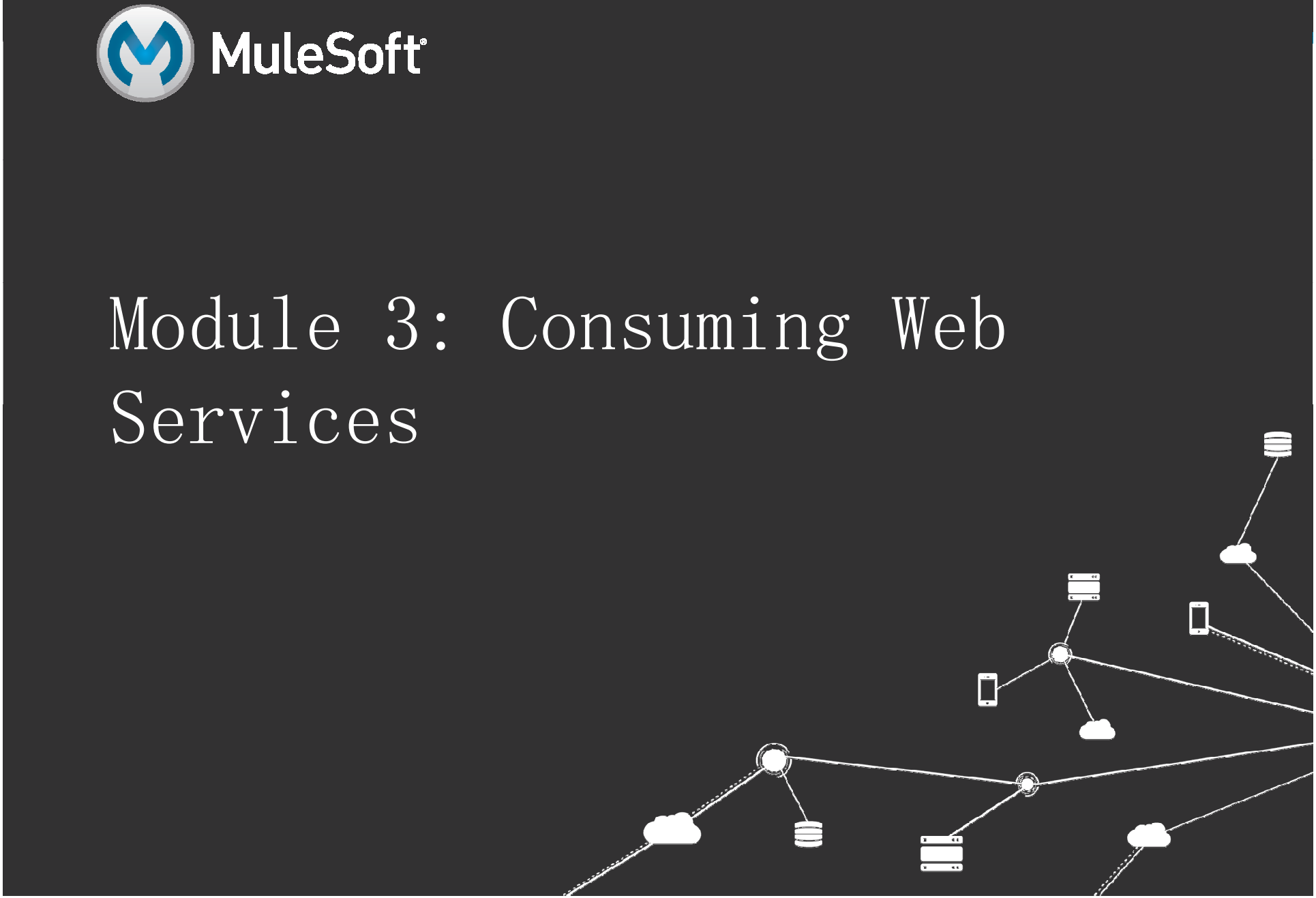
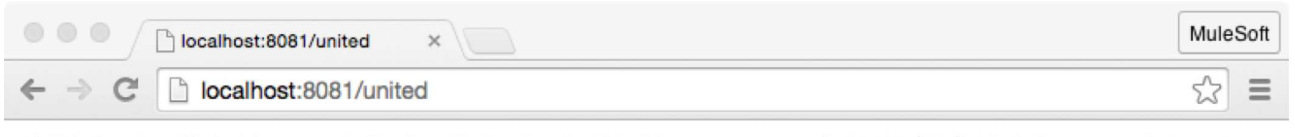
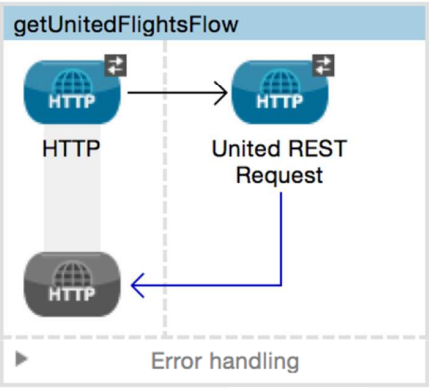


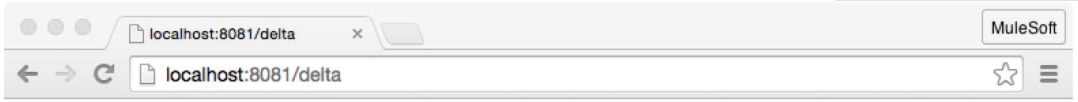
Module 3: Consuming Web Services

Goal



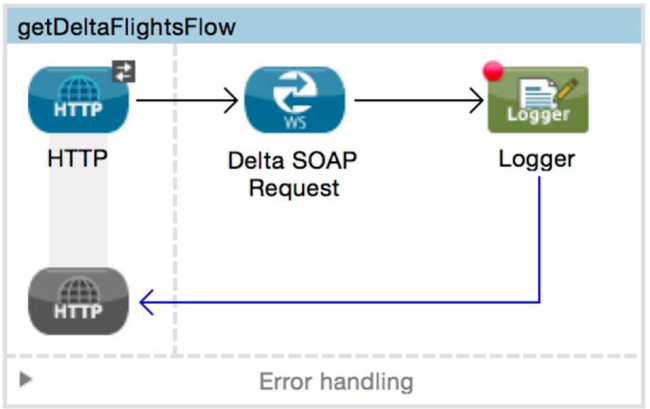

```
{"flights":[{"airlineName":"United","price":400,"departureDate":"2015/03/20","planeType":"Boeing
737","origin":"MUA","code":"ER38sd","emptySeats":0,"destination":"SFO"},
{"airlineName":"United","price":345.99,"departureDate":"2015/02/11","planeT
737","origin":"MUA","code":"ER45if","emptySeats":52,"destination":"LAX"},
{"airlineName":"United","price":346,"departureDate":"2015/04/11","planeType
777","origin":"MUA","code":"ER45jd","emptySeats":12,"destination":"LAX"},
{"airlineName":"United","price":423,"departureDate":"2015/06/11","planeType
707","origin":"MUA","code":"ER0945","emptySeats":0,"destination":"LAX"},
{"airlineName":"United","price":845,"departureDate":"2015/07/11","planeType
727","origin":"MUA","code":"ER9fje","emptySeats":32,"destination":"CLE"},
{"airlineName":"United","price":245,"departureDate":"2015/08/11","planeType
747","origin":"MUA","code":"ER3kfd","emptySeats":13,"destination":"CLE"},
```



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>
     <code>A1B2C4</code>
     <departureDate>2015/02/11</departureDate>
     <destination>LAX</destination>
     <emptySeats>10</emptySeats>
     <origin>MUA</origin>
     <planeType>Boing 737</planeType>
     <price>199.99</price>
   </return>
  ▼<return>
     <airlineName>Delta</airlineName>
     <code>A134DS</code>
     <departureDate>2015/04/11</departureDate>
```

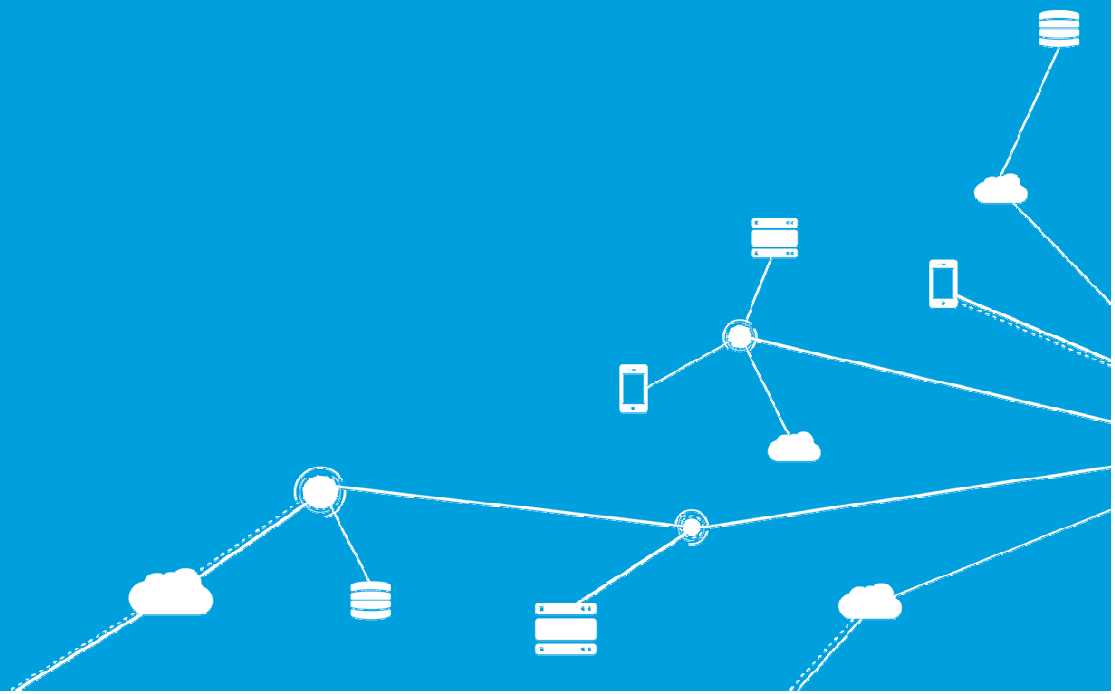



Objectives

- In this module, you will learn:
  - About RESTful and SOAP based web services
  - What RAML is and how it can be used
  - To consume RESTful web services with and without RAML definitions
  - To consume SOAP web services

MuleSoft

Understanding web services



SOAP web services

- Traditional way to expose web services
  - Can bridge protocols, application platforms, programming languages and hardware architectures
- Use SOAP protocol to define a message architecture and message formats
  - Simple Object Access Protocol (SOAP)
- Self-descriptive
  - WSDL (Web Service Description Language)

SOAP protocol

- Based on XML (SOAP envelope uses XML)
- Defines operations, arguments, data types, and more
- Requires tooling to publish and consume
- Usually over HTTP, but can use any protocol
- SOAP request is sent as the body of a HTTP POST

SOAP message example

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap=
"http://www.w3.org/2003/05/soap-envelope">
    <soap:Header>
    </soap:Header>
     <soap:Body>
         <m:GetStockPrice
             xmlns:m="http://www.fooCo.com/stockPrice">
             <m:StockName>FOOC</m:StockName>
         </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```


MuleSoft

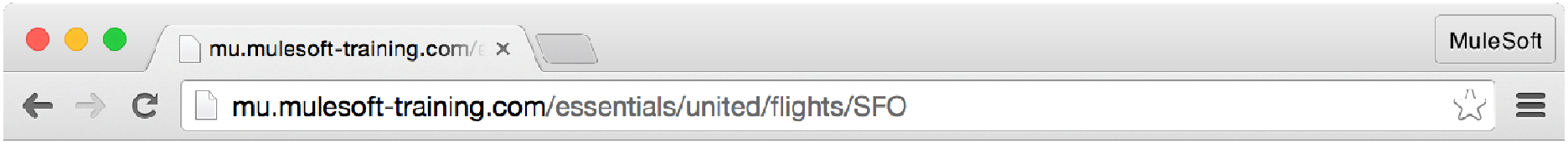
RESTful web services

- Second generation web services
- REST stands for Representational State Transfer
  - An architectural style where clients and servers exchange representations of resources using standardized HTTP protocol
  - The resources are acted upon by using a set of simple, well-defined operations: PUT, GET, POST, DELETE
- Lightweight without a lot of extra XML markup
- Human readable results (usually JSON or XML)
- Easy to build, no toolkits required

RESTful web services

- Data and resources are accessed using URIs
- Resources are manipulated using a fixed set of operations
  - GET retrieves the current state of a resource in some representation (usually JSON or XML)
  - POST creates a new resource
  - PUT transfers a new state onto a resource
  - DELETE deletes a resource

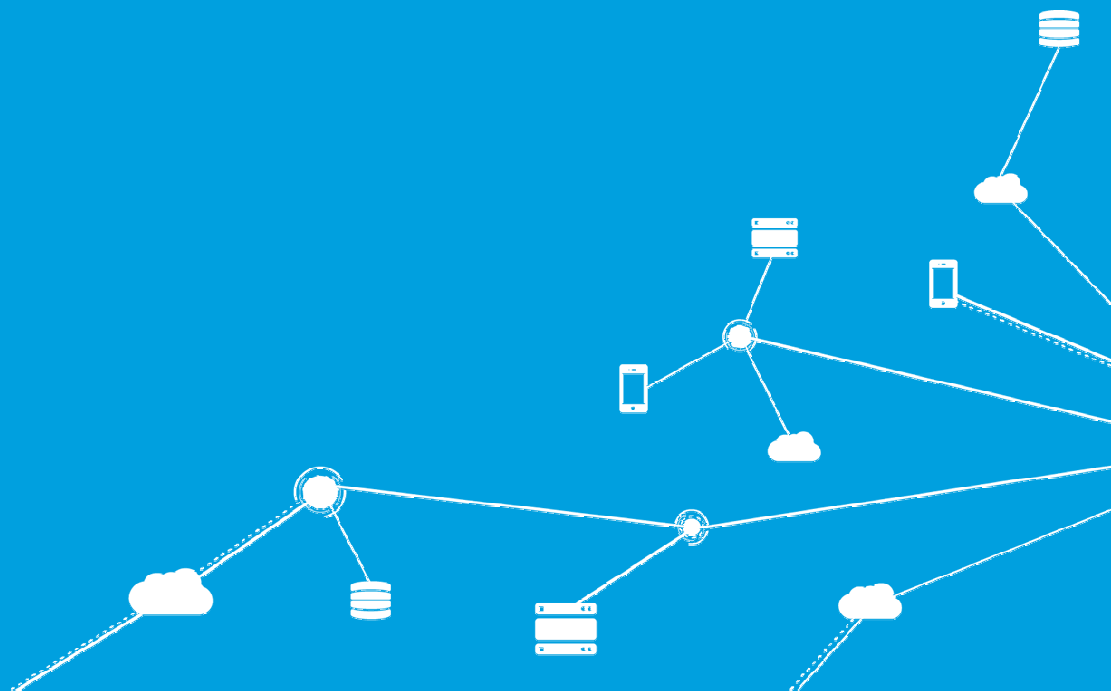
RESTful web service example




```
{"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"}]}
```


MuleSoft

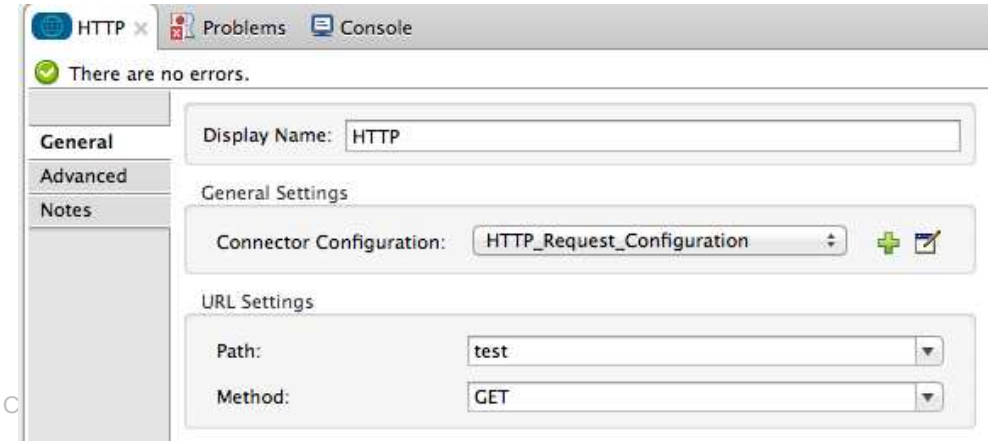
Consuming RESTful web services



Consuming RESTful web services

- First check and see if there is an existing Anypoint Connector to connect to the service provider

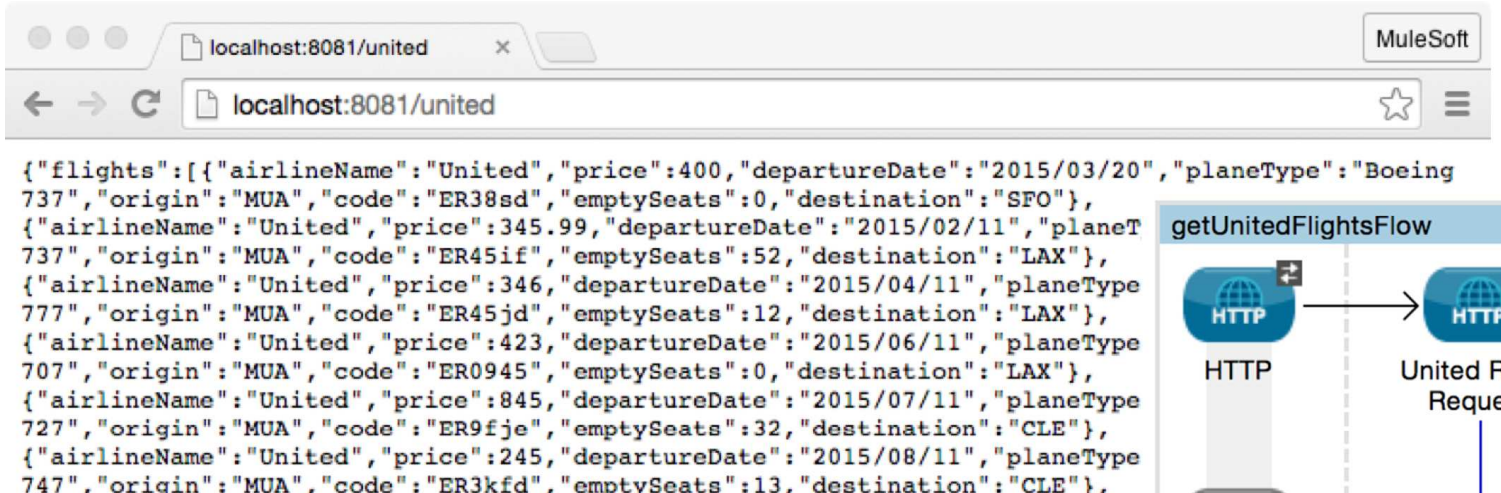
- If there is not, use the HTTP Request connector
  - For the connector
    - Specify host, port, and optionally, a base path
  - For the endpoint
    - Specify path and method



MuleSoft

Walkthrough 3-1: Consume a RESTful web service

- Create a second flow and rename flows
- Add an HTTP Listener connector endpoint to receive requests at http://localhost:8081/united
- Add an HTTP Request connector endpoint to consume a RESTful web service for United flight data



Passing data to a RESTful web service

- For an HTTP Request endpoint, you can add parameters
  - URI parameters
  - Query parameters
  - Headers

- Send form parameters with a request by setting them in the payload

Set Payload

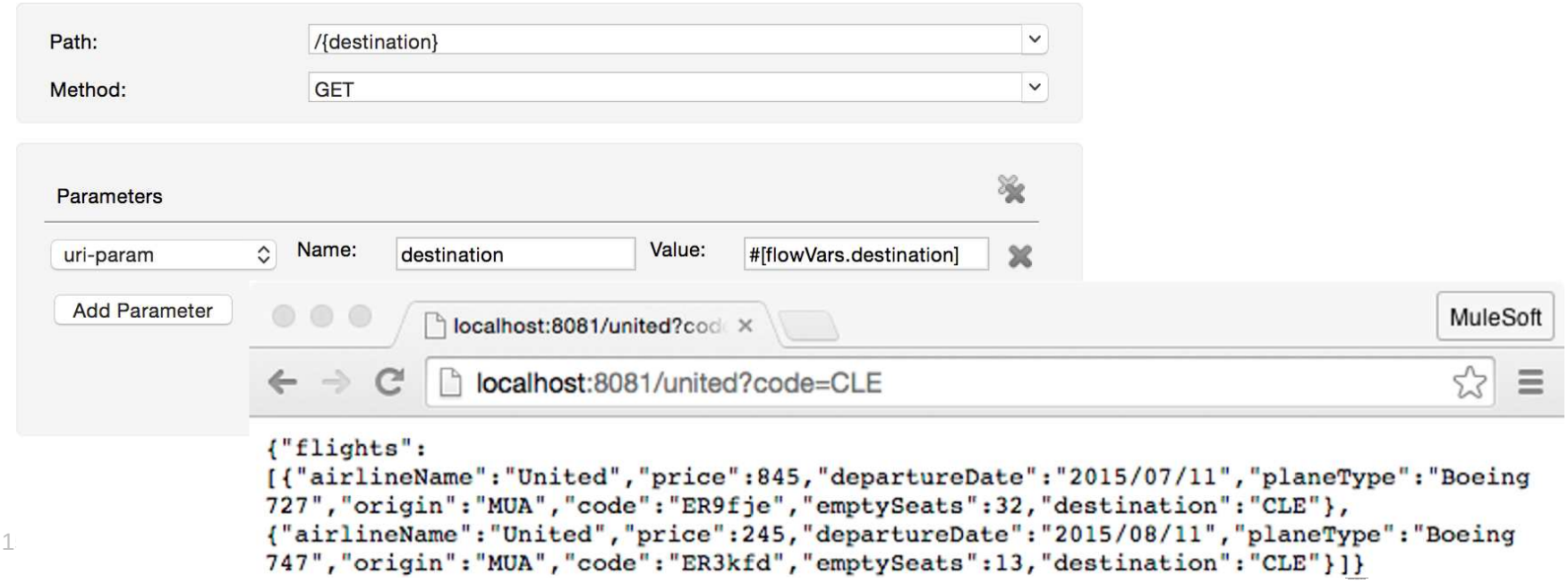
- Include attachments by adding an Attachment transformer to your flow

Attachment

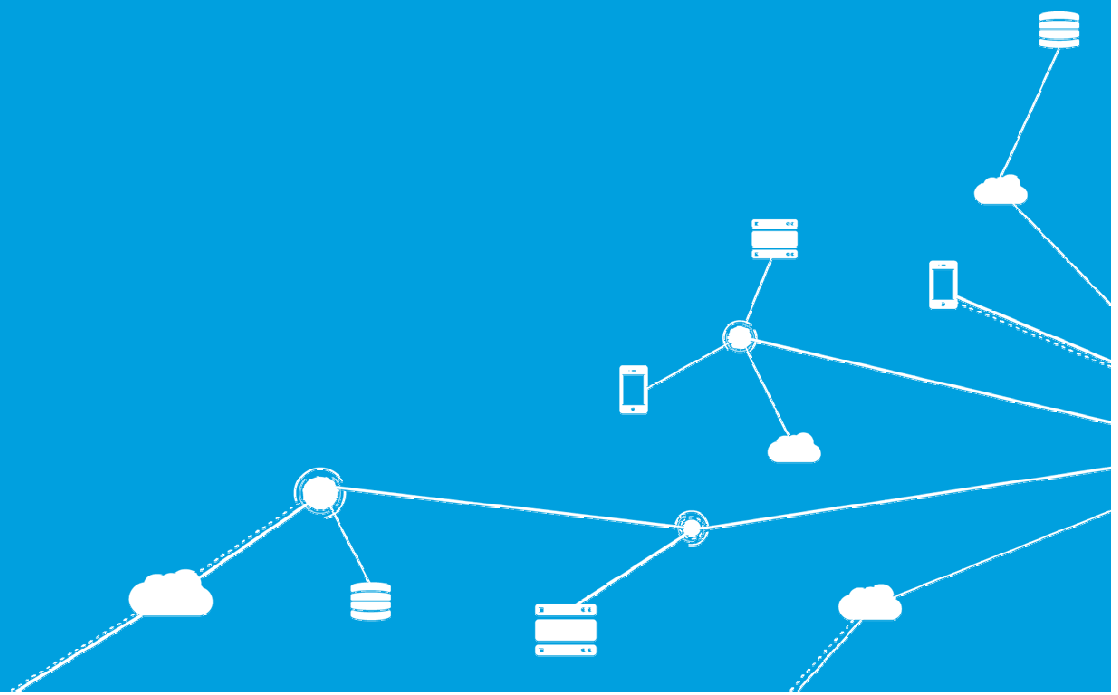
MuleSoft

Walkthrough 3-2: Pass arguments to a RESTful web service

- Modify the HTTP Request connector endpoint to use a URI parameter for the destination
- Set the destination to a static value
- Set the destination to a dynamic query parameter value
- Create a variable to set the destination



Introducing RAML



Approaches to API design

HAND CODING

- Time-consuming
- Maintenance nightmare
- Difficult to read/ consume
- Disconnected from consumer

SWAGGER

- Verbose
- Primarily for doc generation
- Limited reuse
- JSON based

BLUEPRINT

- Markdown
- Design focused
- Limited reuse
- Tooling is proprietary

RAML

- Simple and succinct
- Intuitive
- Open, non-proprietary
- Based on standards

MuleSoft

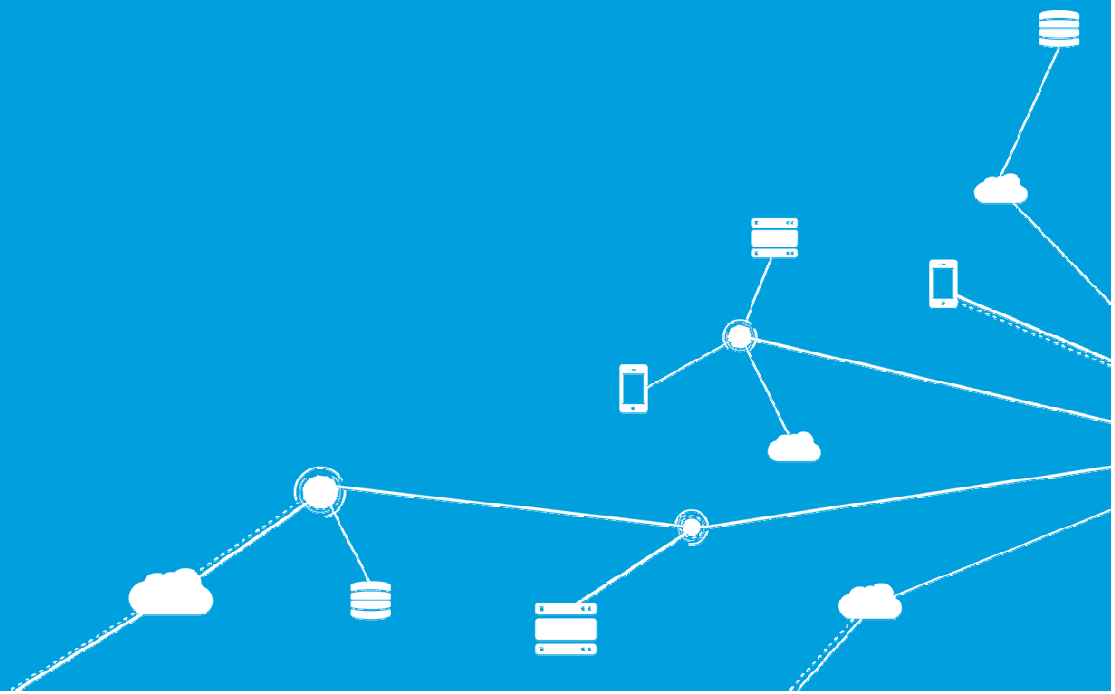
RAML: RESTful API Modeling Language

- A simple and succinct way of describing RESTful APIs
  - Resources, schema, parameters, responses, and more
- Developed to help out the current API ecosystem
  - Encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices
- A non-proprietary, vendor-neutral open spec
- Built on broadly-used standards such as YAML and JSON
  - YAML A'int a Markup Language
    - A human-readable data serialization format where data structure hierarchy is maintained by outline indentation
- http://raml.org

RAML example

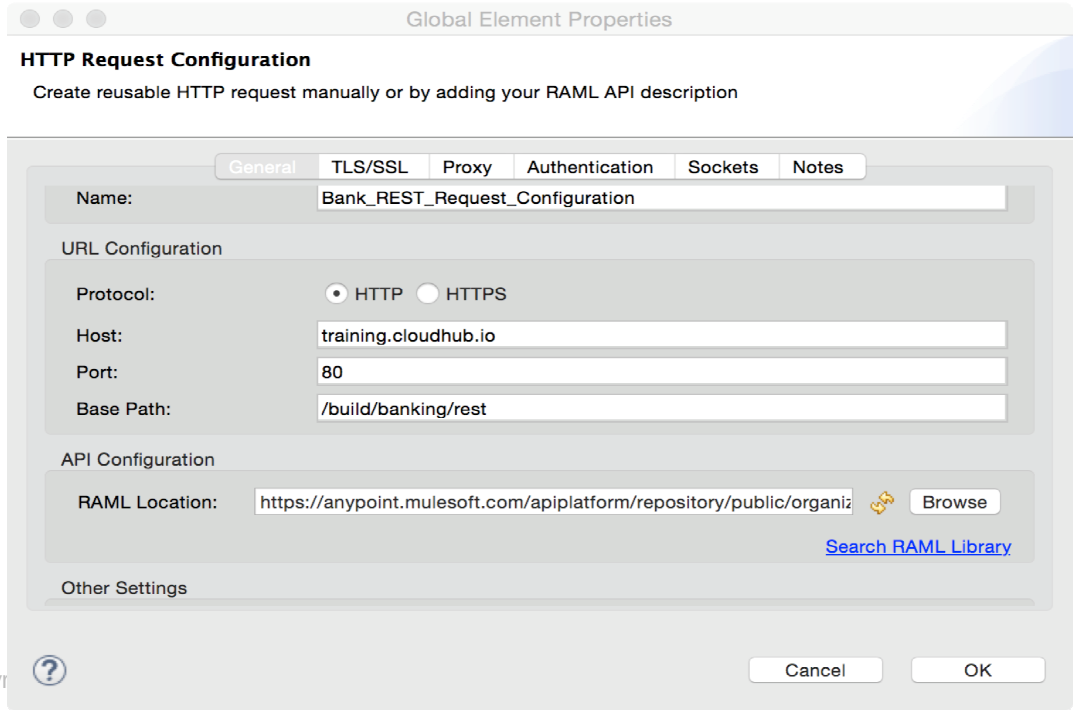
```
#%RAML 0.8

title: World Music API
baseUri: http://example.api.com/{version}
version: v1
traits:
  - paged:
      queryParameters:
        pages:
          description: The number of pages to return
          type: number
  - secured: !include http://raml-example.com/secured.yml
/songs:
  is: [ paged, secured ]
  get:
    queryParameters:
      genre:
        description: filter the songs by genre
  post:
  /{songId}:
    get:
      responses:
        200:
          body:
            application/json:
              schema: |
```

Consuming RESTful web services with RAML definitions



Consuming RESTful web services with RAML definitions

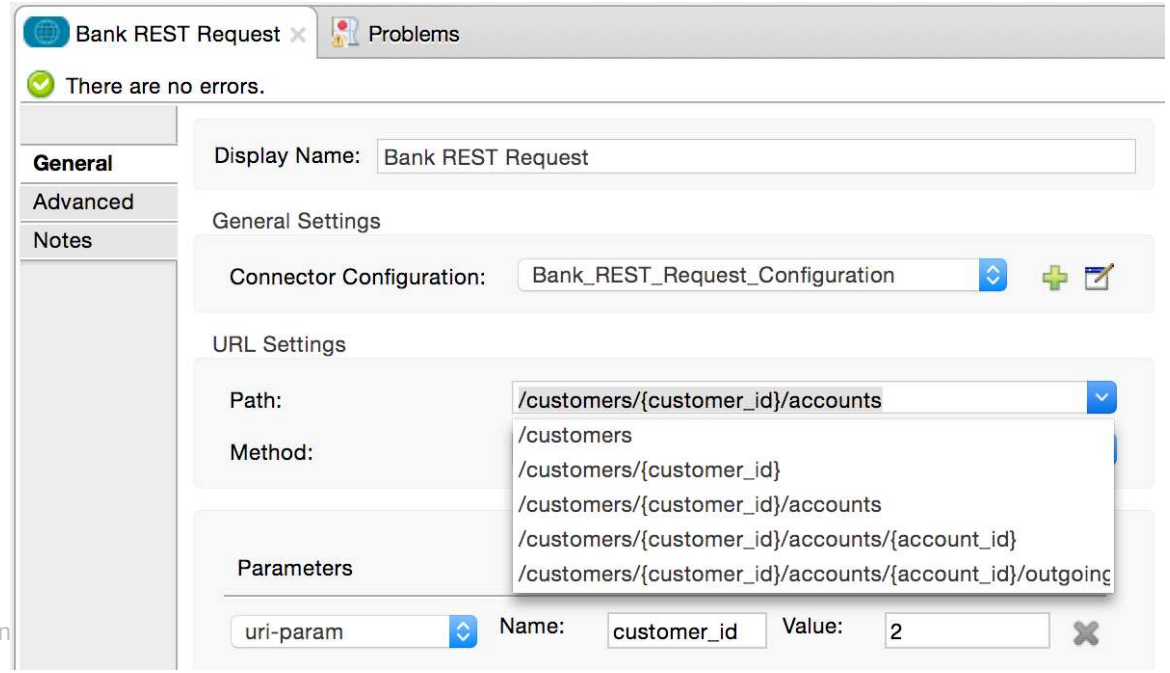
- A RAML location (local file or external URI) can be specified for an HTTP Request connector
- After you specify a RAML location
  - All of the other fields will be automatically populated based on what's specified in the RAML



MuleSoft

RAML metadata

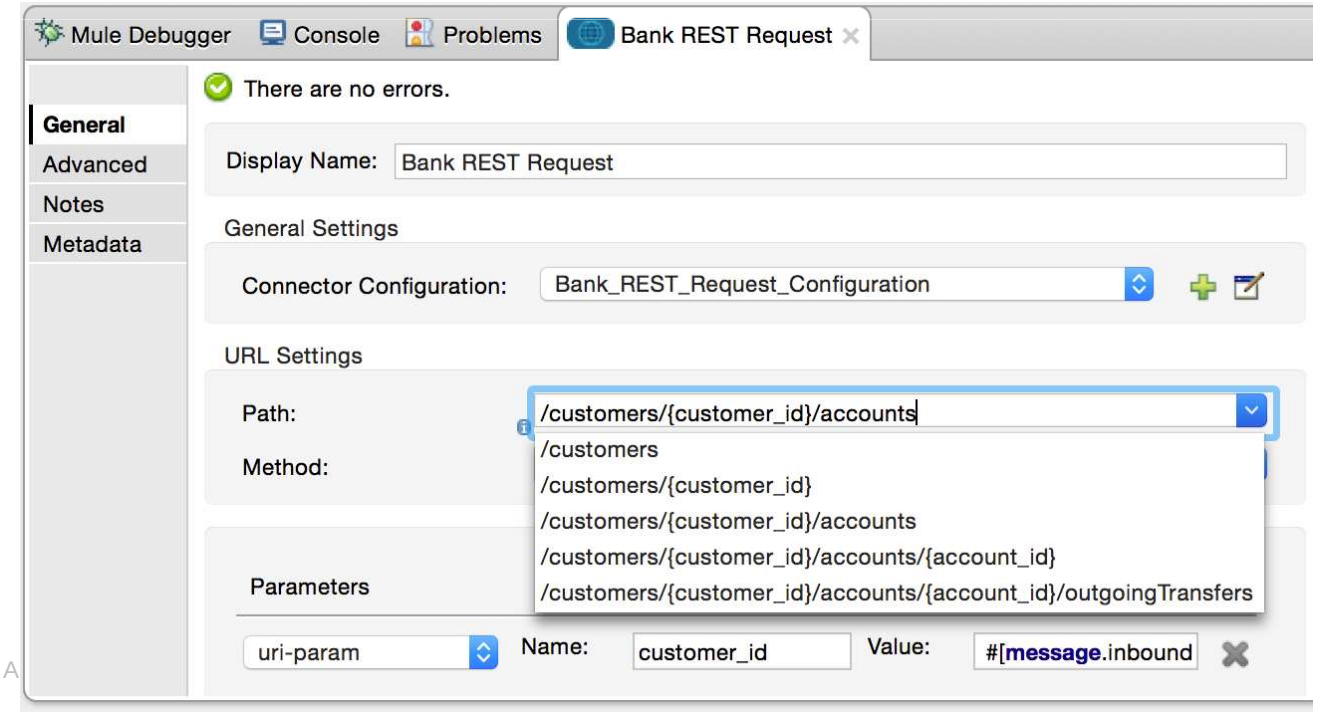
- The RAML metadata
  - Will be obtained every time you open the project in Studio
  - Kept in cache until you close the project
  - Can be refreshed /reloaded if the RAML changes
- Is used by Studio to offer intelligent suggestions



MuleSoft

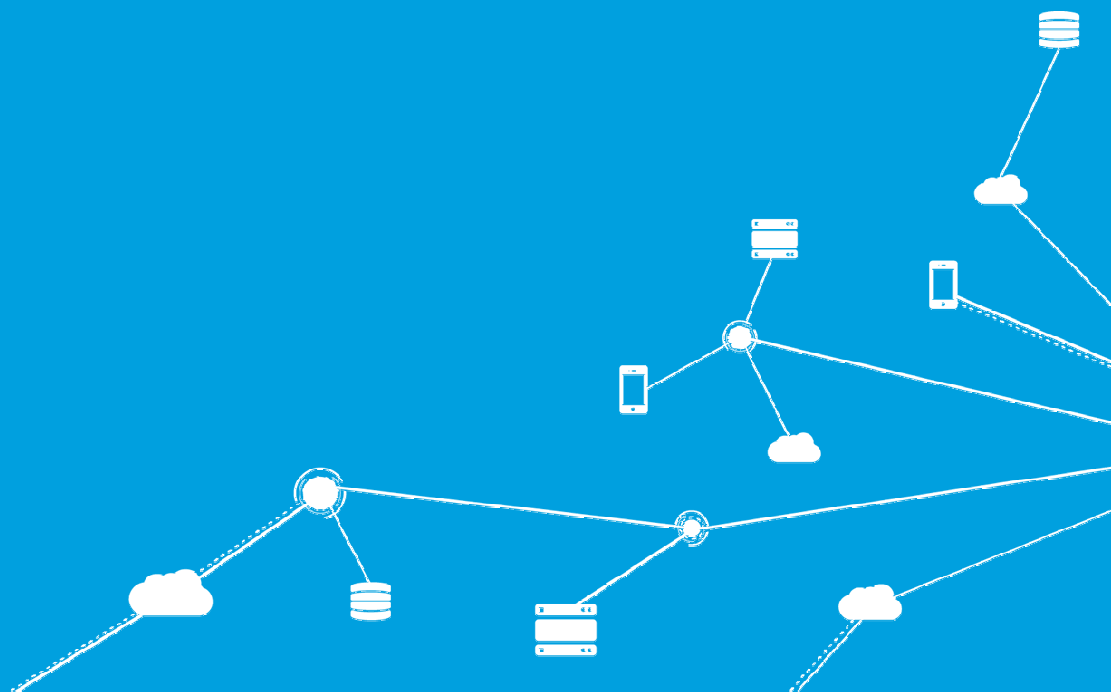
Walkthrough 3-3: Consume a RESTful web service that has a RAML definition

- Add a third flow to the application
- Add an endpoint to receive requests at http://localhost:8081/bank
- Add an HTTP Request endpoint to consume a RESTful web service defined with a RAML file



MuleSoft

Consuming SOAP web services



Apache CXF

- Mule's SOAP support is based on Apache CXF
  - A web services framework in Java for SOAP messaging
  - Handles all serialization and deserialization
  - Handles all SOAP envelope and namespace processing
  - Developer sees only POJOs, etc. - not SOAP XML

Consuming SOAP web services

- First check and see if there is an existing Anypoint Connector to connect to the service provider

- If there is not, use the Web Service Consumer connector
  - Provide the location of the WSDL
  - The rest will be configured for you: host, port, address, available operations

Web Service Consumer

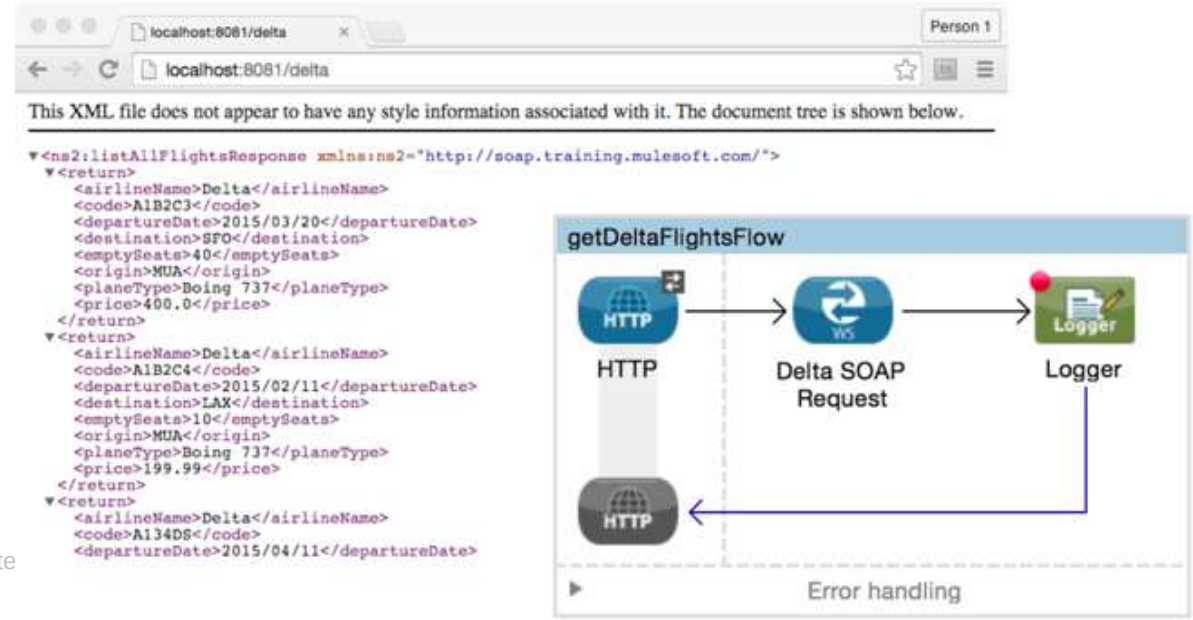
- If you need more features, use the CXF component
  - Also used to expose an endpoint as a SOAP service

CXF

MuleSoft

Walkthrough 3-4: Consume a SOAP Web Service

- Create a fourth flow with an endpoint to receive requests at http://localhost:8081/delta
- Add a Web Service Consumer connector to consume a SOAP web service for Delta flight data
- Use the DOM to XML transformer to display the SOAP response

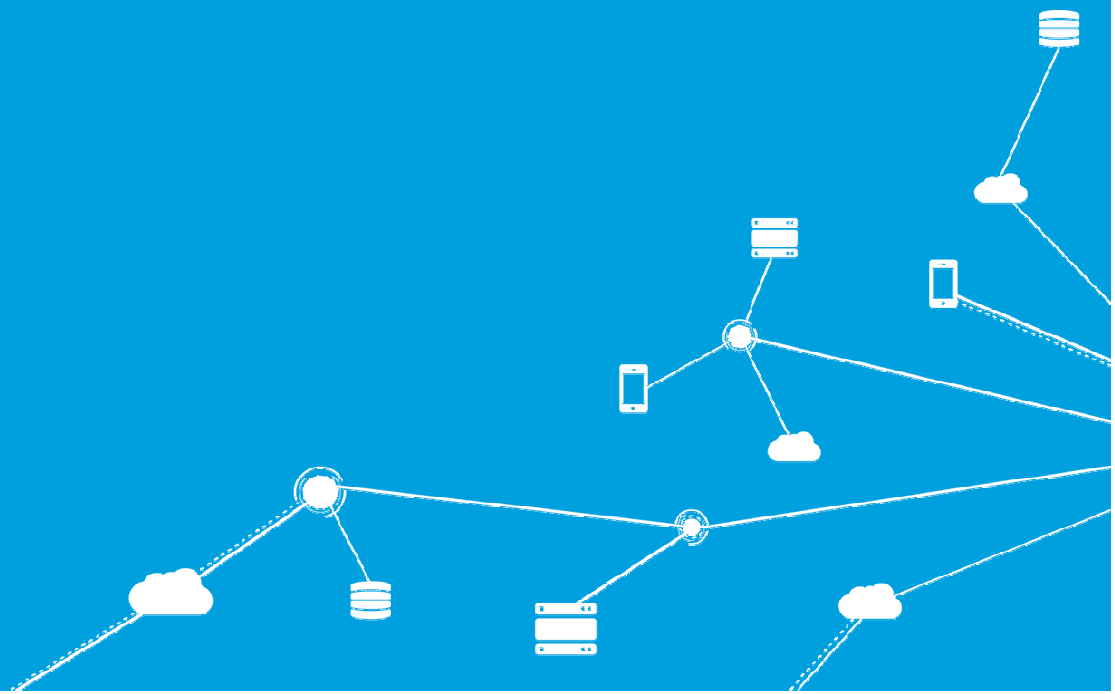


MuleSoft

Passing data to a SOAP web service

- Using the Web Service Consumer
  - Use DataWeave (or the DataMapper transformer) to add and map an input argument
    - You will do this in module 5

Summary



Summary

- In this module, you learned to consume web services
- Use the Web Consumer connector to consume SOAP web services
- Use the HTTP Request connector to consume REST web services
  - With or without URI parameters and query parameters
  - With or without a RAML definition
- RAML is the Restful API Modeling Language
  - A simple, succinct, open-spec standard based way to describe RESTful APIs

