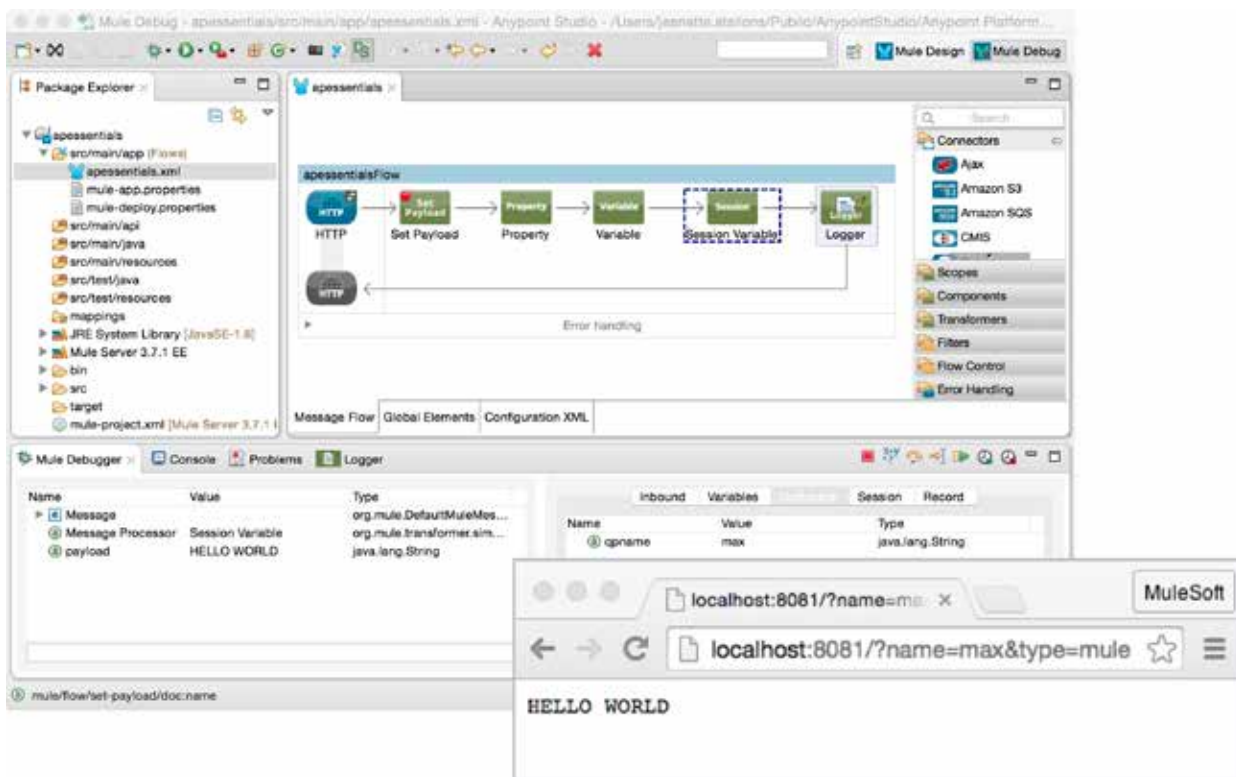


Module 2: Building Integration Applications with Anypoint Studio



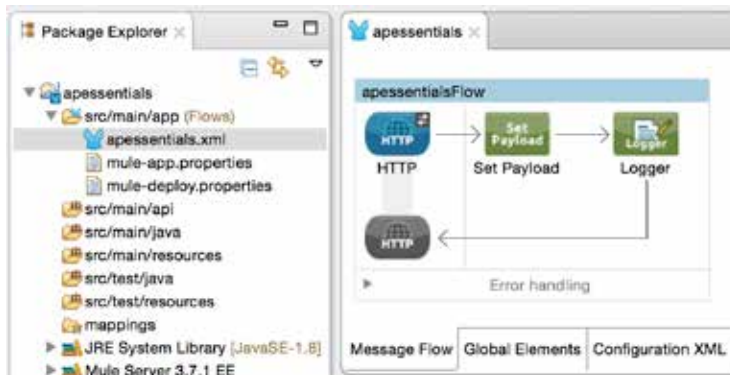
In this module, you will learn:

- About Mule applications, flows, messages, and message processors.
- To use Anypoint Studio to create flows graphically using connectors, transformers, components, scopes, and flow control elements.
- To build, run, test, and debug Mule applications.
- To read and write message properties.
- To write expressions with Mule Expression Language (MEL).
- To create variables.

Walkthrough 2-1: Create your first Mule application

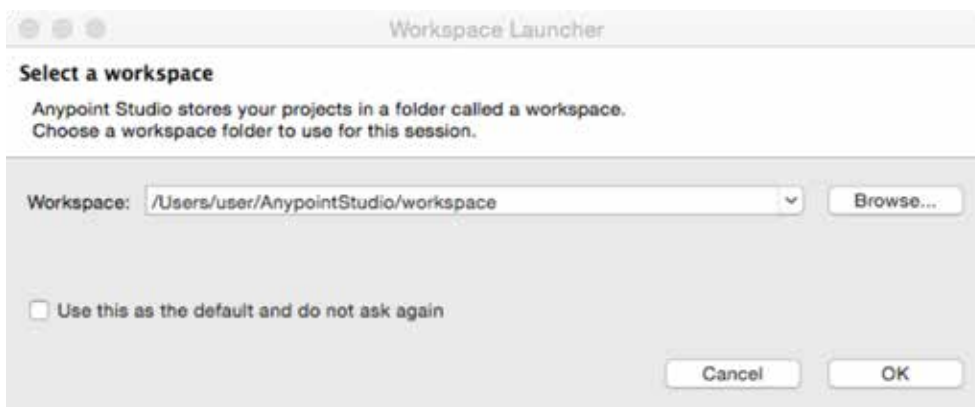
In this walkthrough, you will build your first Mule application. You will:

- Create a new Mule project with Anypoint Studio.
- Add a connector to receive requests at an endpoint.
- Display a message in the Anypoint Studio console.
- Set the message payload.



Launch Anypoint Studio

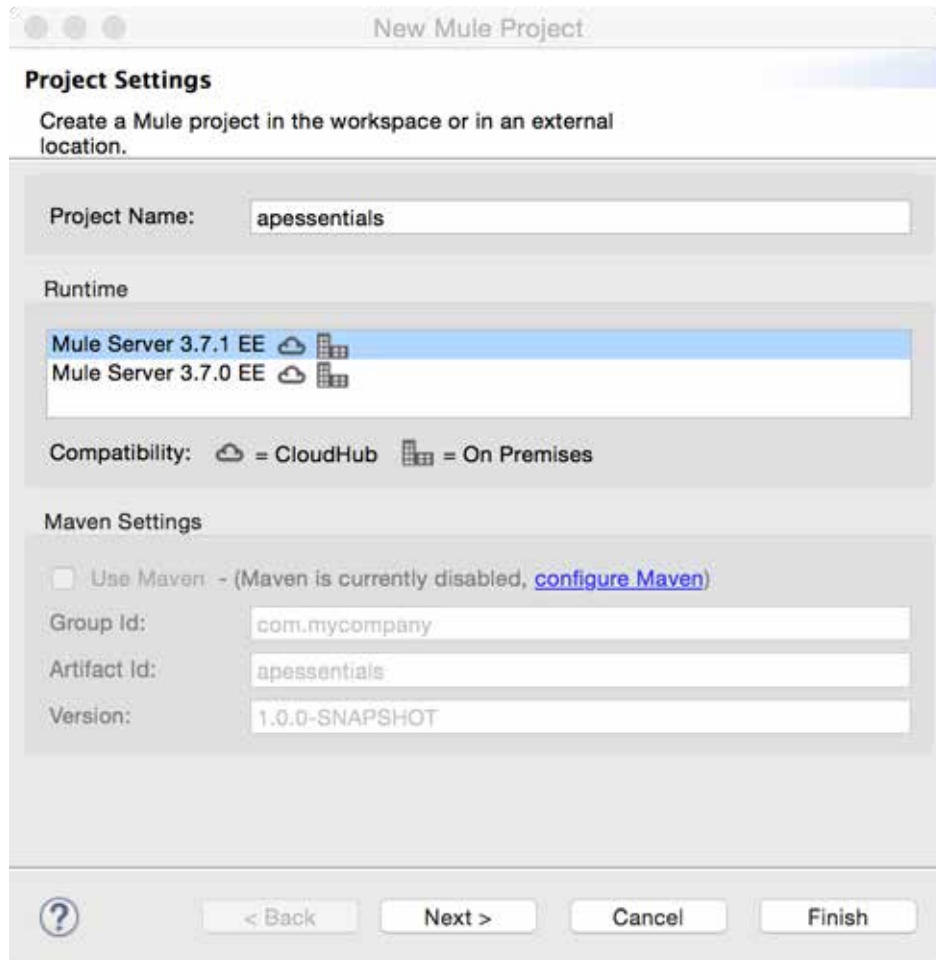
1. Open Anypoint Studio.
2. In the Workspace Launcher dialog box, look at the location of the default workspace; change the workspace location if you want.



3. Click OK to select the workspace; Anypoint Studio should open.
4. If you get a Welcome Page, click the X on the tab to close it.
5. If you get an Updates Available pop-up in the lower-right corner of the application, click it and install the available updates.

Create a project

6. Select File > New > Mule Project.
7. Set the Project Name to apessentials.
8. Ensure the Runtime is set to the latest version of the Mule Server.
9. Click Finish.



Create an HTTP connector endpoint to receive requests

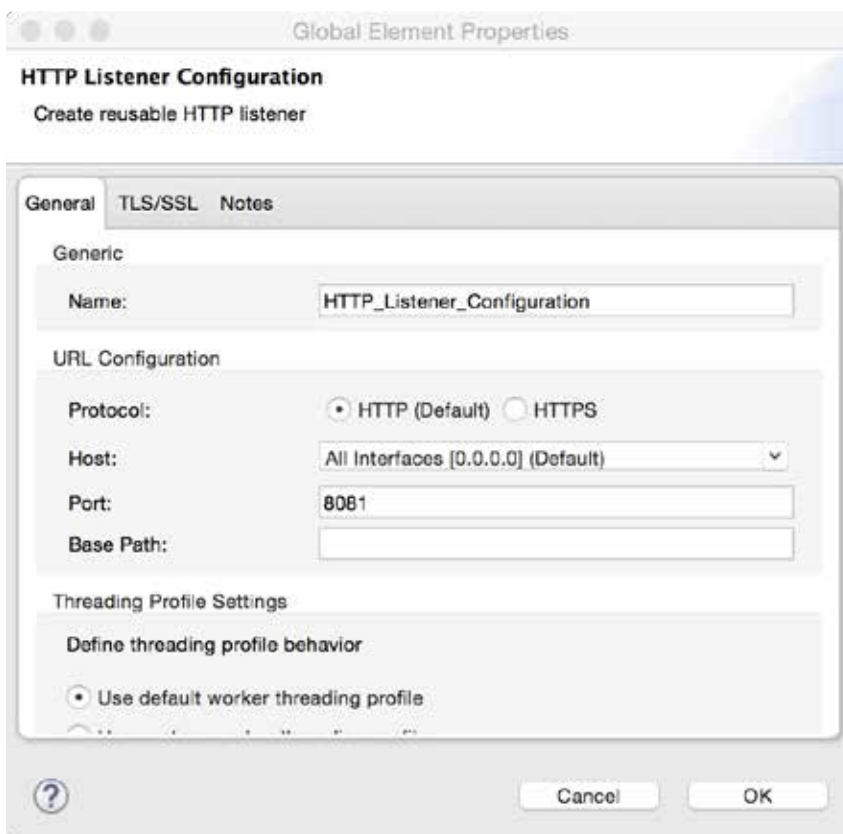
10. Drag an HTTP connector from the palette to the canvas.



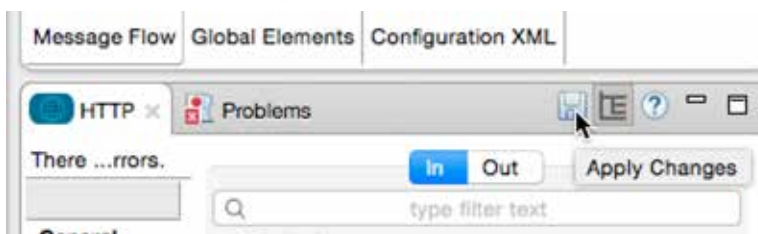
11. Double-click the HTTP connector endpoint.
12. In the Mule Properties view, click the Add button next to connector configuration.



13. In the Global Element Properties dialog box, look at the default values and click OK.

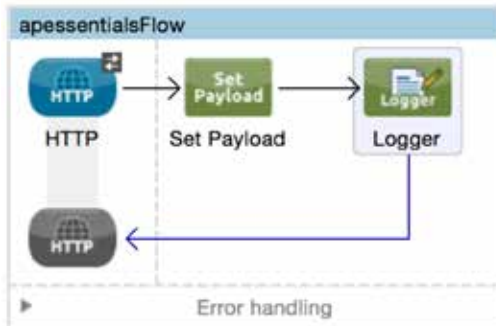


14. Click the Apply Changes button; the errors in the Problems view should disappear.



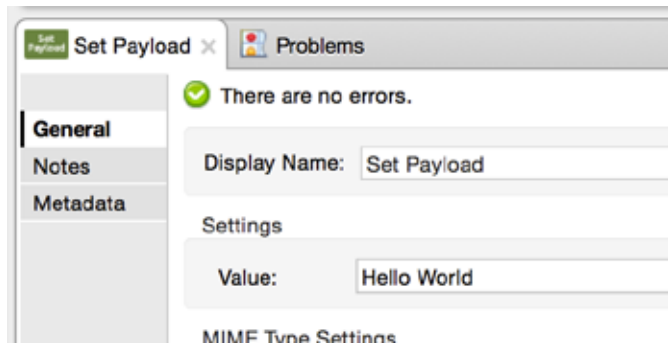
Display data

15. Drag a Set Payload transformer from the palette into the process section of the flow.
16. Drag in a Logger component and drop it after the Set Payload transformer.

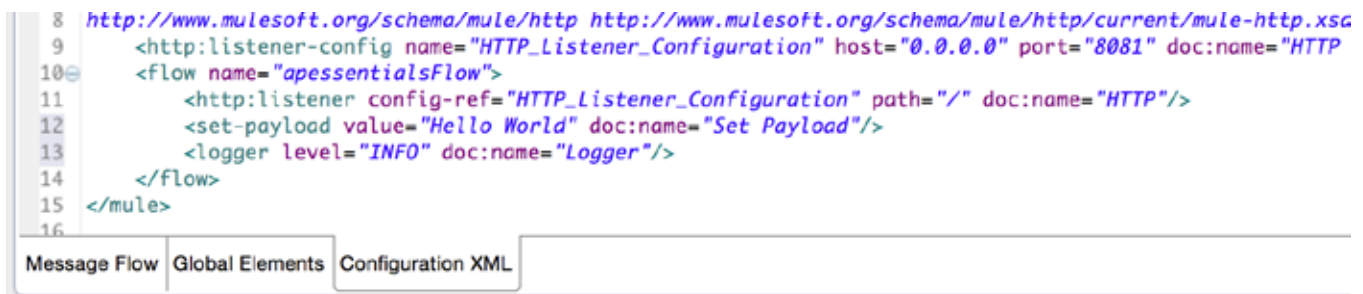


Configure the Set Payload transformer

17. Double-click the Set Payload transformer.
18. In the Properties view, set the value field to Hello World.



19. Click the Configuration XML tab at the bottom of the canvas and examine the corresponding XML.

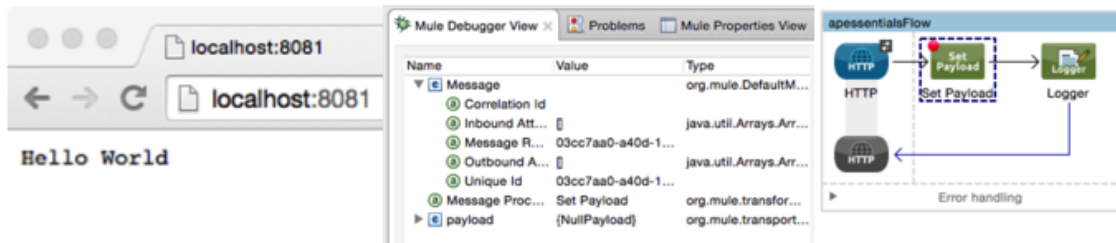


20. Click the Message Flow tab to return to the canvas.
21. Click the Save button or press Ctrl+S to save the file.

Walkthrough 2-2: Run, test, and debug an application

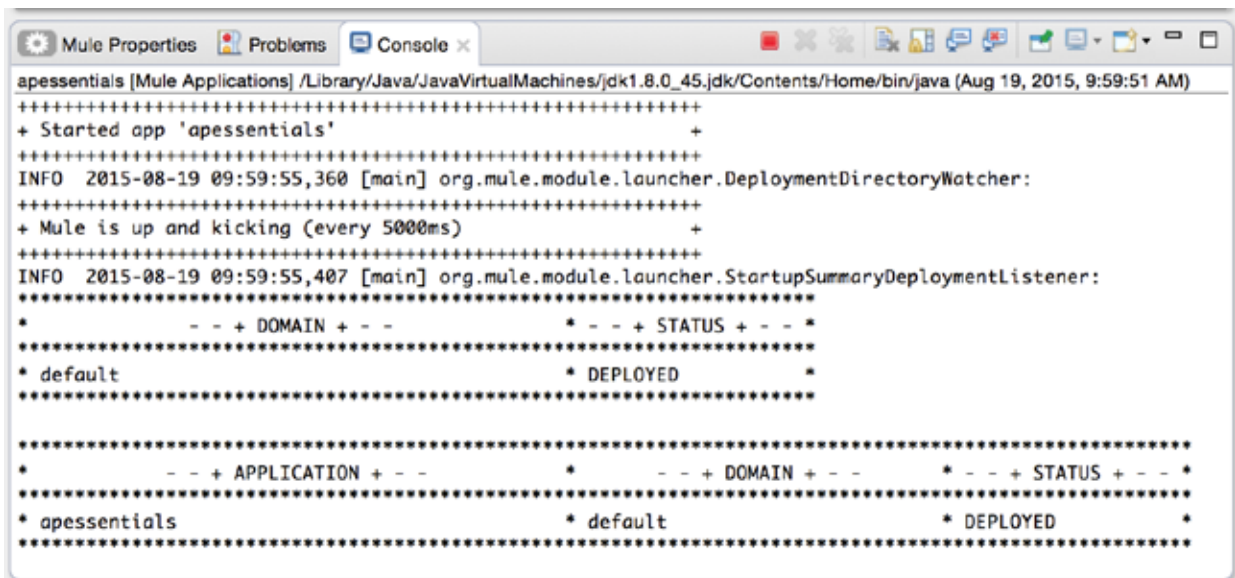
In this walkthrough, you will run, test, and debug your first Mule application. You will:

- Run a Mule application using the embedded Mule runtime.
- Make an HTTP request to the endpoint via a web browser or a tool like cURL or Postman.
- Receive a response with the text Hello World.
- Redeploy an application.
- Use the Mule Debugger to debug an application.



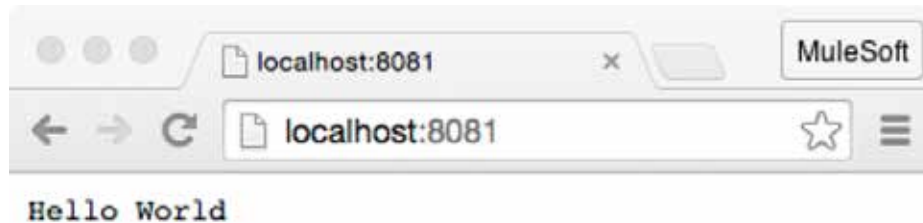
Run the application

1. From the main menu bar, select Run > Run As > Mule Application.
2. Watch the Console view; it should display information letting you know that both the Mule runtime and the apessentials application started.



Test the application

3. Send an HTTP request to <http://localhost:8081> through a browser or tool like cURL or Postman; you should see Hello World displayed.



4. Return to the console in Anypoint Studio.
5. Examine the last entry.

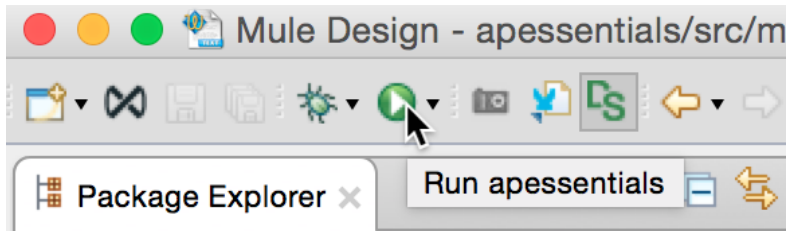


6. Click the red Terminate button to stop the application and the Mule runtime.
7. Answer the following questions.
 - What triggered all the output you saw in the console?
 - What is the last thing displayed?
 - What Java class represents the payload?
 - What are the inbound and outbound properties on the message?

Rerun the application

8. Change the value of the Set Payload transformer to a different value.
9. Save the file.

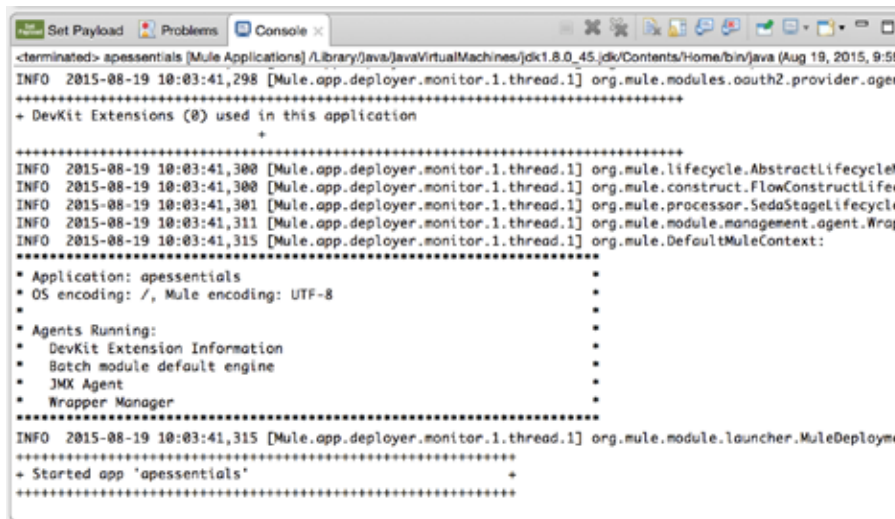
- Click the Run button and watch the console; you should the application is redeployed but the Mule runtime is also restarted.



Note: You may want to modify your perspective so you always see the console. 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.

Redeploy the application

- Change the value of the Set Payload transformer again to a different value.
- Click the Apply Changes button in the upper-right corner of the Properties view and watch the console; you should see the application is redeployed but the Mule runtime is not restarted.



Debug the application

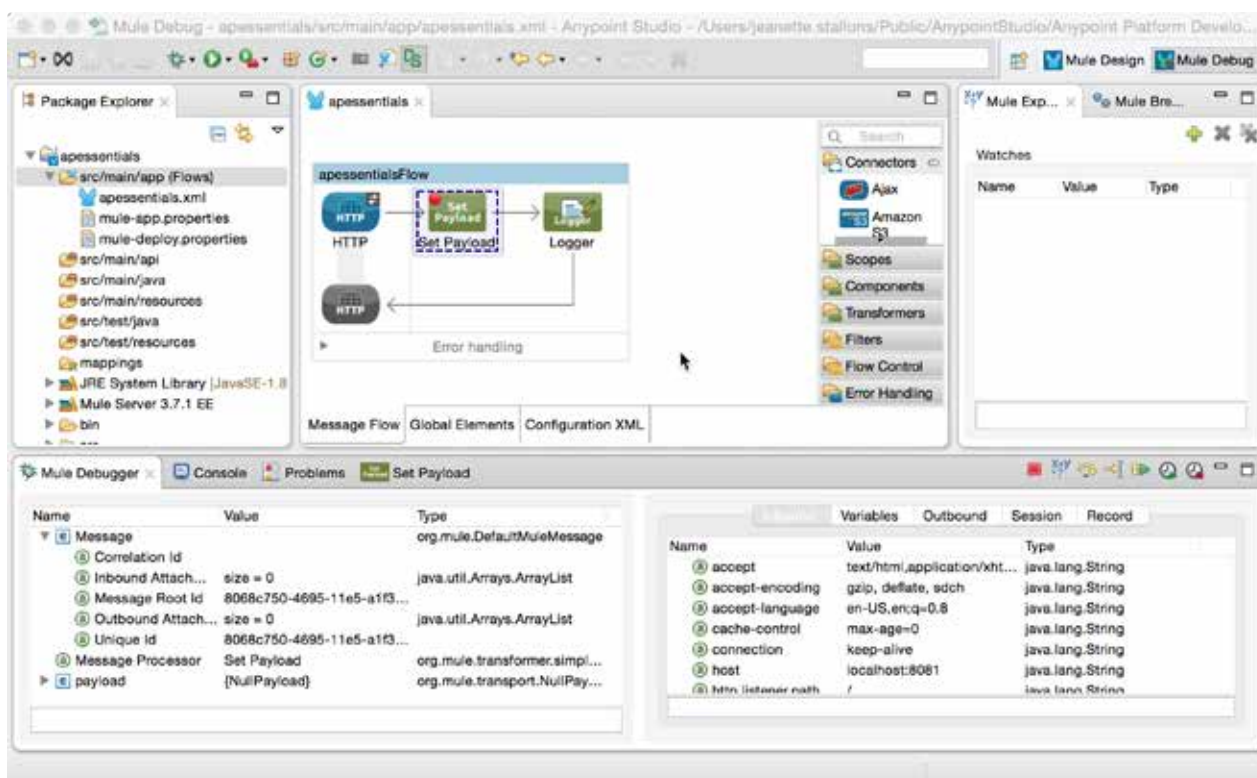
- Right-click the Set Payload component and select Toggle breakpoint.
- Select Run > Debug or click the Debug button in the main menu bar.
- If you get an Accept incoming network connections dialog box, click Allow.
- If you get a Confirm Perspective Switch dialog box, click Yes.

Note: If you do not want to get this dialog box again, check Remember my decision.

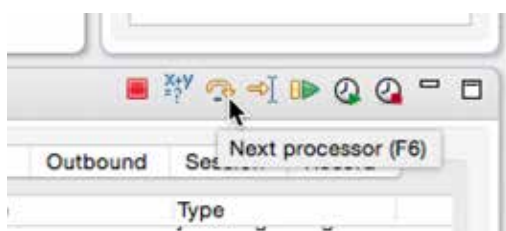
17. In the Mule ESB Runtime window, click Yes to stop the Mule runtime and restart it connected to the Mule Debugger.



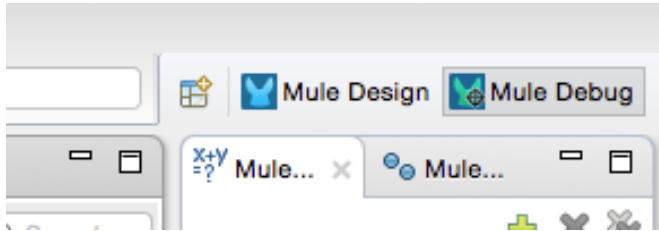
18. Make another request to <http://localhost:8081> with a browser or other tool like cURL or Postman.
19. Return to Anypoint Studio and look at the Mule Debugger view.
20. Drill-down and explore the properties and variables.



21. Click the Next processor button.



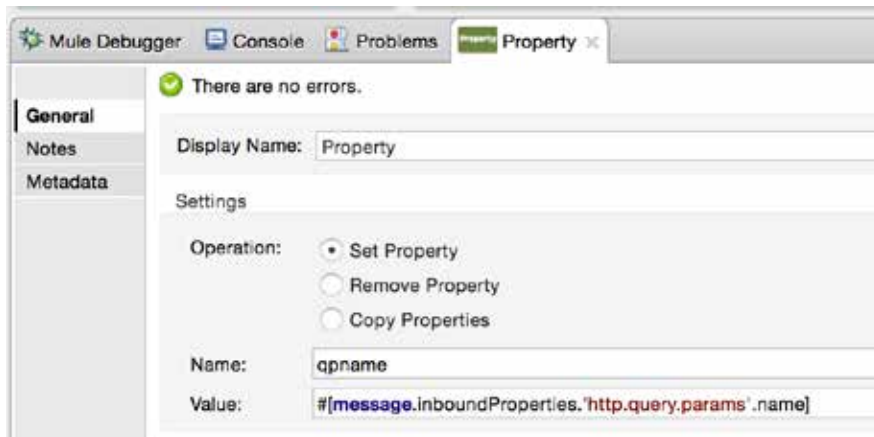
22. Look at the new value of the payload.
23. Step through the rest of the application.
24. Stop the application and the Mule runtime.
25. Click the Mule Design tab in the upper-right corner of the application to switch perspectives.



Walkthrough 2-3: Read and write message properties

In this walkthrough, you will manipulate message properties. You will:

- Write MEL expressions.
- Use the Debugger to read inbound and outbound message properties.
- Use the Property transformer to set outbound message properties.



Use an expression to set the payload

1. Navigate to the Properties view for the Set Payload transformer.
2. Change the value to an expression.

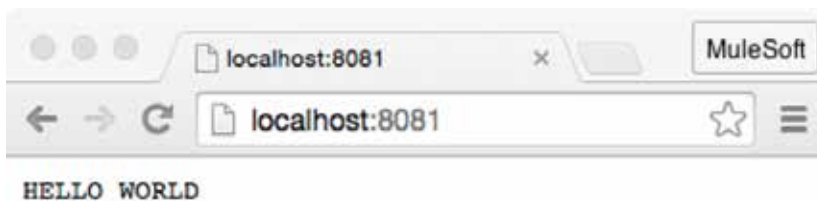
```
#['Hello World']
```

3. Click the Apply Changes button and run the application; you have to run it instead of redeploying it because you stopped the Mule runtime when you stopped the Debugger.
4. Make a request to <http://localhost:8081>; the application should work as before.
5. Return to the Set Payload expression and use autocomplete to use the `toUpperCase()` method to return the value in upper case.

```
#['Hello World'.toUpperCase()]
```

Note: Press Ctrl+Space to trigger autocomplete if it does not appear.

6. Click the Apply Changes button to redeploy the application.
7. Make a request to the endpoint; the return string should now be in upper case.

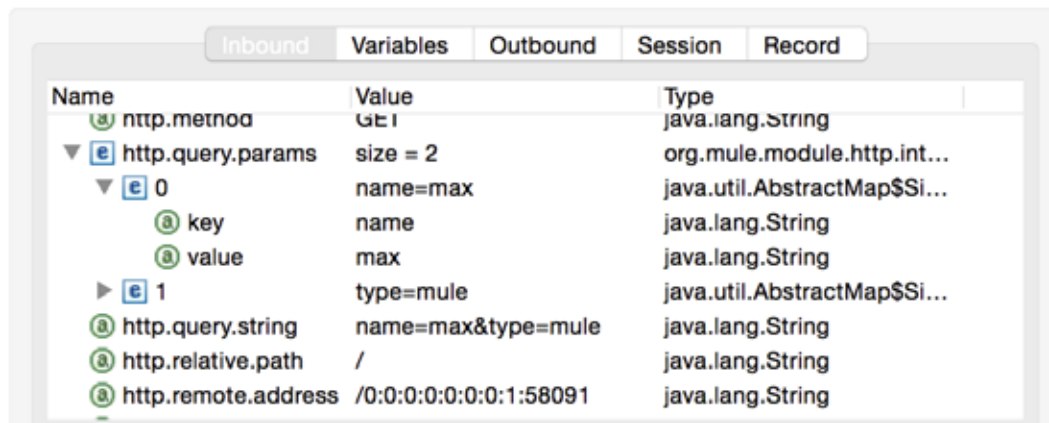


Use an expression to display info to the console

8. Navigate to the Properties view for the Logger component.
9. Set the message to display the http.query.params property of the message inbound properties.

Message:	<code>#[message.inboundProperties.'http.query.params']</code>
Level:	INFO (Default)
Category:	

10. Apply the changes and debug the application.
11. Make a request to the endpoint with a couple of query parameters; for example, <http://localhost:8081/?name=max&type=mule>.
12. Return to the Mule Debugger view and locate your query parameters.



Name	Value	Type
http.method	GET	java.lang.String
http.query.params	size = 2	org.mule.module.http.int...
0	name=max	java.util.AbstractMap\$Si...
key	name	java.lang.String
value	max	java.lang.String
1	type=mule	java.util.AbstractMap\$Si...
http.query.string	name=max&type=mule	java.lang.String
http.relative.path	/	java.lang.String
http.remote.address	/0:0:0:0:0:0:1:58091	java.lang.String

13. Step through the application and watch the property values.
14. Navigate to the console; you should see a ParameterMap object listed.

```
INFO 2015-08-19 10:29:32,305 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.api.processor.LoggerMessageProcessor: ParameterMap{name=[max], type=[mule]}
```

15. Modify the Logger to display one of your query parameters.

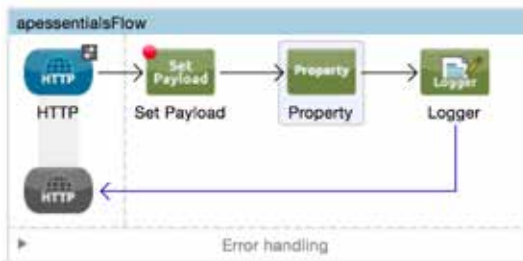
```
#[message.inboundProperties.'http.query.params'.name]
```

16. Click Apply Changes to redeploy the application and make a request to the endpoint with query parameters again.
17. Click the Resume button in the Mule Debugger view.
18. Return to the console; you should now see the value of the parameter displayed.

```
INFO 2015-08-19 10:27:14,586 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.api.processor.LoggerMessageProcessor: max
```

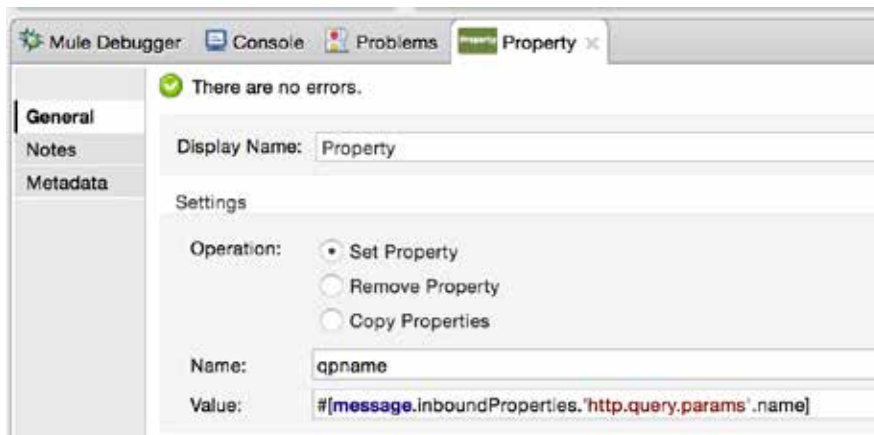
Set an outbound property

19. Add a Property transformer between the Set Payload and Logger processors.



Note: The Message Properties transformer has been deprecated; it has been replaced by the collection of Property, Variable, Session Variable, and Attachment transformers.

20. In the Properties view for the Property transformer, select Set Property.
21. Set the name to qpname (or some other value) and the value to an expression that evaluates one of your query parameters.



22. Modify the Logger to display the value of this new outbound property.
- `#[message.outboundProperties.qpname]`
23. Click the Apply Changes button to redeploy the application with a connection to the Mule Debugger.
24. Make another request to the endpoint with query parameters.
25. Click the Outbound tab in the Mule Debugger view.
26. Step to the Logger; you should see your new outbound property.

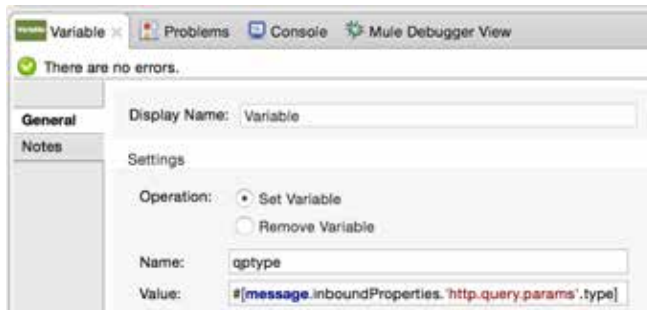


27. Click the Resume button.
28. Look at the console; you should see the value of your variable displayed.

Walkthrough 2-4: Read and write variables

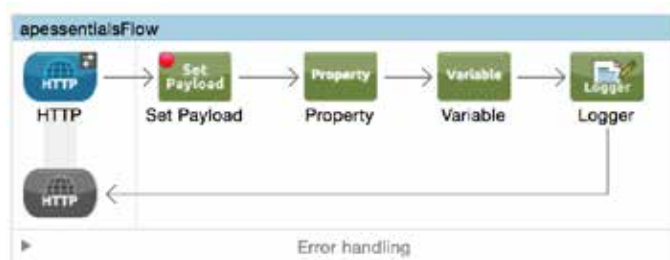
In this walkthrough, you will create flow and session variables. You will:

- Use the Variable transformer to create flow variables.
- Use the Session transformer to create session variables.

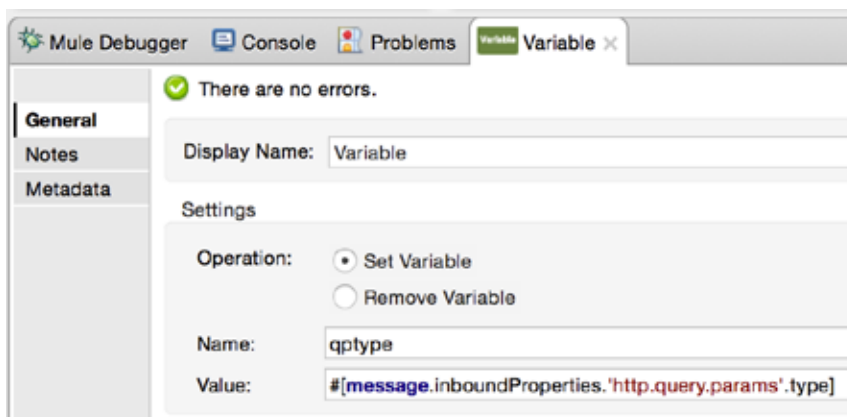


Create a flow variable

1. Add a Variable transformer between the Property and Logger processors.



2. In the Variable Properties view, select Set Variable.
3. Set the name to qptype (or some other value) and the value to your second query parameter.



4. Modify the Logger to also display the value of this new variable.

```
#['Name: ' + message.outboundProperties.qpname + ' Type: ' +  
flowVars.qptype]
```

Note: The flowVars is optional.

5. Save the file (or click Apply Changes) to redeploy the application in debug mode.
6. Make a request to the endpoint with query parameters again.
7. Click the Variables tab in the Mule Debugger view.
8. Step to the Logger; you should see your new flow variable.

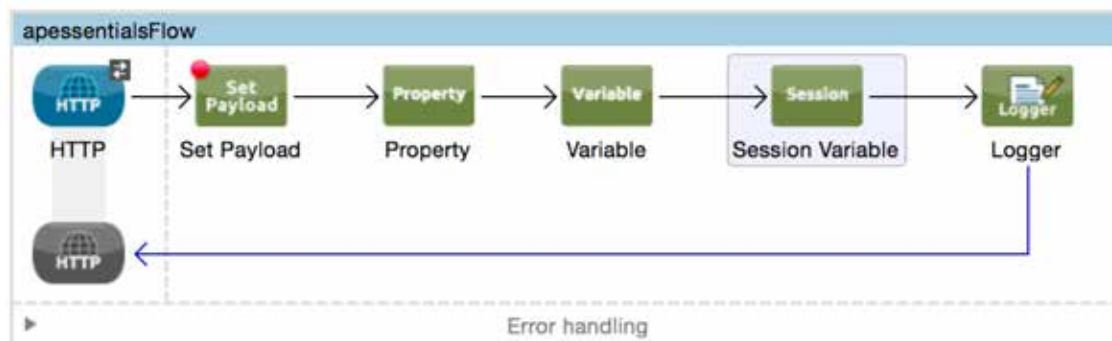
Inbound Variables Outbound Session Record		
Name	Value	Type
qptype	mule	java.lang.String

9. Click the Resume button.
10. Look at the console; you should see the value of your outbound property and the value of your new flow variable displayed.

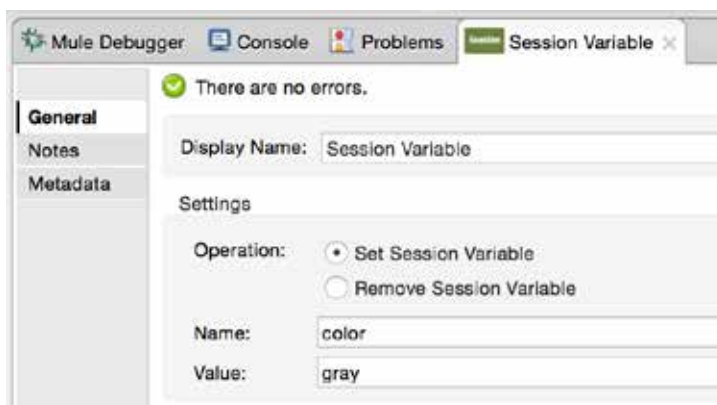
```
INFO 2015-08-19 10:45:33,189 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.api.processor.LoggerMessageProcessor: Name: max Type: mule
```

Create a session variable

11. Add a Session Variable transformer between the Variable and Logger processors.



12. In the Session Variable Properties view, select Set Session Variable.
13. Set the name to color (or some other value) and give it any value, static or dynamic.



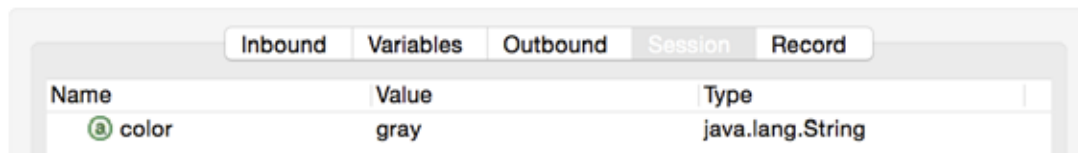
14. Modify the Logger to also display the session variable.

```
#['Name: ' + message.outboundProperties.qpname + ' Type: ' +  
flowVars.qptype + ' Color: ' + sessionVars.color]
```

15. Save and redeploy the application and make a request to the endpoint with query parameters again.

16. Click the Session tab in the Mule Debugger view.

17. Step to the Logger; you should see your new session variable.



18. Click the Resume button.

19. Look at the console; you should see the value of your session variable displayed.

20. Stop the Mule runtime.

Note: You will explore the persistence of these variables in a later module, Refactoring Mule Applications.