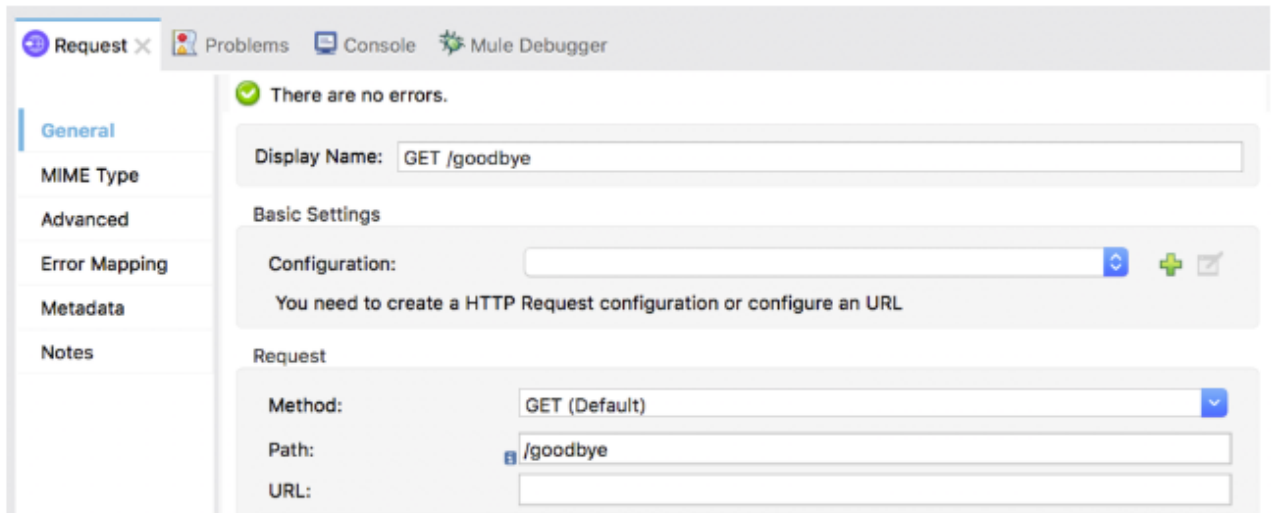
7. Add a Set Payload transformer and a Logger to the flow.
8. In the Set Payload properties view, set the display name and value to Goodbye.

Make an HTTP request

9. From the Mule Palette, drag an HTTP Request to the canvas and drop it before the Logger in helloFlow.

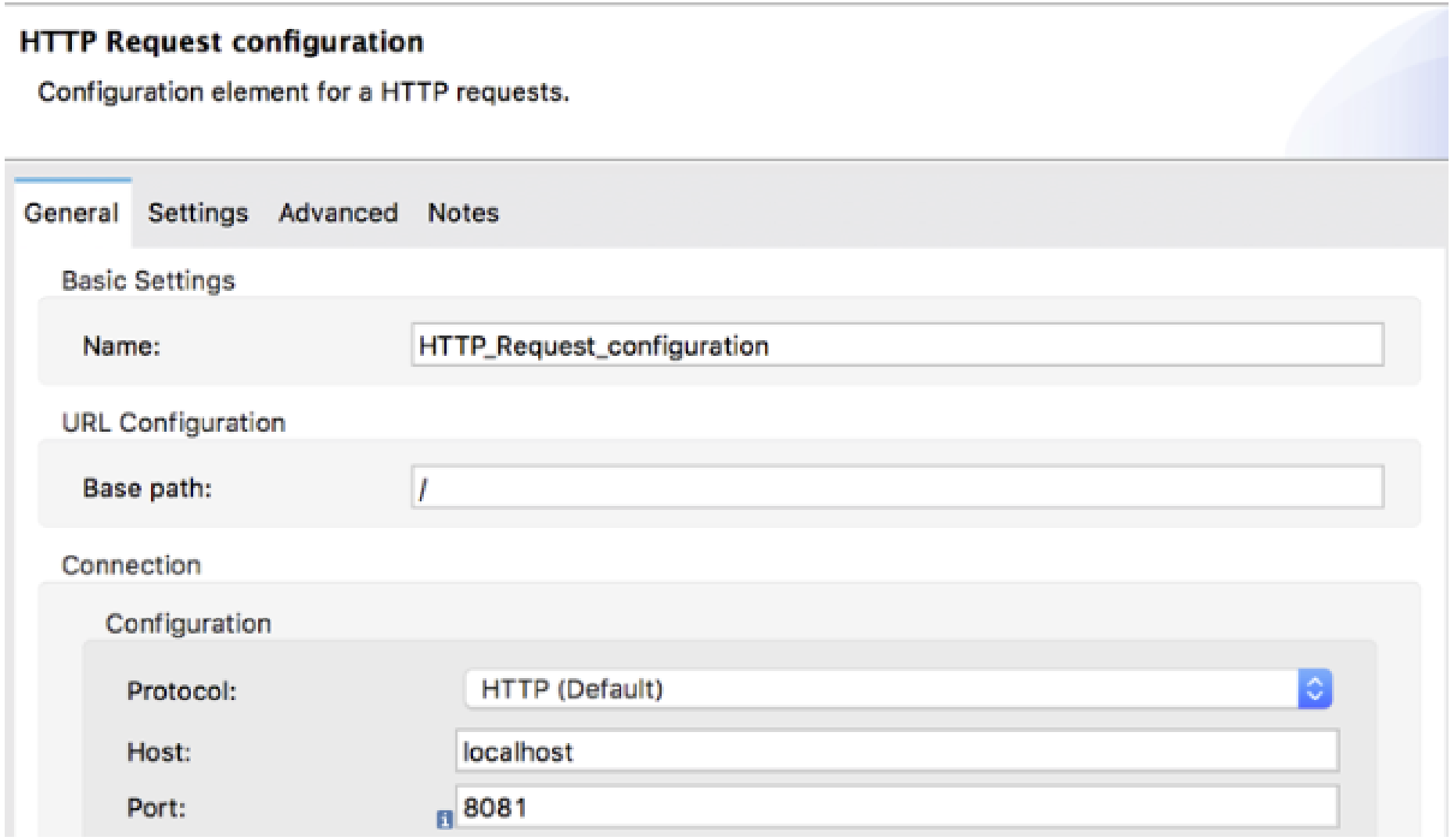


10. In the Request properties view, set the display name to GET /goodbye.
11. Set the path to /goodbye and leave the method set to GET.



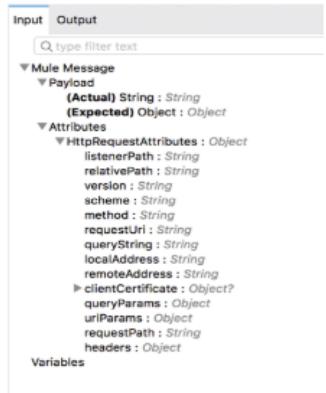
12. Click the Add button next to configuration.

13. In the dialog box, set the host to localhost and the port to 8081 and click OK.



View event structure and metadata in the DataSense Explorer

14. In the DataSense Explorer, expand Payload and Attributes in the Input tab.



15. Select the Output tab and expand Payload and Attributes.

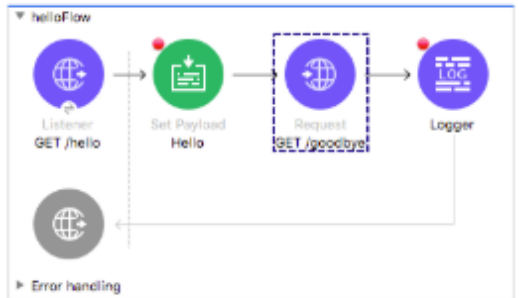
Change timeout for HTTP Request response

16. In the properties view for the GET /goodbye HTTP Request, locate the Response section on the General tab.
17. Set the Timeout to 300000.

    Note: This is being set only for debugging purposes so that the HTTP Request does not timeout when you are stepping through the application and examining data in the Mule Debugger.

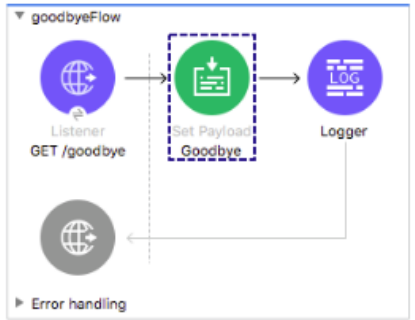
Debug the application

18. Save the file to redeploy the application.
19. In Advanced REST Client, send the same request to http://localhost:8081/hello?fname=max&lname=mule.
20. In the Mule Debugger, step to the GET /goodbye request.



21. Look at the values of the payload and the attributes, including the queryParams.

22. Step into goodbyeFlow.

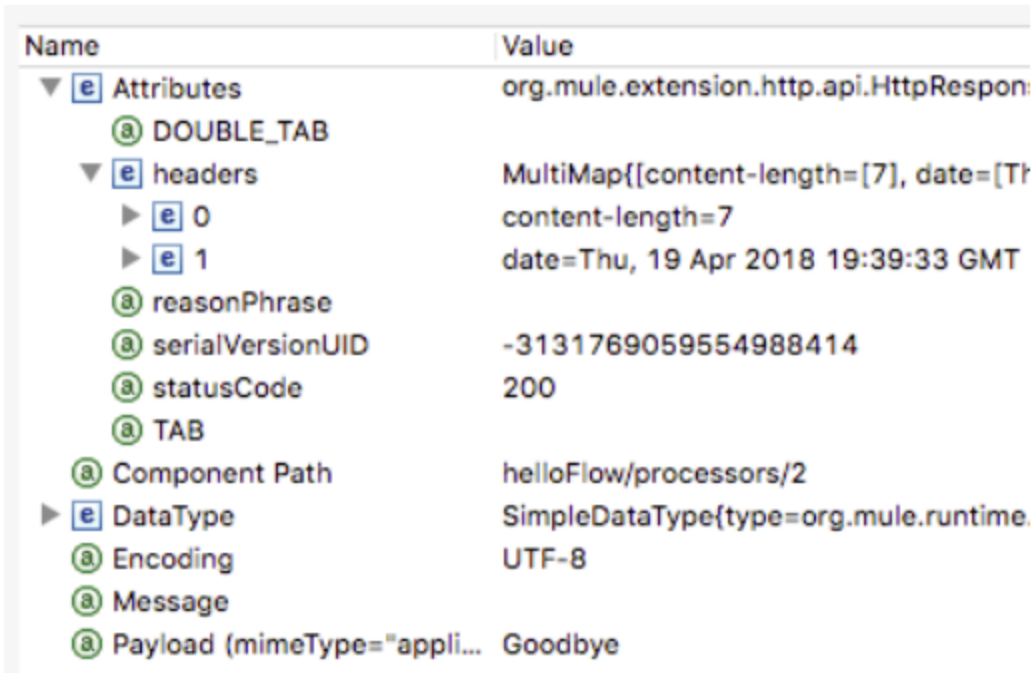


23. Look at the values of the payload and the attributes.

Name	Value
Attributes	org.mule.extension.http.api.HttpReque...
clientCertificate	null
DOUBLE_TAB	
headers	MultiMap{[x-correlation-id=[0-440733...
listenerPath	/goodbye
localAddress	localhost/127.0.0.1:8081
method	GET
queryParams	MultiMap{[]}
queryString	
relativePath	/goodbye
remoteAddress	/127.0.0.1:51473
requestPath	/goodbye
requestUri	/goodbye
scheme	http
serialVersionUID	7227330842640270811
TAB	
uriParams	MultiMap{[]}
version	HTTP/1.1
Component Path	goodbyeFlow/processors/0
DataType	SimpleDataType{type=org.mule.runtim...
Encoding	UTF-8
Message	
Payload (mimeType="*/*; charset=UT...	

24. Step through the flow until the event returns to the Logger in helloFlow.

25. Look at the values of the payload and the attributes.



26. Step through the rest of the application.

Review response data

27. Return to Advanced REST Client and view the return data and the http status code.
28. Click the Details button on the right side to expand this section.
29. Look at the response headers; you should see content-length, content-type, and date headers.

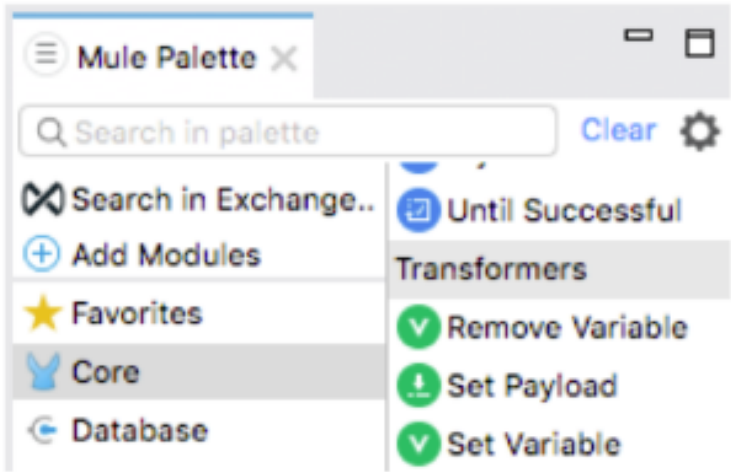
30. Return to Anypoint Studio and switch to the Mule Design perspective.

Setting request and response data

Setting the message payload

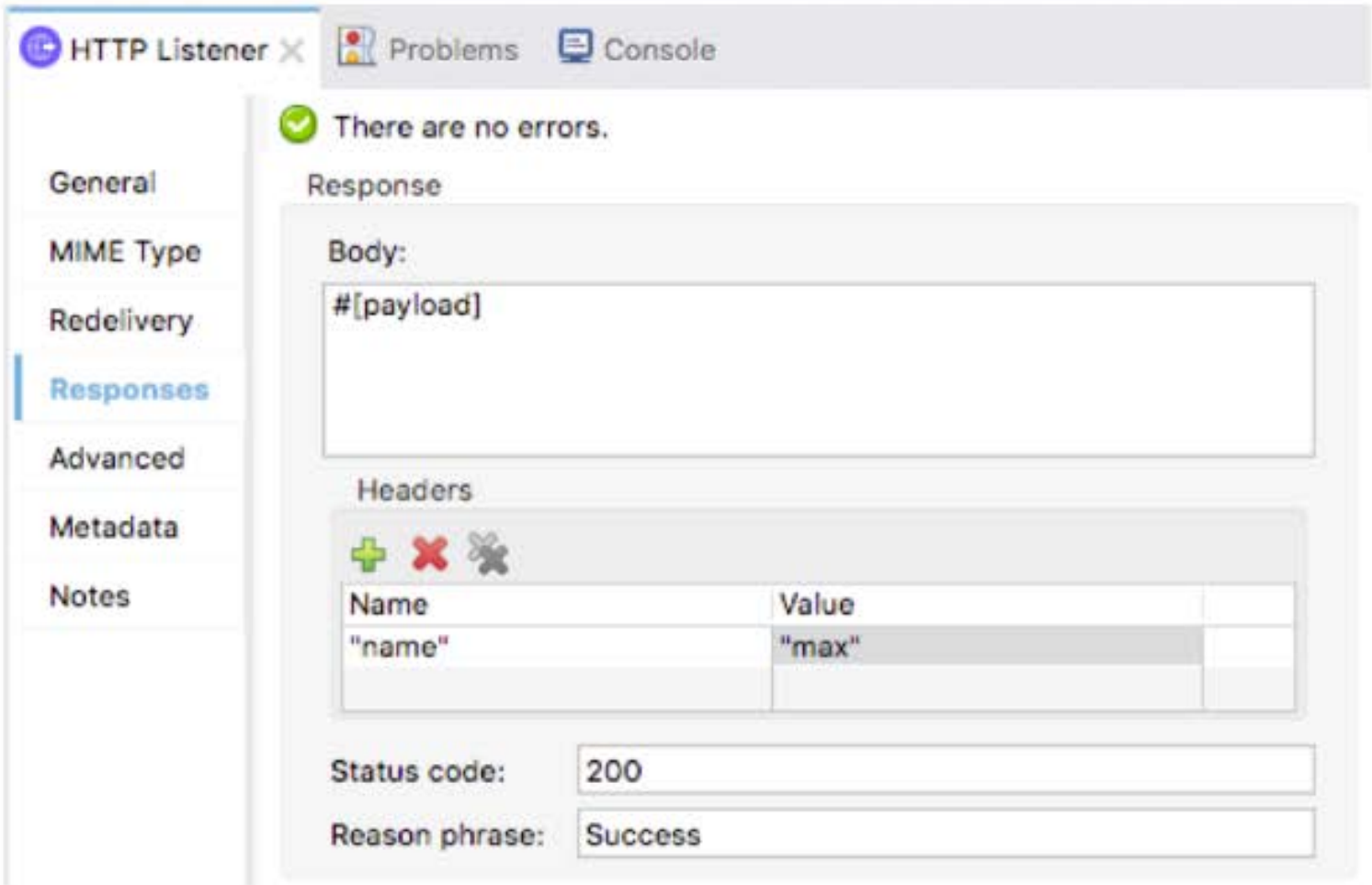
- Use the Set Payload transformer to set the payload

Set Payload



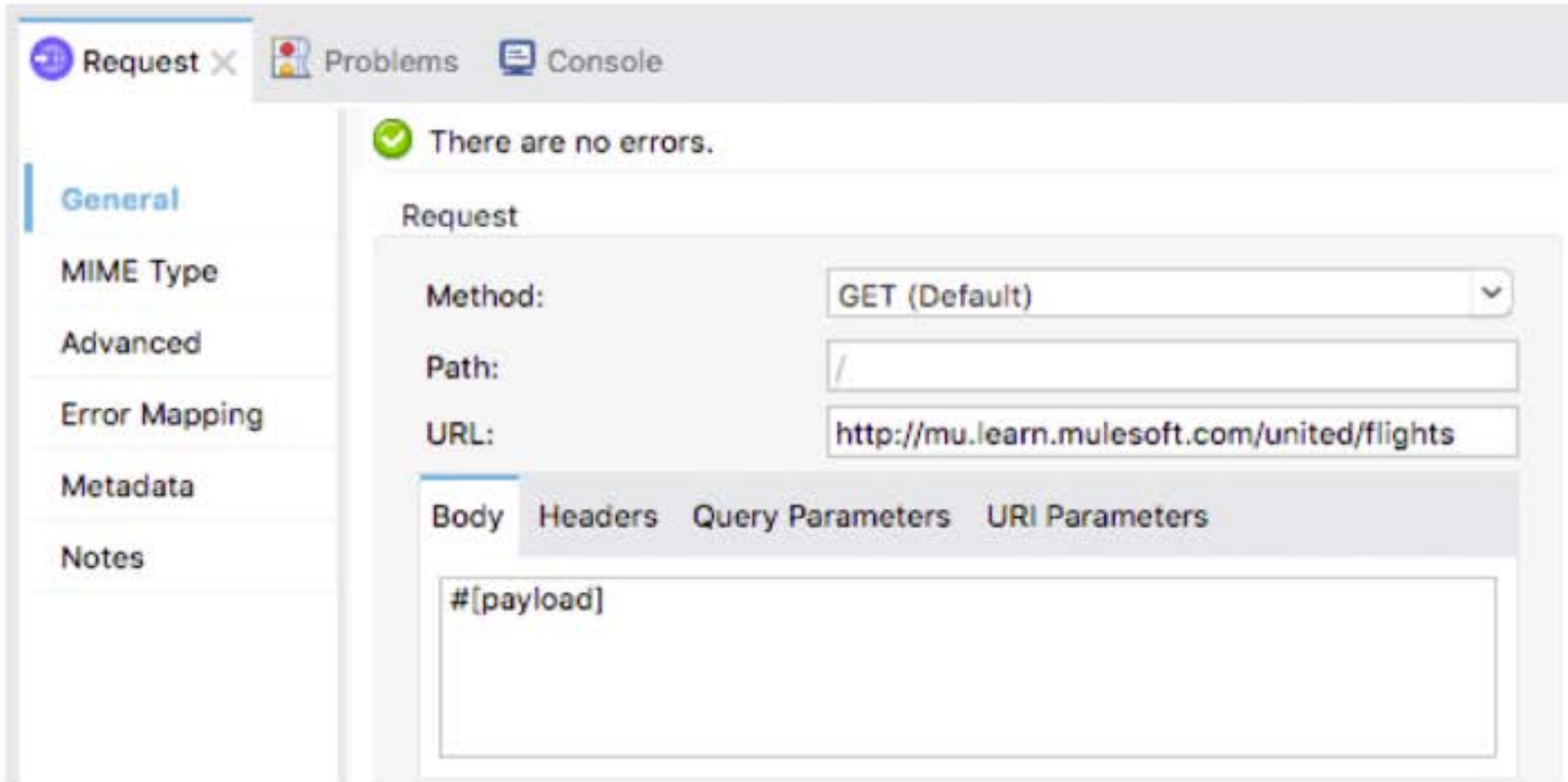
Setting data returned from HTTP listeners

- Set in the HTTP Listener properties view > Responses



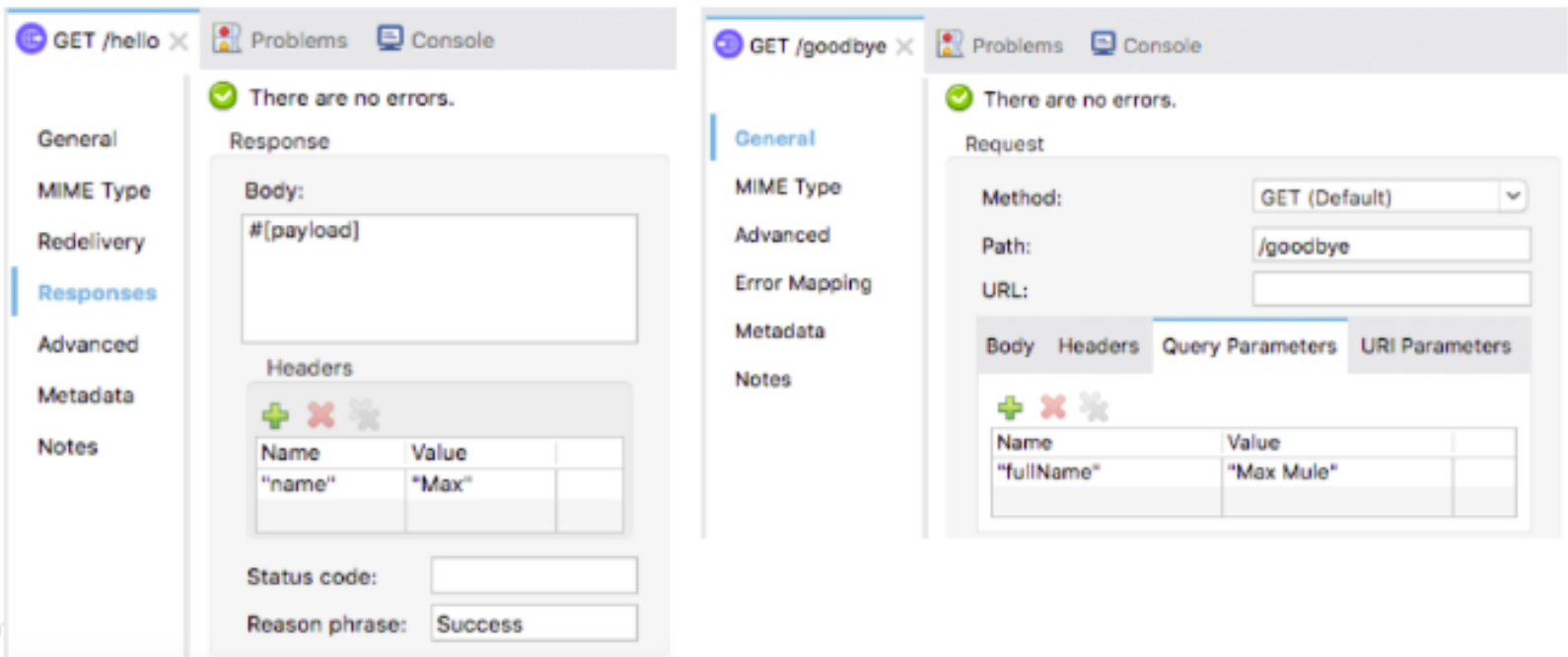
Setting data sent to HTTP requests

- Set in the HTTP Request properties view > General



Walkthrough 6-4: Set request and response data

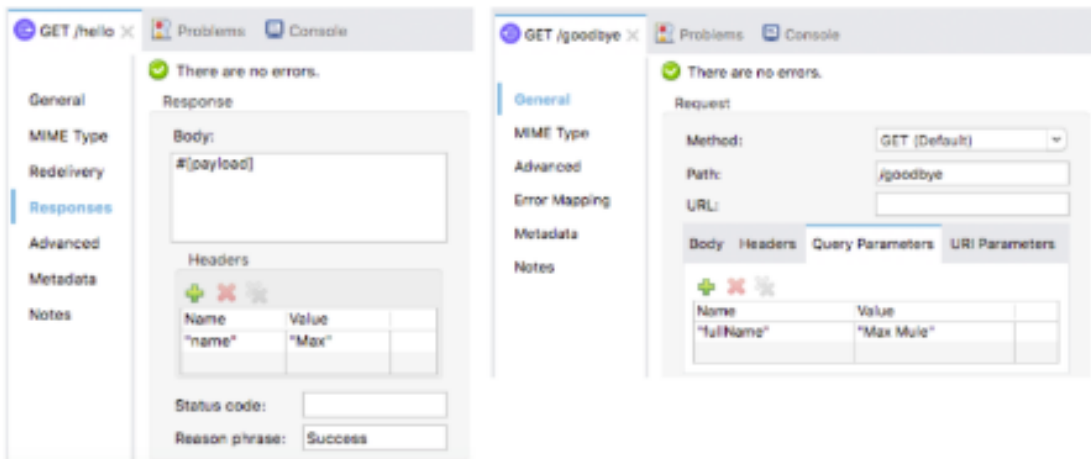
- View the default setting for a response body
- Set a response header
- View the default setting for a request body
- Set a request query parameter



Walkthrough 6-4: Set request and response data

In this walkthrough, you set response and request data for HTTP Listener and HTTP Request operations. You will:

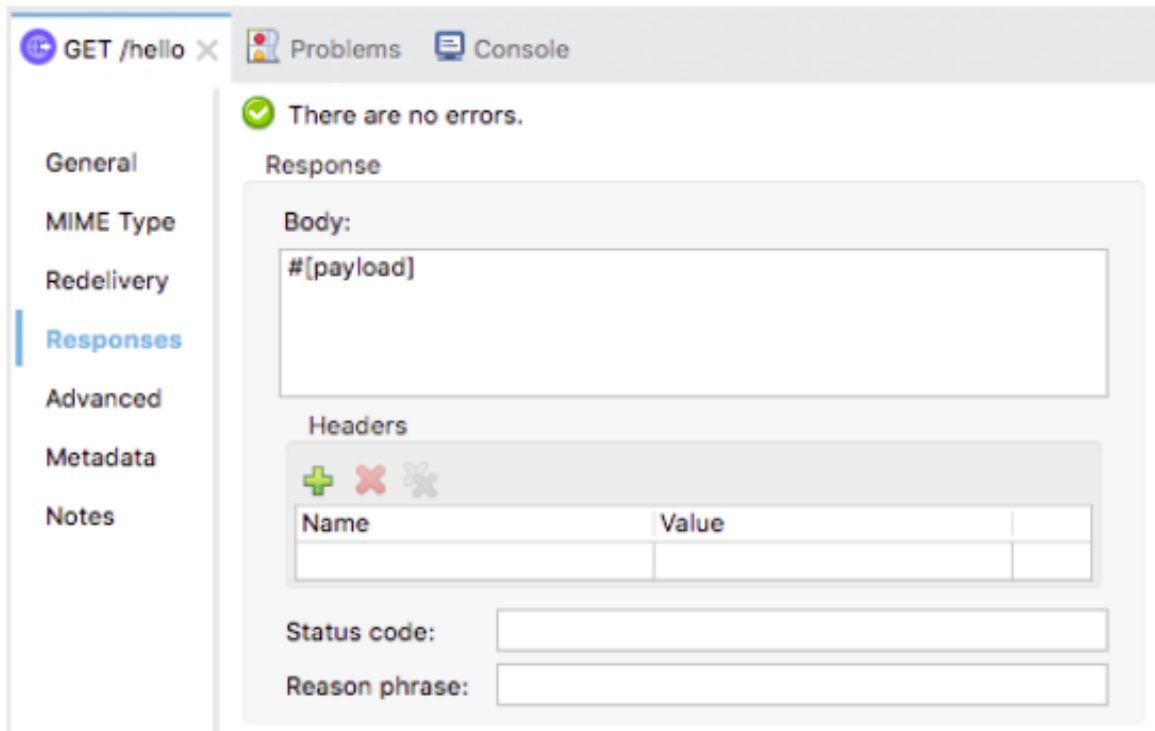
- View the default setting for a response body.
- Set a response header.
- View the default setting for a request body.
- Set a request query parameter.



View default response body

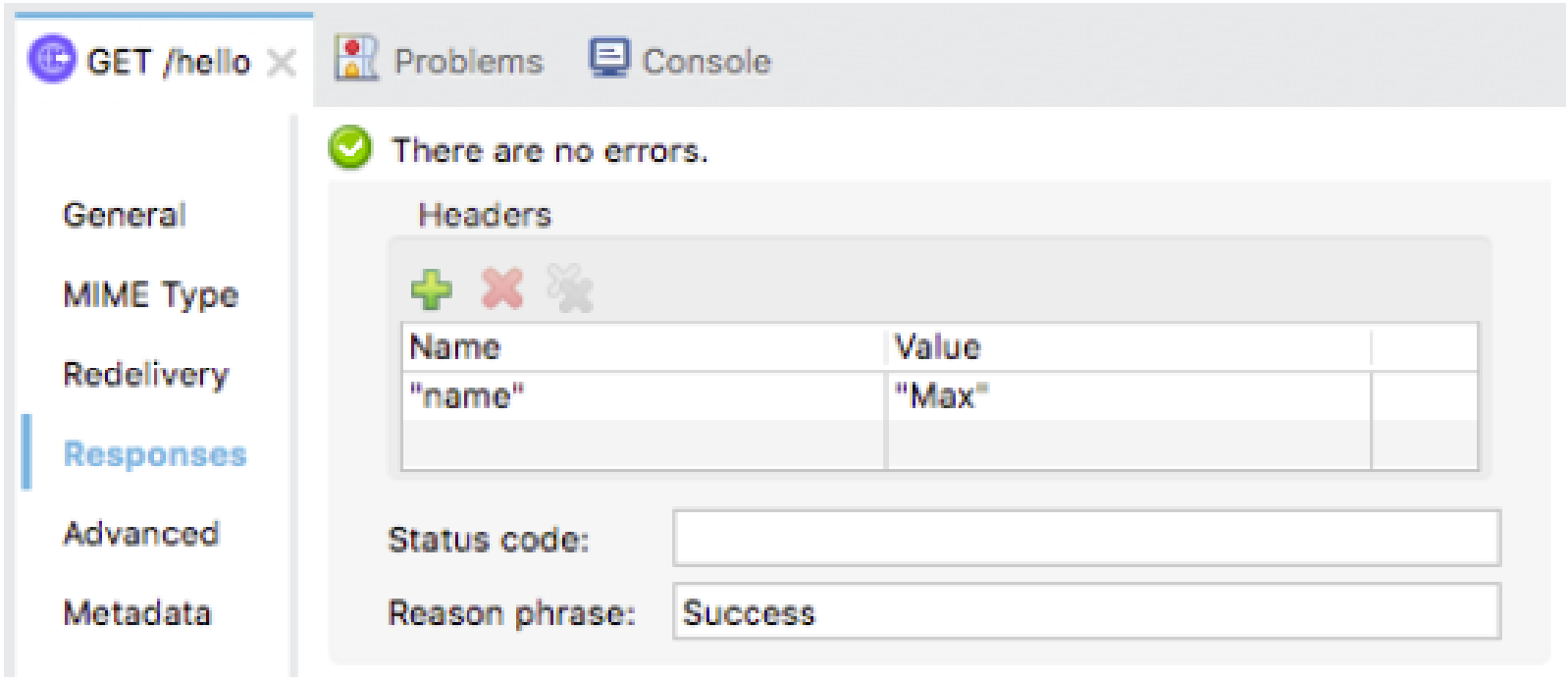
1. Return to apdev-examples.xml.
2. Double-click the GET /hello HTTP Listener in helloFlow.
3. In the GET /hello properties view, select the Responses tab.
4. In the Response section, locate the expression that sets the response body by default to the value of the payload.

   Note: The syntax for expressions will be covered in the next walkthrough.



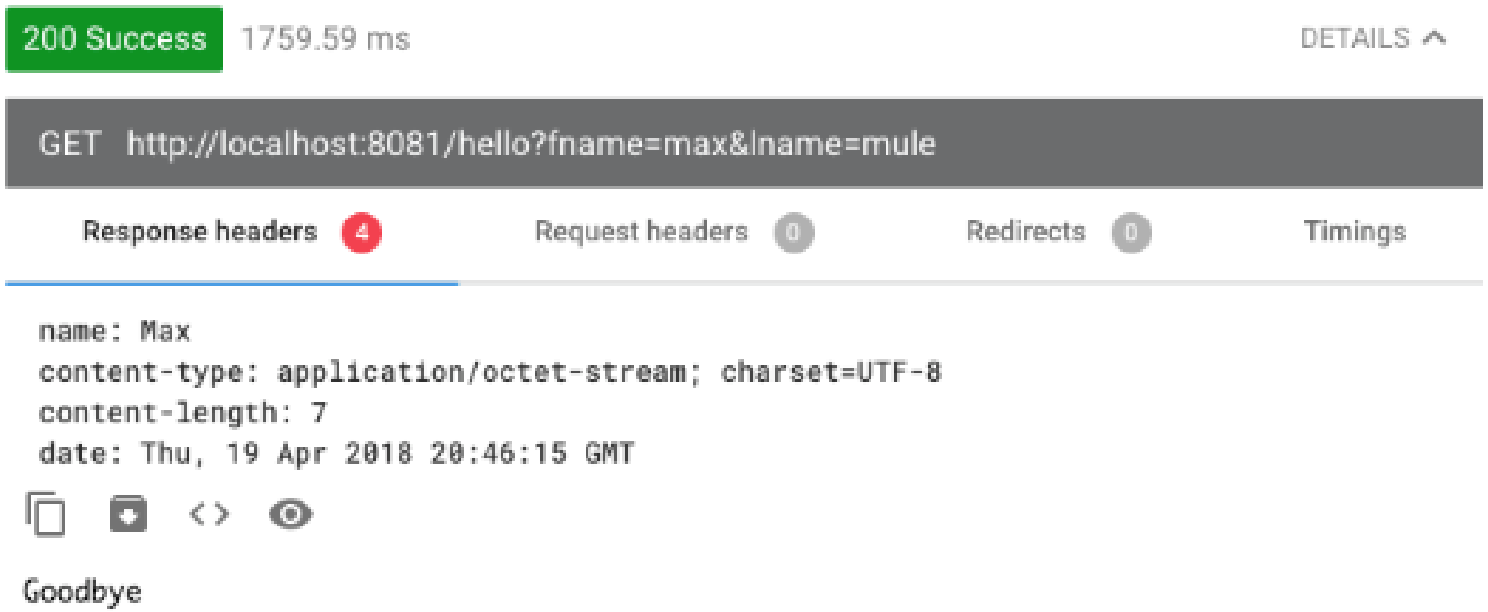
Set a response header

5. In the Headers section for the response, click the Add button.
6. Set the name to "name" and the value to "Max".
7. Locate the status code field and leave it blank so the default value 200 is returned.
8. Set the reason phrase to Success.



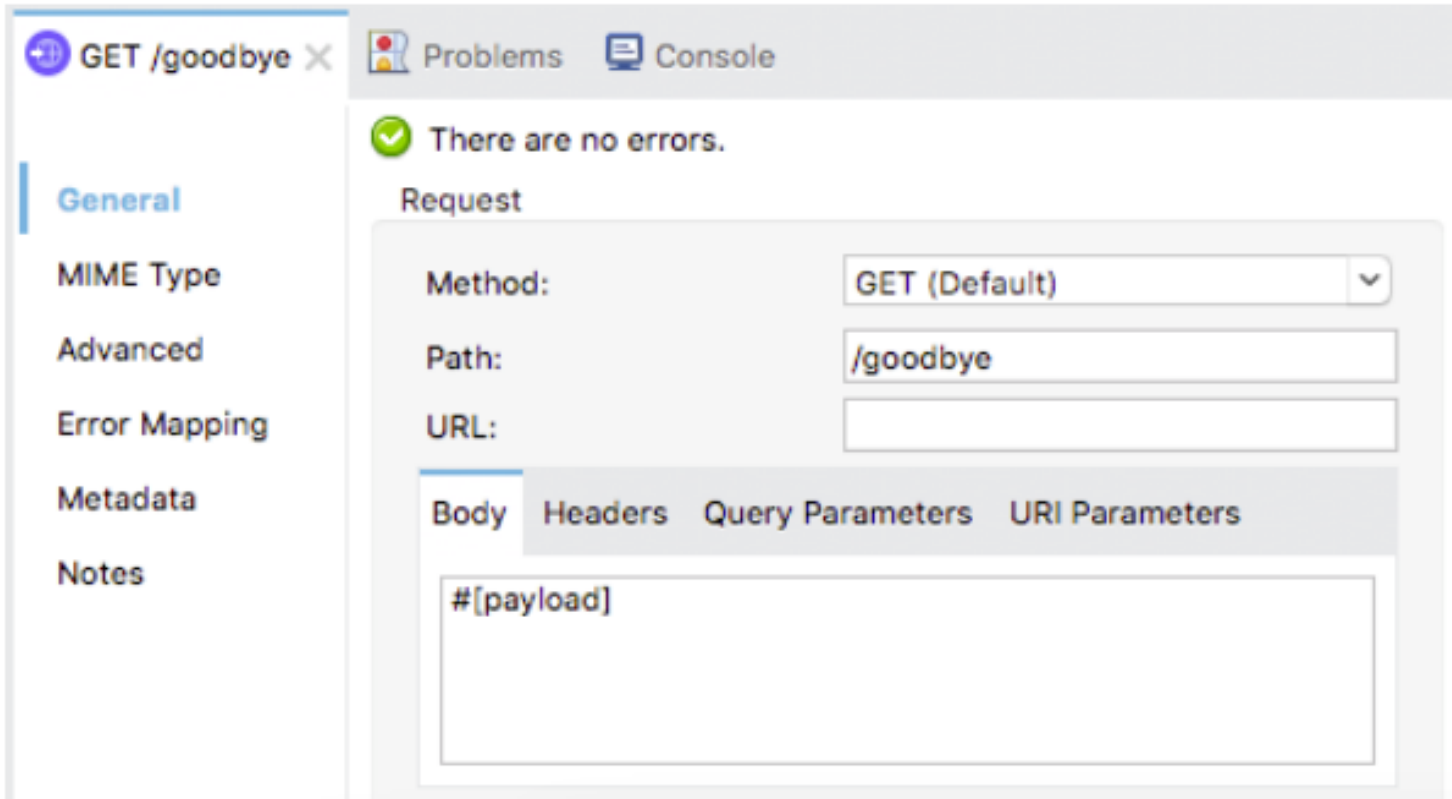
Run the application and review response data

9. Save the file to deploy the project.
10. Return to Advanced REST Client and send the same request.
11. In the Mule Debugger, click Resume until you step through the application.
12. In Advanced REST Client, locate your new status code reason phrase.
13. Review the response headers; you should now see the new name header.



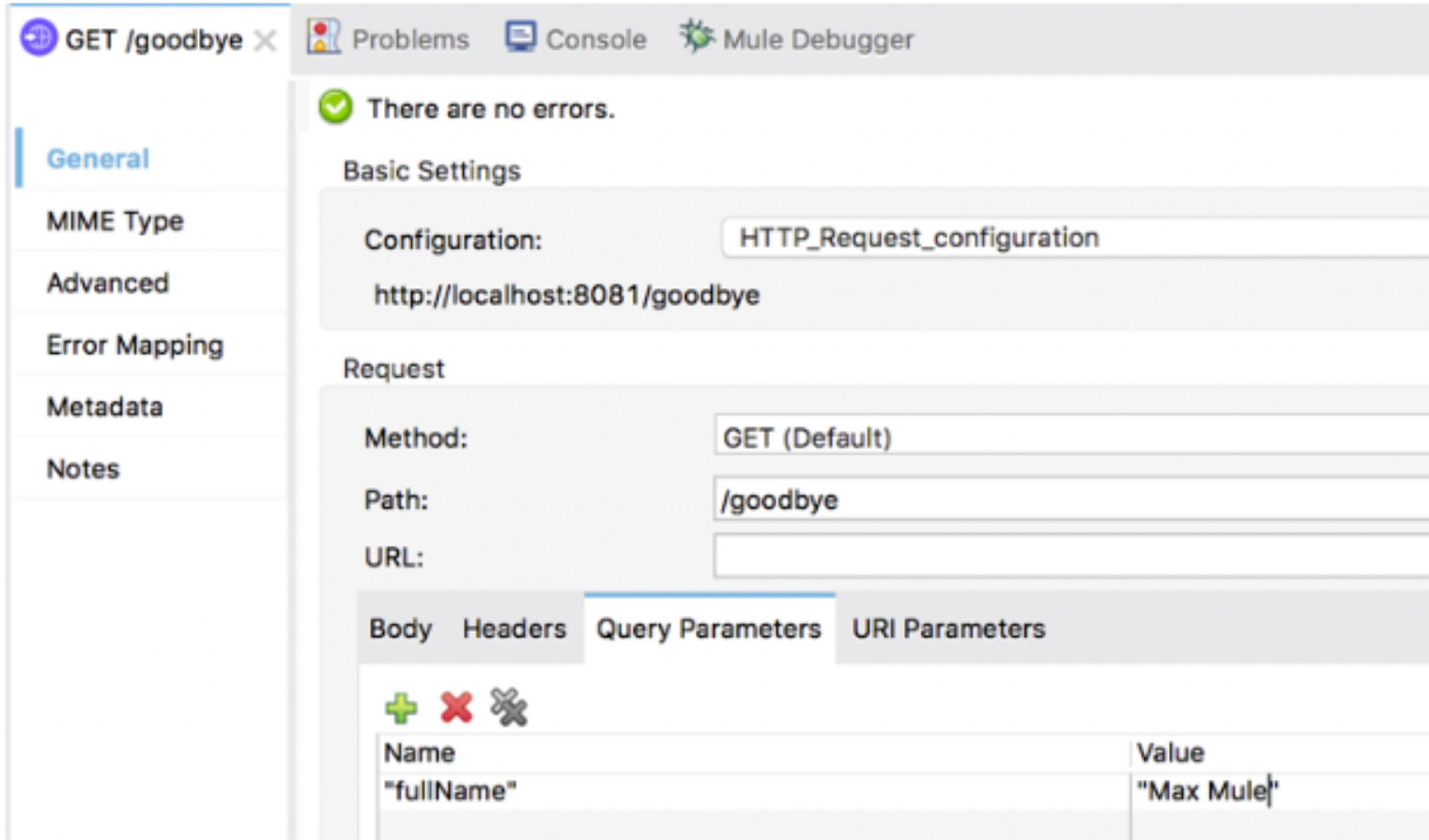
View default request body

14. Return to Anypoint Studio and switch perspectives.
15. Double-click the GET /goodbye HTTP Request in helloFlow.
16. In the GET /goodbye properties view, locate the Request section on the General tab.
17. On the Body tab, locate the expression that sets the request body by default to the value of the payload.



Set a request query parameter

18. Select the Query Parameters tab and click the Add button.
19. Set the name to "fullName" and the value to "Max Mule".



Debug and verify the query parameter is sent with the request

20. Save the file to redeploy the project.
21. Return to Advanced REST Client and send the same request.
22. Return to the Mule Debugger and step into goodbyeFlow.
23. Expand Attributes and locate the fullName query parameter.

24. Step through the rest of the application.
25. Switch to the Mule Design perspective.
26. Stop the project.

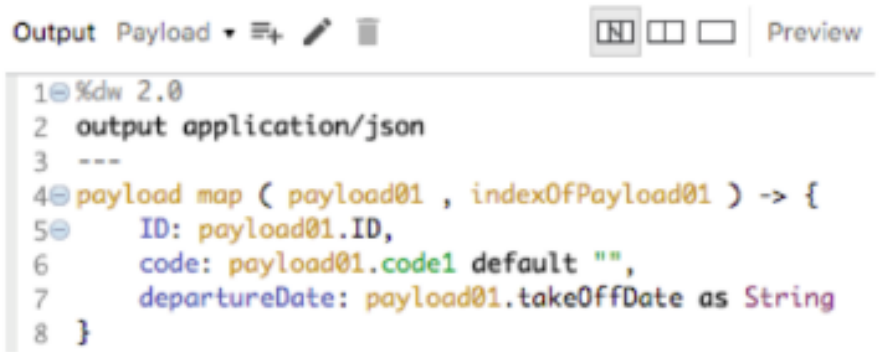
Using DataWeave expressions to read and write event data

The DataWeave expression language

- A Mule-specific expression and transformation language
- Can be used to access and evaluate the data in the payload, attributes, and variables of a Mule event
- Accessible and usable from all event processors and global elements
  - Is used to modify the way the processors act upon the event such as routing
- Case-sensitive
- Easy to use with auto-complete everywhere

Types of DataWeave expressions

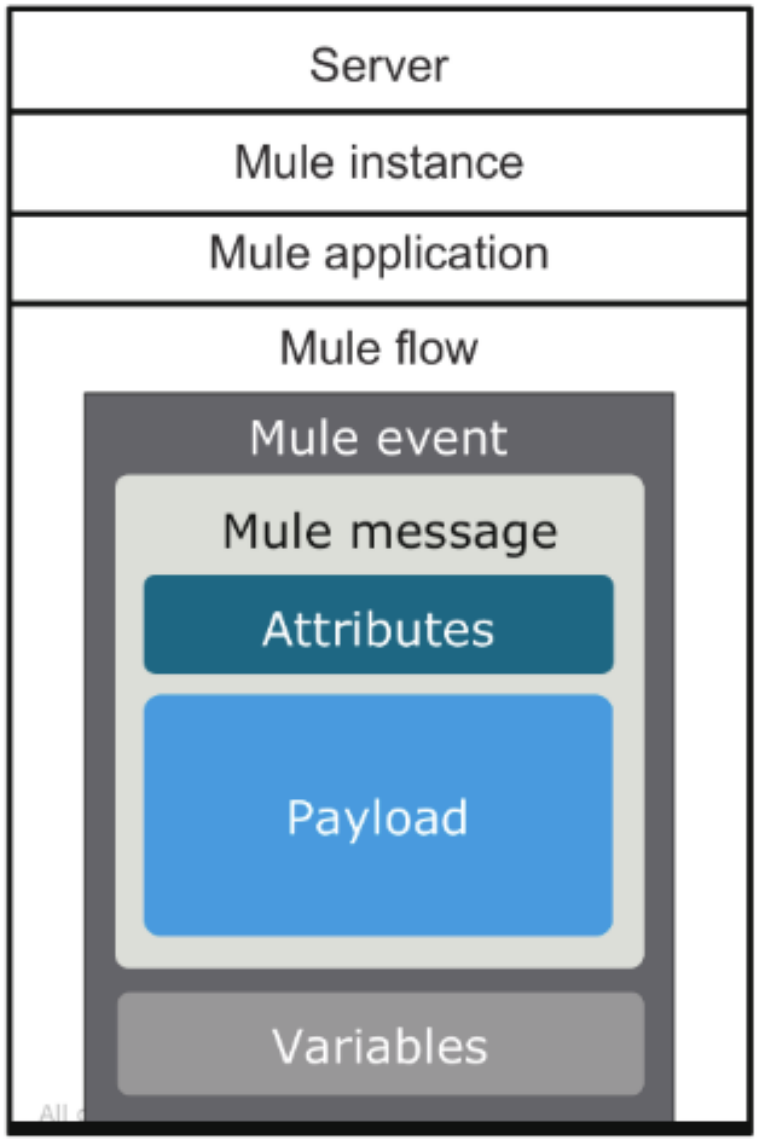
- **Standalone scripts**
  - Were generated using the Transform Message graphical editor in Module 4
  - We will write these from scratch in Module 11



- **Inline expressions**
  - Are used to dynamically set the value of properties in an event processor or global configuration element
  - Are enclosed in #[]

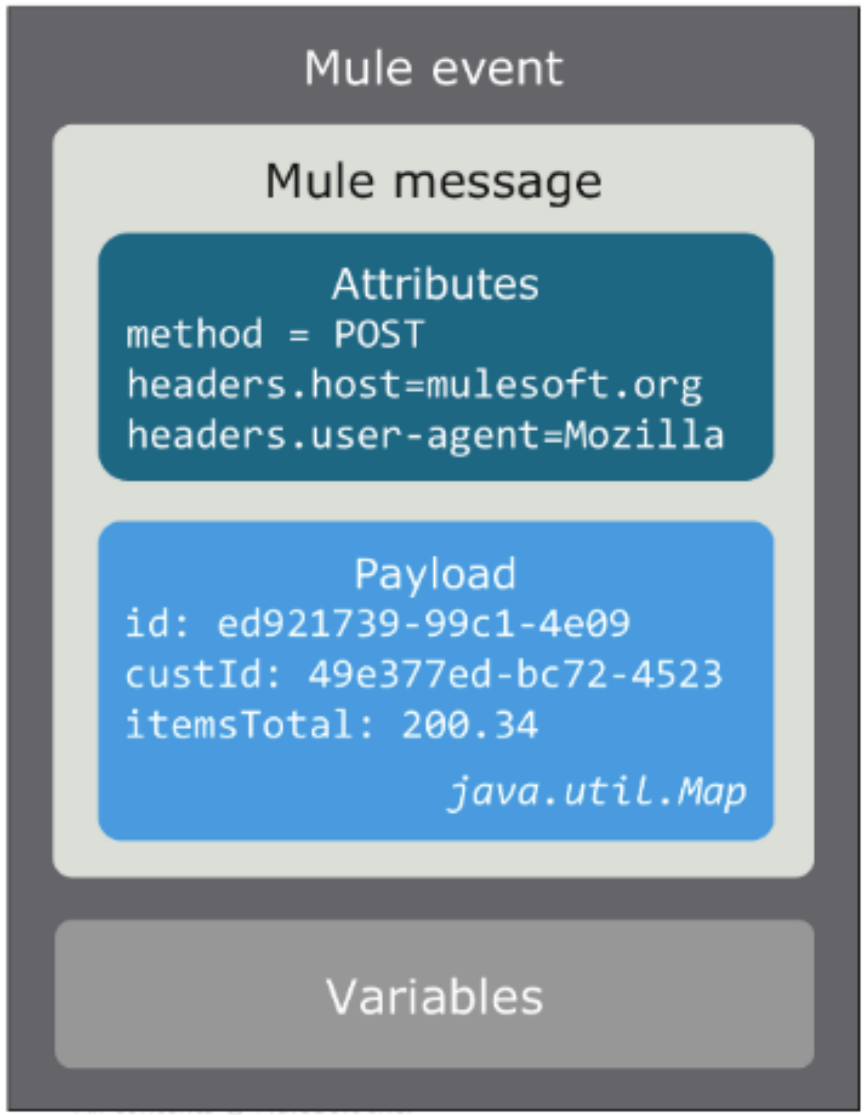
#[]

Referencing Mule objects in DataWeave expressions



server	#[server.osName]
mule	#[mule.version]
app	#[app.name]
flow	#[flow.name]
message	#[message.payload]
attributes	#[attributes.queryParams.param1]
payload	#[payload]
vars	#[vars.foo]

Accessing event attributes



`#[message.attributes.method]`	POST
`#[attributes.method]`	POST
`#[attributes.headers.host]`	mulesoft.org
`#[attributes.header['user-agent']]`	Mozilla

`#[message.payload.id]`	ed921739-99c1-4e09
`#[payload.id]`	ed921739-99c1-4e09
`#[payload['id']]`	ed921739-99c1-4e09
`#[payload.itemsTotal]`	200.34

Using selectors in DataWeave expressions

Description	Selector	Example
Value of a key:value pair	Single Value selector	`#[payload.name]` `#[attributes.queryParams]`
Value at selected array index	Indexed selector	`#[payload[0].name]`
Array with values of key:value pairs	Multi Value selector	`#[payload.*name]`
Array with values of key:value pairs	Descendants selector	`#[payload..zip]`

Note: You will learn additional selectors in Module 11

Using operators in DataWeave expressions

Description	Operators	Example
Arithmetic	+, -, /, *	`#[payload.age * 2]`
Equality	==, !=, ~=	`#[payload.name == "max"]`
Relational	>, >=, <, <=, is	`#[payload.age > 30]`
Conditional	and, or, not	`#[(payload.name=="max") and (payload.age>30)]`
Type coercion	as	`#[(payload.age as Number) * 2]`
Default value	default	`#[payload.type  default "student"]`

Note: For operator precedence, see https://docs.mulesoft.com/mule4-user-guide/v/4.1/dataweave-flow-control-precedence

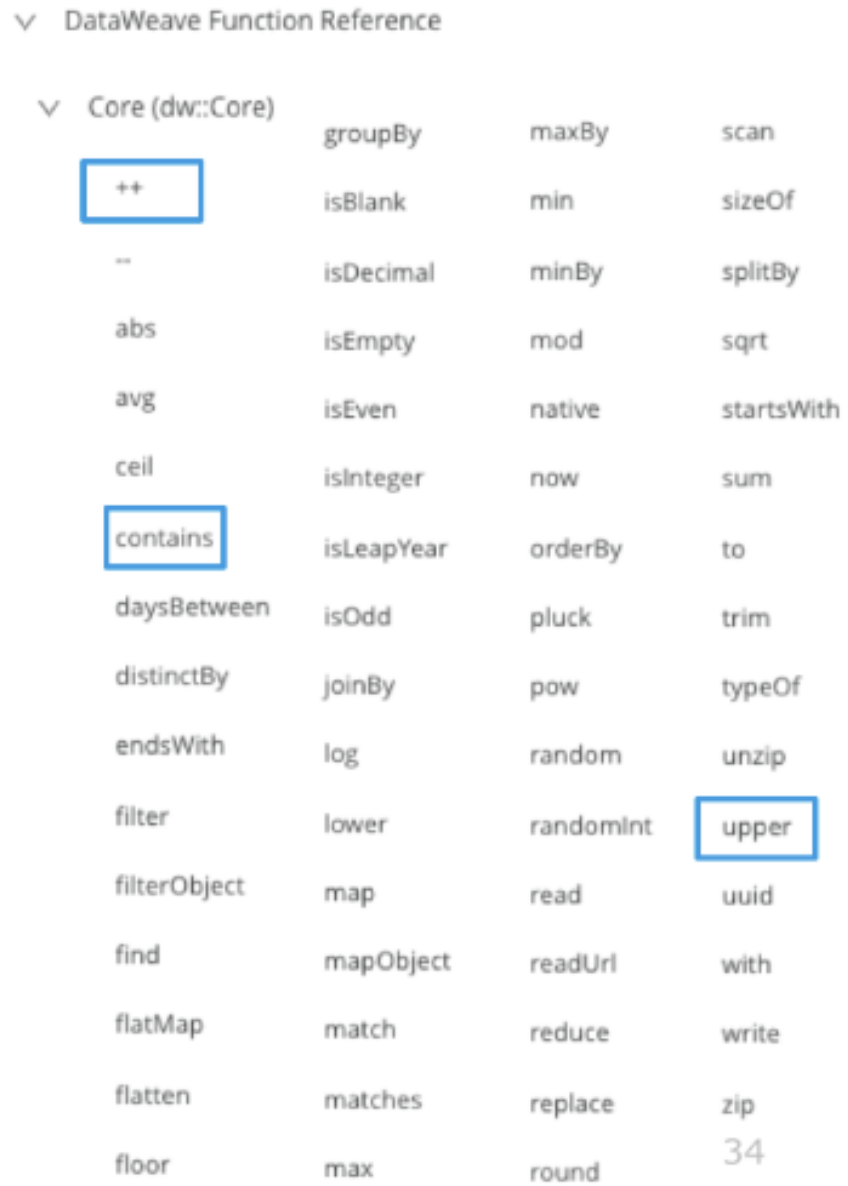
Using conditional logic statements in DataWeave

- Use if / else if / else statements

```
if (payload.age < 15)
  group: "child"
else if (payload.age < 25)
  group: "youth"
else if (payload.age < 65)
  group: "adult"
else
  group: "senior"
```

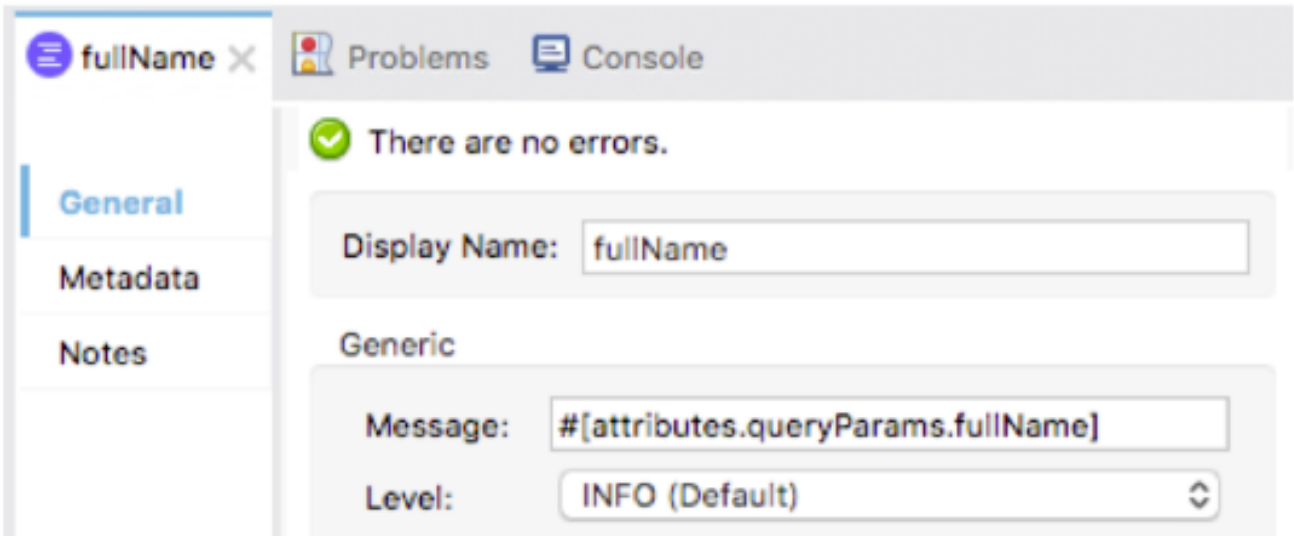
Using DataWeave functions

- Functions are packaged in modules
- Functions in the Core module are imported automatically into DataWeave scripts
- For function reference, see
  - https://docs.mulesoft.com/mule4-user-guide/v/4.1/dataweave
- For functions with 2 parameters, an alternate syntax can be used
  - #[contains(payload,max)]
  - #[payload contains "max "]



Walkthrough 6-5: Get and set event data using DataWeave expressions

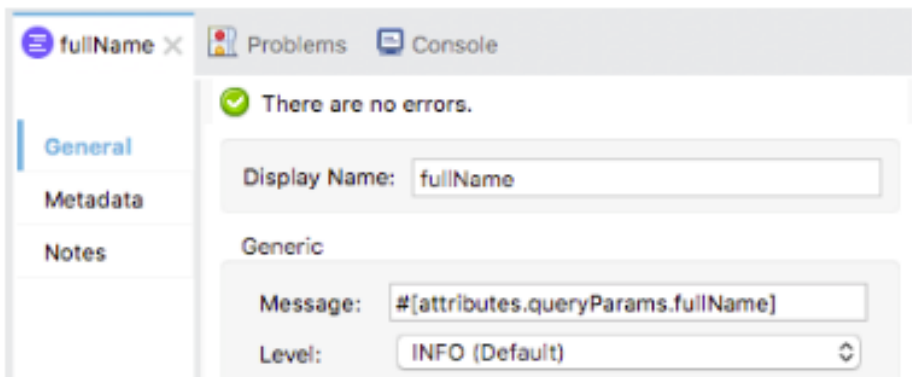
- Use expressions to set the payload and a logged value
- Use expressions to set a response header and a request query param
- In expressions, reference values for the event payload and attributes
- Use the DataWeave upper() function and the concatenation, as, and default operators



Walkthrough 6-5: Get and set event data using DataWeave expressions

In this walkthrough, you get and set event data using DataWeave expressions. You will:

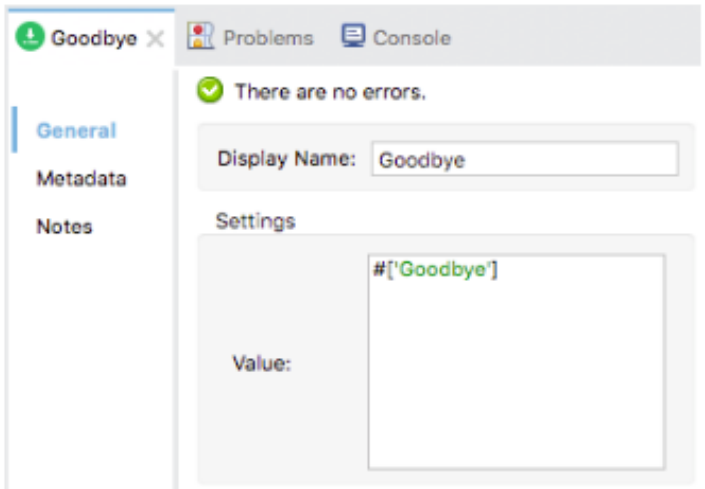
- Use expressions to set the payload and a logged value.
- Use expressions to set a response header and a request query parameter.
- In expressions, reference values for the event payload and attributes.
- Use the DataWeave upper() function and the concatenation, as, and default operators.



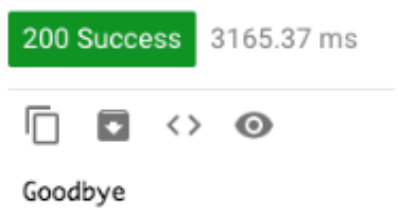
Use an expression to set the payload

1. Return to apdev-examples.xml.
2. Navigate to the properties view for the Goodbye Set Payload transformer in goodbyeFlow.
3. Change the value to an expression.

   `#['Goodbye']`

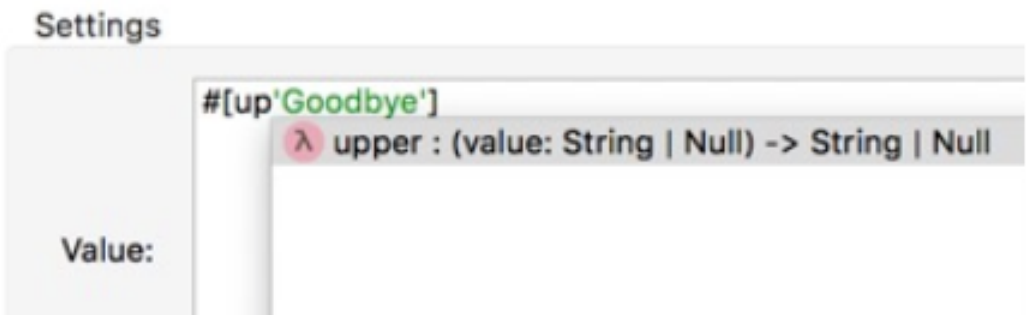


4. Run the project.
5. In Advanced REST Client, send the same request; you should get the same result of Goodbye as before.



Use a DataWeave function

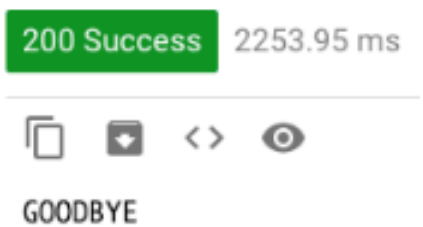
6. In Anypoint Studio, return to the Goodbye Set Payload properties view.
7. Inside the brackets and before the string, type the word up and press Ctrl+Spacebar.
8. In the auto-completion menu, select the upper function.



9. Add parentheses around the Goodbye string.

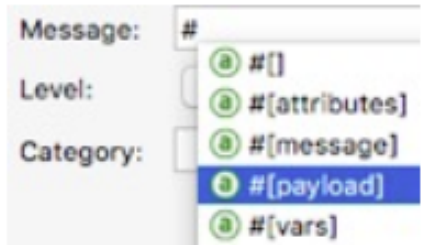
```
#[upper('Goodbye')]
```

10. Save the file to redeploy the application.
11. In Advanced REST Client, send the same request; the return string should now be in upper case.



Use an expression that references the payload in a Logger

12. Return to Anypoint Studio.
13. Navigate to the properties view for the Logger in helloFlow.
14. Change the display name to payload.
15. Type # in the message field and select #[payload] in the auto-completion menu.



16. Save the file to redeploy the application.
17. In Advanced REST Client, send the same request.
18. Return to the console in Anypoint Studio; you should see GOODBYE displayed for the second Logger instead of the entire event object.

```
INFO  2018-04-19 14:10:48,008 [[MuleRuntime].cpuLight.06: [apdev-examples].helloFlow.CPU_LITE @5dc0c3a3] [event: 0-216c8
710-4416-11e8-861e-8c85900da7e5] org.mule.runtime.core.internal.processor.LoggerMessageProcessor: GOODBYE
```

Use a string literal and a DataWeave expression in a property value

19. Return to the properties view for the Logger in helloFlow.
20. Change the value to display the string Message in front of the payload.

```
Message: #[payload]
```

21. Save the file to redeploy the application.
22. In Advanced REST Client, send the same request.
23. Return to the console in Anypoint Studio; you should see the new string displayed.

```
INFO  2018-04-19 14:15:43,640 [[MuleRuntime].cpuLight.16: [apdev-examples].helloFlow.CPU_LITE @3dd1f9a1] [event: 0-d1a23
d00-4416-11e8-b833-8c85900da7e5] org.mule.runtime.core.internal.processor.LoggerMessageProcessor: Message: GOODBYE
```

24. Return to the properties view for the Logger in helloFlow.

Use the DataWeave concatenation operator

25. Change the value so the string is part of the evaluated expression and use the concatenation operator.

```
#['Message: ' ++ payload]
```

26. Add \n in front of the message to display it on a new line.

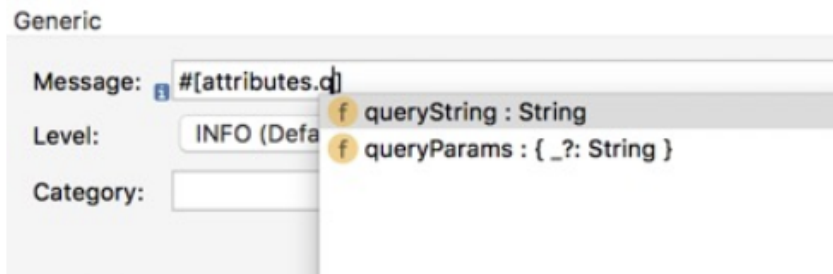
```
#['\nMessage: ' ++ payload]
```

27. Save the file to redeploy the application.
28. In Advanced REST Client, send the same request.
29. Return to the console in Anypoint Studio; you should see the string displayed on a new line.

```
INFO  2018-04-19 14:18:29,925 [[MuleRuntime].cpuLight.09: [apdev-examples].helloFlow.CPU_LITE @1d096383] [event: 0-34ddd
550-4417-11e8-b833-8c85900da7e5] org.mule.runtime.core.internal.processor.LoggerMessageProcessor:
Message: GOODBYE
```

Use an expression that references an attribute in a Logger

30. Return to the properties view for the Logger in goodbyeFlow.
31. Change the display name to fullName.
32. Type # in the message field and select #[attributes] in the auto-completion menu.
33. At the end of attributes, add a period, type q and select queryParams in the auto-completion menu.




```
#[attributes.queryParams]
```

34. Click Apply Changes to redeploy the application.
35. In Advanced REST Client, send the same request.
36. Return to the Anypoint Studio console; you should see an object displayed by the first Logger.

```
INFO  2018-04-19 14:25:27,821 [[MuleRuntime].cpuLight.15: [apdev-examples].goodbyeFlow.CPU_LITE @7c601de9] [event: 0-2df
95920-4418-11e8-b833-8c85900da7e5] org.mule.runtime.core.internal.processor.LoggerMessageProcessor: {fullName=Max Mule}
```

37. Modify the message for the Logger in goodbyeFlow to display the value of the query parameter.

```
#[attributes.queryParams.fullName]
```

38. Click Apply Changes to redeploy the application.
39. In Advanced REST Client, send the same request.
40. Return to the console; you should now see the value of the parameter displayed.

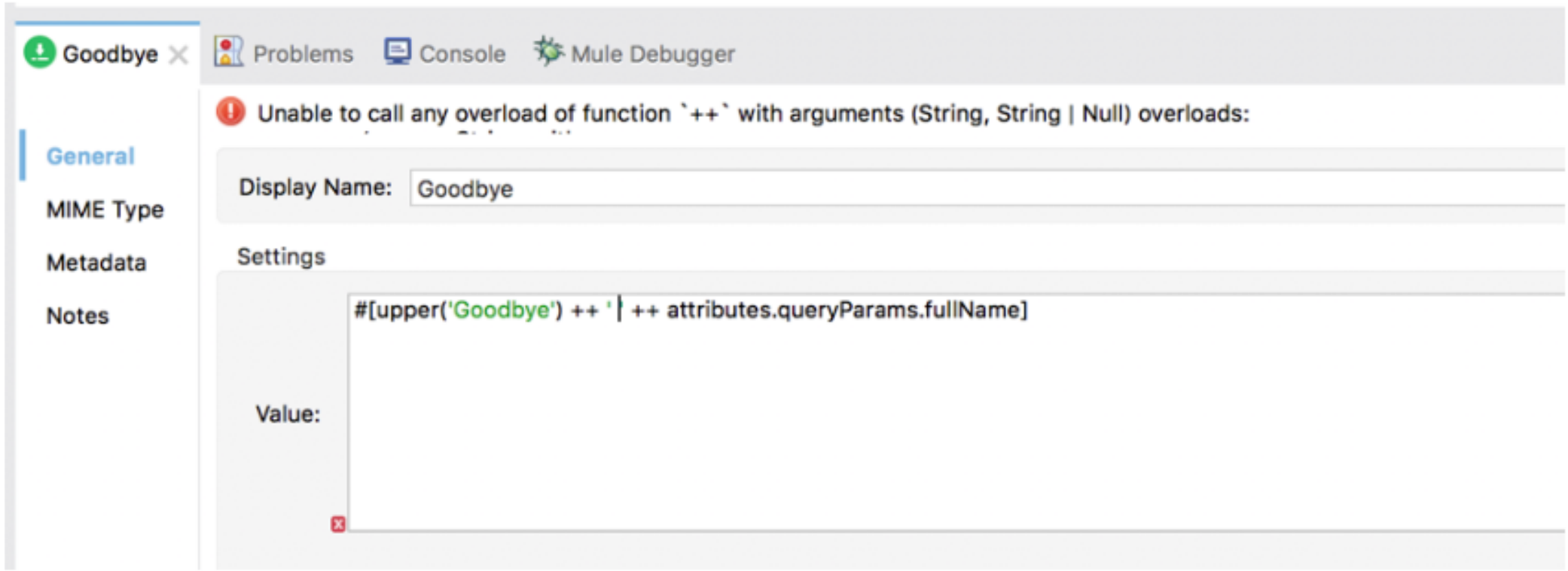
```
INFO  2018-04-19 14:26:22,192 [[MuleRuntime].cpuLight.03: [apdev-examples].goodbyeFlow.CPU_LITE @1e7dbbd3] [event: 0-4e6
55dd0-4418-11e8-b833-8c85900da7e5] org.mule.runtime.core.internal.processor.LoggerMessageProcessor: Max Mule
```

Use an expression that references an attribute when setting the payload

41. Navigate to the properties view for the Goodbye Set Payload in goodbyeFlow.
42. Use the concatenation operator to also display the value of the query parameter separated by a space.

```
#[upper('Goodbye') ++ ' ' ++ attributes.queryParams.fullName]
```

43. Review the error.



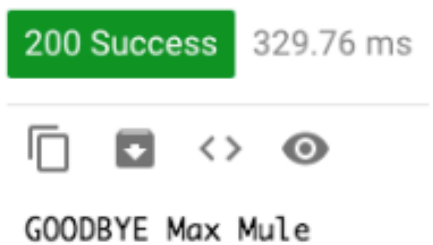
Use the as operator to coerce the attribute to a String

44. Use the as operator to also display the value of the query parameter separated by a space.

    ```
    #[upper('Goodbye') ++ ' ' ++ attributes.queryParams.fullName as String]
    ```

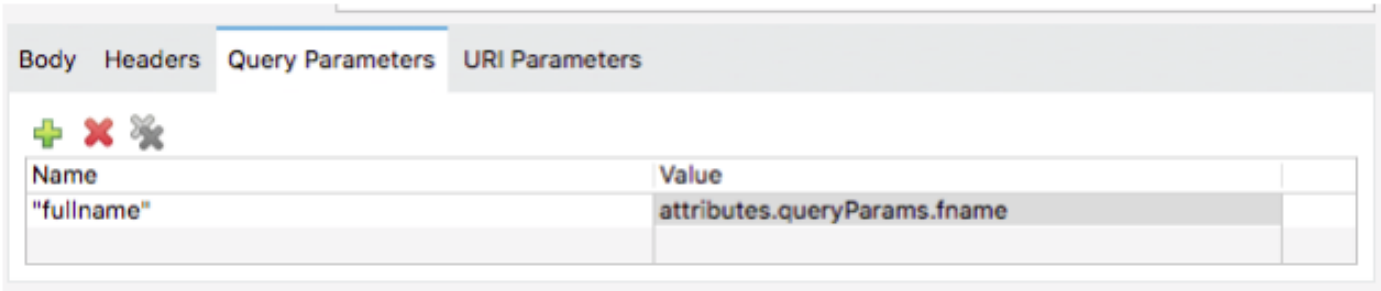
    Note: This step coerces a null value to a String, to avoid UI errors. The recommended way to handle null values is by using a default value which is explained later in this walkthrough.

45. Click Apply Changes to redeploy the application.
46. In Advanced REST Client, send the same request; you should now also see the name displayed.

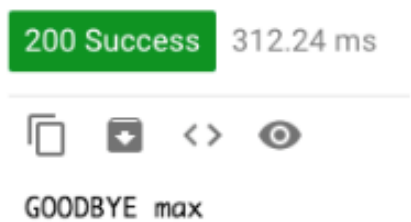


Use an expression to set a request header

47. Return to the Anypoint Studio and navigate to the properties view for the GET /goodbye HTTP Request in helloFlow.
48. In the Request section, select the Query Parameters tab.
49. Change the value of fullName to the value of the fname query parameter.

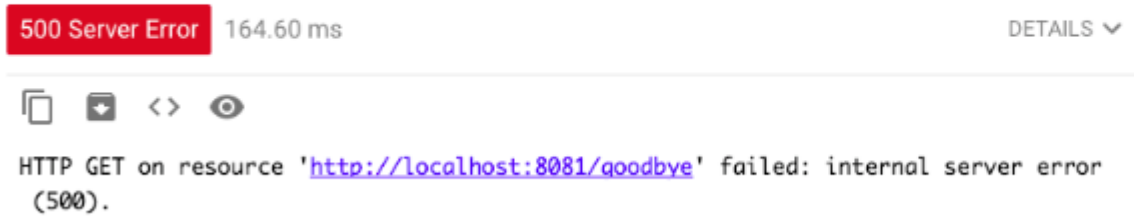


50. Click Apply Changes to redeploy the application.
51. In Advanced REST Client, send the same request; you should now see the name of the fname query parameter displayed.



Make a request and do not send a query parameter

52. Remove the query parameters and make a request to http://localhost:8081/hello; you should get an error.

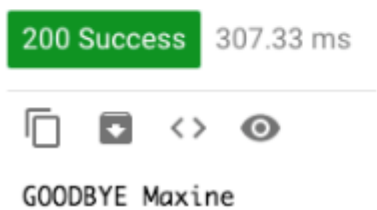


Set a default value in an expression

53. Return to the Anypoint Studio and navigate to the properties view for the Goodbye Set Payload in goodbyeFlow.
54. Remove as String.
55. Use the default operator to add a default value of Maxine.

```
#[upper('Goodbye') ++ ' ' ++ (attributes.queryParams.fullName default 'Maxine')]
```

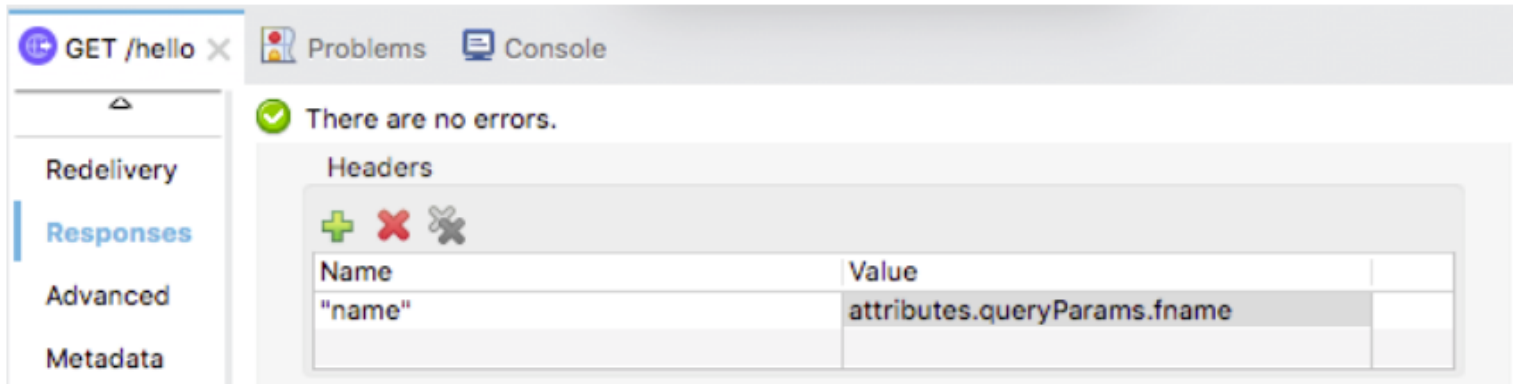
56. Click Apply Changes to redeploy the application.
57. In Advanced REST Client, send the same request; you should now see the default value Maxine displayed.



Use a query parameter in the expression for a response header

58. Return to the Anypoint Studio and navigate to the properties view for the GET /hello HTTP Listener.
59. Select the Responses tab.
60. Change the name header value to the value of the fname query parameter.

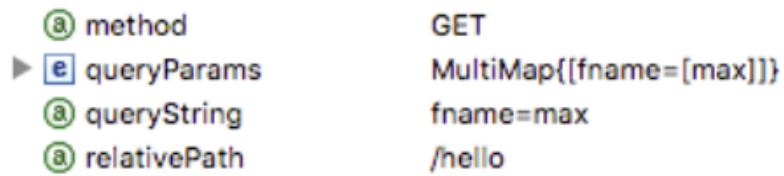
    `attributes.queryParams.fname`



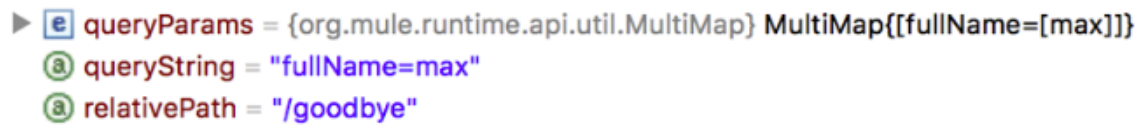
61. Click Apply Changes to redeploy the application.
62. In Advanced REST Client, add a fname and set it equal to max or some other value and send the request.
63. Look at the response headers; you should no longer see a name header.

Debug and verify the query parameter is sent with the request

64. Return to Anypoint Studio.
65. Stop and then debug the project.
66. Return to Advanced REST Client and send the same request.
67. Return to the Mule Debugger and look at the attributes and query parameters; you should see the fname query parameter.



68. Step into goodbyeFlow and look at the attributes and query parameters; you should see the fullName query parameter.



69. Step back to helloFlow and look at the value of the attributes; you should not see any query parameters.

70. Step through the rest of the application.
71. Switch to the Mule Design perspective.

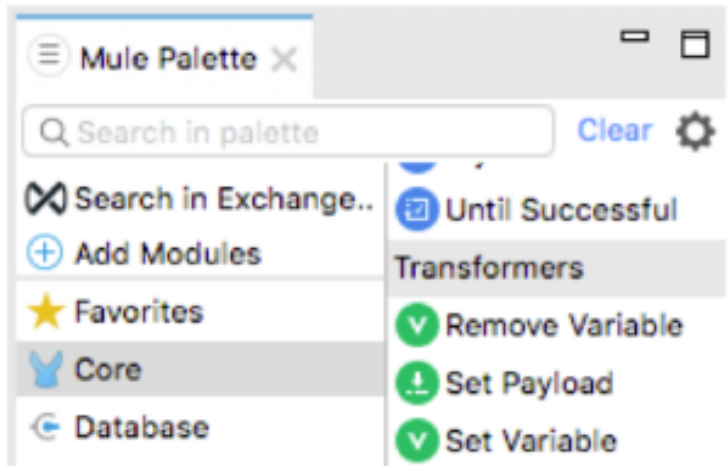
Creating variables

Creating variables

- Create variables to store metadata for the Mule event
- Use the Set Variable transformer to create a variable

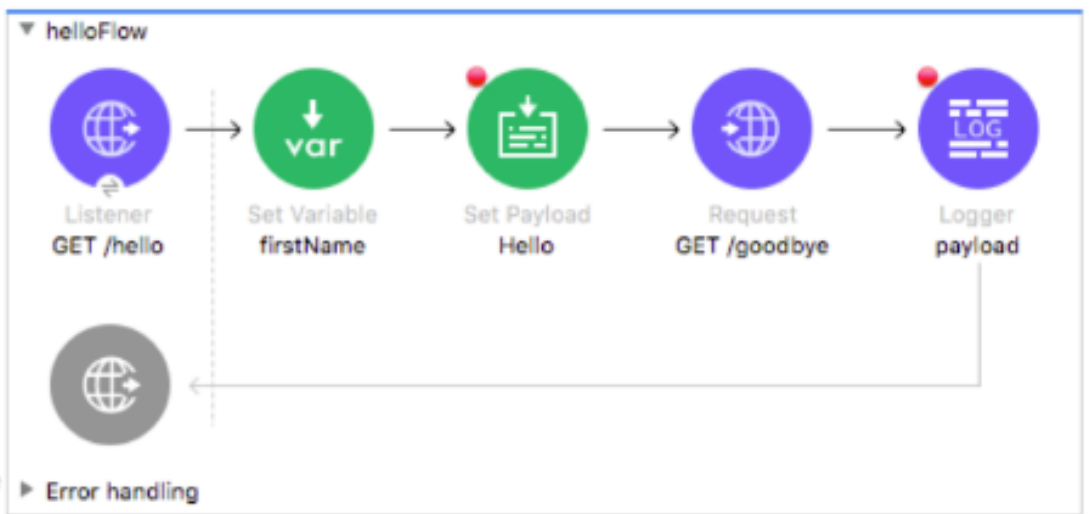
Set Variable

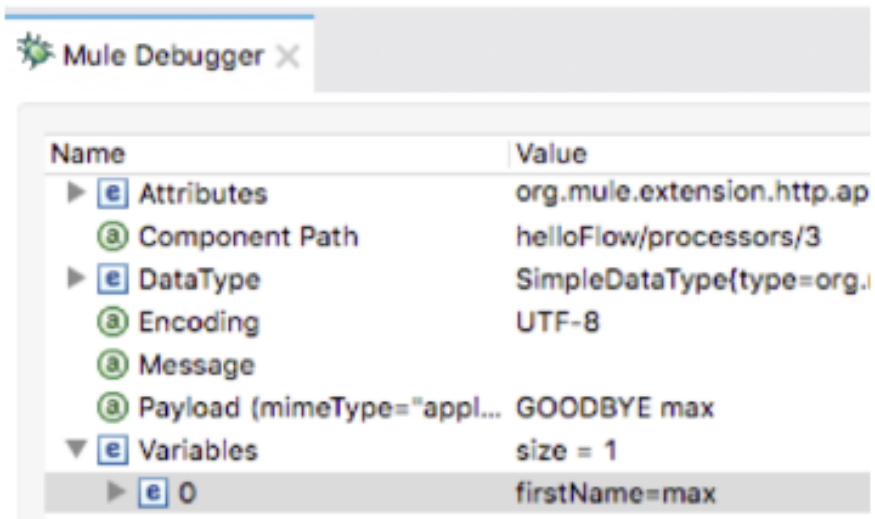
- In expressions, reference as vars
  - #[vars.foo]



Walkthrough 6-6: Set and get variables

- Use the Set Variable transformer to create a variable
- Reference a variable in a DataWeave expression
- Use a variable to dynamically set a response header
- Use the Mule Debugger to see the value of a variable
- Track variables across a transport boundary

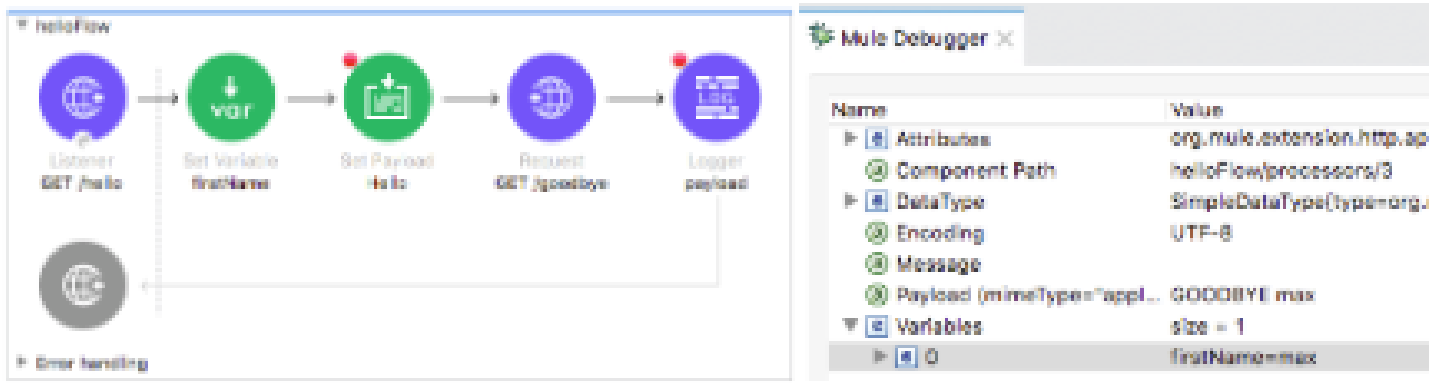




Walkthrough 6-6: Set and get variables

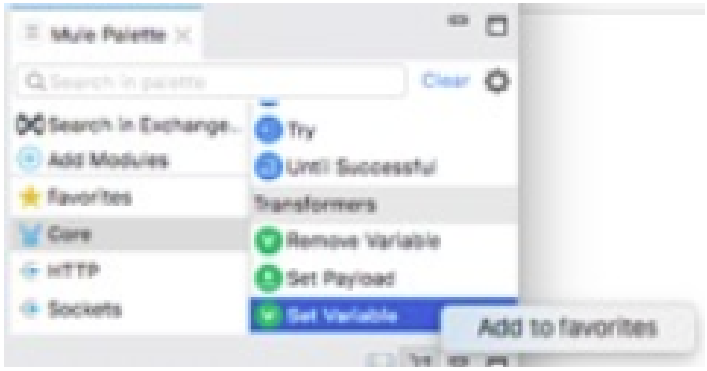
In this walkthrough, you create variables associated with an event. You will:

- Use the Set Variable transformer to create a variable.
- Reference a variable in a DataWeave expression.
- Use a variable to dynamically set a response header.
- Use the Mule Debugger to see the value of a variable.
- Track variables across a transport boundary.



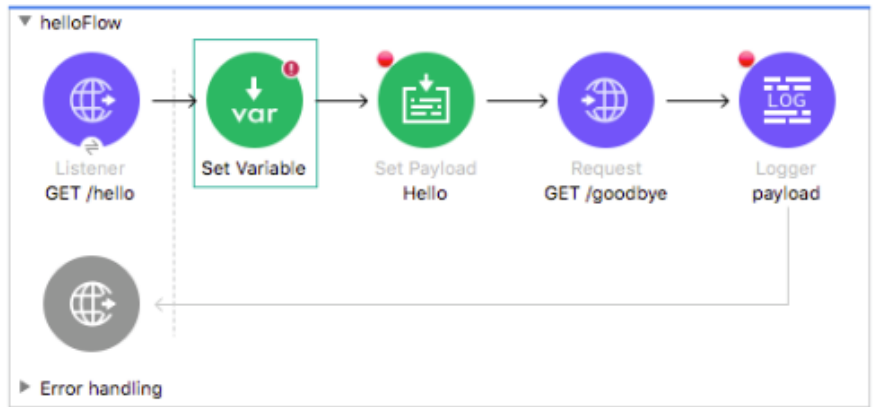
Add the Set Variable transformer to the Favorites section of the Mule Palette

1. Return to apdev-examples.xml.
2. In the Mule Palette, select Core.
3. Locate the Set Variable transformer and right-click it and select Add to favorites.

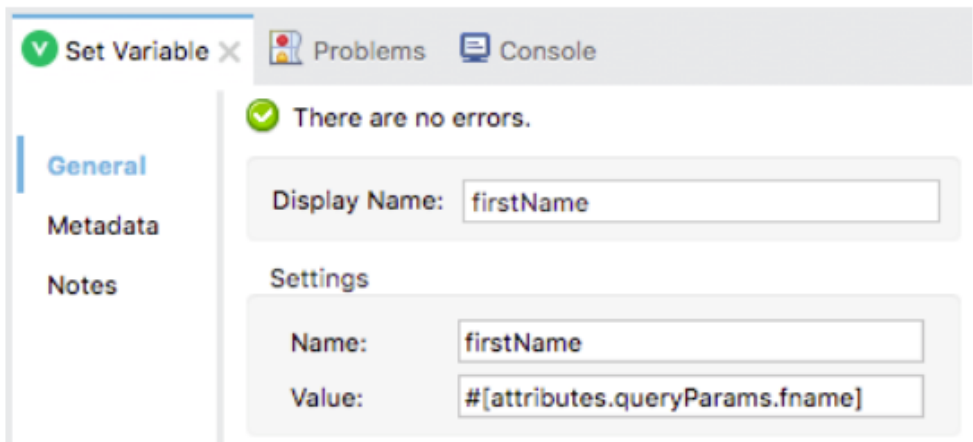


Create a variable

4. Drag the Set Variable transformer from the Favorites section of the Mule Palette and drop it after the GET /hello HTTP Listener in helloFlow.



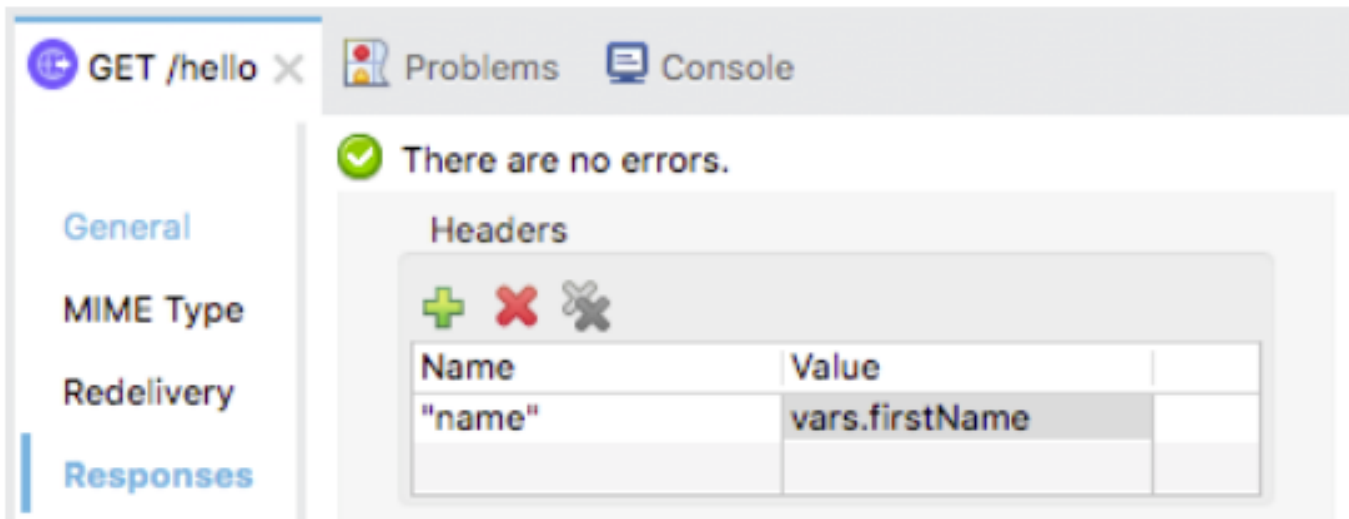
5. In the Set Variable properties view, set the display name and name to firstName.
6. Set the value to your query parameter, fname.



Use an expression that references the variable to set a response header

7. Return to the properties view for the GET /hello HTTP Listener in helloFlow.
8. Select the Responses tab.
9. Change the name header value from attributes.queryParams.fname to the value of the firstName variable.

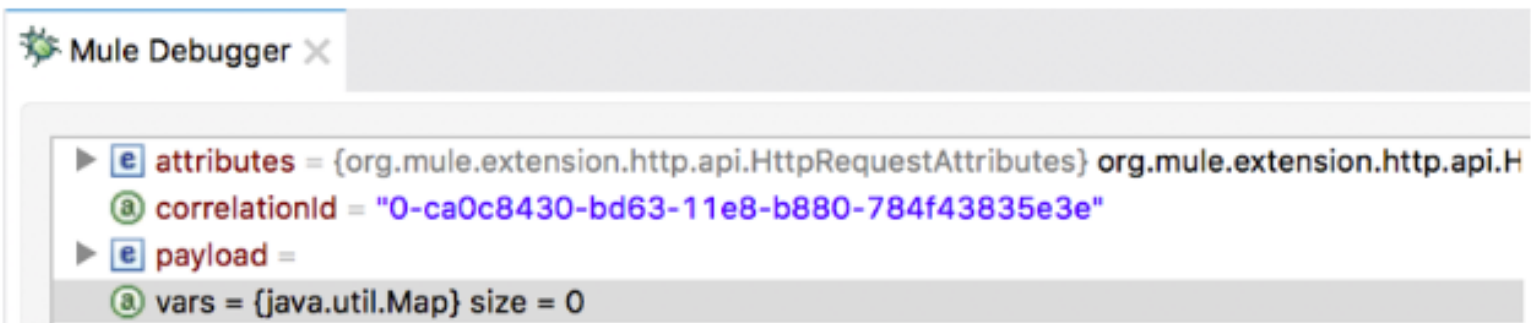
```
vars.firstName
```



View variables in the Mule Debugger

10. Save the file to redeploy the project in debug mode.
11. In Advanced REST Client, send the same request.
12. In the Mule Debugger, locate and expand the new Variables section; you should see your firstName variable.

13. Step into goodbyeFlow; you should see no variables in the vars section.

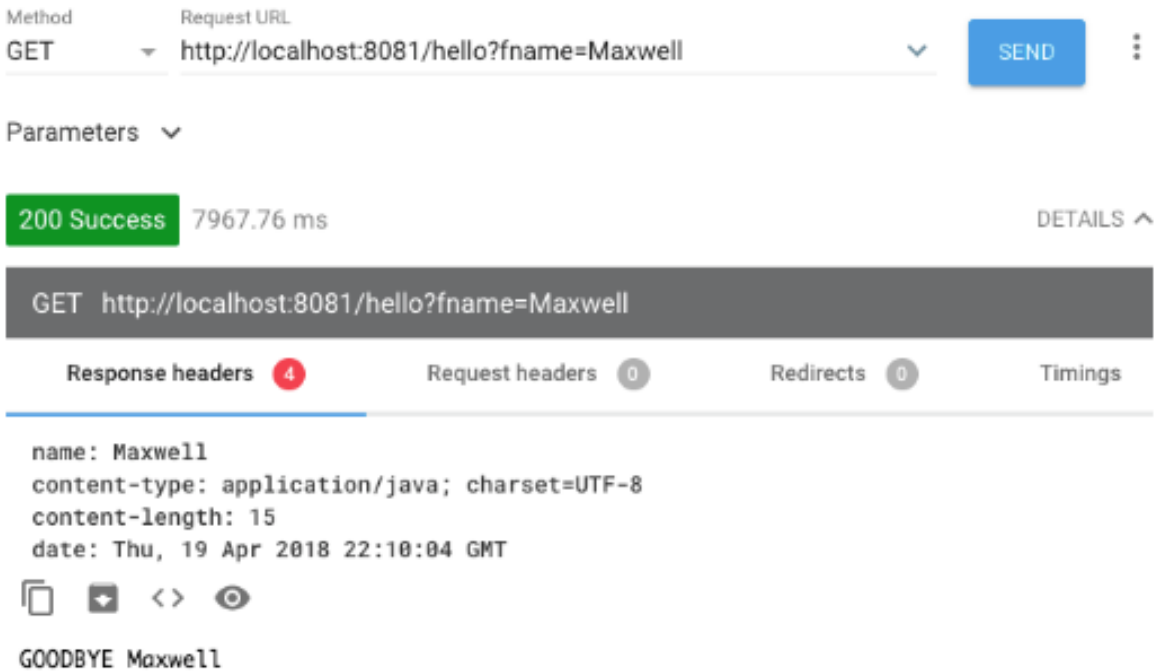


14. Step back to helloFlow; you should see the Variables section again with your firstName variable.

15. Step through the rest of the application.
16. Return to Advanced REST Client; you should see the header with the value of the query parameter.

17. Change the value of the query parameter and send another request.
18. In the Mule Debugger, click the Resume button until you step through the application.

19. Return to Advanced REST client; you should see the new query parameter value returned in both the body and the header.



20. Return to Anypoint Studio and switch perspectives.

Summary

Summary

- The best way to view event data is to add breakpoints to a flow and use the **Mule Debugger**
- Use the **Logger** component to display data in the console
- Use the **Set Payload** transformer to set the payload
- Use the properties view to set response data for an HTTP Listener and request data for an HTTP Request operation
- Use the **DataWeave language** to write inline expressions in **#[]**
- Use the **Set Variable** transformer to create variables

DIY Exercise 6-1: Troubleshoot and refactor a Mule application

Time estimate: 2 hours

Objectives

In this exercise, you troubleshoot and refactor a Mule application. You will:

- Diagnose and repair common Mule event-related, RAML-related, and connector-related issues.
- Organize and refactor Mule flows.

Scenario

Your colleagues need assistance; they have modified the Accounts API specification to include a new salesID field and broken the Accounts API implementation when trying to update it. Review their changes and then troubleshoot and fix the API implementation.

Import the starting project

Import /files/module06/accounts-mod06-debugging-starter.jar (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources) into Anypoint Studio.

Note: This Mule project file has errors specifically designed to help you learn to diagnose common errors. Do not use your previous solution.

Review the modified API in the starter project

Review the updated files in the api folder located in the accounts-mod06-debugging-starter project. Look for any issues with the updated RAML API. Compare the RAML data type files with the examples and identify any differences that might be causing errors.

Review the existing (broken) implementation

Review implementation.xml in the accounts-mod06-debugging-starter project. Try to identify any issues before you debug the Mule application.

Troubleshoot and fix the Mule application

Debug the Mule application and fix all the errors that prevent it from starting up.

Hint: The project references a flow that is in another Mule application called templates. You will need to import the template application as a dependency into your local m2 repository. For help, follow the step-by-step Install a Mule application as a dependency walkthrough.

Troubleshoot and fix database connectivity errors

Verify the database driver and connection information is correct. If you get database connectivity errors, you can use the following MySQL database credentials:

- Host: mudb.learn.mulesoft.com
- Port: 3306
- User: mule
- Password: mule
- Database: training
- Table: flights_customers

Here is the expected output from the Mule application where salesId is a concatenation of id, first name, last name, postal, and creationDate:

```
[
 {
  "salesId": "100Alice Green941082018-10-01T23:57:59Z+0:00"
  "id": "100"
  "firstName": "Alice",
  "lastName": "Green",
  "address": "77 Geary St., San Francisco"
  "postal": "94108"
  "country": "USA",
  "creationDate": "2018-10-01T23:57:59Z+0:00",
  "accountType": "business",
  "miles": 1000
 }
]
```

Refactor the Mule application to use parameterized queries

Refactor the Mule application so that the SQL statements refer to parameterized queries instead of directly inserting query parameters in the database components.

Test the APIkit Console and make sure it works

Make sure the APIkit Console's view is able to load without any errors and that you can use it to make a GET request to /accounts.

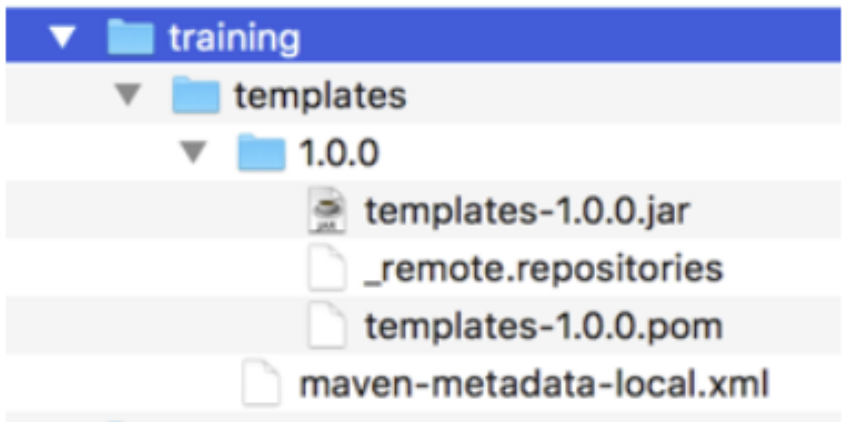
Verify your solution

Load the solution /files/module06/accounts-mod06-debugging-solution.jar (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources) and compare your solution.

DIY Walkthrough: Install a Mule application as a Maven dependency

In this walkthrough, you reuse the flows of one Mule application in another Mule application. First, you install a (reusable) Mule application into your local Maven repository folder, then you refer to the installed reusable Mule application's flows from another (main) Mule application. You will:

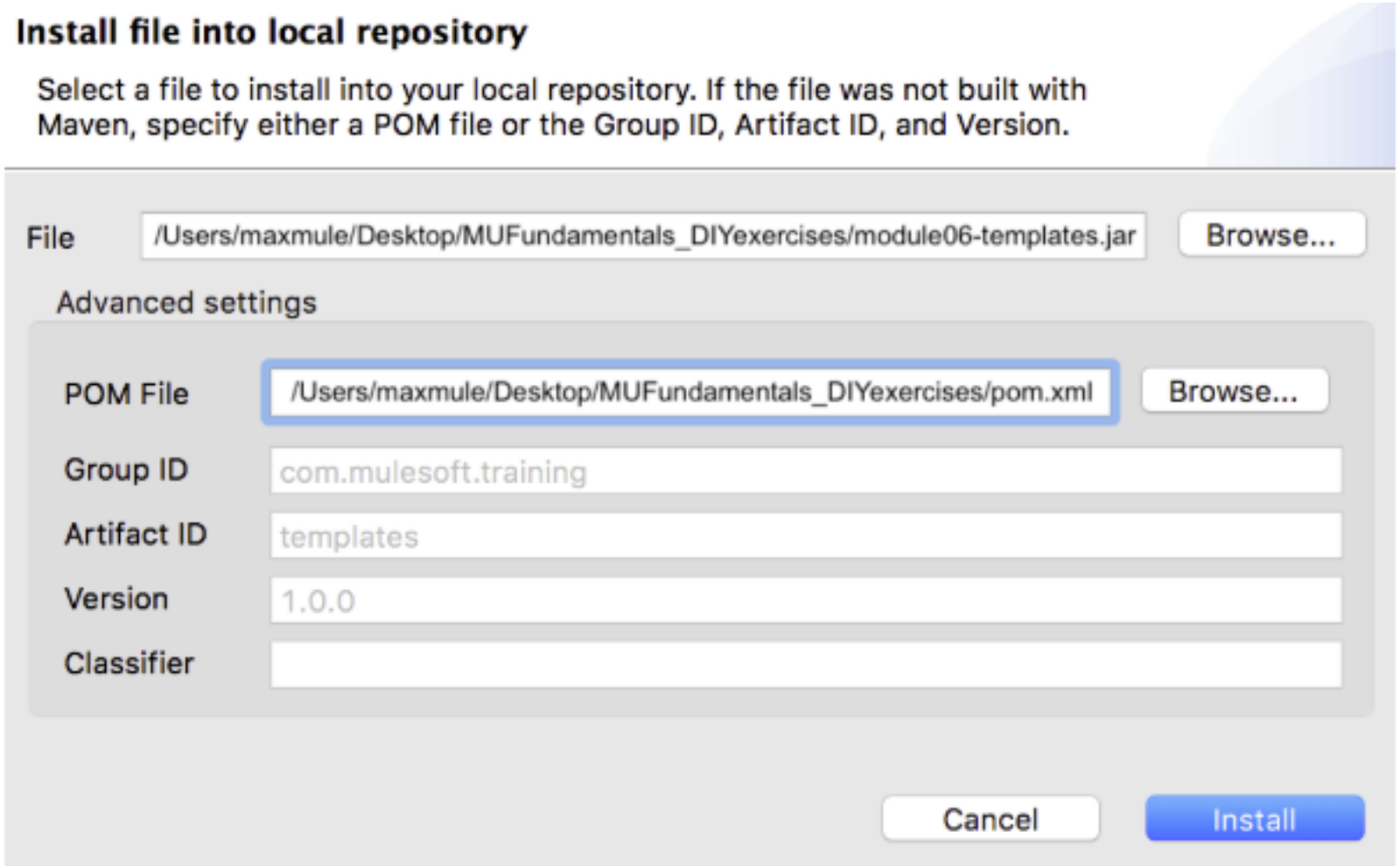
- Install a reusable Mule application's jar file into a local Maven repository m2 folder.
- Add the reusable Mule application dependency to the main Mule application's pom.xml file.
- Verify the installation of the reusable Mule application into the main Mule application.
- Call a flow of the installed reusable Mule application from the main Mule application.



Install a Mule application into the local Maven repository

1. In Anypoint Studio, create a new Mule project named main.
2. Click the Maven icon in the upper-left corner of Anypoint Studio.

3. In the Install file into local repository dialog box, click Browse for the POM file.
4. In the file dialog box, browse to the pom.xml file located in the /files/module06/templates_app/ folder located in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources.
5. Click Open.
6. In the Install file into local repository dialog box, click Browse for the file.
7. In the file dialog box, select the module06-templates.jar file in the /files/module06/templates_app/ folder.
8. Click Open.
9. In the Install file into local repository dialog box, verify that the group ID, artifact ID, and version values are set, but the classifier is empty.



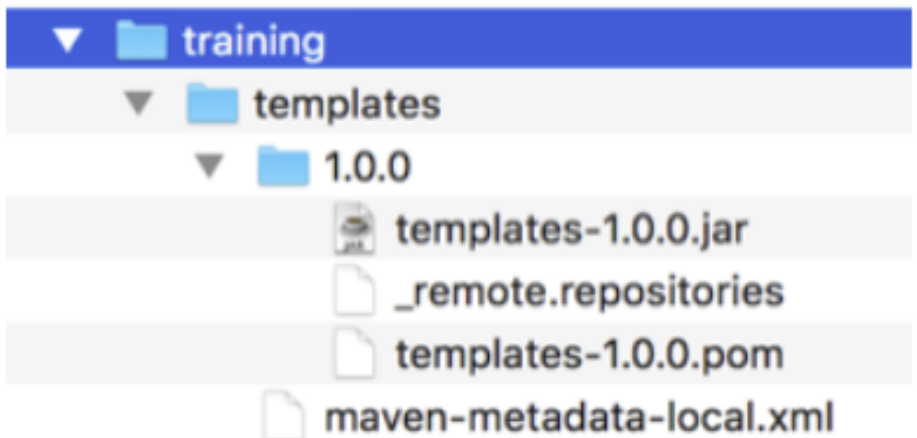
10. Click Install.
11. In the Install file into local repository dialog box, click OK.

Verify the Mule application is installed in your local Maven repository m2 folder

12. In the computer's file explorer, navigate to your user directory, which is often located at /Users/{*your_user_name*}.
13. In your user home directory, navigate to:

    `.m2/repository/com/mulesoft/training/templates`

14. Verify that the following files are present in the templates folder with the exact names listed below.



Note: The reusable Mule application is called templates and its flows are stored in the templates-1.0.0.jar file.

Add a Mule application dependency to another Mule application's pom.xml file

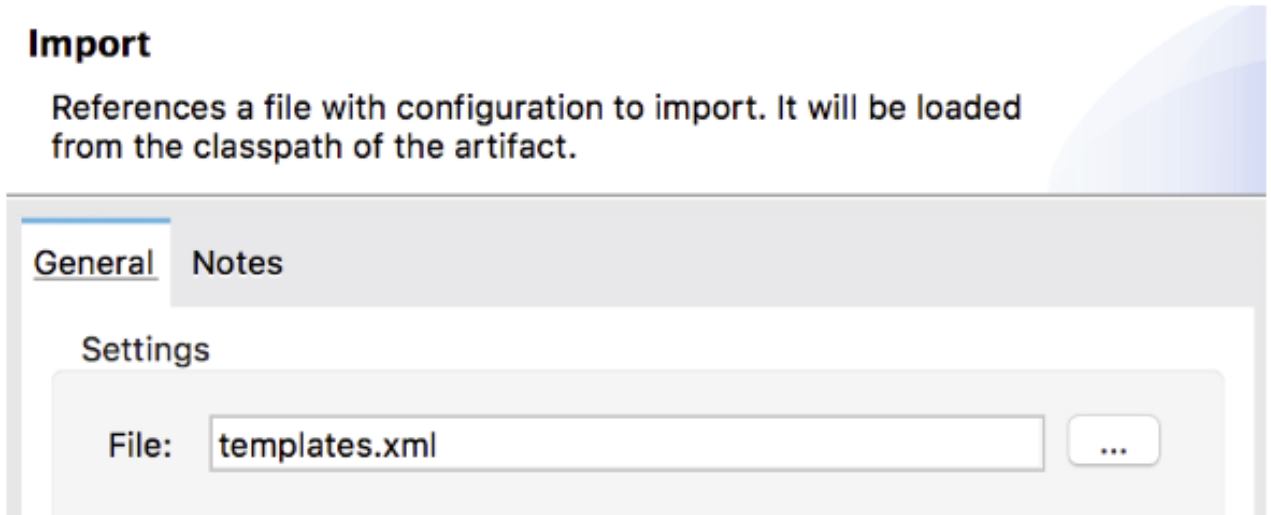
15. Return to Anypoint Studio.
16. From the Package Explorer, open the main Mule application's pom.xml.
17. Select the Source tab at the bottom of pom.xml.
18. In the dependencies section of pom.xml, add a new dependency to refer to the templates artifact defined in the templates Mule application's pom.xml file.

```
...
  <dependency>
    <groupId>com.mulesoft.training</groupId>
    <artifactId>templates</artifactId>
    <version>1.0.0</version>
  </dependency>
</dependencies>
```

19. Save pom.xml.

Import the templates.xml file contained in the templates Mule application

20. Return to main.xml.
21. Select the Global Elements tab at the bottom of the canvas.
22. Click Create.
23. In the Choose Global Type dialog box, select Global Configurations > Import.
24. Click OK.
25. In the Import dialog box, set the file to templates.xml.



26. Click OK.

Note: The templates.xml file is the Mule configuration file for the templates Mule application that contains flows that can be referenced from the main Mule application.

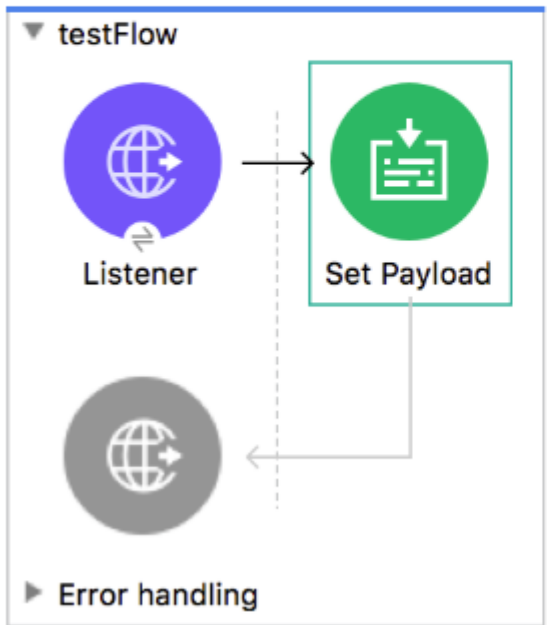
Create an HTTP Listener to receive requests

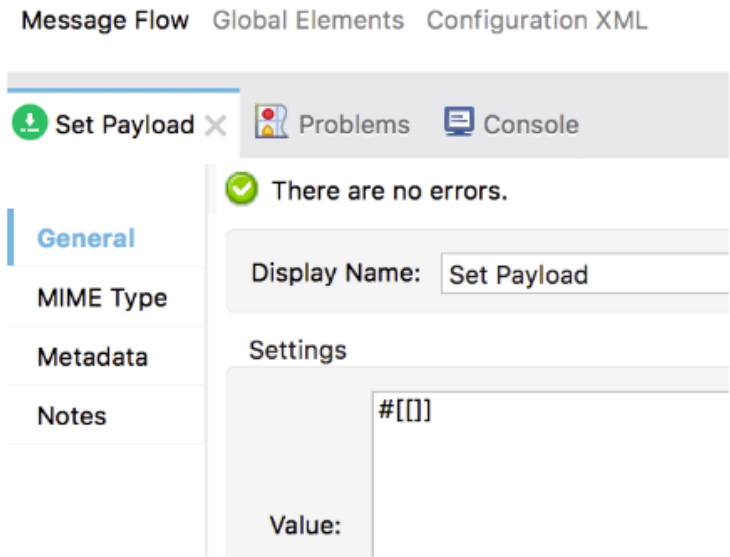
27. Return to the Message Flow view in main.xml.
28. Drag an HTTP Listener from the Mule Palette to the canvas.
29. In the Listener properties view, click the Add button next to Connector configuration.
30. In the Global Element Properties dialog box, set the host to 0.0.0.0 and the port to 8081.
31. Click OK.
32. In the Listener properties view, set the path to /test.

Set the payload to an empty array

33. Drag a Set Payload transformer from the Mule Palette into the process section of the flow.
34. In the Set Payload properties view, set the value to an empty array.

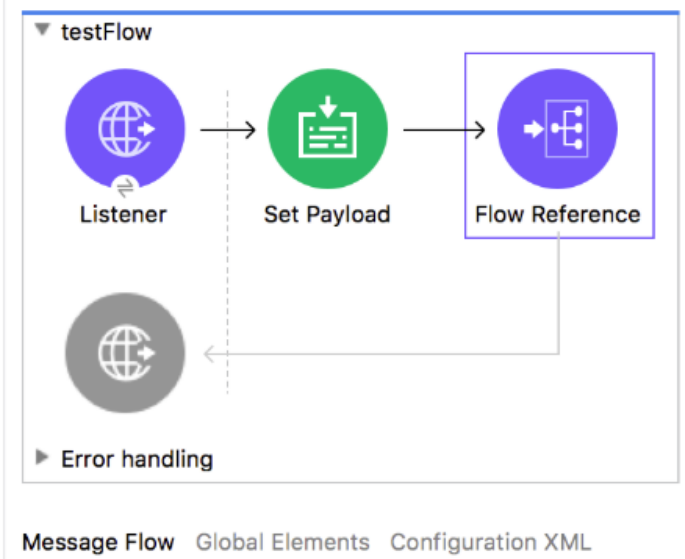
```
#[[]]
```

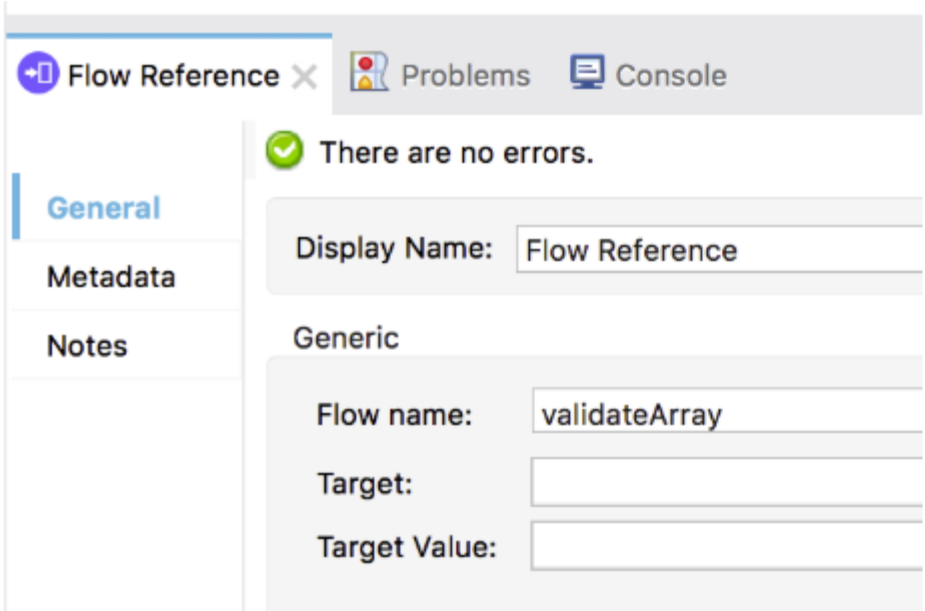




Call a flow in the templates Mule application from the main Mule application

35. Drag a Flow Reference component from the Mule Palette and drop it at the end of the flow.
36. In the Flow Reference properties view, set the flow name to validateArray.





Note: The validateArray flow is defined in templates.xml in the templates Mule application. The flow will print out an error message in the console if the payload is an empty array.

Test the application

37. Save the file and run the project.
38. In a web browser, navigate to http://localhost:8081/test.
39. Return to Anypoint Studio.
40. Verify a Payload is empty error message appears in the Console view.

```
ERROR 2018-08-20 18:05:01,679 [[MuleRuntime].cpuLight.05: [test].validateArray.CPU_LITE @dd267cf] [event
********************************************************************************
Message               : Payload is empty, which it should not be empty. Logging the incident.
Error type            : VALIDATION:EMPTY_COLLECTION
Element               : validateArray/processors/0 @ test:templates.xml:8 (Is not empty collection)
Element XML           : <validation:is-not-empty-collection doc:name="Is not empty collection" doc:id="f

  (set debug level logging or '-Dmule.verbose.exceptions=true' for everything)
********************************************************************************
```

41. Stop the project.