

Seminario 2. Segunda Parte: Anypoint Studio - primer proyecto

Sistemas Distribuidos

Juan Boubeta Puig

Antonio Balderas Alberico. Editadas por Pablo García Sánchez

Departamento de Ingeniería Informática

Marzo de 2018



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



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



¿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. The title bar reads "Mule Design - mybroker/src/main/app/mybroker.xml - Anypoint Studio". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, running, and debugging. The left sidebar shows the Package Explorer with a tree view of the project structure, including "mybroker" and its sub-packages. The main workspace shows a Mule flow diagram titled "mybrokerFlow". The flow starts with an "HTTP" connector, followed by a "Byte Array to String" transformer, and then a "JMS" connector. An error handling section is visible at the bottom of the flow. The right sidebar contains a search bar and a list of connectors, scopes, components, transformers, filters, flow control, and error handling. The bottom console shows the following log output:

```
mybroker [Mule Applications] /usr/lib/jvm/java-7-openjdk-1386/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:
```

Anypoint Studio - Editor XML

Mule Design - mybroker/src/main/app/mybroker.xml - Anypoint Studio

File Edit Source Navigate Search Project Run Window Help

Package Explorer

- asperadelgadojuanmanu
- domotica
- mybroker
 - src/main/app (Flows)
 - mybroker.xml
 - mule-app.properties
 - mule-deploy.properties
 - src/main/api
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources

XML Editor

```
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/mule
8     http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule
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
```

Message Flow | Global Elements | Configuration XML

Mule Properties View | Problems | Console

mybroker [Mule Applications] /usr/lib/jvm/java-7-openjdk-1386/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:

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



Requisitos de software y hardware

Hardware

- 3GB de RAM
- 2GHz de CPU
- 4GB libres de espacio de disco

Software

- *Java Runtime Environments:*
 - Oracle Java 1.6
 - Oracle Java 1.7
 - IBM Java 1.6
- *Sistemas operativos:*
 - Windows (32 o 64 bit)
 - Mac OS (32 o 64 bit)
 - Linux (32 o 64 bit)

Más información: <http://www.mulesoft.org/documentation/display/current/Hardware+and+Software+Requirements>

Pasos para instalar y ejecutar Anypoint Studio

- 1 Descargar la versión Anypoint Studio v3.4 para Windows, Linux o Mac:
<http://www.mulesoft.org/download-mule-esb-community-edition>.
- 2 Nota: la última versión en linux está en
<https://www.mulesoft.com/ty/dl/studio-linux>.
- 3 Descomprimir el archivo *MuleStudio-for-*.zip* en un directorio cuya ruta no sea muy extensa.
- 4 Una vez descomprimido, ejecutar el fichero MuleStudio:
MuleStudio.exe (Windows), MuleStudio.app (Mac OSX) o MuleStudio (Linux).

También puede descargarse e instalarse Anypoint Studio como plugin de Eclipse. Más información en: <http://www.mulesoft.org/documentation/display/current/Studio+in+Eclipse>

Uso de Anypoint Studio con sistema de control de versiones

- **Subclipse:** <http://www.mulesoft.org/documentation/display/33X/Using+Subversion+with+Studio>.
- **Git:** <http://www.mulesoft.org/documentation/display/33X/Using+Git+with+Studio>.

Es necesario registrarse en la web para poder acceder a dicha información.

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

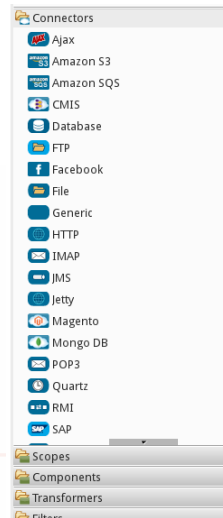


Connectors

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.

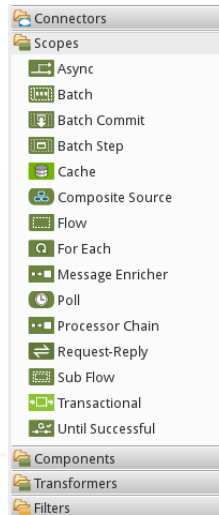


Scopes

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

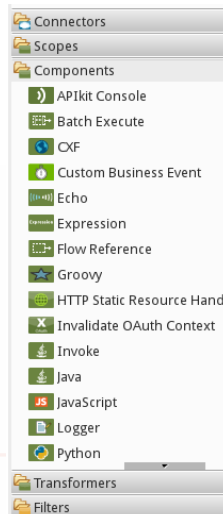
- Mejorar la legibilidad del código XML.
- Implementar procesamiento paralelo.
- Crear secuencias de bloques reusables.

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



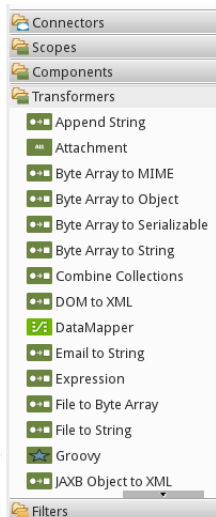
Components

- 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.



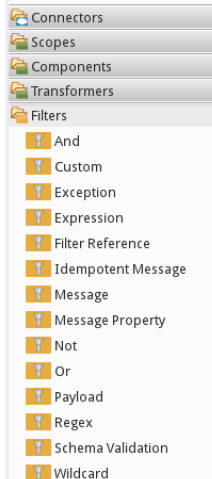
Transformers

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



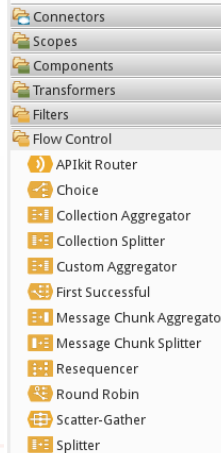
Filters

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



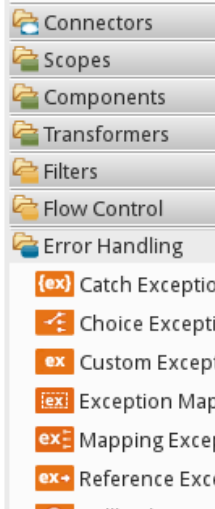
Flow controls

- 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.



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



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).

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



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

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

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

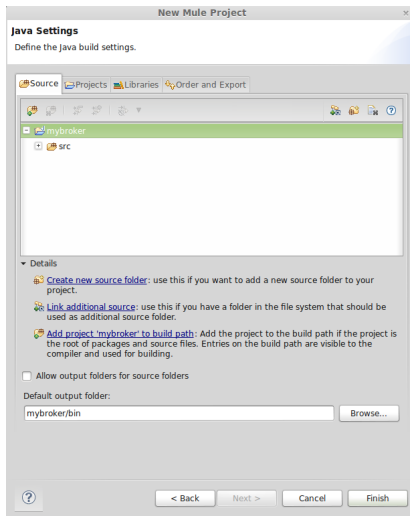
Crear un proyecto Mule (cont.)

■ Siguiente ...

The screenshot shows the 'New Mule Project' dialog box. The title bar says 'New Mule Project'. Below the title bar, it says 'Create a Java Project' and 'Create a Java project in the workspace or in an external location.' The 'Project name' field is filled with 'mybroker'. The 'Use default location' checkbox is checked. The 'Location' field shows '/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 selected JRE is 'javaSE-1.7'. 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 a checkbox 'Add project to working sets' which is unchecked. The 'Working sets' field is empty with a 'Select...' button. At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Finish'.

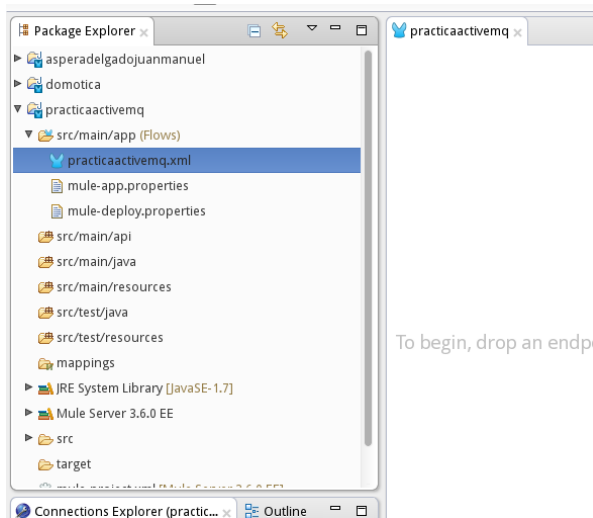
Crear un proyecto Mule (cont.)

■ Y finalizar



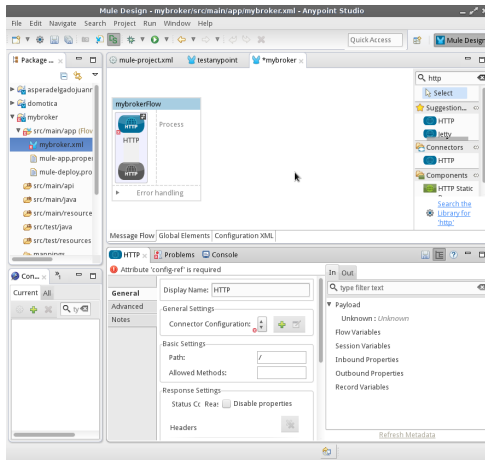
Crear un proyecto Mule (cont.)

■ Nuestro proyecto creado



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 configuration window for the HTTP connector in Mule Studio. The interface includes a top bar with tabs for 'HTTP', 'Problems', and 'Console'. Below the tabs, a green checkmark icon indicates 'There are no errors.' On the left side, there is a sidebar with three tabs: 'General' (selected), 'Advanced', and 'Notes'. The main configuration area is divided into four sections: 'General Settings' with a 'Display Name' field set to 'HTTP' and a 'Connector Configuration' dropdown set to 'HTTP_Listener_Configuration'; 'Basic Settings' with a 'Path' field set to 'mybroker' and an empty 'Allowed Methods' field; and 'Response Settings' with 'Status Code' and 'Reason' fields, and a 'Disable protocol' checkbox.

HTTP x Problems Console

✓ There are no errors.

General

Advanced

Notes

Display Name: HTTP

General Settings

Connector Configuration: HTTP_Listener_Configuration

Basic Settings

Path: mybroker

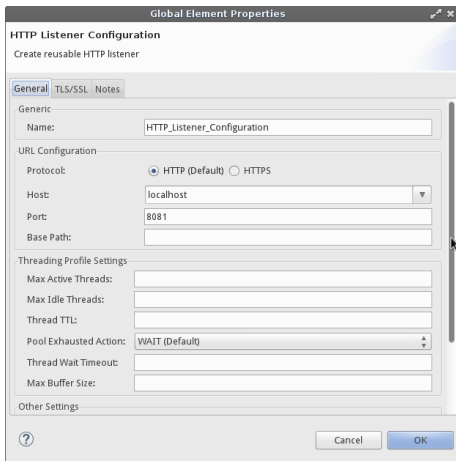
Allowed Methods:

Response Settings

Status Code: Reason: ☐ Disable protocol

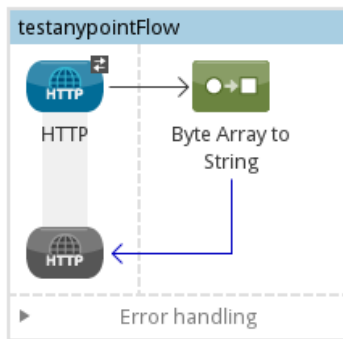
Crear un proyecto Mule: HTTP (cont.)

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



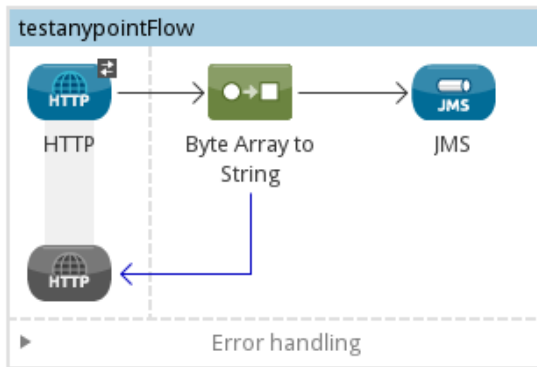
Crear un proyecto Mule: Byte Array to String

- 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 'JMS' configuration window in Anypoint Studio. The 'General' tab is selected, and a red warning icon indicates that the 'action' attribute is required. The 'Display Name' is set to 'JMS'. Under 'Basic Settings', the 'Exchange Pattern' is set to 'one-way (Default)', and the 'Queue' is named 'mybroker'. The 'Connector Configuration' is set to 'jmsConnector'. Under 'Transaction', the 'Type' is 'No Transaction (Default)' and the 'Action' is 'NONE (Default)'. The 'Timeout' field is empty.

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

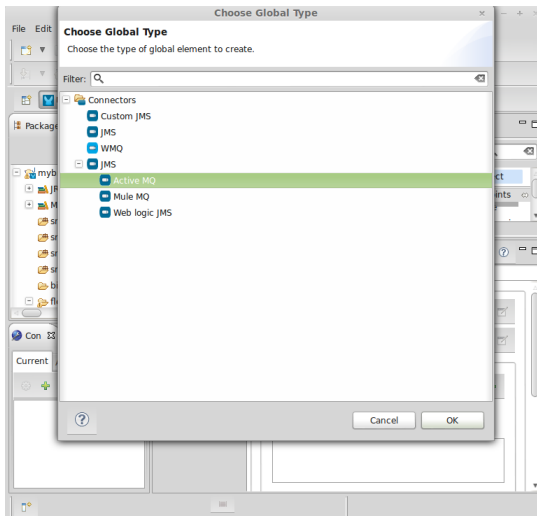
Type: No Transaction (Default)

Action: NONE (Default)

Timeout:

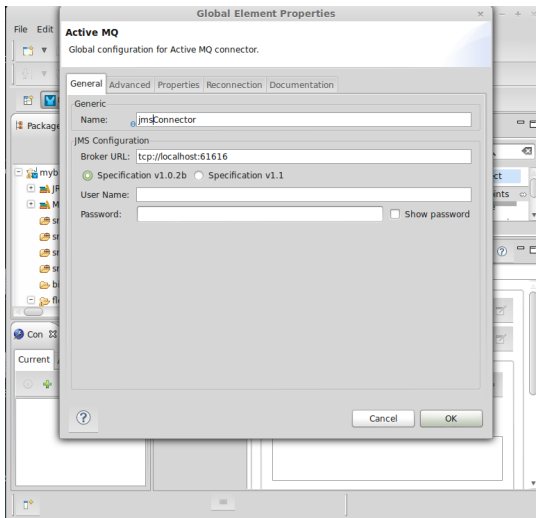
Crear un proyecto Mule: JMS (cont.)

- Seleccionamos **Active MQ**

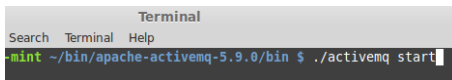


Crear un proyecto Mule: JMS (cont.)

- Definimos la configuración del conector



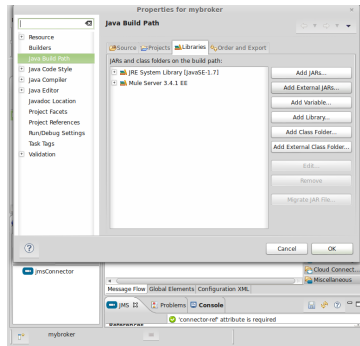
- Ahora vamos a configurar una instancia de ActiveMQ local para probar con nuestro proyecto Mule.
- Descarga ActiveMQ:
<http://activemq.apache.org/download-archives.html>
- Descomprimir el archivo, vaya al directorio bin y ejecute: ActiveMQ start



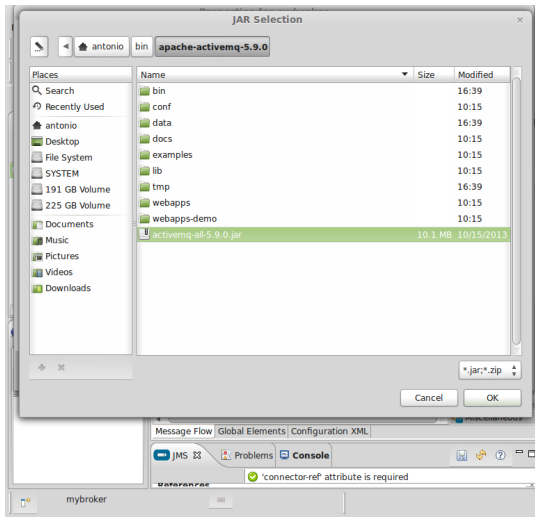
```
Terminal
Search Terminal Help
-mint ~/bin/apache-activemq-5.9.0/bin $ ./activemq start
```

Crear un proyecto Mule: JMS (cont.)

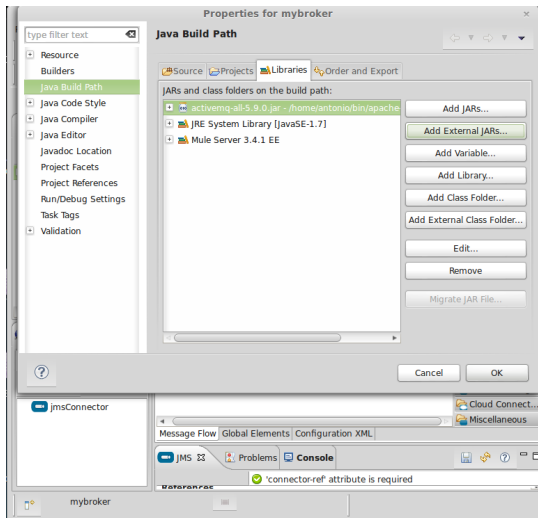
- Antes de poder ejecutar la aplicación, tendrás que añadir el JAR de ActiveMQ a su proyecto.
- Clic derecho en Mule Runtime en el panel Explorador de proyectos, seleccione **Build Path** y seleccione **Configure Build Path**.



Crear un proyecto Mule (cont.)

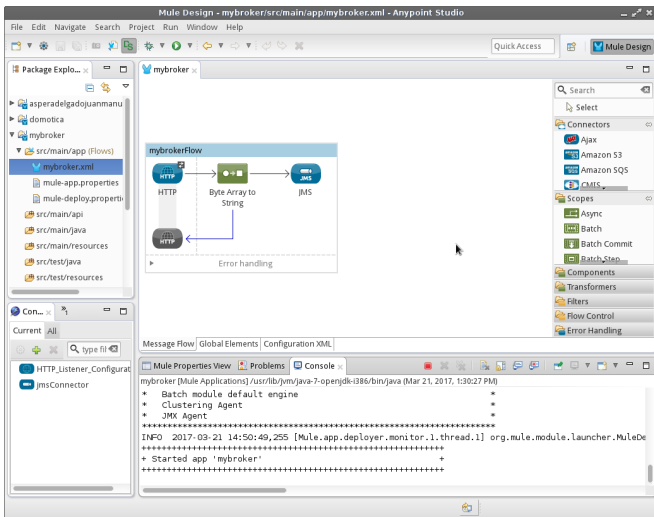


Crear un proyecto Mule (cont.)



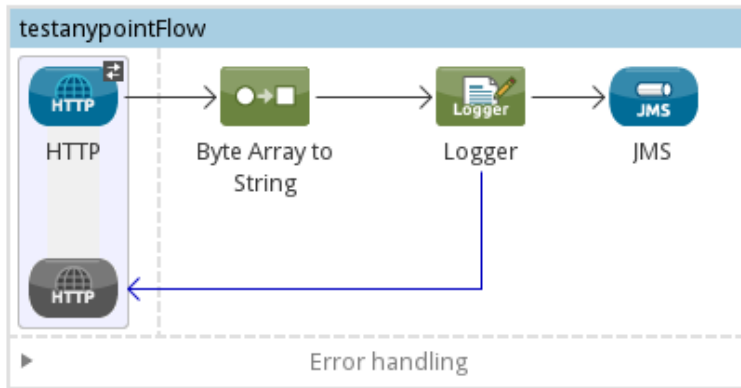
Crear un proyecto Mule (cont.)

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



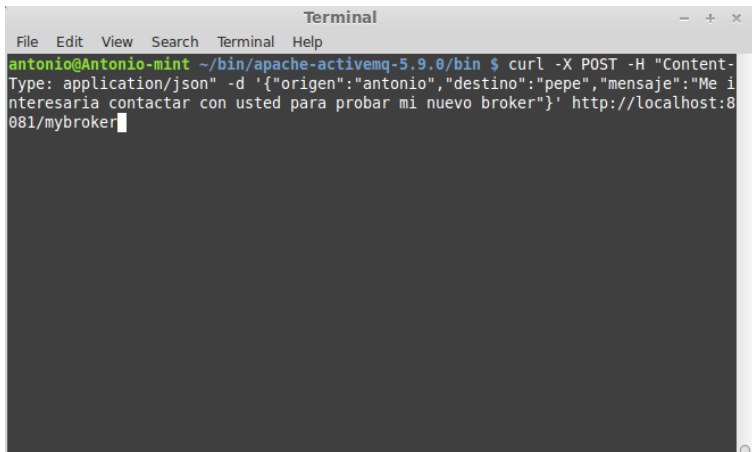
Crear un proyecto Mule: Logger (cont.)

- 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.)

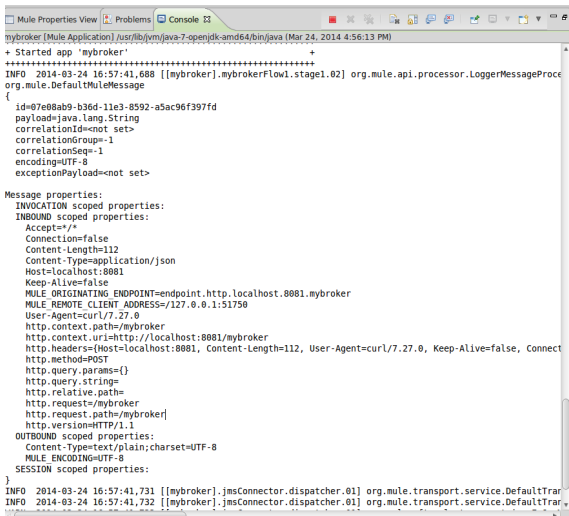
- Enviamos datos JSON a la entrada del flujo usando curl
- Puede enviar varios mensajes para ver a continuación como se acumulan en la cola



```
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_ORIGINAL_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=false)
  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
SESSION scoped properties:
}
INFO  2014-03-24 16:57:41,731 [[mybroker].jmsConnector.dispatcher.01] org.mule.transport.service.DefaultTransportService:
INFO  2014-03-24 16:57:41,732 [[mybroker].jmsConnector.dispatcher.01] org.mule.transport.service.DefaultTransportService:
  
```

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

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

ACTIVEMQ Software <http://www.activemq.org>

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

Queue Name

Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
mybroker	1	0	1	0	Browse Active Consumers <input type="button" value="atom"/> <input type="button" value="rss"/>	Send To Purge Delete
products	0	0	0	0	Browse Active Consumers <input type="button" value="atom"/> <input type="button" value="rss"/>	Send To Purge Delete

■ Que
 ■ Grap
 ■ XML

■ Top
 ■ XML

■ Sub
 ■ XML

■ Use
 ■ Docu

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 Buló

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.

