

# Seminario 8: Introducción a Mule con Anypoint

## Sistemas Distribuidos

Juan Boubeta (editado por Antonio Balderas, Pablo García y Salvador Gutiérrez)

Departamento de Ingeniería Informática  
Universidad de Cádiz



Escuela Superior de Ingeniería  
Dpto. de Ingeniería Informática



Curso 2019 – 2020

# Índice

- 1 Introducción
- 2 Instalación de Anypoint Studio
- 3 Visión general de la herramienta
- 4 Esquema de un flujo típico de Mule
- 5 Primer proyecto Mule

# Sección 1 | Introducción

# ¿Qué es Anypoint Studio?

- Interfaz gráfica que abstrae al usuario de los detalles más técnicos de Mule ESB.
- En lugar de tener que escribir “a mano” el código XML para crear aplicaciones Mule; Anypoint Studio se encarga de ello.
- Los elementos necesarios para modelar y configurar aplicaciones Mule se incorporan al canvas del editor mediante *drag and drop*.
- Una aplicación Studio puede ser incluso desplegada en la nube (véase *CloudHub* para más información).
- Está basado en Eclipse y proporciona dos entornos de desarrollo que pueden utilizarse para crear aplicaciones Mule:
  - Un editor *drag and drop* visual.
  - Un editor XML.
- Lo que se desarrolle o configure en uno de los editores se actualizará automáticamente en el otro.

# Anypoint Studio - Editor visual

The screenshot displays the Anypoint Studio IDE interface for Mule Design. The main canvas shows a flow named 'mybrokerFlow' with the following components:

- HTTP Connector:** Receives incoming requests.
- Byte Array to String Transformer:** Converts the received data from a byte array to a string.
- JMS Connector:** Sends the processed data to a JMS queue.
- Error handling:** A path is shown for handling errors, leading back to the HTTP connector.

The left sidebar (Package Explorer) shows the project structure, including the 'mybroker' project and its source files. The right sidebar (Search and Connectors) provides a search bar and a list of available connectors and scopes. The bottom console shows the startup logs for the Mule application, indicating that the application is running successfully.

# Anypoint Studio - Editor XML

The screenshot displays the Anypoint Studio IDE with the following components:

- Package Explorer (Left):** Shows the project structure for 'mybroker', including 'src/main/app (Flows)' with 'mybroker.xml', 'mule-app.properties', and 'mule-deploy.properties', as well as other source folders like 'src/main/api', 'src/main/java', 'src/main/resources', 'src/test/java', and 'src/test/resources'.
- Main Editor (Center):** Displays the XML content of 'mybroker.xml'. The XML is a Mule configuration file with the following structure:
 

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns:jms="http://www.mulesoft.org
4   xmlns:spring="http://www.springframework.org/schema/beans" version="EE-3.6.0"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org
7   http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mul
8   http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mul
9   http://www.mulesoft.org/schema/mule/jms http://www.mulesoft.org/schema/mule/jms/current/mule-
10  <http:listener-config name="HTTP_Listener_Configuration" host="localhost" port="8081" doc
11  <jms:activemq-connector name="jmsConnector" brokerURL="tcp://localhost:61616" validateConn
12  <flow name="mybrokerFlow">
13    <http:listener config-ref="HTTP_Listener_Configuration" path="/mybroker" doc:name="HTT
14    <byte-array-to-string-transformer doc:name="Byte Array to String"/>
15    <jms:outbound-endpoint queue="mybroker" connector-ref="jmsConnector" doc:name="JMS"/>
16  </flow>
17 </mule>
18
      
```
- Bottom Console:** Shows the Mule application logs, including the startup message:
 

```

mybroker [Mule Applications] /usr/lib/jvm/java-7-openjdk-i386/bin/java (Mar 21, 2017, 1:30:27 PM)
+ Mule is up and kicking (every 5000ms)
+
*****
INFO 2017-03-21 13:30:40,074 [main] org.mule.module.launcher.StartupSummaryDeploymentListener:
*****
      
```

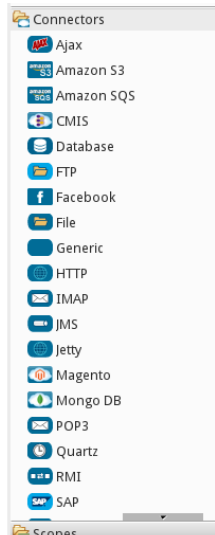
## Sección 2 | Visión general de la herramienta

## Connectors

Permiten que las aplicaciones Mule puedan comunicarse con el “mundo” exterior. Se clasifican en:

**Inbound** La aplicación recibirá información del exterior.

**Outbound** La aplicación enviará información al exterior.



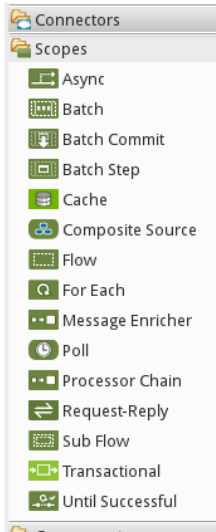


# Scopes

Proporcionan diferentes formas de combinar (agrupar) varios procesadores de mensajes con el objetivo de:

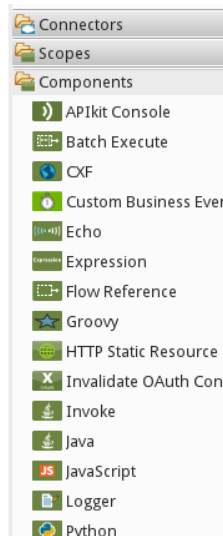
- Mejorar la legibilidad del código XML.
- Implementar procesamiento paralelo.
- Crear secuencias de bloques reusables.

Denominaremos “procesadores de mensajes” a los bloques que permiten filtrar, enriquecer, encaminar o validar los mensajes.



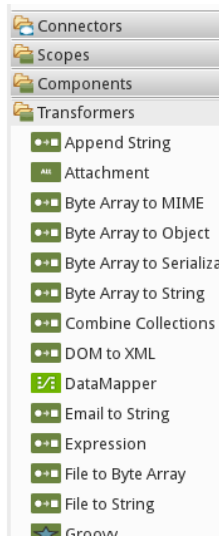
# Components

- Añaden funcionalidad a un flujo como *logging* e impresión por pantalla.
- Además, también facilitan la integración *Software as a Service* (SaaS) proporcionando “shells” específicos de lenguaje que permiten definir una lógica de negocio con código personalizado para las aplicaciones Mule.
- Un componente recibe, procesa y devuelve mensajes.
- Es un objeto en el que uno de sus métodos será invocado cuando reciba un mensaje.



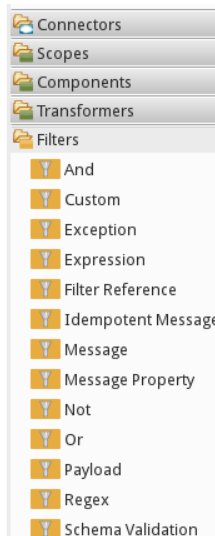
# Transformers

Se encargan de transformar o enriquecer los mensajes (cabecera y cuerpo del mensaje).



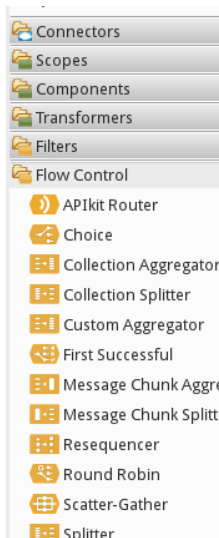
# Filters

Determinan si un mensaje puede continuar a través del flujo de la aplicación, o si debe rechazarse.



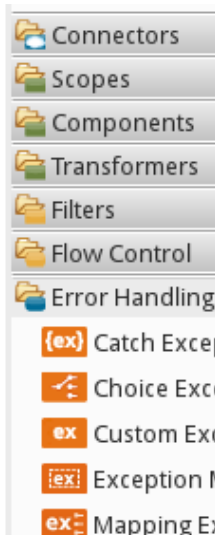
# Flow controls

- Especifican cómo los mensajes serán encaminados hacia distintos procesadores de mensajes dentro de un flujo.
- También pueden procesar mensajes (agregación, separación...) antes de encaminarlos a otros procesadores de mensajes.



# Error handlers

Ofrecen varios procedimientos para manejar excepciones bajo ciertas circunstancias.



## Sección 3 | Esquema de un flujo típico de Mule

# Esquema de un flujo típico de Mule I

- 1 **Una fuente de mensajes:** uno o más *endpoints* activan el flujo cada vez que llega un mensaje.
- 2 **Un filtro:** puede ser embebido en la fuente de mensajes o conectado a esta fuente; debe identificar mensajes inválidos y rechazar su paso al resto del flujo.
- 3 **Un transformador:** puede convertir los mensajes de entrada en un formato de datos consumible por otros procesadores de mensajes del flujo.
- 4 **Un enriquecedor de mensajes:** puede añadir información relevante en un mensaje. Por ejemplo, si el mensaje llega solamente con el DNI de una persona, podría añadirse al mensaje su nombre y apellidos.



# Esquema de un flujo típico de Mule II

- 5 **Un componente:** una vez preparado el mensaje para ser procesado, normalmente será enviado a un componente que se encargará de procesarlo de una determinada forma según su contenido. A veces también se utilizan BD externas o API (ej. Salesforce) como *cloud connectors*.
- 6 Los últimos “pasos” de un flujo pueden ser muy distintos, por ejemplo:
  - Se devuelve una respuesta al emisor original del mensaje.
  - Los resultados del procesamiento son almacenados en una base de datos o enviados a terceros (ej. correo electrónico).

## Sección 4 | Primer proyecto Mule

# Enunciado

Como primer proyecto crearemos un bróker de mensajería:

- El ESB recibirá por POST los datos de un formulario HTML
- Transformará el mensaje recibido
- Y lo enviará a una cola que implemente la especificación de JMS (Java Message Service)

# Crear un proyecto Mule

- Empezamos creando un nuevo proyecto: File → New → Mule project
- Damos un nombre al proyecto y pulsamos en siguiente

**New Mule Project**

Create a Mule project in the workspace or in an external location.

**Project Settings**

Project Name:

**Runtime**

CloudHub Mule Runtime (Dec 2013) ☁  
Mule Server 3.4.1 EE ☁

Compatibility: ☁ = CloudHub = On Premise

**Maven Settings**

☐ Create POM file for project and maintain with Maven (enable Maven Support in Preferences)

Group Id:   
Artifact Id:   
Version:

**Version Control System support**

☐ Create a .gitignore file for the project with default ignores for Studio projects

# Crear un proyecto Mule (cont.)

- Siguiente ...

The screenshot shows the 'New Mule Project' dialog box. It has a title bar 'New Mule Project' with a close button. Below the title bar is a section 'Create a Java Project' with the instruction 'Create a Java project in the workspace or in an external location.' The 'Project name' field contains 'mybroker'. The 'Use default location' checkbox is checked. The 'Location' field shows the path '/home/antonio/MuleStudio/workspace/mybroker' with a 'Browse...' button. The 'JRE' section has three radio buttons: 'Use an execution environment JRE:' (selected), 'Use a project specific JRE:', and 'Use default JRE (currently 'java-7-openjdk-amd64')'. The first option has a dropdown menu showing 'javaSE-1.7'. The second option has a dropdown menu showing 'java-7-openjdk-amd64'. There is a 'Configure JREs...' link. The 'Project layout' section has two radio buttons: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (selected). There is a 'Configure default...' link. The 'Working sets' section has an 'Add project to working sets' checkbox and a 'Working sets:' field with a dropdown menu and a 'Select...' button.

**New Mule Project**

**Create a Java Project**  
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:

**JRE**

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-7-openjdk-amd64') [Configure JREs...](#)

**Project layout**

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

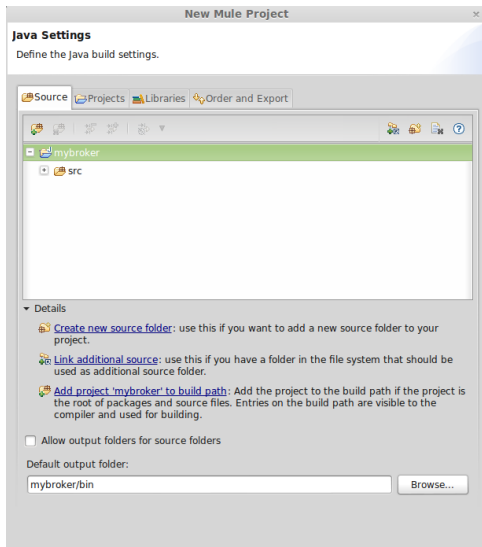
**Working sets**

☐ Add project to working sets

Working sets:

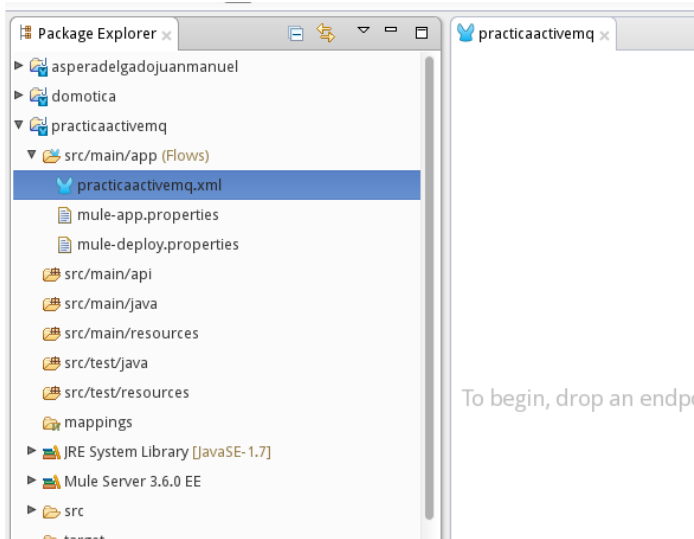
# Crear un proyecto Mule (cont.)

- Y finalizar



# Crear un proyecto Mule (cont.)

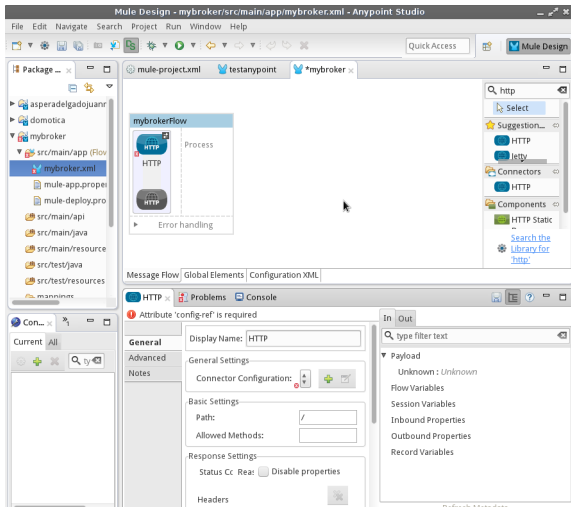
- Nuestro proyecto creado



To begin, drop an endpo

# Crear un proyecto Mule: HTTP (cont.)

- Seleccionamos un **Connector** de tipo **HTTP** y lo arrastramos al area de trabajo





# Crear un proyecto Mule: HTTP (cont.)

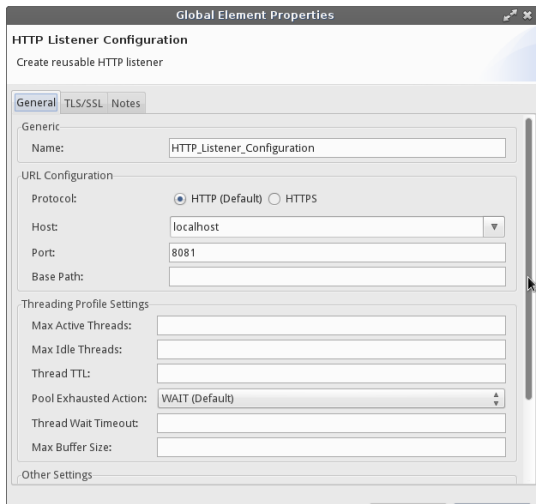
- Seleccionando el objeto **HTTP** Sobre el area de trabajo accedemos en la parte inferior a las propiedades y establecemos el PATH (la URL que será llamada desde fuera)

The screenshot shows the Mule IDE interface with the HTTP connector selected. The top bar includes tabs for HTTP, Problems, and Console. A green checkmark icon indicates that there are no errors. The configuration panel on the left has three tabs: General, Advanced, and Notes. The General tab is active, displaying the following settings:

- Display Name:** HTTP
- General Settings**
  - Connector Configuration:** HTTP\_Listener\_Configuration
- Basic Settings**
  - Path:** mybroker
  - Allowed Methods:** (empty field)
- Response Settings**
  - Status Code:** (empty field)
  - Reason:** (empty field)

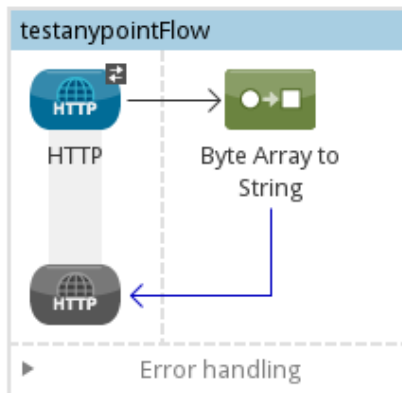
# Crear un proyecto Mule: HTTP (cont.)

- Añadimos un nuevo Connector Configuration: host (localhost) y port (8081)



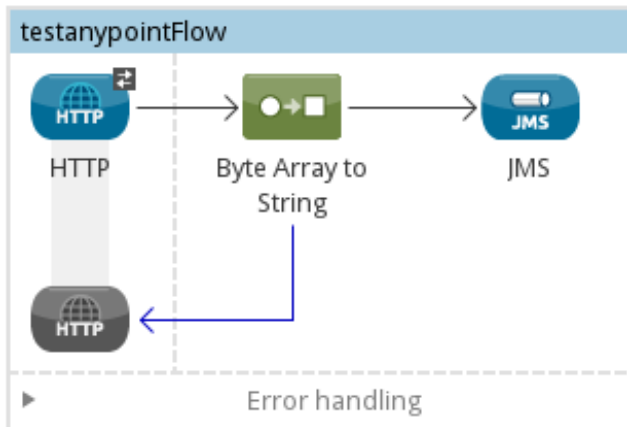
# Crear un proyecto Mule: Byte Array to String

- Seleccionamos un **Transformer** de tipo **Byte Array to String** y lo arrastramos al area de trabajo a continuación del endpoint **HTTP**
- Se necesita el transformador para convertir la entrada HTTP POST a una instancia de tipo cadena.



# Crear un proyecto Mule: JMS

- Seleccionamos un **Connector** de tipo **JMS** y lo arrastramos al area de trabajo a continuación del transformador
- La salida del HTTP enviará la cadena a la cola JMS especificada



# Crear un proyecto Mule: JMS (cont.)

- Establecemos el nombre de la cola (mybroker) y creamos un nuevo connector configuration

The screenshot shows the Mule IDE configuration window for a JMS connector. The window has tabs for 'JMS', 'Problems', and 'Console'. A red warning icon and the message 'Attribute 'action' is required' are visible at the top. On the left, there is a sidebar with tabs for 'General', 'Advanced', 'Transformers', and 'Notes'. The 'General' tab is selected, showing the 'Basic Settings' section. The 'Display Name' is set to 'JMS'. The 'Exchange Pattern' is set to 'one-way (Default)'. The 'Queue' is set to 'mybroker'. The 'Connector Configuration' is set to 'jmsConnector'. The 'Transaction' section is partially visible at the bottom, showing 'Type' set to 'No Transaction (Default)'.

JMS x Problems Console

Attribute 'action' is required

**General**

Advanced

Transformers

Notes

Display Name: JMS

Basic Settings

Exchange Pattern: ☒ one-way (Default) ☐ request-response

☒ Queue: mybroker

☐ Topic:

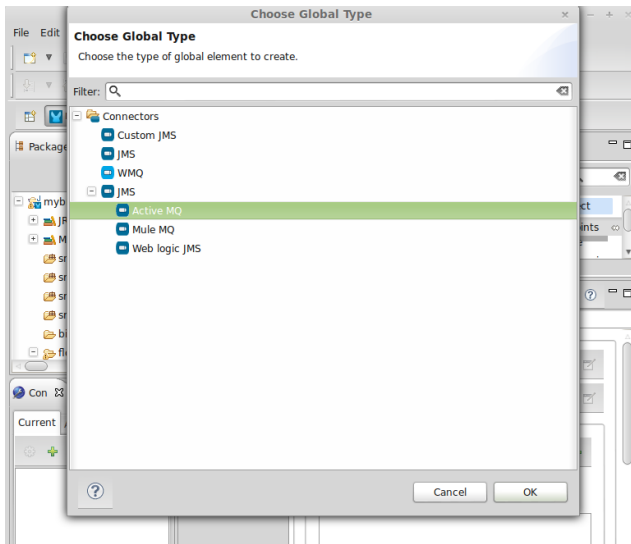
Connector Configuration: jmsConnector

Transaction

Type: No Transaction (Default)

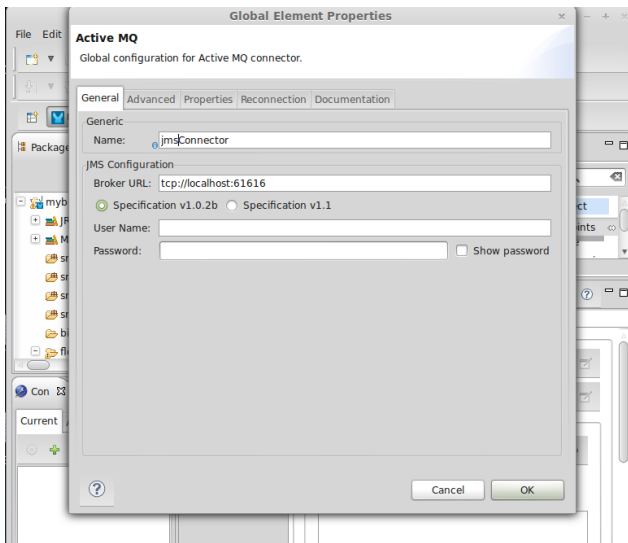
# Crear un proyecto Mule: JMS (cont.)

- Seleccionamos **Active MQ**

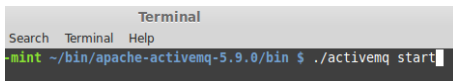


# Crear un proyecto Mule: JMS (cont.)

- Definimos la configuración del conector



- Ahora vamos a configurar una instancia de ActiveMQ local para probar con nuestro proyecto Mule.
- Descarga ActiveMQ:  
<http://activemq.apache.org/download-archives.html>
- Descomprimir el archivo, vaya al directorio bin y ejecute: ActiveMQ start

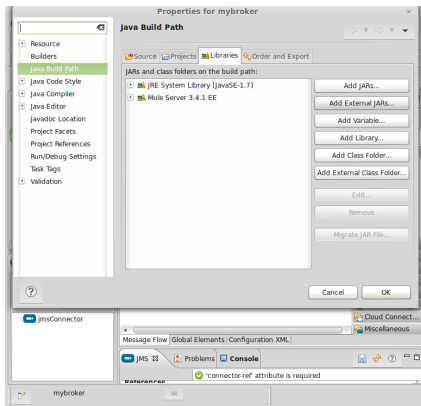


```
Terminal
Search Terminal Help
-mint ~/bin/apache-activemq-5.9.0/bin $ ./activemq start
```

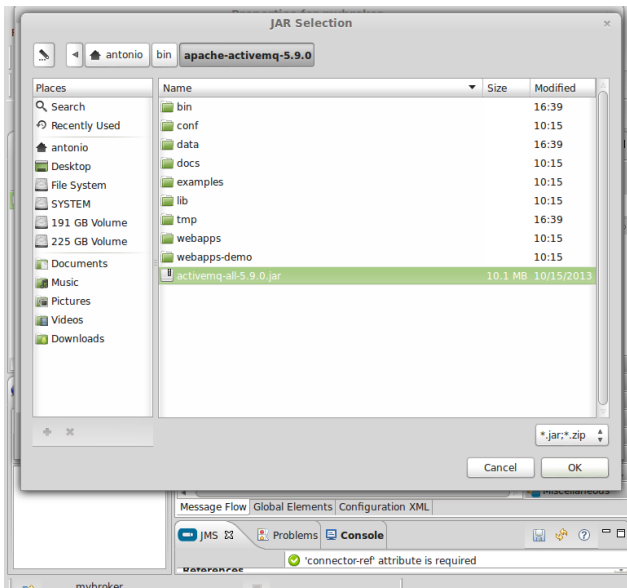


# Crear un proyecto Mule: JMS (cont.)

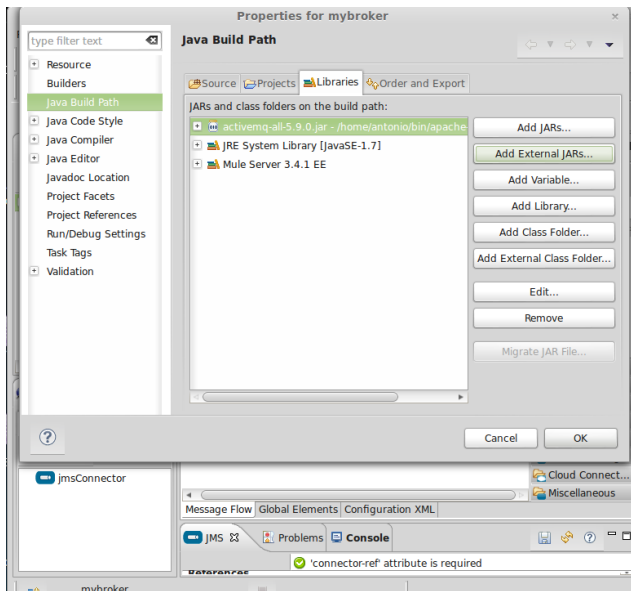
- Antes de poder ejecutar la aplicación, tendrás que añadir el JAR de ActiveMQ a su proyecto.
- Clic derecho en Mule Runtime en el panel Explorador de proyectos, seleccione **Build Path** y seleccione **Configure Build Path**.



# Crear un proyecto Mule (cont.)

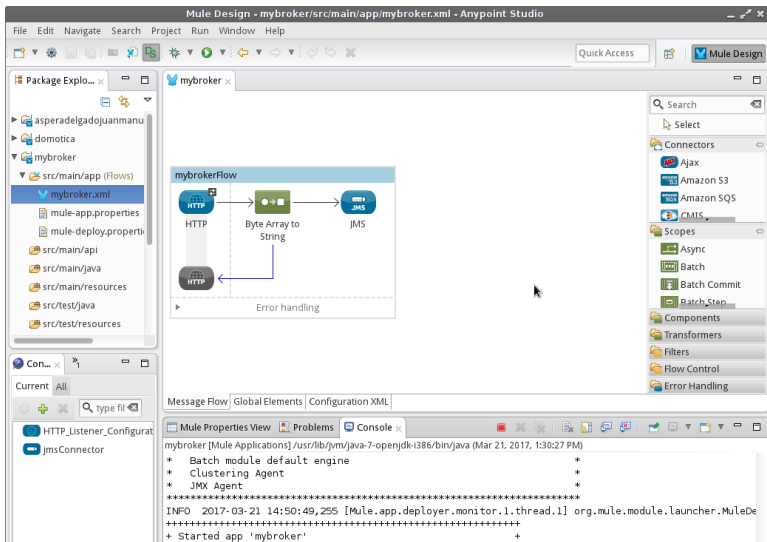


# Crear un proyecto Mule (cont.)



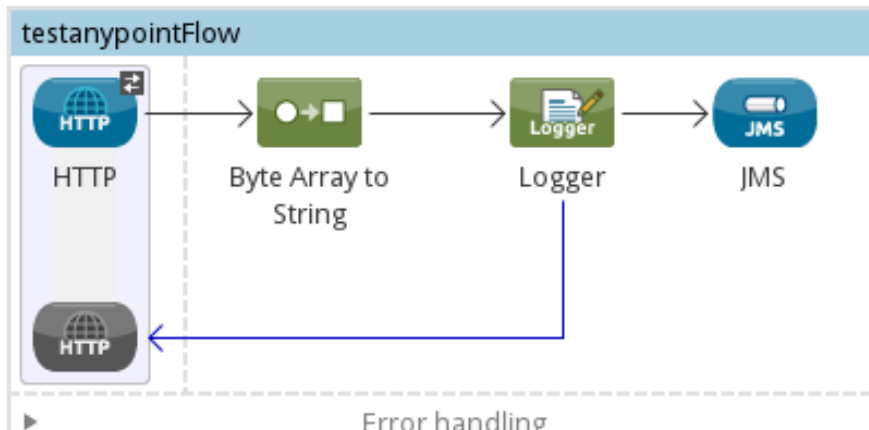
# Crear un proyecto Mule (cont.)

- Clic derecho sobre el proyecto: Run as → Mule Application



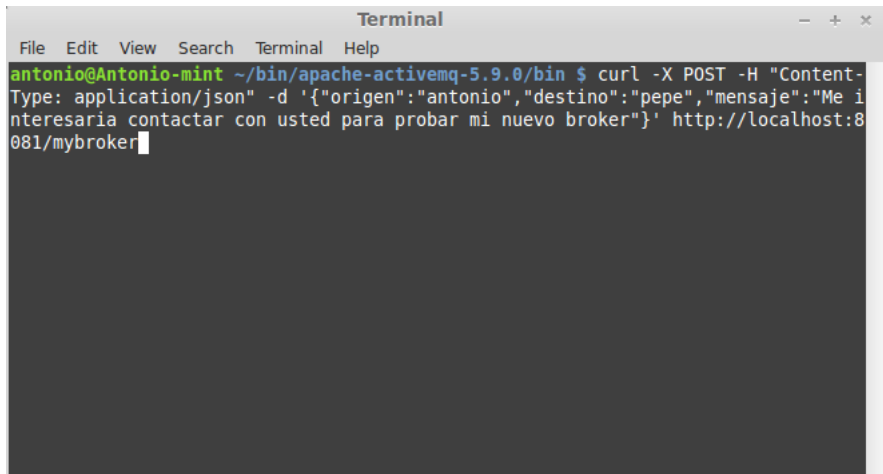
## Crear un proyecto Mule: Logger (cont.)

- Para ver el registro en la consola añadimos un objeto **Logger** al flujo
- Detenemos la aplicación y volvemos a ejecutar



## Crear un proyecto Mule (cont.)

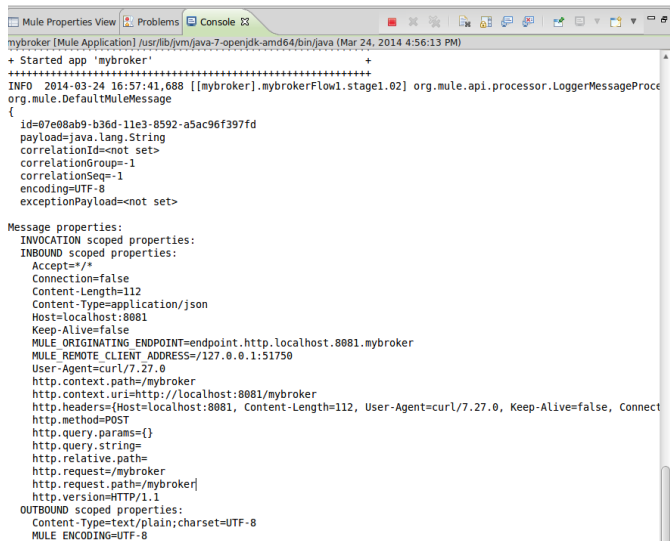
- Enviamos datos JSON a la entrada del flujo usando curl
- Puede enviar varios mensajes para ver a continuación como se acumulan en la cola



```
Terminal
File Edit View Search Terminal Help
antonio@Antonio-mint ~/bin/apache-activemq-5.9.0/bin $ curl -X POST -H "Content-Type: application/json" -d '{"origen":"antonio","destino":"pepe","mensaje":"Me interesaría contactar con usted para probar mi nuevo broker"}' http://localhost:8081/mybroker
```

# Crear un proyecto Mule (cont.)

- Datos de monitorización obtenidos gracias al componente **Logger**



```

mybroker [Mule Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Mar 24, 2014 4:56:13 PM)

+ Started app 'mybroker' +
+++++
INFO 2014-03-24 16:57:41,688 [[mybroker].mybrokerFlow1.stage1.02] org.mule.api.processor.LoggerMessageProcessor:
org.mule.DefaultMuleMessage
{
  id=07e08ab9-b36d-11e3-8592-a5ac96f397fd
  payload=java.lang.String
  correlationId=<not set>
  correlationGroup=-1
  correlationSeq=-1
  encoding=UTF-8
  exceptionPayload=<not set>

Message properties:
  INVOCATION scoped properties:
  INBOUND scoped properties:
    Accept=/*/*
    Connection=false
    Content-Length=112
    Content-Type=application/json
    Host=localhost:8081
    Keep-Alive=false
    MULE_ORIGINATING_ENDPOINT=endpoint.http.localhost.8081.mybroker
    MULE_REMOTE_CLIENT_ADDRESS=/127.0.0.1:51750
    User-Agent=curl/7.27.0
    http.context.path=/mybroker
    http.context.uri=http://localhost:8081/mybroker
    http.headers=(Host=localhost:8081, Content-Length=112, User-Agent=curl/7.27.0, Keep-Alive=false, Connection=close)
    http.method=POST
    http.query.params={}
    http.query.string=
    http.relative.path=
    http.request=/mybroker
    http.request.path=/mybroker
    http.version=HTTP/1.1
  OUTBOUND scoped properties:
    Content-Type=text/plain;charset=UTF-8
    MULE_ENCODING=UTF-8
  
```

# Crear un proyecto Mule (cont.)

- Acceso a `http://localhost:8161/admin/queues.jsp` para ver colas.
- Por defecto usuario y clave son admin y admin respectivamente.

localhost:8161/admin/queues.jsp

Accesso privado - ... Save to Mendeley XeoWeb: 19) [Ko... MOD 145 Tabla de mareas ... Aprendiendo exp... Documentación X

**ACTIVE MQ**

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Queue Name  Create

## Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
mybroker	1	0	1	0		<a href="#">Browse Active Consumers</a> <a href="#">atom</a> <a href="#">rss</a> <a href="#">Send To Purge Delete</a>



# Referencias bibliográficas I



D.Dossot; J.DEmic; V.Romero  
Mule in Action, Second Edition  
Manning Publications, 2014.



MuleSoft Inc.  
Mule Studio  
<http://www.mulesoft.org/download-mule-esb-community-edition>,  
mayo 2013.



LogMeIn, Inc.  
Xively – Public Cloud for the Internet of Things  
<https://xively.com/>, mayo 2013.



D. Luckham  
The Power of Events: An Introduction to Complex Event Processing in  
Distributed Enterprise Systems  
Addison-Wesley, 2001.

## Referencias bibliográficas II



D. Luckham

Event Processing for Business: Organizing the Real-Time Enterprise  
Wiley, 2012.



EsperTech Inc.

Esper - Complex Event Processing  
<http://esper.codehaus.org/>, mayo 2013.



J. Boubeta Puig; G. Ortiz; I. Medina Bulo

Procesamiento de Eventos Complejos en Entornos SOA: Caso de Estudio para  
la Detección Temprana de Epidemias  
Actas de las VII Jornadas de Ciencia e Ingeniería de Servicios  
A Coruña, septiembre, 2011.



L. Atzori; A. Iera; G. Morabito

The Internet of Things: A Survey  
Computer Networks (15), pp. 2787-2805, octubre, 2010.