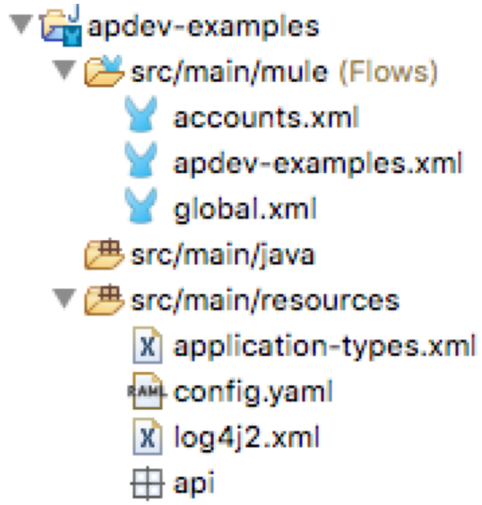
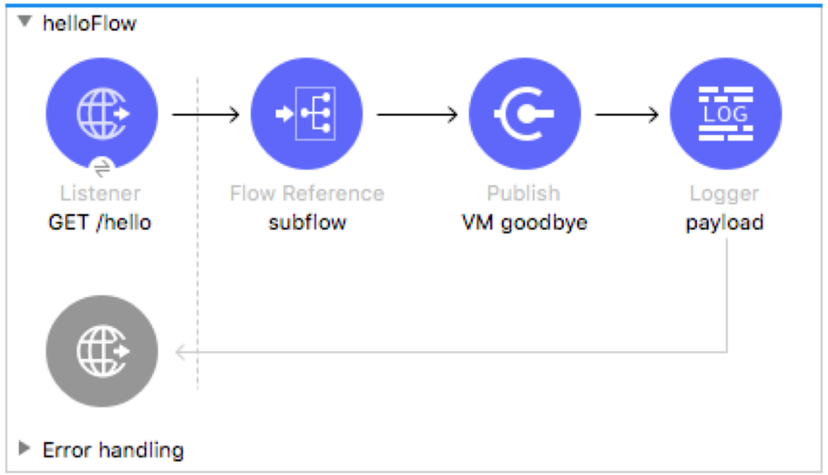
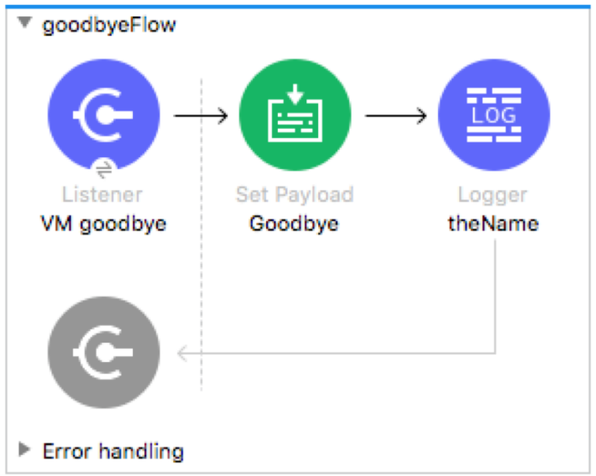
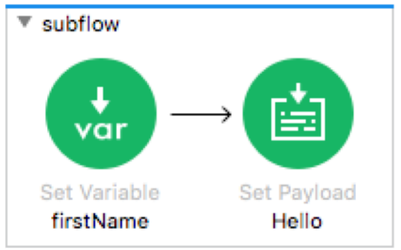
Module 7: Structuring Mule Applications

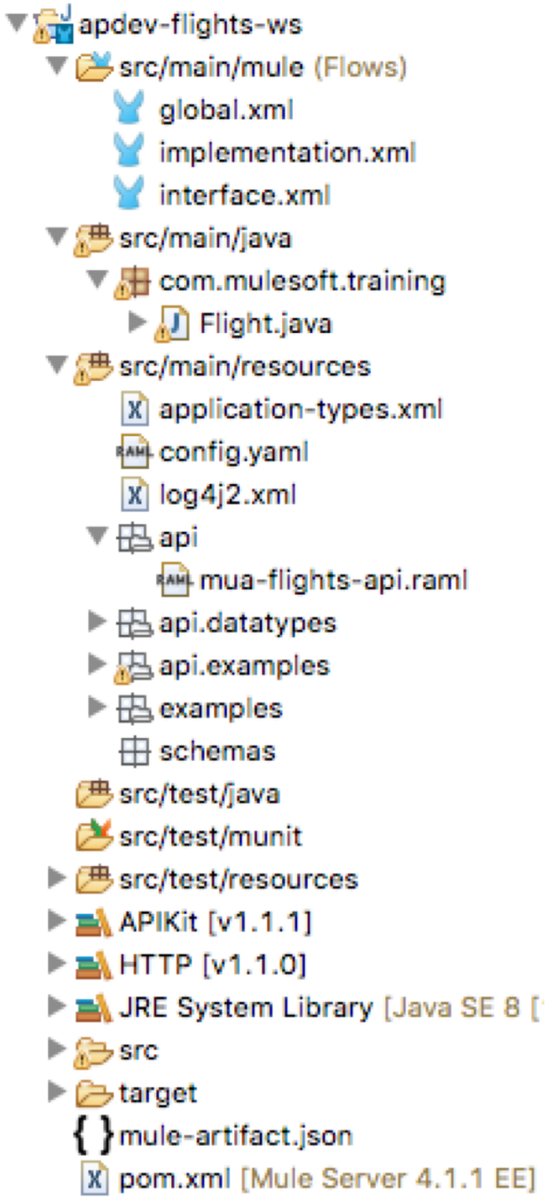
Goal


apdev-examples
src/main/mule (Flows)
accounts.xml
apdev-examples.xml
global.xml
src/main/java
src/main/resources
application-types.xml
config.yaml
log4j2.xml
api


helloFlow
Listener
GET /hello
Flow Reference
subflow
Publish
VM goodbye
Logger
payload
LOG
Error handling


goodbyeFlow
Listener
VM goodbye
Set Payload
Goodbye
Logger
theName
LOG
Error handling


subflow
var
Set Variable
firstName
Set Payload
Hello

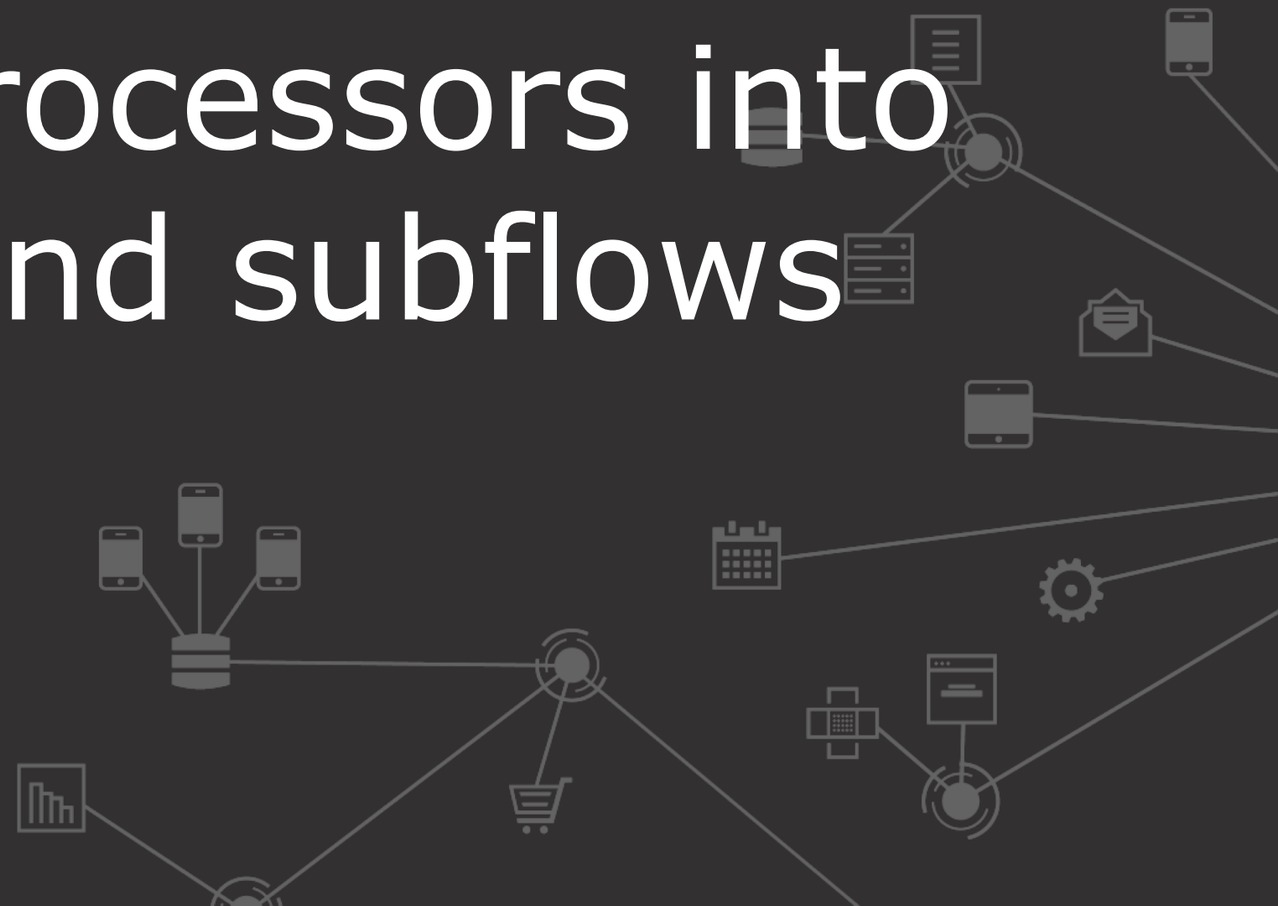

apdev-flights-ws
src/main/mule (Flows)
global.xml
implementation.xml
interface.xml
src/main/java
com.mulesoft.training
Flight.java
src/main/resources
application-types.xml
config.yaml
log4j2.xml
api
mua-flights-api.raml
api.datatypes
api.examples
examples
schemas
src/test/java
src/test/munit
src/test/resources
APIKit [v1.1.1]
HTTP [v1.1.0]
JRE System Library [Java SE 8 [
src
target
mule-artifact.json
pom.xml [Mule Server 4.1.1 EE]

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

- Create applications composed of multiple flows and subflows
- Pass messages between flows using asynchronous queues
- Encapsulate global elements in separate configuration files
- Specify application properties in a separate properties file and use them in the application
- Describe the purpose of each file and folder in a Mule project
- Define and manage application metadata

All contents © MuleSoft Inc.

Encapsulating processors into separate flows and subflows



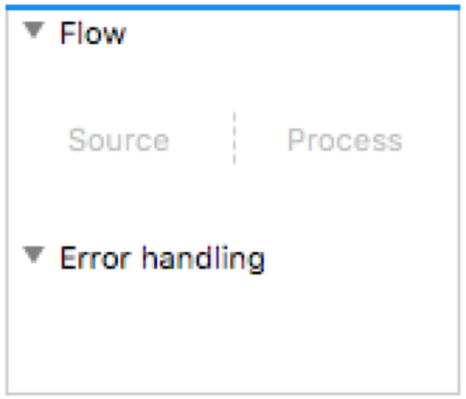
Break up flows into separate flows and subflows

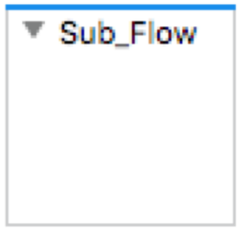
- Makes the graphical view more intuitive
  - You don’t want long flows that go off the screen
- Makes XML code easier to read
- Enables code reuse
- Provides separation between an interface and implementation
  - We already saw this
- Makes them easier to test

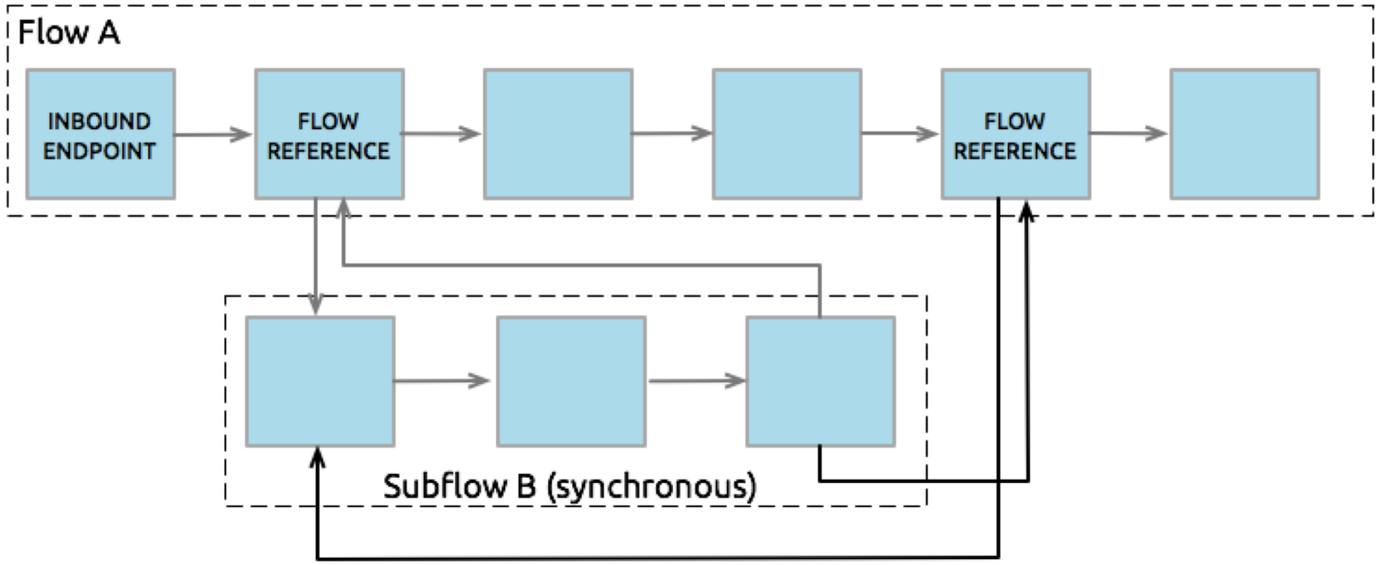
All contents © MuleSoft Inc.

Flows vs subflows

- **Flows** can have their own error handling strategy, **subflows** cannot
- Flows without event sources are sometimes called **private** flows
- Subflows are executed exactly as if the processors were still in the calling flow



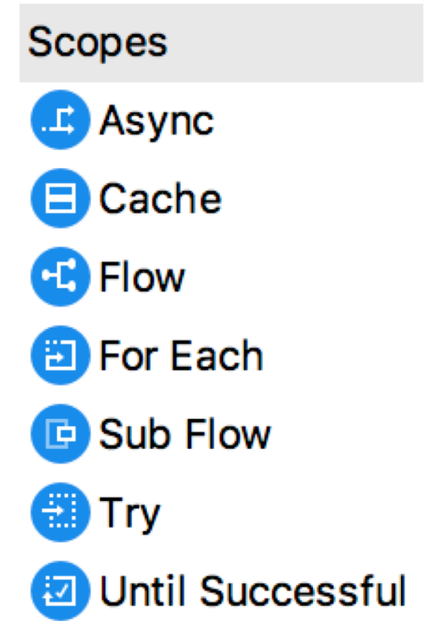


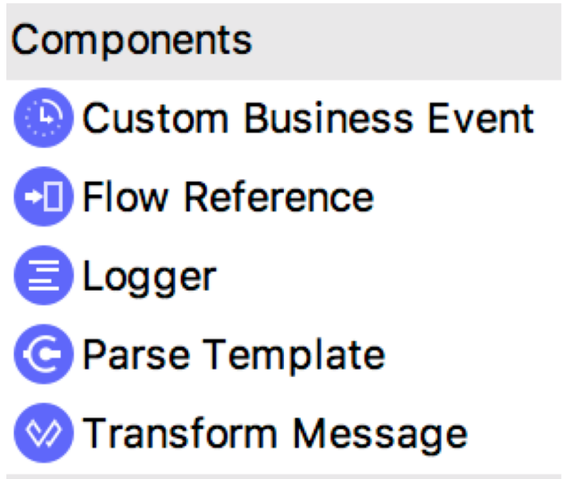


All contents © MuleSoft Inc.

Creating flows and subflows

- Several methods
  - Add a new scope: Flow or Sub Flow
  - Drag any event processor to the canvas – creates a flow
  - Right-click processor(s) in canvas and select Extract to
- Use Flow Reference component to pass events to other flows or subflows
- Variables persist through all flows unless the event crosses a transport boundary
  - We saw this in the last module

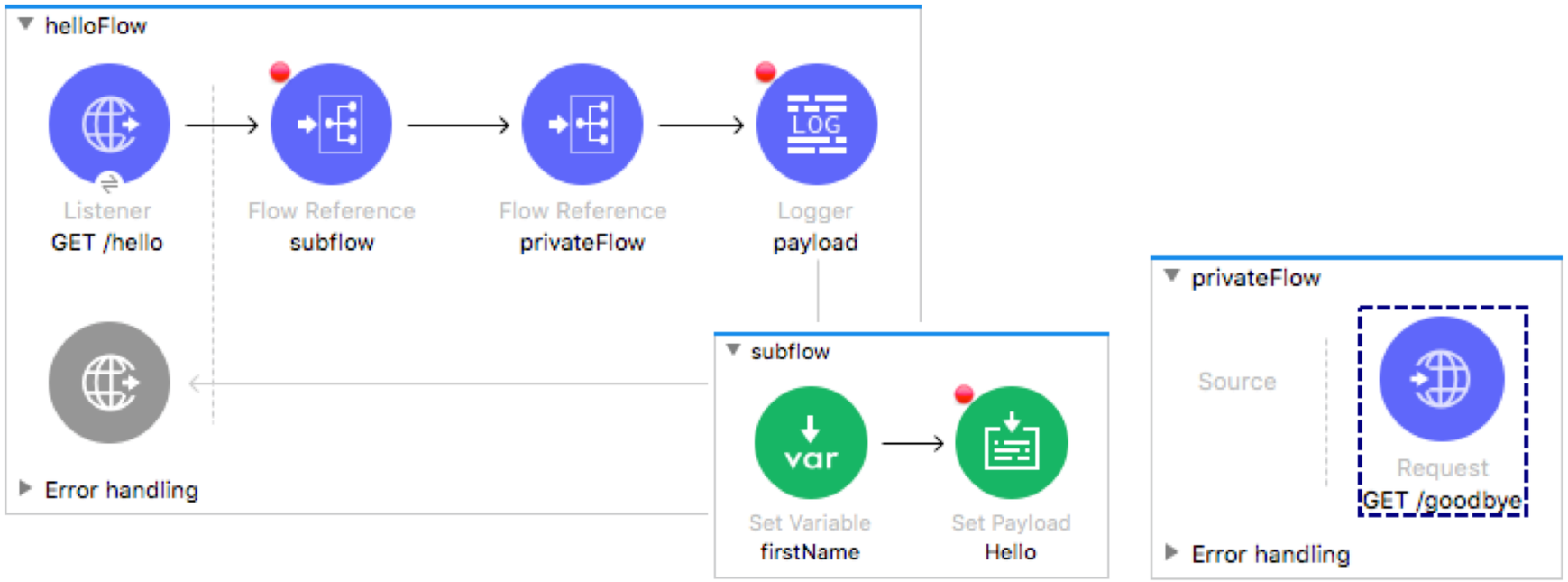




All contents © MuleSoft Inc.

Walkthrough 7-1: Create and reference subflows and private flows

- Extract processors into separate subflows and private flows
- Use the Flow Reference component to reference other flows
- Explore event data persistence through subflows and private flows



All contents © MuleSoft Inc.

Passing messages between flows using asynchronous queues

Passing messages between flows using asynchronous queues

- When using Flow Reference, events are passed synchronously between flows

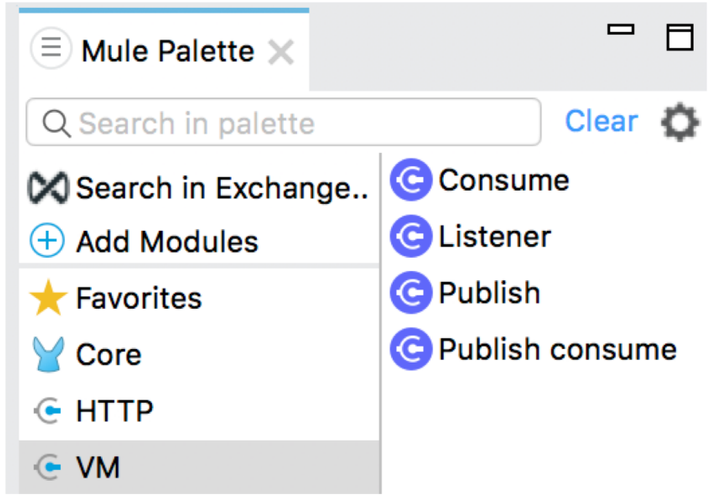
- You may want to pass events asynchronously between flows to
  - Achieve higher levels of parallelism in specific stages of processing
  - Allow for more-specific tuning of areas within a flow's architecture
  - Distribute work across a cluster
  - Communicate with another application running in the same Mule domain
    - Domains will be explained later this module
  - Implement simple queueing that does not justify a full JMS broker
    - JMS is covered in Module 12

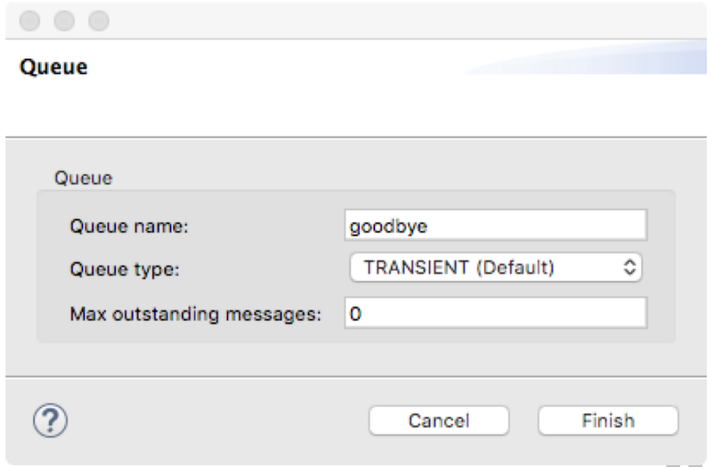
- This can be accomplished using the **VM connector**

All contents © MuleSoft Inc.

Using the VM connector

- Use the connector for intra and inter application communication through asynchronous queues
- Add the VM module to the project
- Configure a global element configuration
  - Specify a queue name and type
    - Queues can be transient or persistent
    - By default, the connector uses in-memory queues
    - **Transient** queues are faster, but are lost in the case of a system crash
    - **Persistent** queues are slower but reliable
- Use operations to publish and/or consume messages

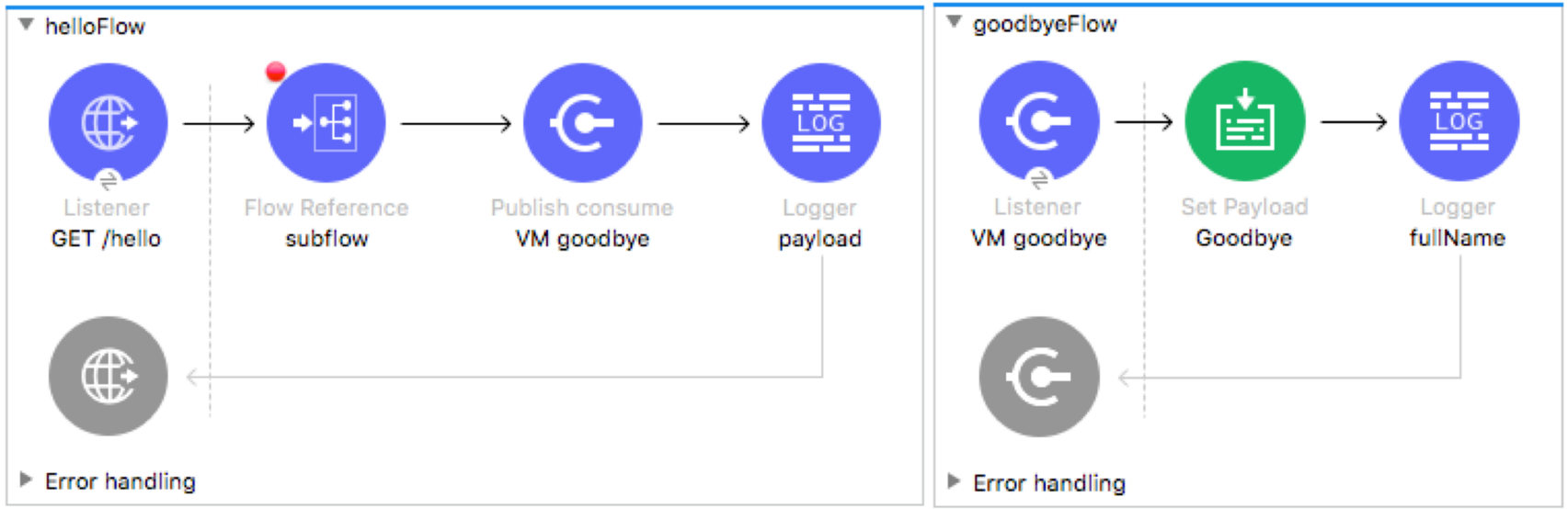




All contents © MuleSoft Inc.

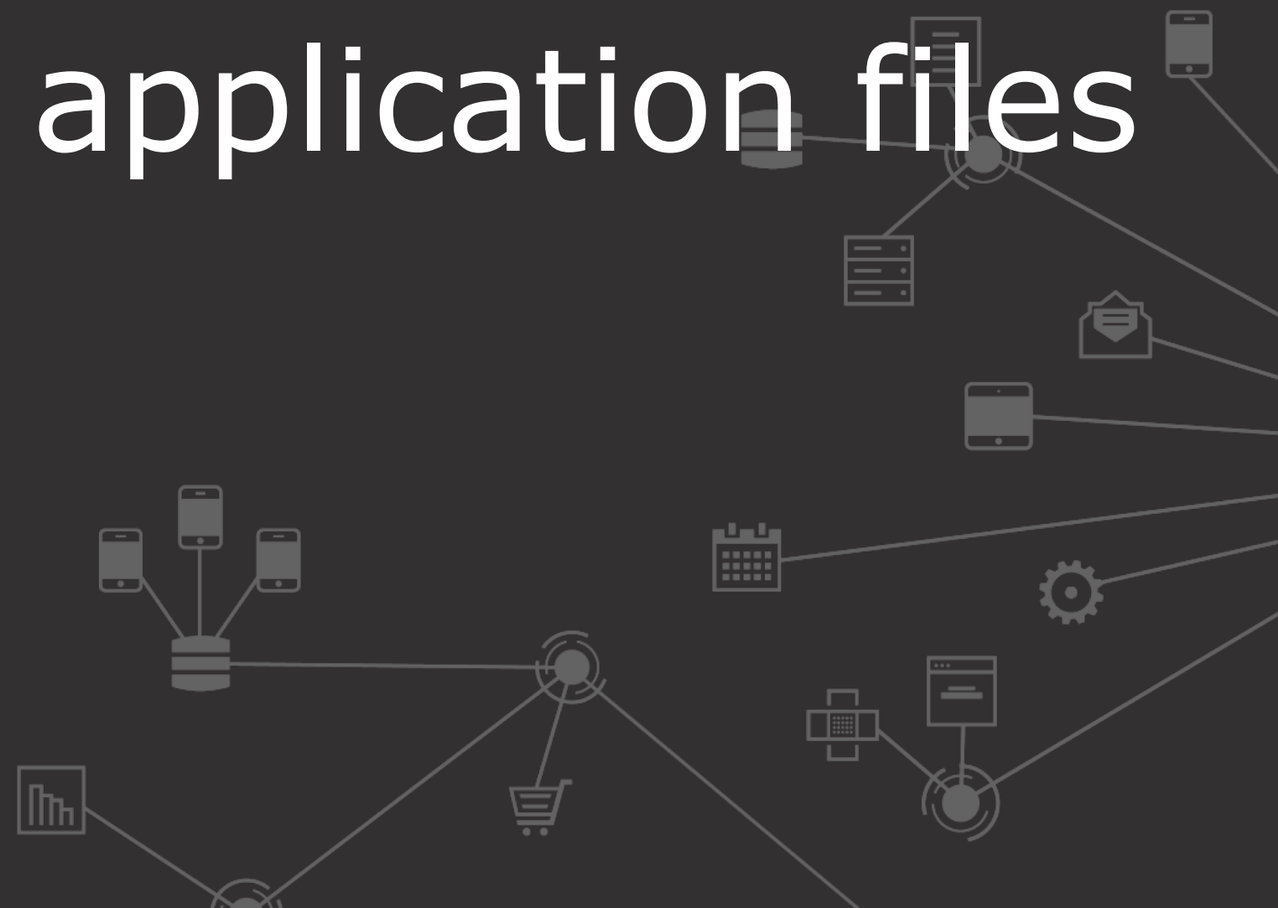
Walkthrough 7-2: Pass messages between flows using the VM connector

- Pass messages between flows using the VM connector
- Explore variable persistence with VM communication
- Publish content to a VM queue and then wait for a response
- Publish content to a VM queue without waiting for a response



All contents © MuleSoft Inc.

Organizing Mule application files



Separating apps into multiple configuration files

- Just as we separated flows into multiple flows, we also want to separate configuration files into multiple configuration files
- Monolithic files are difficult to read and maintain
- Separating an application into multiple configuration files makes code
  - Easier to read
  - Easier to work with
  - Easier to test
  - More maintainable

All contents © MuleSoft Inc.

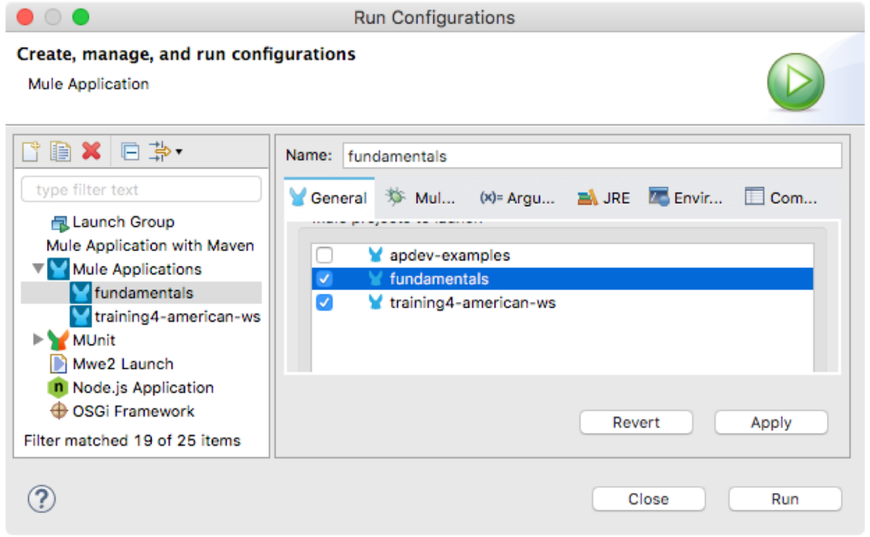
Encapsulating global elements in a configuration file

- If you reference global elements in one file that are defined in various, unrelated files
  - It can be confusing
  - It makes it hard to find them
- A good solution is to put most global elements in one config file
  - All the rest of the files reference them
  - If a global element is specific to and only used in one file, it can make sense to keep it in that file

All contents © MuleSoft Inc.

Creating multiple applications

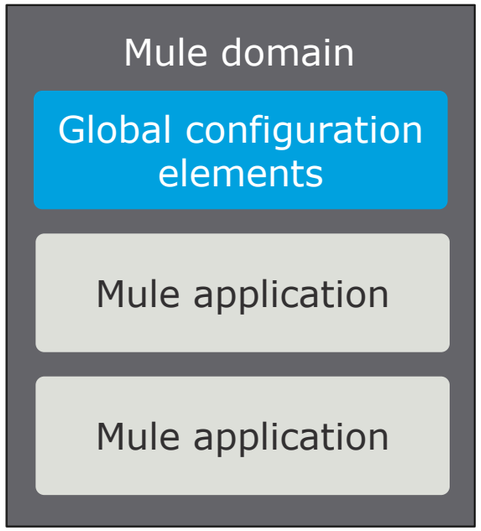
- You are also not going to want all your flows in one application/project
- Separate functionality into multiple applications to
  - Allow managing and monitoring of them as separate entities
  - Use different, incompatible JAR files
- Run more than one application at a time in Anypoint Studio by creating a run configuration



All contents © MuleSoft Inc.

Sharing global elements between applications

- **A domain project** can be used to share global configuration elements between applications, which lets you
  - Ensure consistency between applications upon any changes, as the configuration is only set in one place
  - Expose multiple services within the domain on the same port
  - Share the connection to persistent storage (Module 12)
  - Call flows in other applications using the VM connector



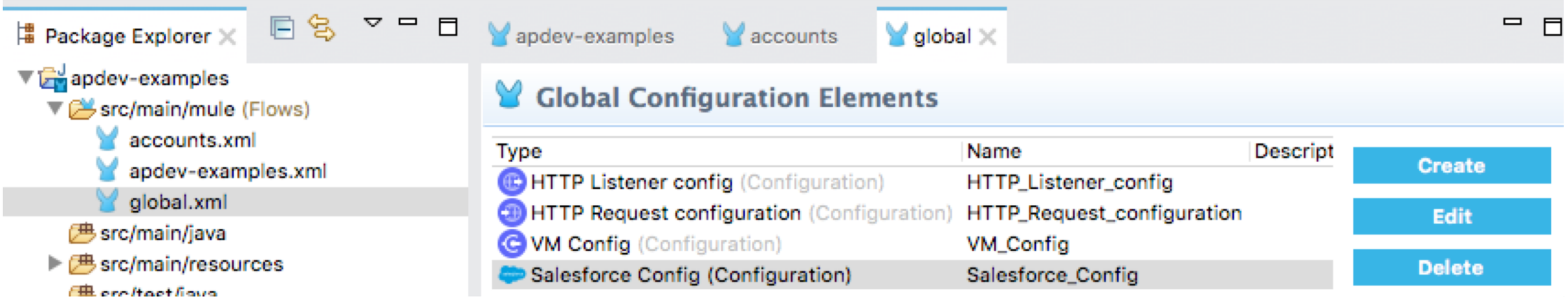
- Only available for customer-hosted Mule runtimes, not on CloudHub

- The general process
  - Create a Mule Domain Project and associate Mule applications with a domain
  - Add global element configurations to the domain project

All contents © MuleSoft Inc.

Walkthrough 7-3: Encapsulate global elements in a separate configuration file

- Create a new configuration file with an endpoint that uses an existing global element
- Create a configuration file global.xml for just global elements
- Move the existing global elements to global.xml
- Create a new global element in global.xml and configure a new connector to use it



All contents © MuleSoft Inc.

Organizing and parameterizing application properties

Application properties

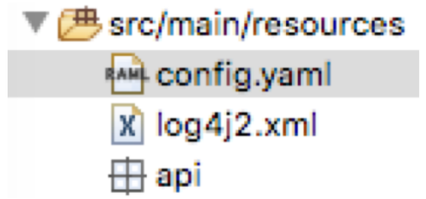
- Provide an easier way to manage connector properties, credentials, and other configurations
- Replace static values
- Are defined in a configuration file
  - Either in a .yaml file or a .properties file
- Are implemented using property placeholders
- Can be encrypted
- Can be overridden by system properties when deploying to different environments

All contents © MuleSoft Inc.

Defining application properties

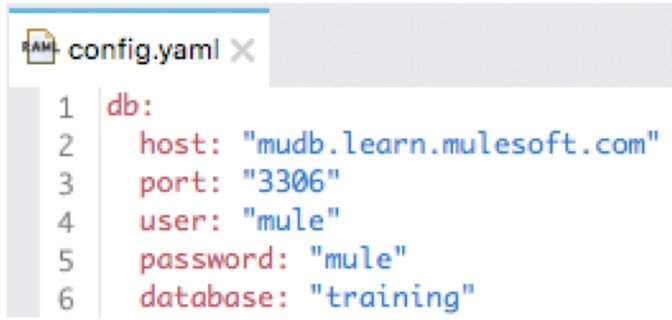
- Create a YAML properties file in the src/main/resources folder

```
config.yaml
```

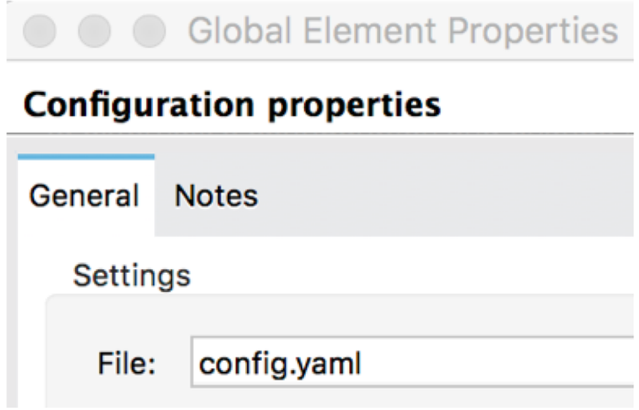


- Define properties in the hierarchical YAML file

```
db:
  port: "3306 "
  user: "mule"
```



- Create a Configuration properties global element



All contents © MuleSoft Inc.

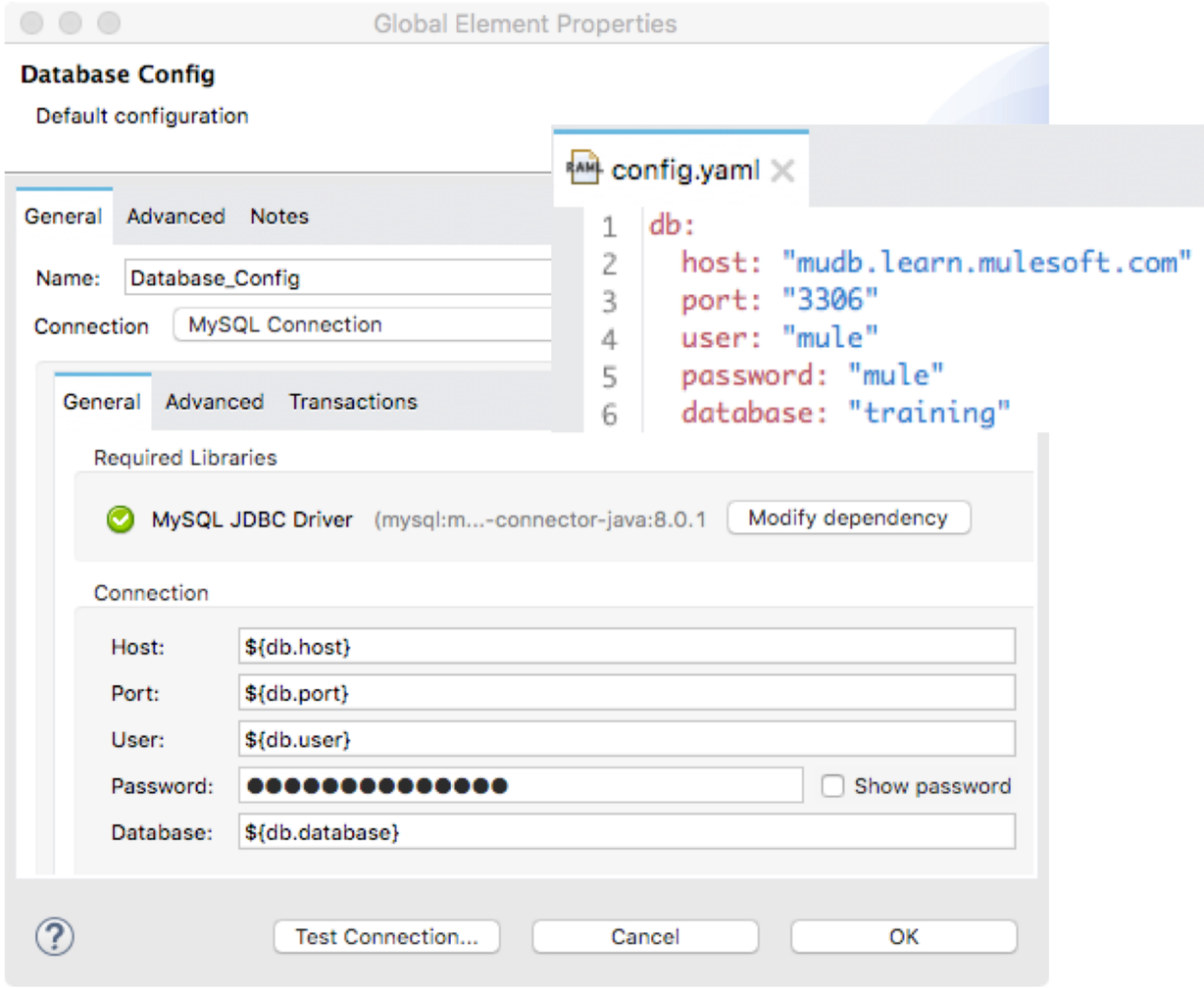
Using application properties

- In global element configurations and event processors

```
${db.port}
```

- In DataWeave expressions

```
{port: p('db.port')}
```



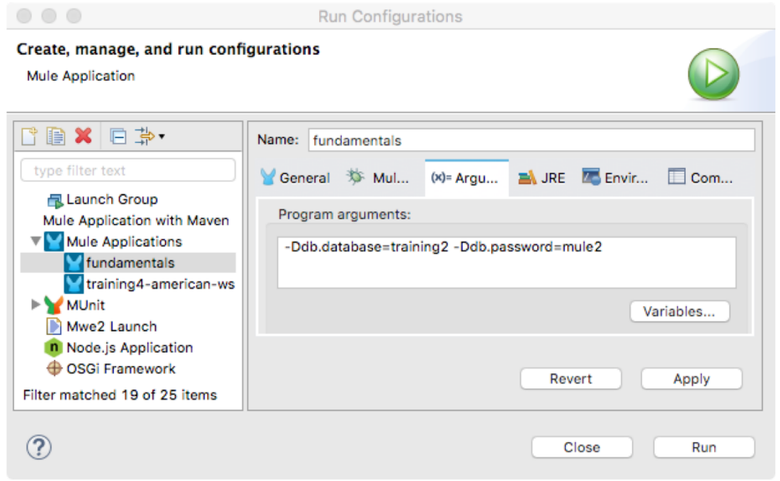
All contents © MuleSoft Inc.

Overriding property values in different environments

- Use **system properties** to override property values when deploying an application to a different environment (like dev, qa, production),

- Set system properties (JVM parameters) from

  - Anypoint Studio
    in Run > Run Configurations > Arguments



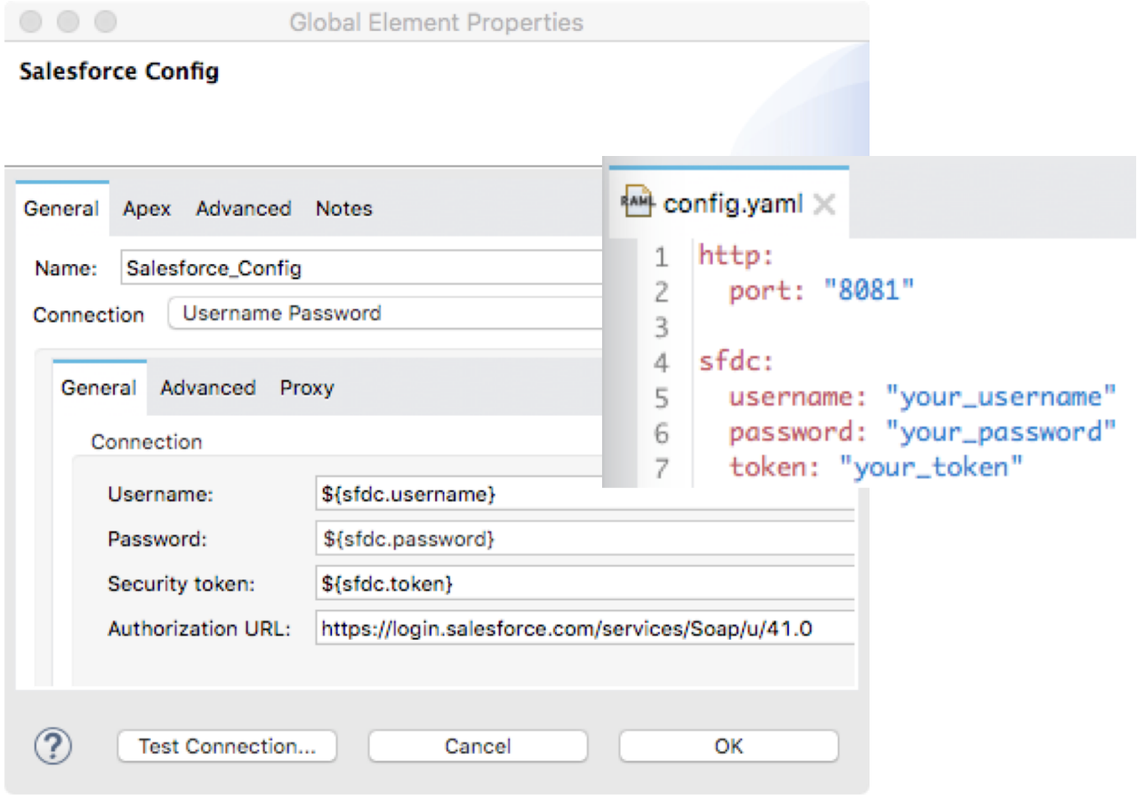
  - The command line for a standalone Mule instance

```
mule -M-Ddb.database=training2 -M-Ddb.password=mule2
```

All contents © MuleSoft Inc.

Walkthrough 7-4: Use property placeholders in connectors

- Create a YAML properties file for an application
- Configure an application to use a properties file
- Define and use HTTP and Salesforce connector properties

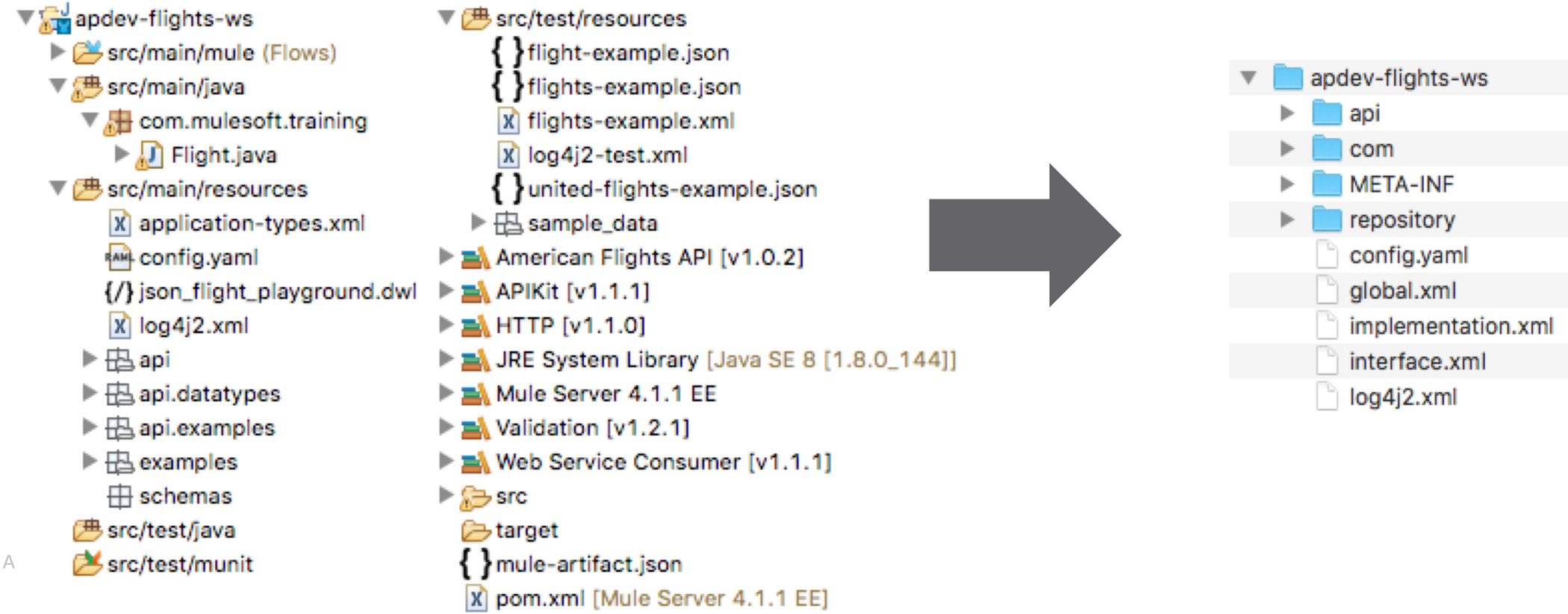


All contents © MuleSoft Inc.

Organizing Mule project files

Examining the folder structure for a Mule project

- The names of folders indicate what they should contain
- src/test folders should contain files only needed at development time
  - Like schema and example files for metadata types, sample data for transformations
  - They are not included in the application JAR when it is packaged



In Mule 4, Mule applications are Maven projects

- **Maven** is a tool for building and managing any Java-based project that provides
  - A standard way to build projects
  - A clear definition of what a project consists of
  - An easy way to publish project information
  - A way to share JARs across several projects

- Maven manages a project's build, reporting, and documentation from a central piece of information – the **project object model (POM)**

- A Maven build produces one or more **artifacts**, like a compiled JAR
  - Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string

All contents © MuleSoft Inc.

The POM (Project Object Model)

- Is an XML file that contains info about the project and configuration details used by Maven to build the project including
  - Project info like its version, description, developers, and more
  - Project dependencies
  - The plugins or goals that can be executed

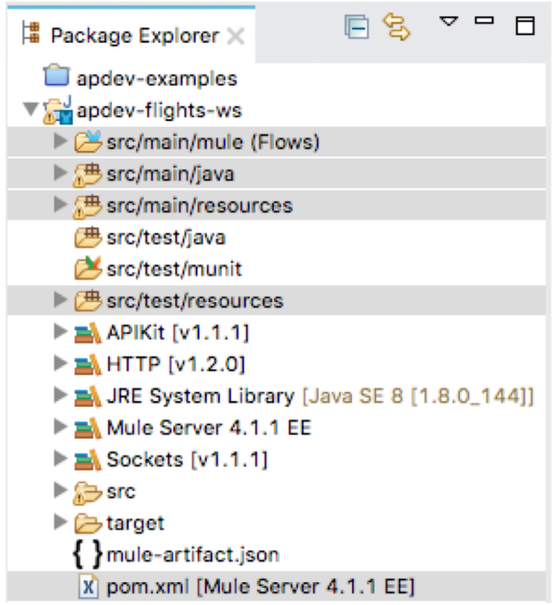
pom.xml

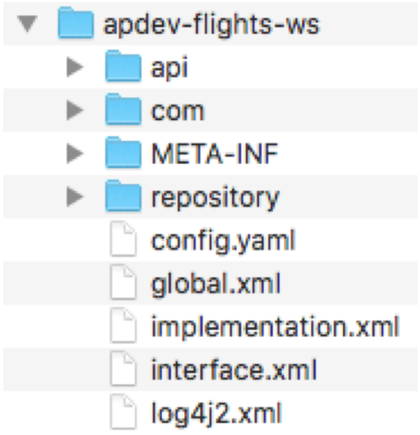
```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:/
3      <modelVersion>4.0.0</modelVersion>
4      <groupId>com.mulesoft</groupId>
5      <artifactId>apdev-flights-ws</artifactId>
6      <version>1.0.0-SNAPSHOT</version>
7      <packaging>mule-application</packaging>
8      <name>apdev-flights-ws</name>
9
10     <properties>..
16
17     <build>..
30
31     <dependencies>
32         <dependency>
33             <groupId>org.mule.modules</groupId>
34             <artifactId>mule-apikit-module</artifactId>
35             <version>1.1.1</version>
36             <classifier>mule-plugin</classifier>
37         </dependency>
38     <dependency>
39             <groupId>74922056-9245-48e0-99df-a8141d1d3e9f</groupId>
40             <artifactId>american4-flights-api</artifactId>
41             <version>1.0.2</version>
42             <classifier>mule-plugin</classifier>
43         </dependency>
44     <dependency>
45             <groupId>org.mule.connectors</groupId>
46             <artifactId>mule-http-connector</artifactId>
```

All contents © MuleSoft Inc.

Walkthrough 7-5: Create a well-organized Mule project

- Create a project with a new RAML file that is added to Anypoint Platform
- Review the project's configuration and properties files
- Create an application properties file and a global configuration file
- Add Java files and test resource files to the project
- Create and examine the contents of a deployable archive for the project



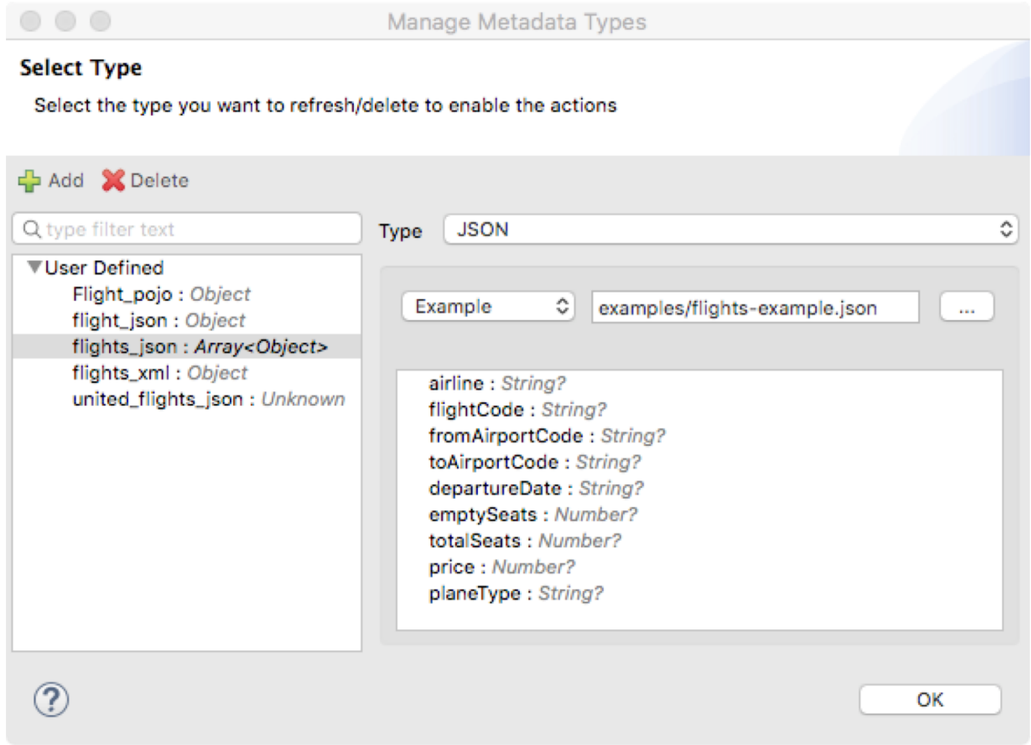


All contents © MuleSoft Inc.

Managing metadata for a project

Defining metadata

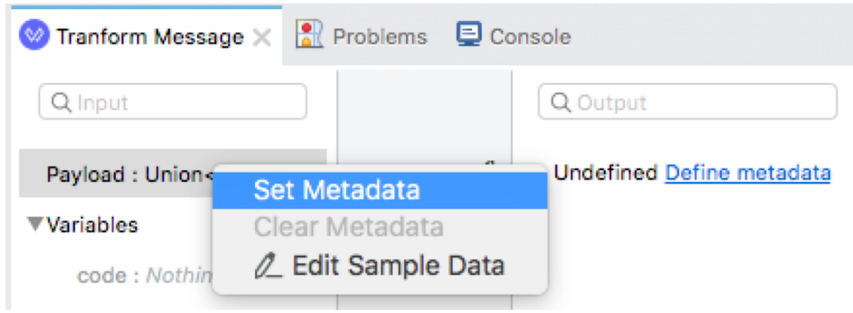
- It is often beneficial to define metadata for an application
  - For the output structures required for transformations
    - You did this in Module 4 when transforming database output to JSON defined in the API
  - For the output of operations that can connect to data sources of different structures
    - Like the HTTP Request connector
  - For the output of connectors that are not DataSense enabled
    - And do not automatically provide metadata about the expected input and output



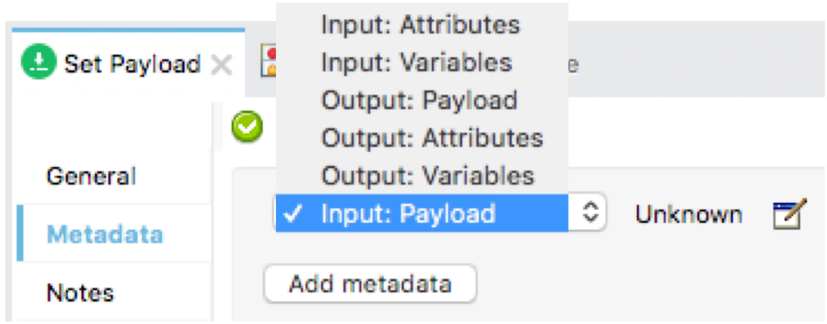
All contents © MuleSoft Inc.

Ways to access the Metadata Editor

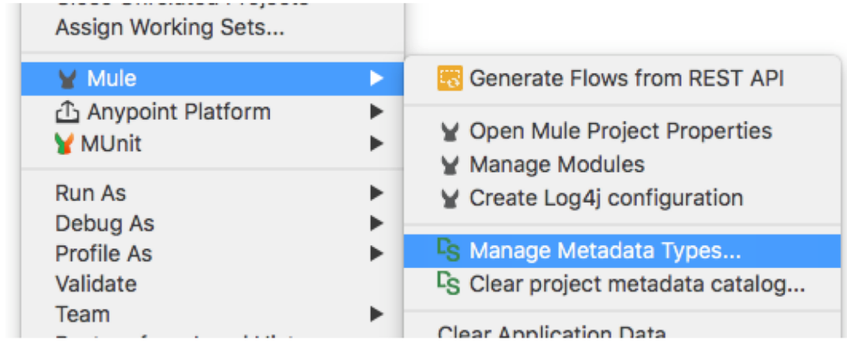
- From the Transform Message component



- From the Metadata tab in the properties view for most event processors



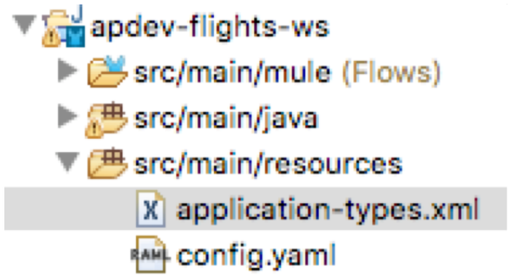
- From a project's menu in the Package Explorer



All contents © MuleSoft Inc.

Where is metadata stored?

- In application-types.xml in src/main/resources



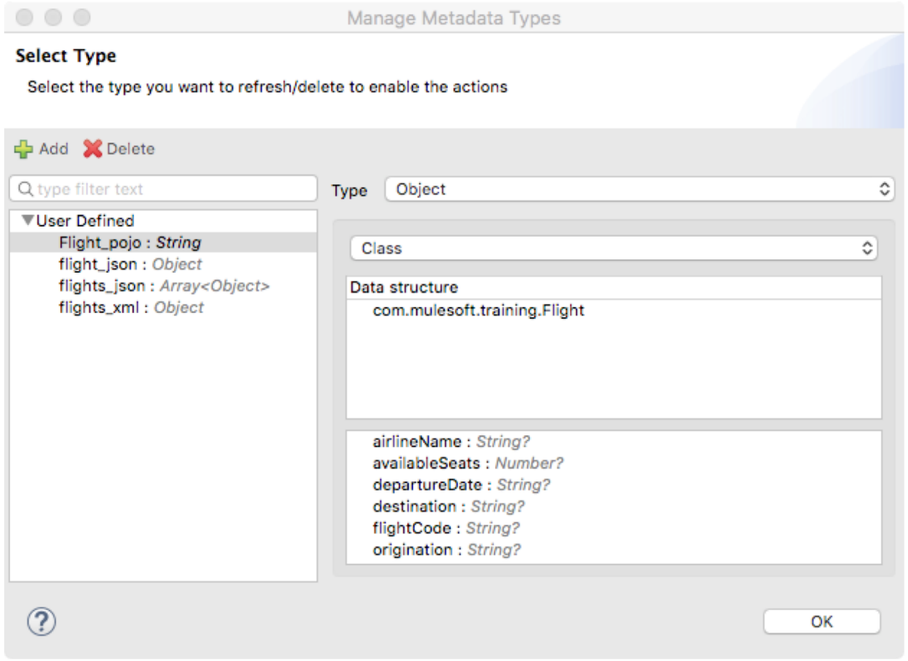
application-types.xml

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <types:mule xmlns:types="http://www.mulesoft.org/schema/mule/types">
3    <types:catalog>
4      <types:type name="united_flights_json" format="json">..
7      <types:type name="Flight_pojo" format="java">..
10     <types:type name="flights_json" format="json">..
13     <types:type name="flights_xml" format="xml">..
16     <types:type name="flight_json" format="json">
17       <types:example format="json" location="examples/flight-example.json"/>
18     </types:type>
19   </types:catalog>
20   <types:enrichment select="#d1d1a734-0050-43cf-baba-a52bfdbba85f">..
29   <types:enrichment select="#f17bea7e-21f3-42a2-a94e-926a936a6f01">..
38   <types:enrichment select="#659fe94d-369d-4f7b-b9e0-733914e41624">..
47   <types:enrichment select="#09e05593-c069-49e2-8562-4aac7e67cf10">..
56   <types:enrichment select="#6309e615-ede9-4808-b78b-ec75b0a58778">..
59   <types:enrichment select="#f963e3e4-5b3a-41bf-8c32-ef79bdce4a32">..
68   <types:enrichment select="#be12bc7d-22f8-49c5-8566-68741d80c449">
69     <types:processor-declaration>
70       <types:input-event>
71         <types:message>
72           <types:payload type="flight_json"/>
73         </types:message>
74       </types:input-event>
75     </types:processor-declaration>
76   </types:enrichment>
77   <types:enrichment select="#edcf1fd1-6237-40d1-ad8d-c196a9b00f21">..
86 </types:mule>
```

All contents © MuleSoft Inc.

Walkthrough 7-6: Manage metadata for a project

- Review existing metadata for the training-american-ws project
- Define new metadata to be used in transformations in the new apdev-flights-ws project
- Manage metadata



All contents © MuleSoft Inc.

Summary

Summary

- Separate functionality into **multiple applications** to allow managing and monitoring of them as separate entities
- Mule applications are **Maven** projects
  - A project's **POM** is used by Maven to build, report upon, and document a project
  - Maven builds an artifact (a Mule deployable archive JAR) from multiple dependencies (module JARs)
- Separate application functionality into **multiple configuration files** for easier development and maintenance
  - Encapsulate **global elements** into their own separate configuration file
- Share resources between applications by creating a **shared domain**

All contents © MuleSoft Inc.

Summary

- Define **application properties** in a YAML file and reference them as ${prop}
- Application **metadata** is stored in application-types.xml
- Create applications composed of multiple **flows** and **subflows** for better readability, maintenance, and reusability
- Use **Flow Reference** to calls flows synchronously
- Use the **VM connector** to pass messages between flows using asynchronous queues

All contents © MuleSoft Inc.