Aplicaciones con Enfoque Orientado a Servicios

Tema Nº10:Trabajando con Mule ESB - III

Indicador de logro Nº10:Realiza pruebas unitarias a una aplicación distribuida previamente configurada, usando MUnit

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº10:**

Trabajando con Mule ESB - III

**Subtema 10.1:**

Configuración de *properties*

Los archivos de propiedades corresponden a una clase de archivos de texto, que generalmente son usados para almacenar parámetros configurables de una aplicación como traducciones, valores, direcciones web, etc.

Cada parámetro de un archivo .properties, se almacena en dos partes, la primera la *key* o clave que nos sirve para identificar nuestro parámetro en Java y el segundo es el valor almacenado para dicho parámetro. Por ejemplo:

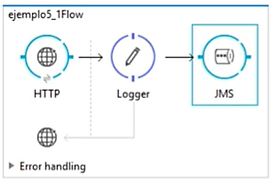
port.queue=6116

En un proyecto Mule ESB, también podemos aprovechar el uso de las propiedades, en especial, en escenarios donde los parámetros pueden cambiar. Así evitamos que se encuentren *harcodeados* dentro de la misma aplicación.

**Ejemplos:**

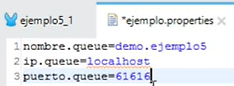
**Uso de archivo .properties**

Para este ejemplo, se usará como base el proyecto de mensajería de colas de Mule ESB con ActiveMQ del tema N° 8 “Trabajando con Mule ESB I”.

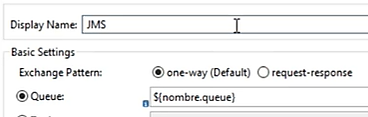


Como pre requisito, cargue dicho proyecto e inicie el ActiveMQ. A partir de ahí realice las siguientes modificaciones:

1. En *Anypoint Studio*, cree un nuevo archivo de propiedades **ejemplo.properties** dentro de la carpeta *src / main / resources*.
2. El archivo debe contener los siguientes parámetros:



1. En el componente *JMS* realice el cambio al nombre de cola utilizando el parámetro del archivo .properties.



1. De igual forma, modifique la configuración de conexión de la cola (ip y puerto). Para ello primero haga clic en el botón  y luego realice el cambio.



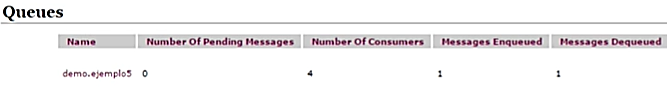
1. Ahora es necesario indicar al flujo la ubicación del archivo properties. Para ello seleccione la opción *Configuration XML*  del flujo.



1. Agregue la siguiente línea indicando el nombre del archivo .properties. Por defecto se considera la ruta *src / main / resources.*



1. Ejecute y pruebe su aplicación.
2. Verifique la cola de ActiveMQ.



**Uso de archivo .properties externo**

El uso de archivo .properties dentro del proyecto ha ido decayendo en uso, y se está optando por contar con una única ubicación de propiedades ajena a un proyecto en específico. Esto permite que, si se desean cambiar los parámetros, no es necesario ingresar al proyecto a modificarlos, sino realizar el cambio en el archivo directamente.

1. Cree una carpeta para las propiedades:

C:\Carpetas\_Irving\Mule\properties\"

1. En la ruta del *Mule StandAlone*, ubíquese en la carpeta **conf** y abra el archivo **wrapper.conf** con un editor de texto.
2. Dentro del archivo busque la línea:

# Java Classpath

1. Verá las siguientes líneas:

wrapper.java.classpath.1=%MULE\_LIB%

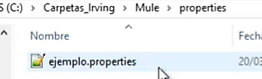
wrapper.java.classpath.2=%MULE\_BASE%/conf

wrapper.java.classpath.3=%MULE\_HOME%/lib/boot/\*.jar

1. Luego de esas líneas, para indicar la ruta de los archivos de propiedades agregue la siguiente:

wrapper.java.classpath.4=C:\Carpetas\_Irving\Mule\properties\

1. Guarde los cambios.
2. Realice la copia del archivo ejemplo.properties del proyecto hacia la carpeta de propiedades recién creada.



1. Luego, elimine el archivo del proyecto para evitar conflictos.
2. Realice el despliegue del aplicativo con Mule StandAlone.
3. Pruebe su aplicación.
4. Verifique la cola de ActiveMQ nuevamente.



**Subtema 10.2:**

Pruebas unitarias en MUnit

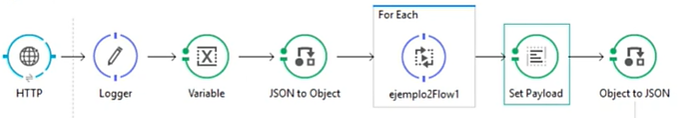
MUnit es un marco de prueba de aplicaciones de Mule que permite crear fácilmente pruebas automatizadas para integraciones y API. Proporciona un conjunto completo de capacidades de prueba de unidad e integración, y está completamente integrado con Maven y Surefire para la integración con su entorno de implementación continua.

Esta herramienta sirve para:

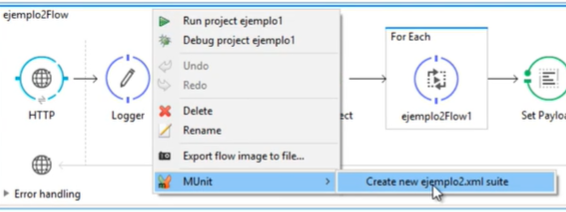
* Crear pruebas escribiendo código Mule
* Simular diversos procesadores
* Verificar las llamadas del procesador
* Habilitar o ignorar pruebas particulares
* Realizar pruebas de etiquetas
* Verificar la cobertura visual en AnypointStudio
* Generar informes de cobertura

**Ejemplos:**

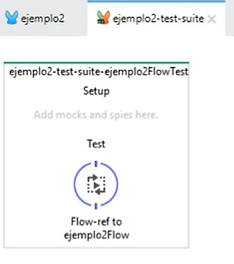
Para este ejemplo, se utilizará un flujo del primer tema de Mule ESB. Utilizaremos esta prueba para enviar un request y validar que nos devuelva la respuesta esperada.



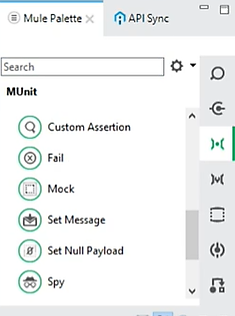
1. Haga clic derecho dentro del flujo y seleccione la opción *MUnit > Create new ejemplo2.xml suite*.



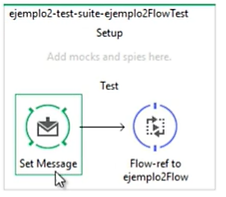
1. Verificamos el archivo que ha sido creado en la carpeta *src / test / munit*:



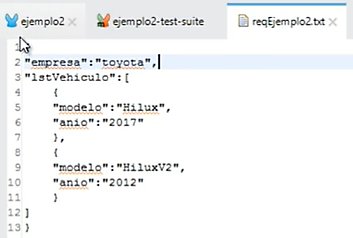
1. En la paleta de Mule, ubique los componentes de MUnit



1. Añada el componente *Set Message* antes de la referencia al flujo.



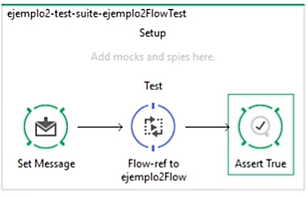
1. Para enviar la petición de prueba, utilizaremos una plantilla. Para ello, añada un archivo de nombre **reqEjemplo2.txt** en *src / test / resources*.



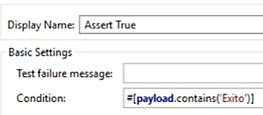
1. Configure el componente *Set Message* para que el *payload* sea cargado a partir del JSON de la plantilla creada.



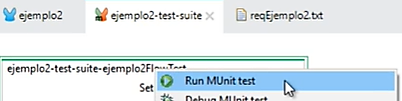
1. Para verificar que la respuesta sea la esperada, utilizaremos el componente *Assert True*. Añada uno de estos al final de la referencia del flujo.



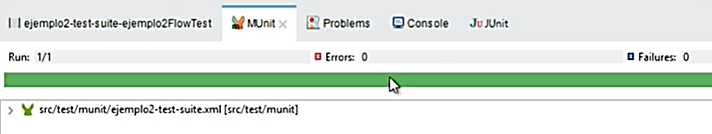
1. Configure la condición del componente para evaluar si el mensaje de la respuesta contiene la palabra “Exito” (este mensaje es el esperado)



1. Ejecute su prueba unitaria haciendo clic derecho en el flujo de prueba y seleccionando la opción *Run MUnit Test*



1. En la pestaña de MUnit verifique los resultados.



**Subtema 10.3:**

Elaboración y consumo de archivo RAML

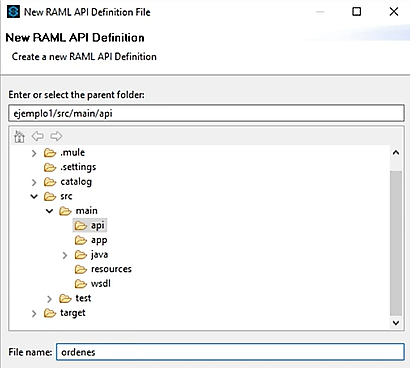
El Lenguaje de Modelado de API de RESTful (RAML) es un lenguaje basado en YAML que describe las API de RESTful y que proporciona capacidades para documentar ampliamente una API de RESTful, activar las herramientas de generador de documentación para extraer la documentación del usuario y, a continuación, traducirla a formatos visuales, tales como PDF, HTML, entre otros.

La acción RAML se utiliza para acceder a las API y realizar llamadas de API a los servicios para los que se crean las API mediante RAML.

**Ejemplos:**

**Elaborar archivo RAML**

1. En la carpeta *src / main / api* haga clic derecho y seleccione la opción *New > RAML API Definition* para crear un nuevo archivo RAML con la siguiente configuración:



1. Modifique el archivo indicando un título, la versión de RAML a usar y la Uri base que tendrá la API.



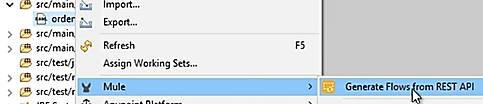
1. Ahora defina el cuerpo que debe tener el request del método POST del path **ordenes**



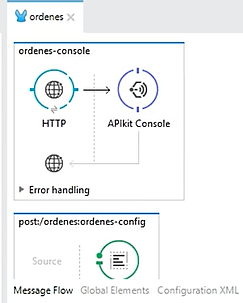
1. Añada un ejemplo de respuesta exitosa (código HTTP 200) que tendrá el método POST. En este caso, será un arreglo de órdenes.



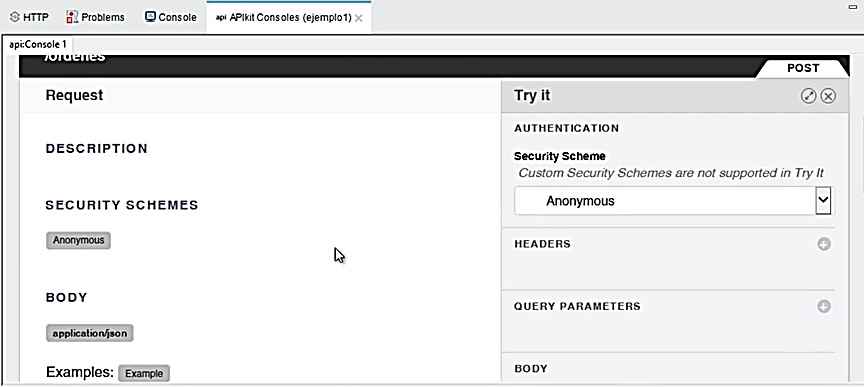
1. A partir de esta definición RAML, genere los flujos correspondientes. Para ello, seleccione el archivo ordenes.raml, haga clic derecho y seleccione la opción *Mule > Generate Flows from REST API*.



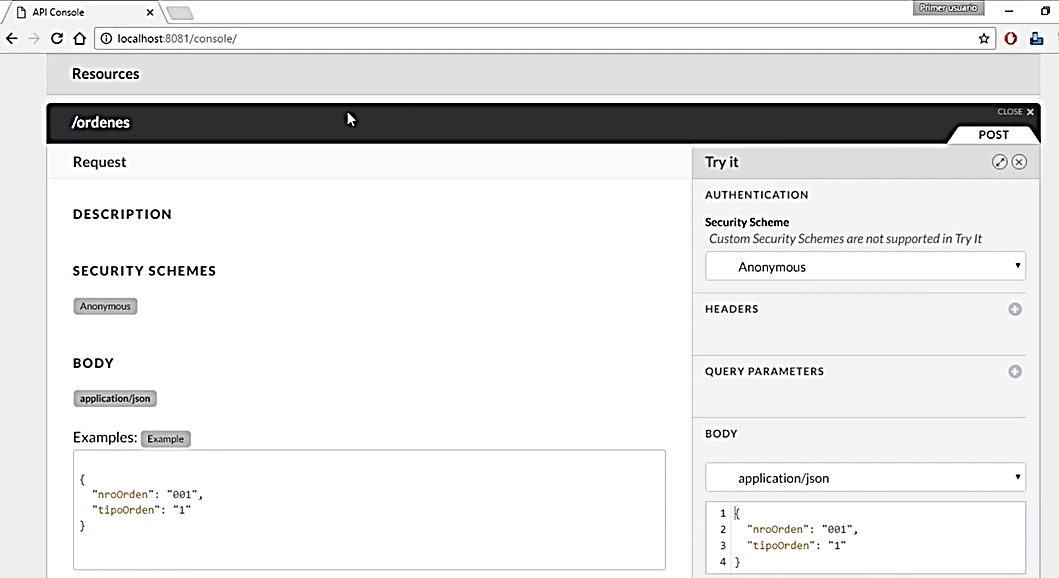
1. Verifique el archivo ordenes.xml que se ha generado con los flujos creados.



1. Ejecute su proyecto.
2. Verifique la pestaña *APIKit Consoles*.



1. También lo puede visualizar en el navegador con la documentación generada.



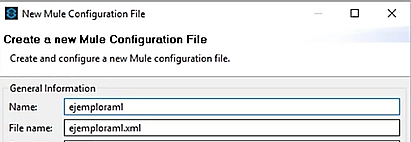
1. Para verificar que el servicio se haya creado, haga pruebas con Postman a la URL de la API.



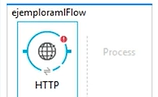
**Consumo de servicio RAML**

En base al servicio RAML creado en el ejemplo anterior, ahora realizaremos el consumo del mismo.

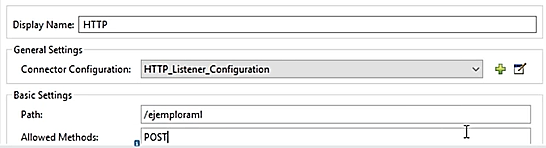
1. Cree un nuevo archivo de configuración de Mule *(New Mule Configuration File)*.



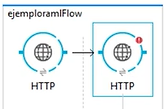
1. Añada un componente *HTTP* al flujo.



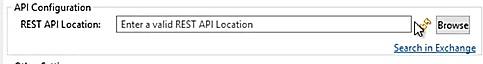
1. Configure el componente *HTTP*.



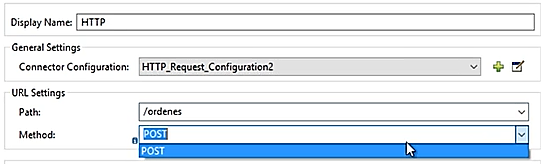
1. Añada otro componente *HTTP* y ubíquelo en la sección *Process* del flujo.



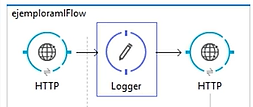
1. Para este segundo componente, configure el conector haciendo clic en el botón **+**.
2. En la ventana de configuración, en la sección *API Configuration* haga clic en el botón *Browse* para ubicar el archivo ordenes.raml



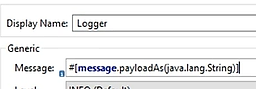
1. Termine de configurar el segundo componente HTTP



1. Añada un componente *Logger* en medio de los dos componentes.



1. Configure el componente para mostrar el mensaje que se está enviando.



1. Despliegue el proyecto en Mule StandAlone.
2. Pruebe el proyecto teniendo en cuenta que la URL es diferente a la del servicio RAML



1. Envíe una petición JSON.



1. Verifique la respuesta.



**Actividad:**

Teniendo como base las 2 actividades de la Guía 9 – Trabajando con Mule ESB II, implemente:

1. Para el proyecto de consumo de servicio SOAP, configure las rutas y nombres de métodos a partir de archivos de propiedades dentro del proyecto.
2. Para el proyecto de consumo de servicio REST, configure la ruta y el nombre del país a consultar en un archivo de propiedades externo.
3. Para ambos proyectos, realice pruebas unitarias.