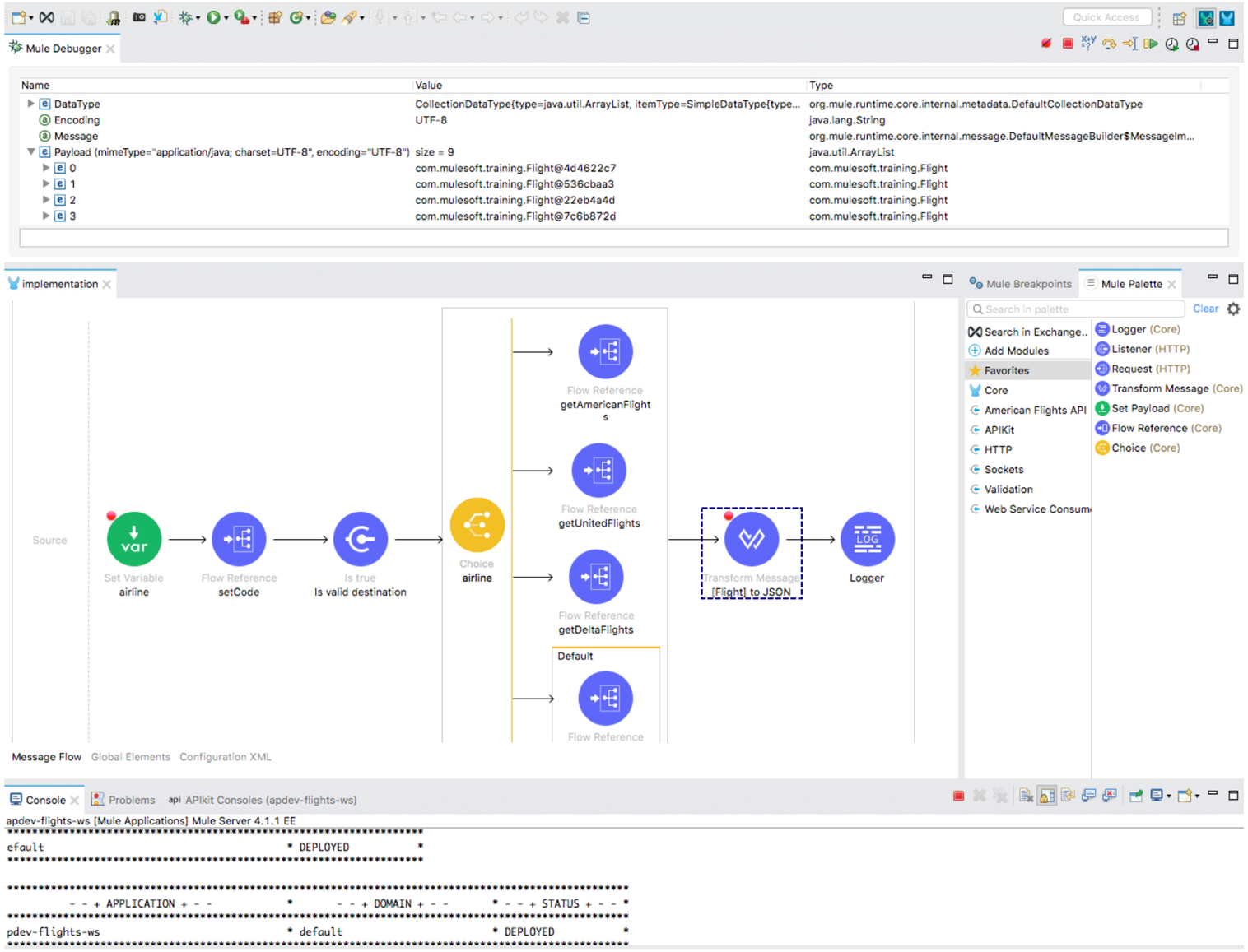
PART 2: Building Applications with Anypoint Studio

Goal

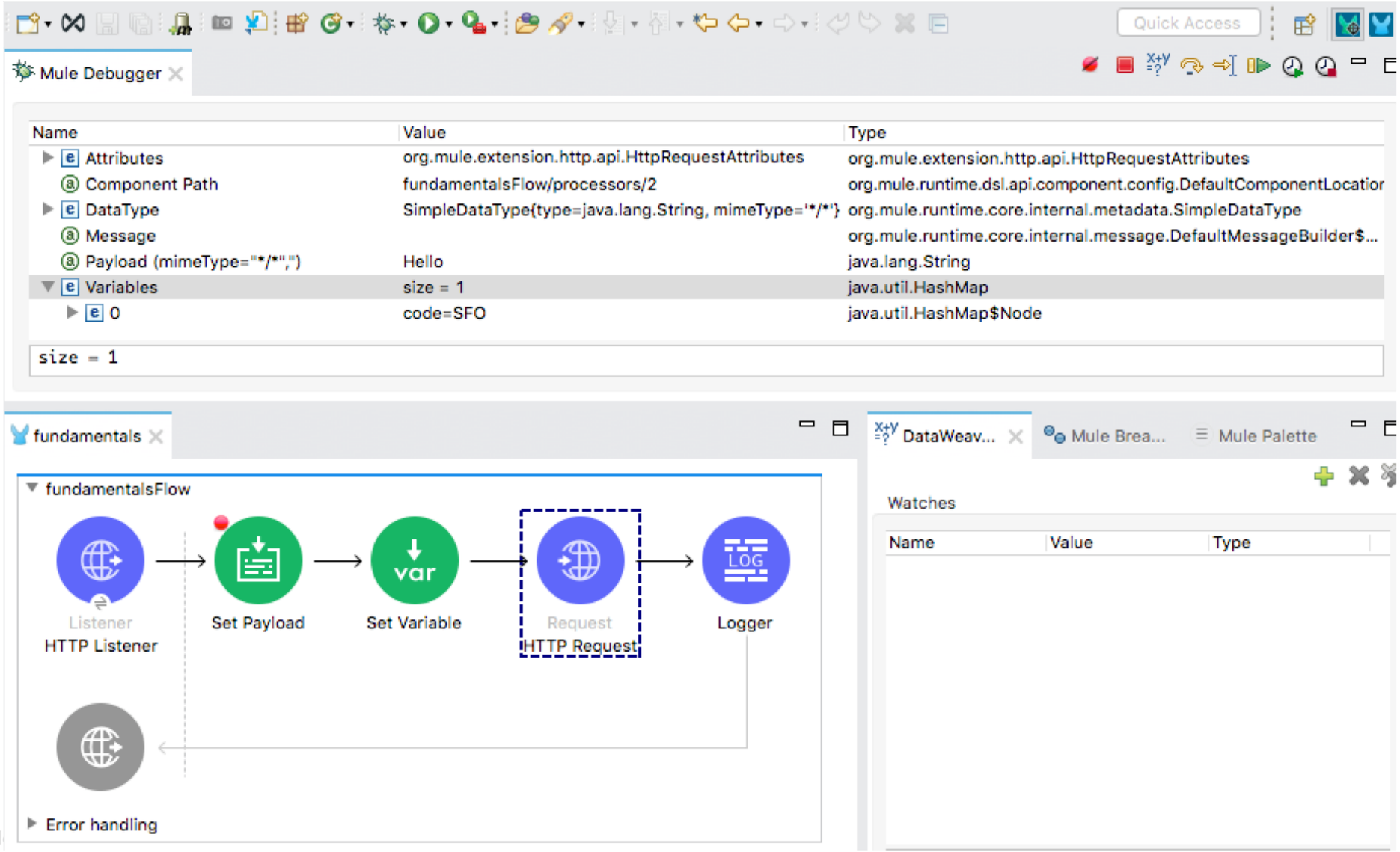


At the end of this part, you should be able to

- 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

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

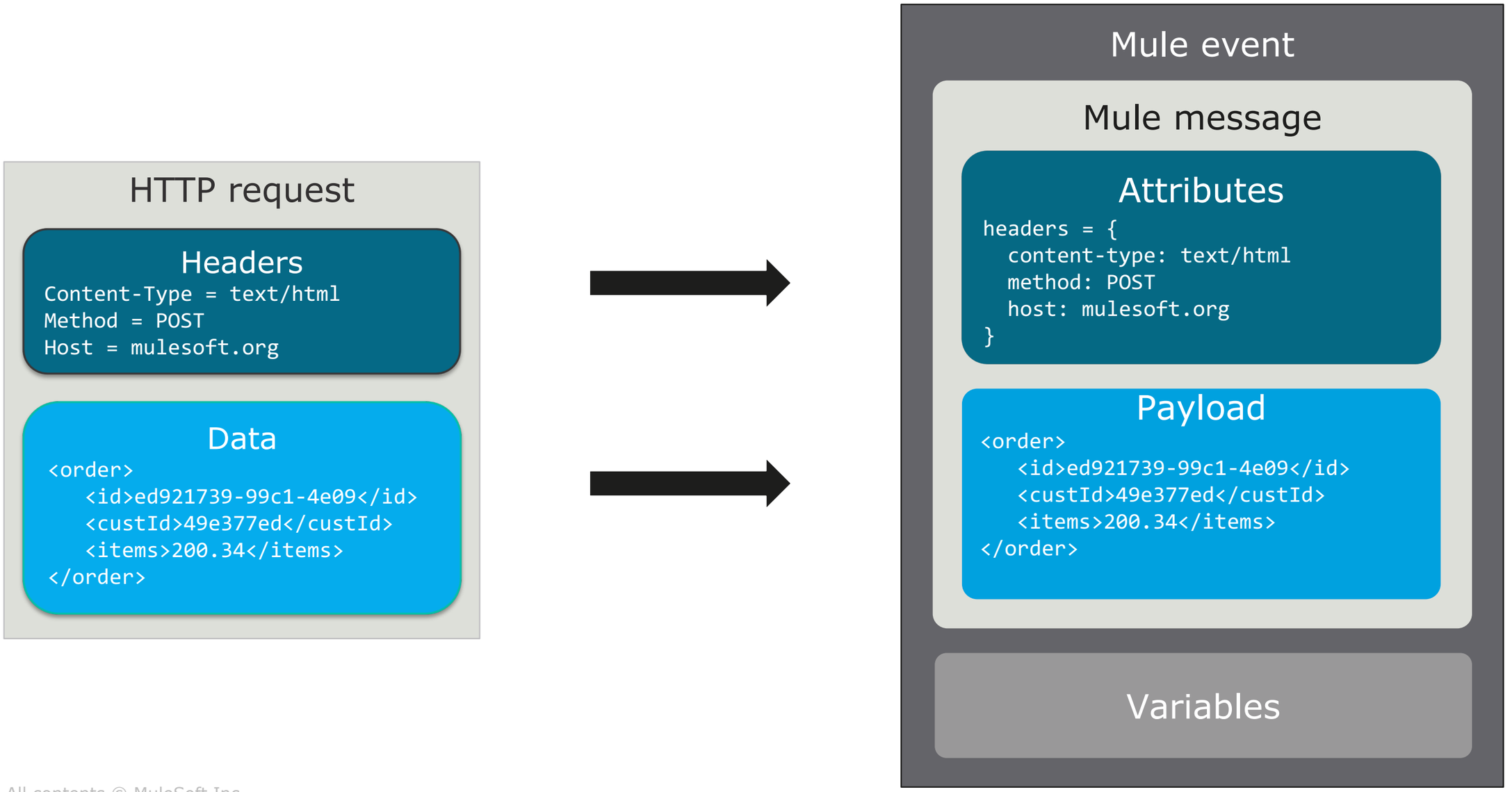
The data that passes through flows in the app

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



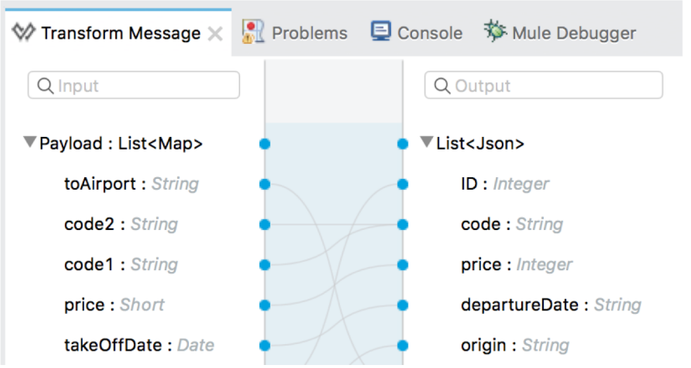
Viewing event data

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

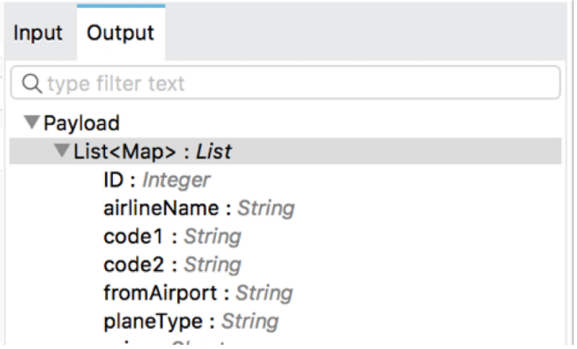
- At **run time**
  - What did we see so far in the course and where?

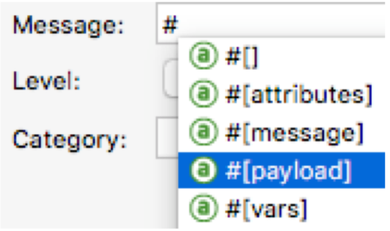
Viewing event data at design time

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



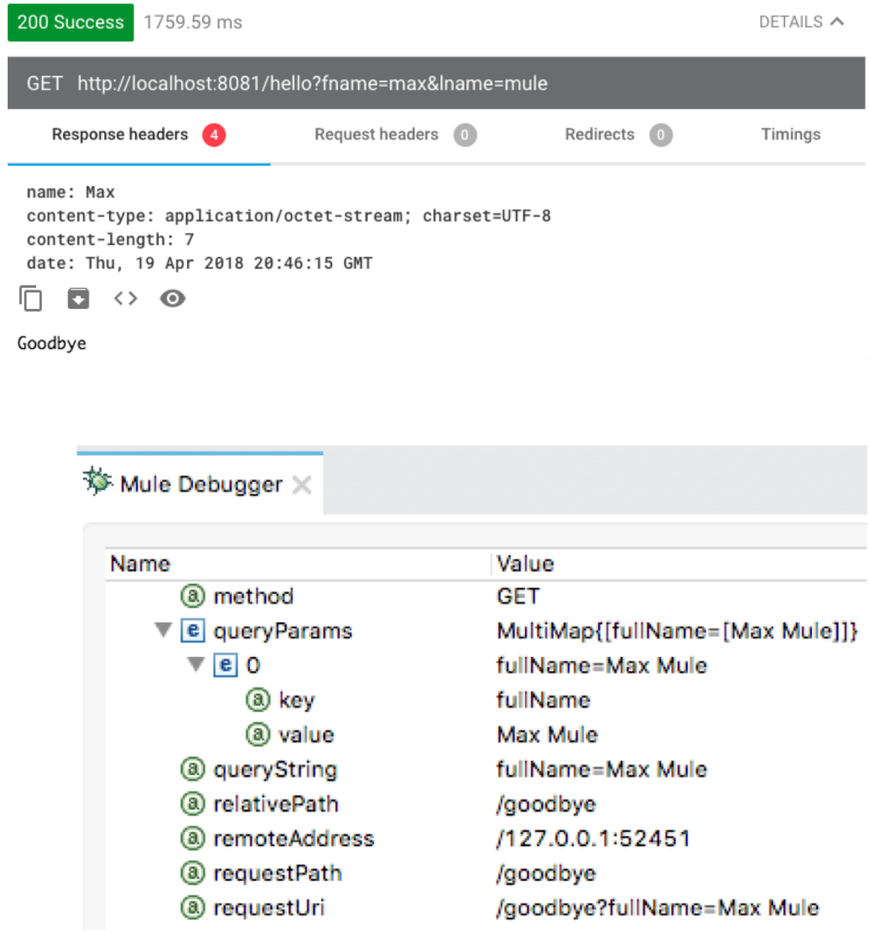
- **DataSense** is the ability to proactively discover metadata from internal and external resources
  - Keeps you from having to manually discover info about the data
  - Facilitates transformations by providing DataWeave expected input or output





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

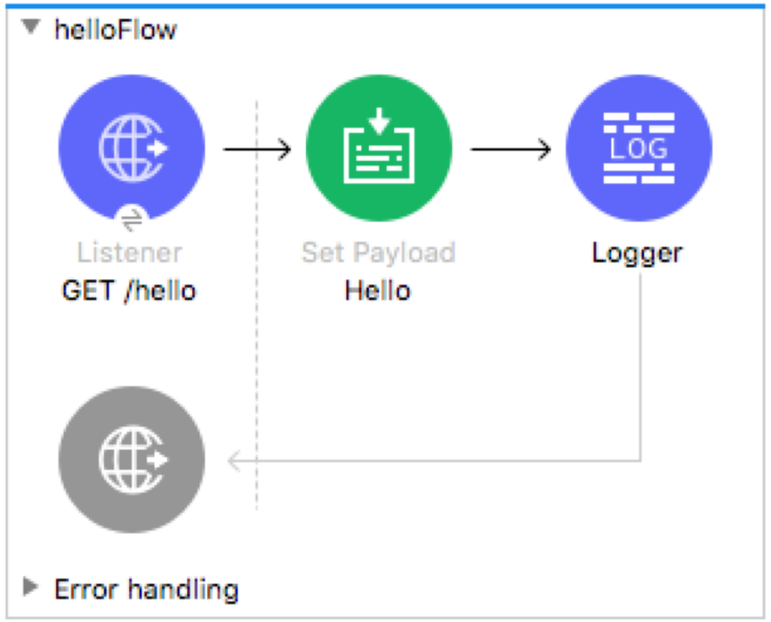
- Add a **Logger** component to a flow and view its output

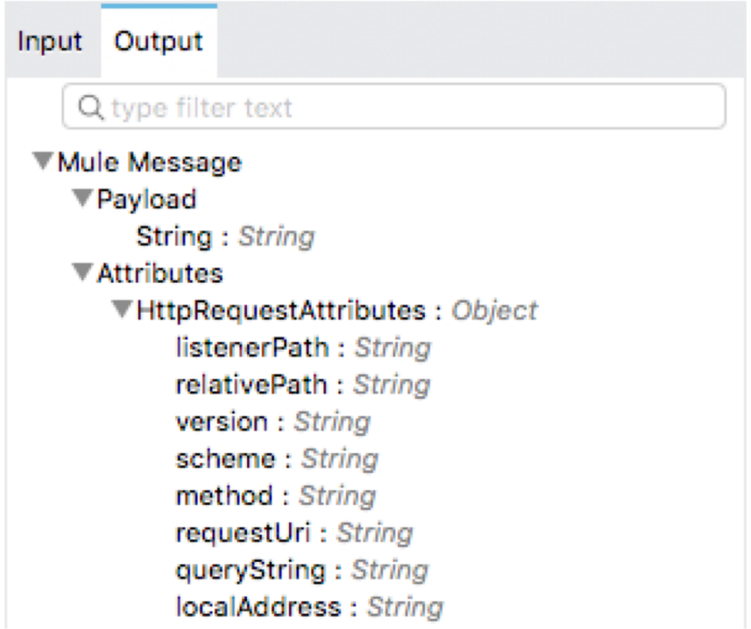
Logger

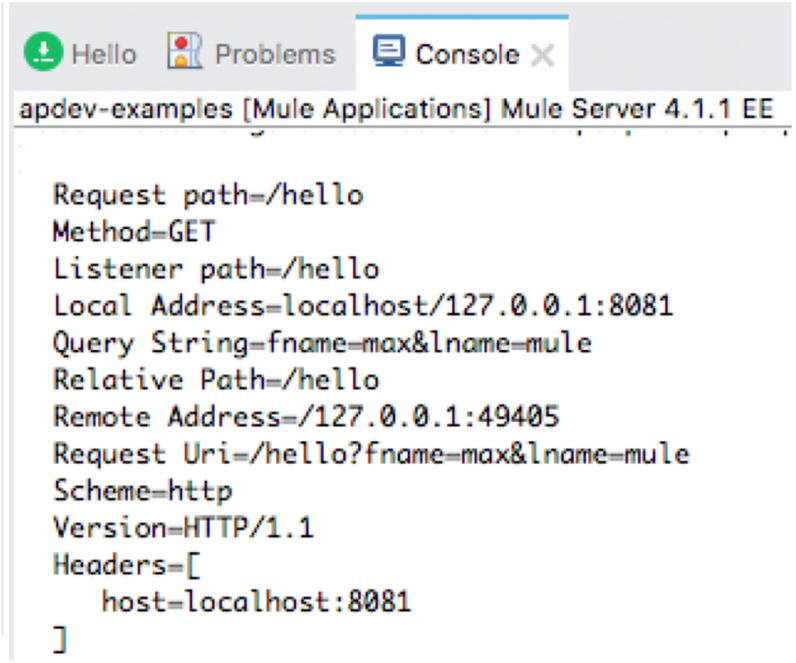
- Logged valued are displayed
  - For an application run from Anypoint Studio with embedded runtime, in the **Console view**
  - For applications deployed to CloudHub or customer-hosted runtimes, in the **log files**

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



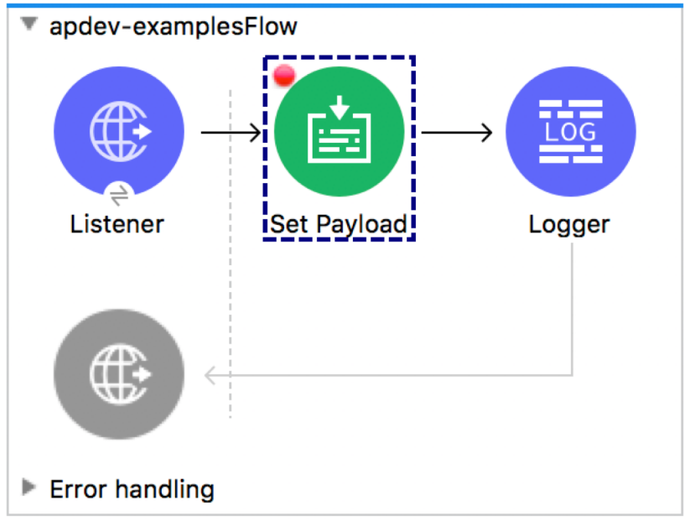




Debugging Mule applications

Debugging applications with the Mule Debugger

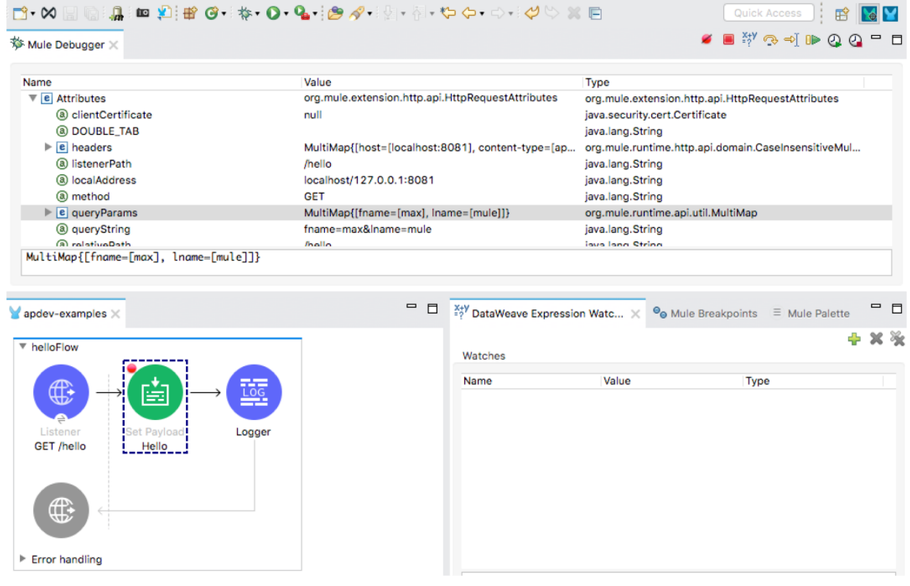
- Can add breakpoints to processors and step through the application
  - Watch event properties and values
  - Watch and evaluate DataWeave expressions



- 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

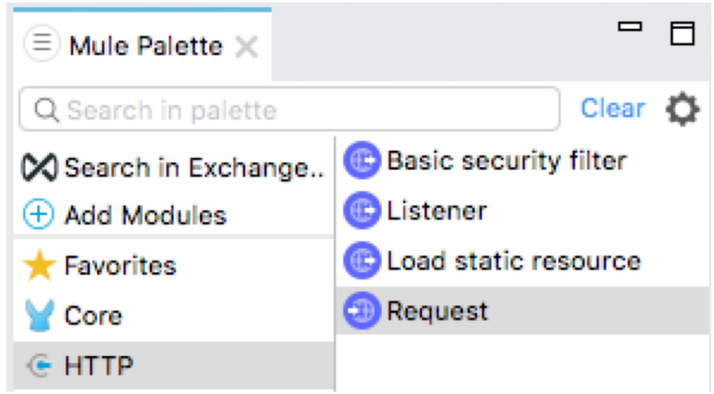


Tracking event data as it moves in and out of Mule applications

Changes to the event object

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

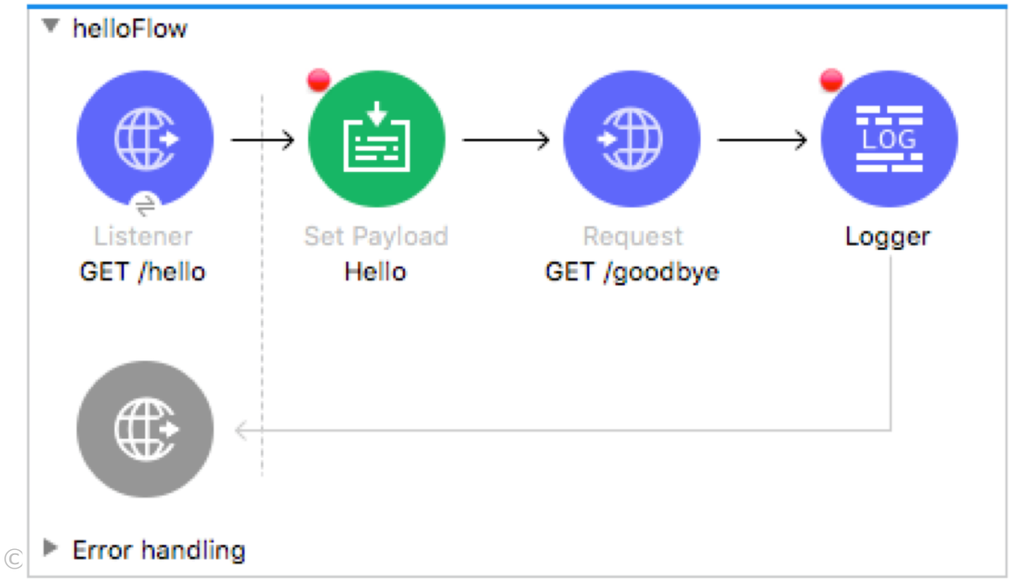
- What happens to the event object when calls are made to an external resource from a flow?
  - For example, you call a web service, and get return data?

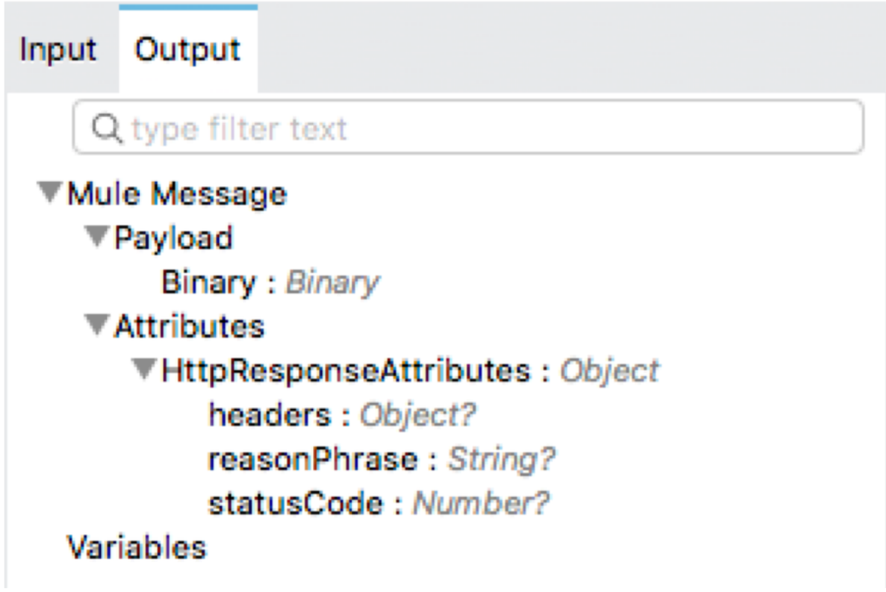


Walkthrough 6-3: Track event data as it moves in and out of a Mule application

- 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.*



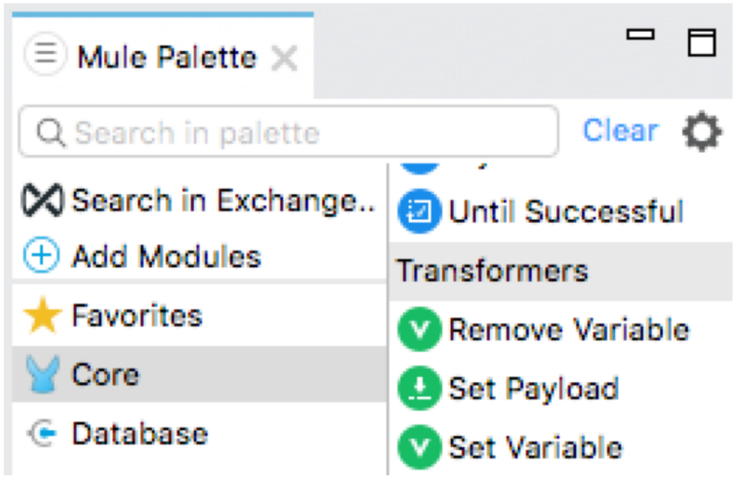


Setting request and response data

Setting the message payload

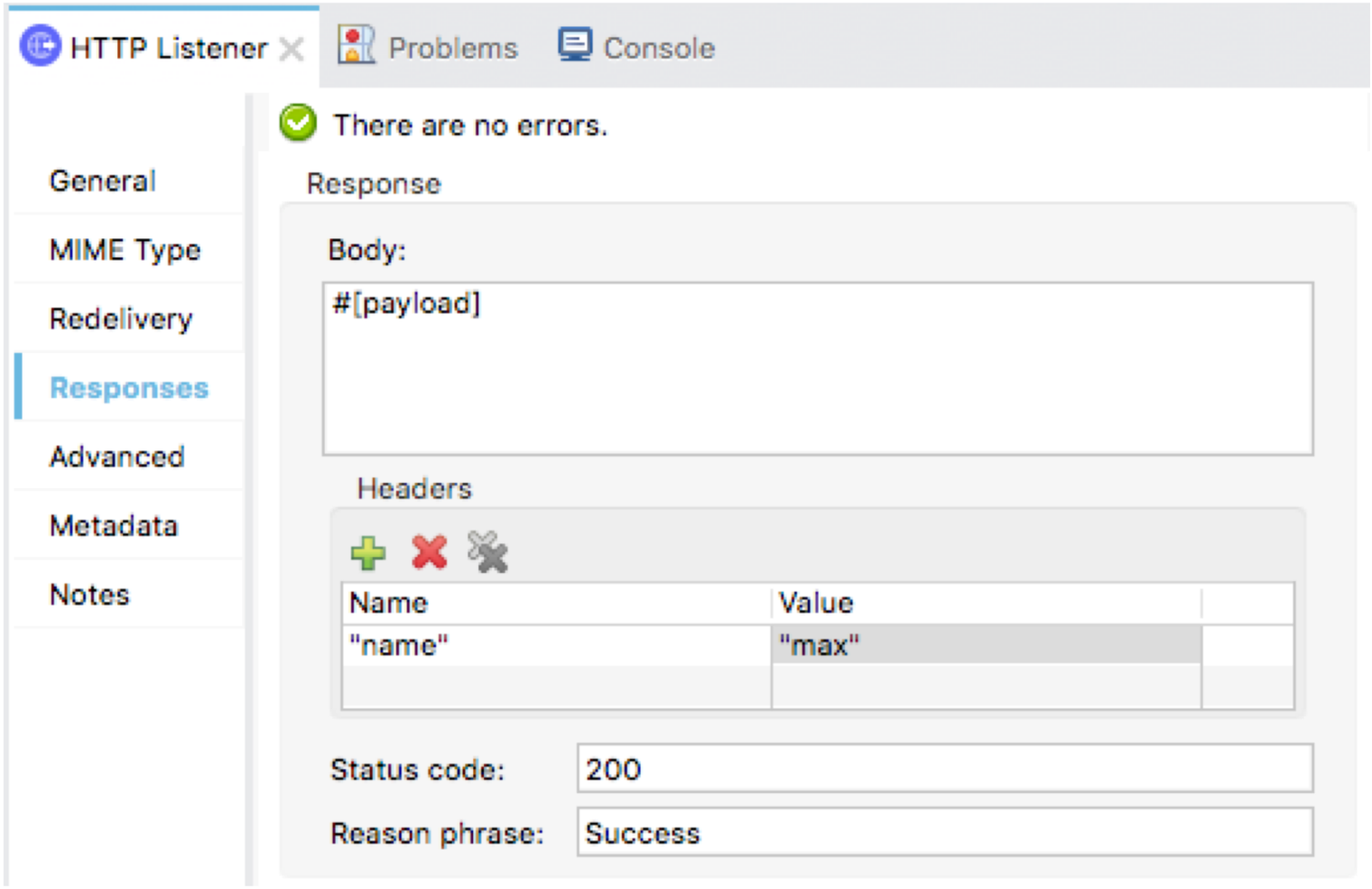
- Use the Set Payload transformer to set the payload

Set Payload



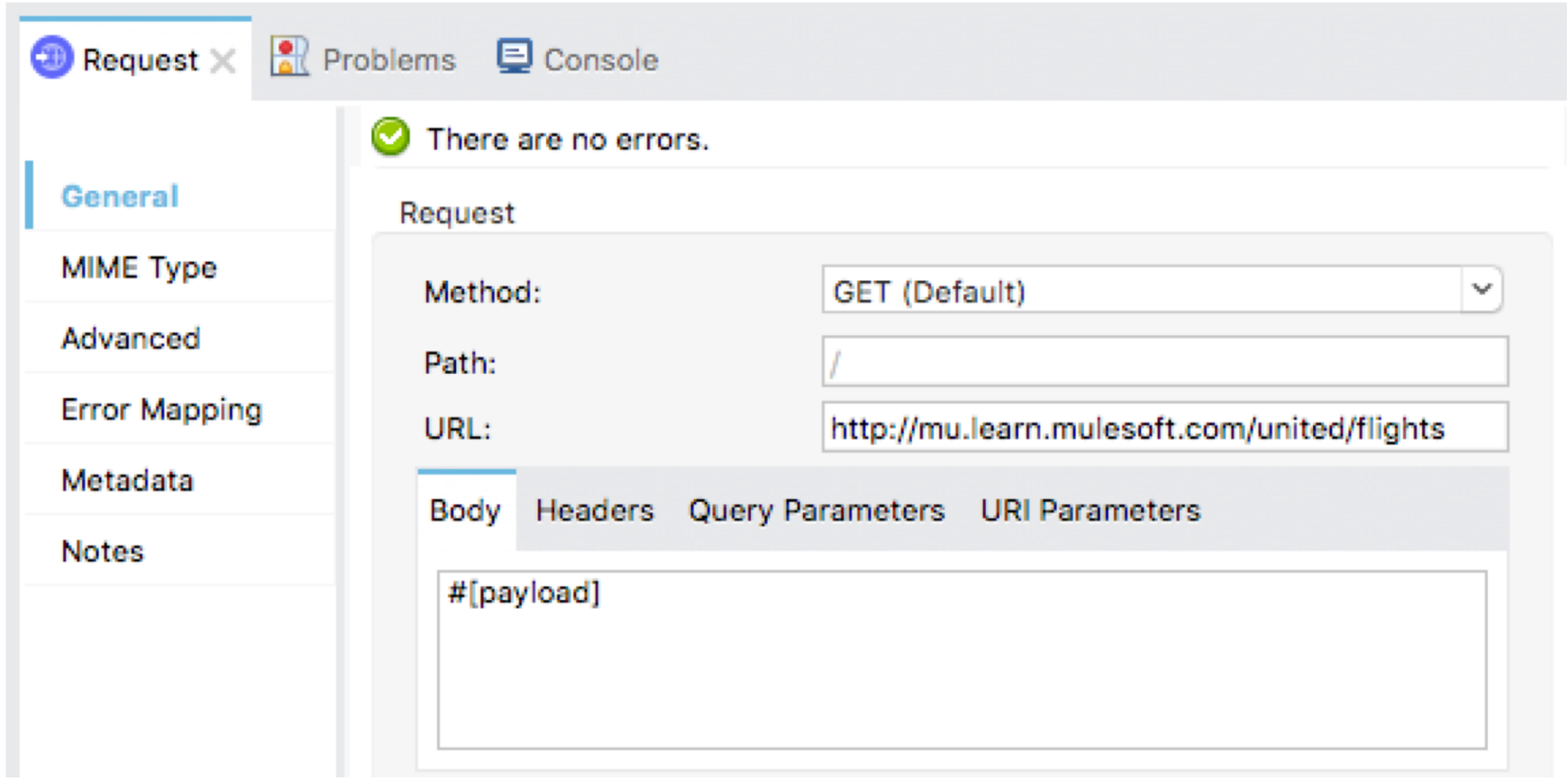
Setting data returned from HTTP listeners

- Set in the HTTP Listener properties view > Responses



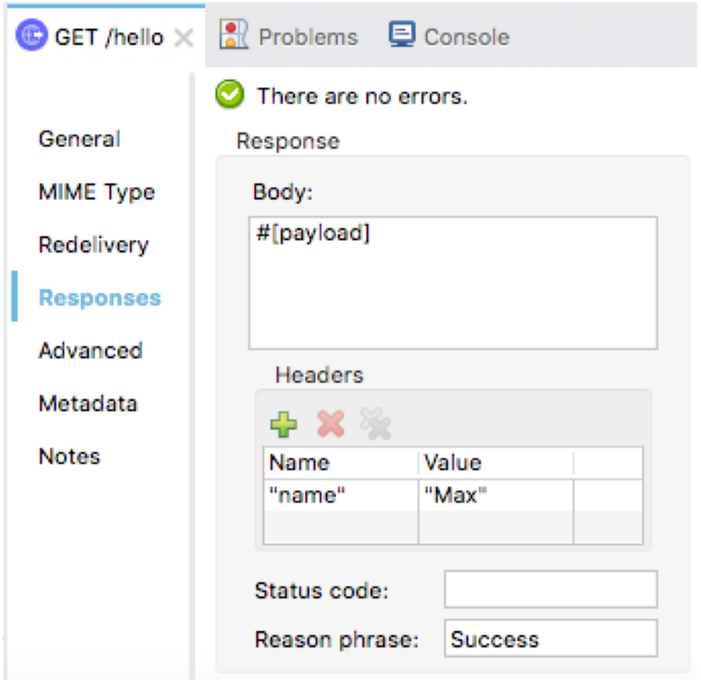
Setting data sent to HTTP requests

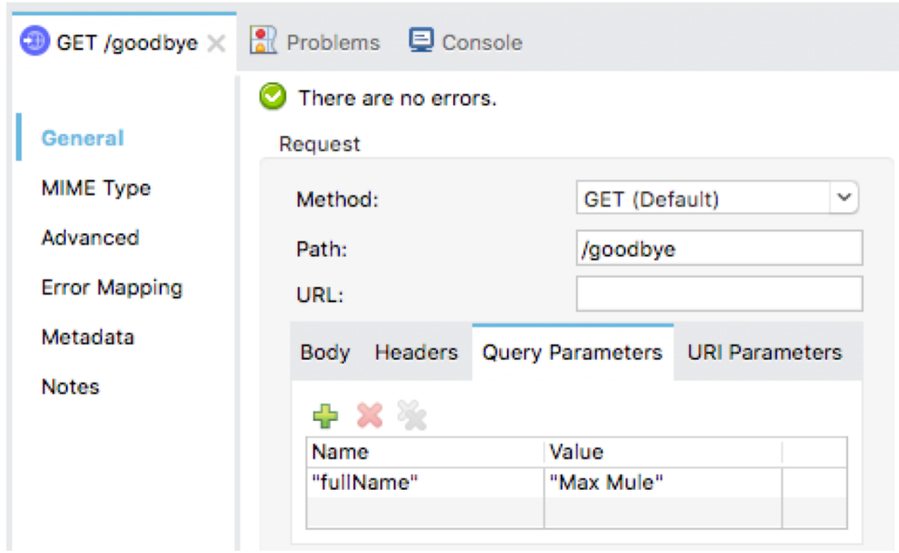
- Set in the HTTP Request properties view > General



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





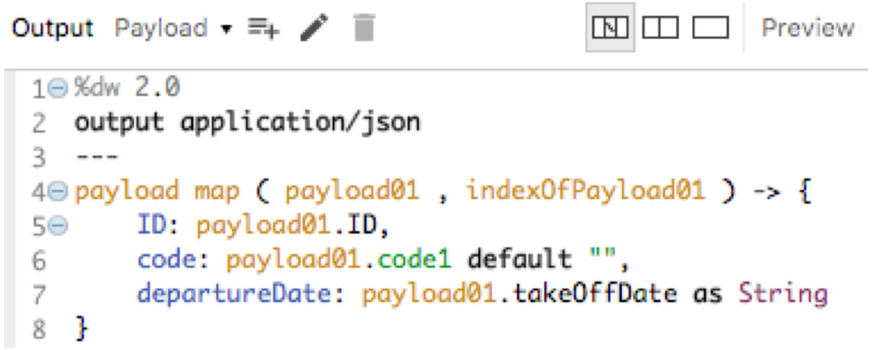
Using DataWeave expressions to read and write event data

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

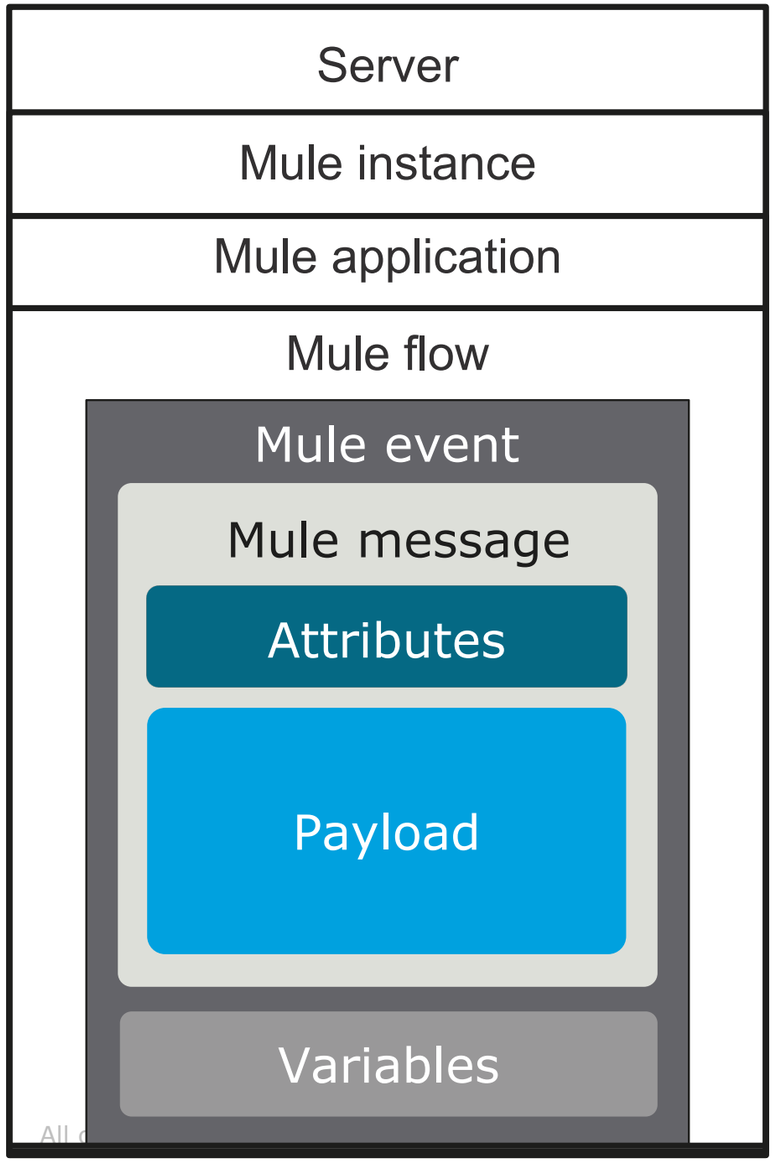
- **Standalone scripts**
  - Were generated using the Transform Message graphical editor in Module 4
  - We will write these from scratch in Module 11



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

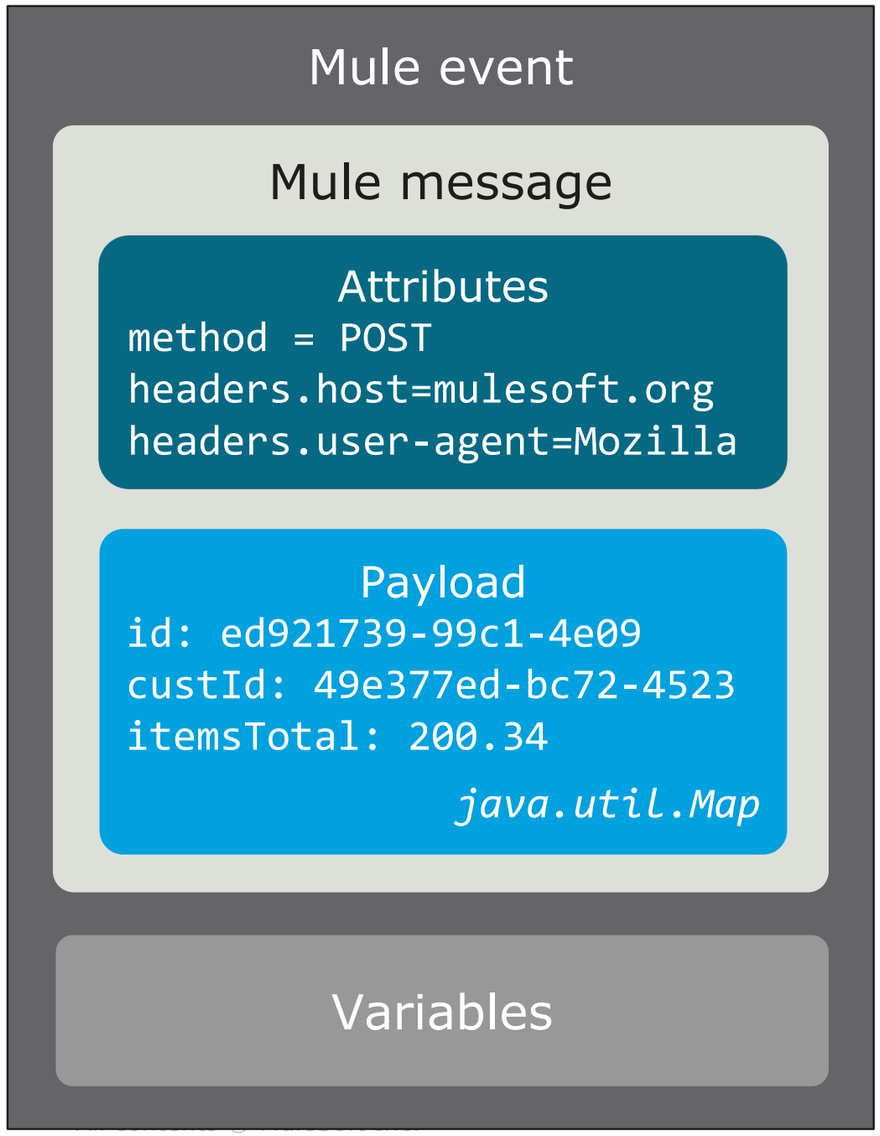
#[]

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

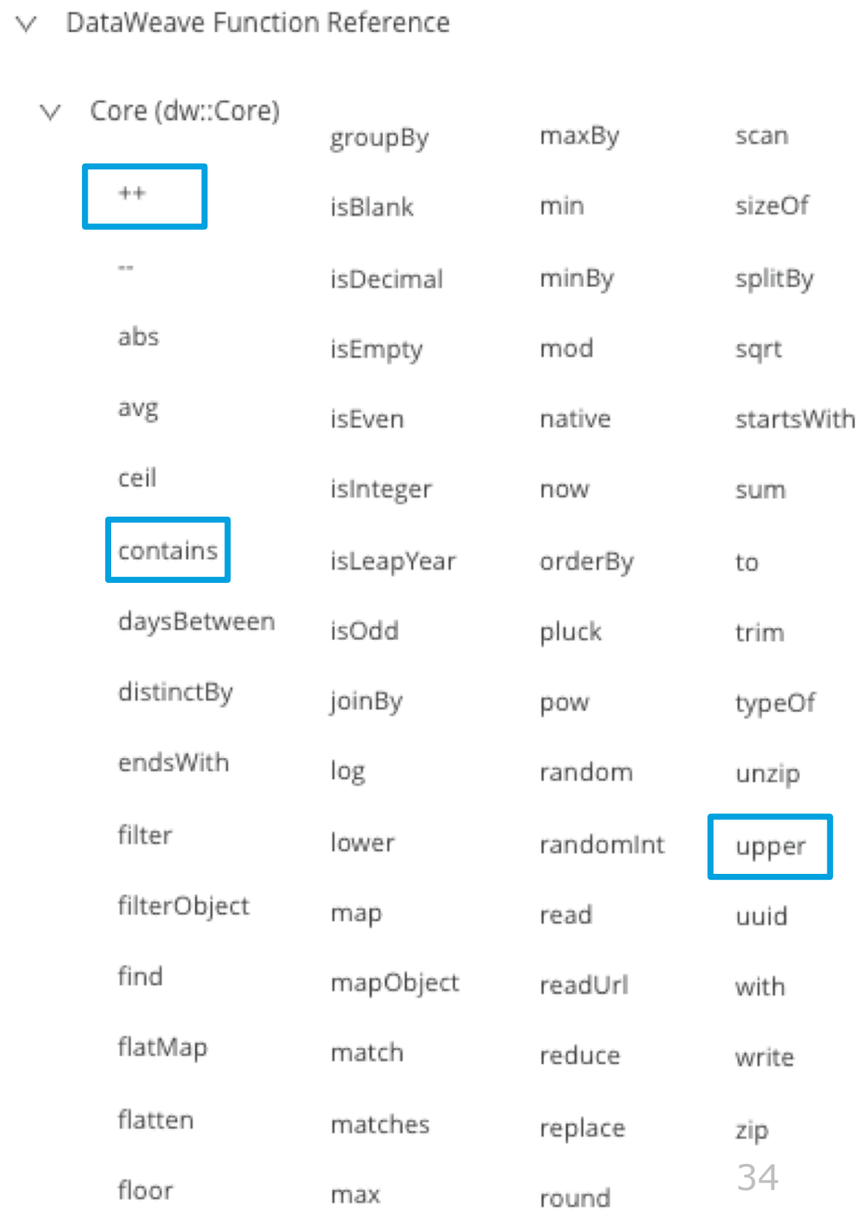
```
if (payload.age < 15)
  group: "child"
else if (payload.age < 25)
  group: "youth"
else if (payload.age < 65)
  group: "adult"
else
  group: "senior"
```

Using DataWeave functions

- Functions are packaged in modules
- Functions in the Core module are imported automatically into DataWeave scripts

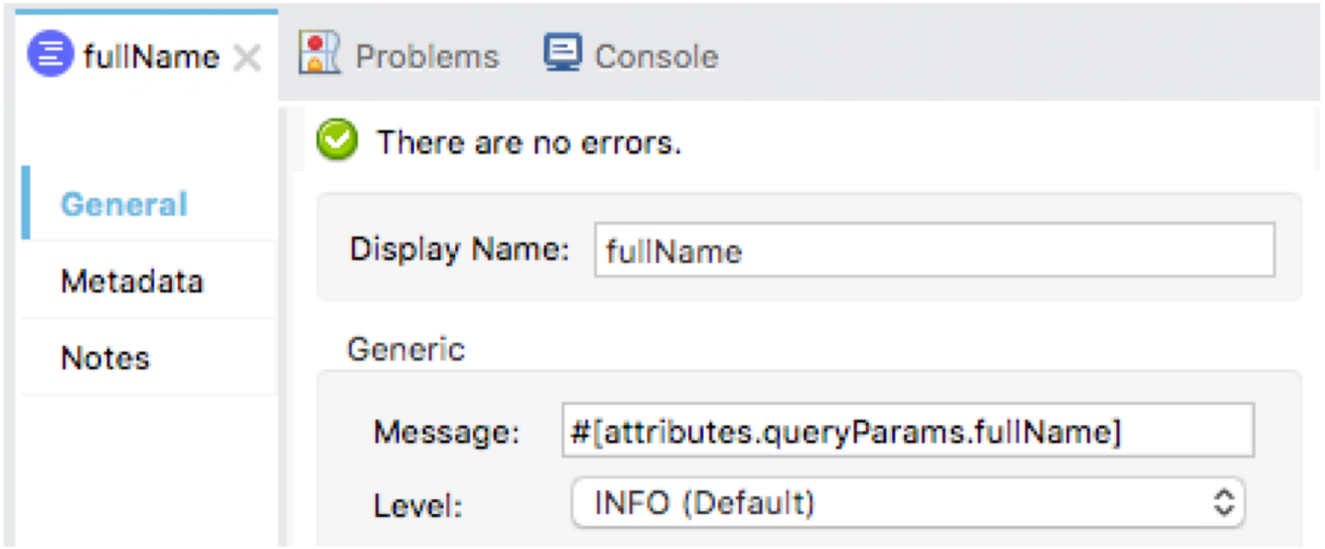
- For function reference, see
  - https://docs.mulesoft.com/mule4-user-guide/v/4.1/dataweave

- 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



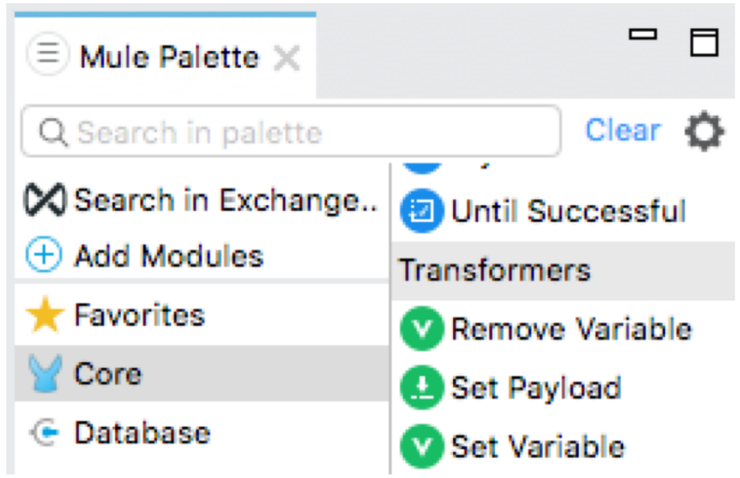
Creating variables

Creating variables

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

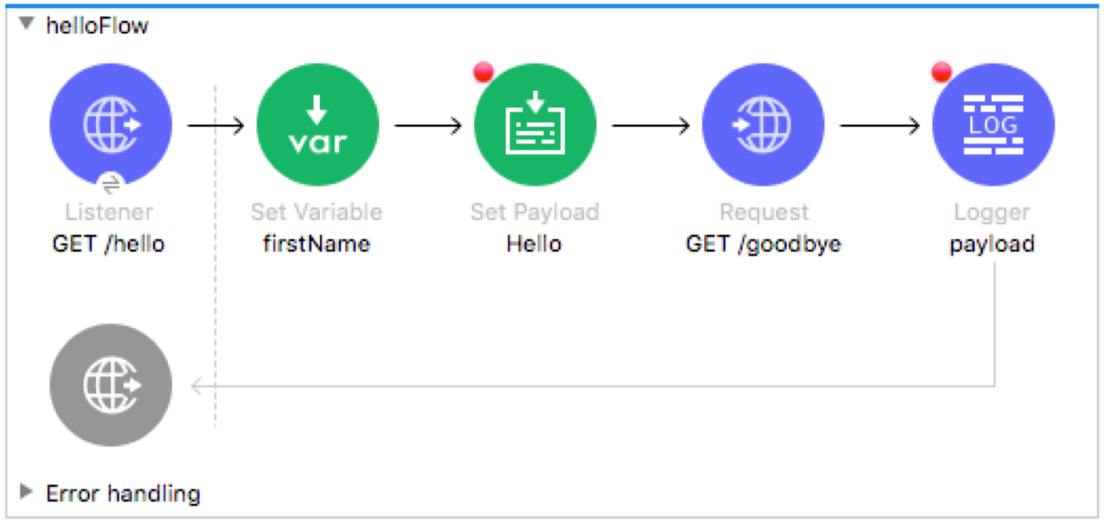
Set Variable

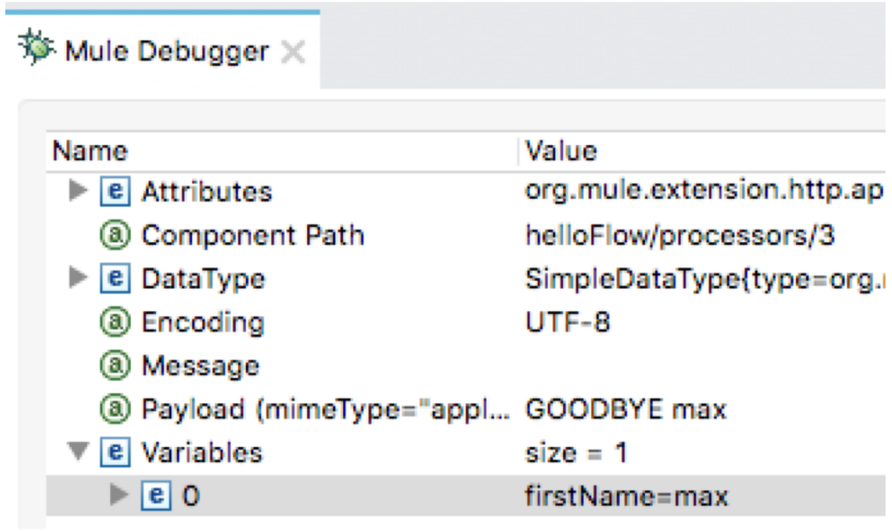
- In expressions, reference as vars
  - #[vars.foo]



Walkthrough 6-5: 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





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