Práctica BM008 – Bluemix AlchemyAPI

Abril 2015

AlchemyAPI es un servicio web que analiza contenido no estructurado (noticias, blogs, imágenes, etc.) exponiendo la riqueza semántica de tus datos. Sus algoritmos de procesamiento de lenguaje natural y visión artificial analizan texto, imágenes o contenido web, identificando entidades nombradas (personas, lugares, compañías, etc.), hechos y relaciones, palabras claves, sentimiento del texto, clasificación taxonómica, y más.

En este documento te mostraremos cómo construir una aplicación en Bluemix utilizando Node-RED para acceder AlchemyAPI y analizar el contenido de una imagen.

Prerrequisitos

1. Cuenta de Bluemix.
2. Para acceder los servicios de AlechemyAPI necesitas una clave de acceso.

   Puedes obtener una clave gratis registrándote en la siguiente dirección: http://www.alchemyapi.com/api/register.html

   (La clave te permitirá un número limitado de llamadas por día)

Creando la aplicación

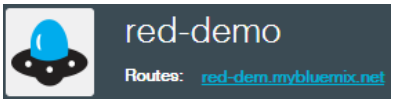
1. En el catálogo de servicios, selecciona el ícono Node-RED Starter de la sección Boilerplates para crear una nueva aplicación **y asígnale un nombre único** en el dominio.

   Node-RED es una herramienta de código abierto que te permite construir aplicaciones ligando elementos entre sí en un ambiente gráfico. Puedes obtener más información en http://nodered.org/

2. Accede al editor gráfico.

   a. Ingresa la URL generada para tu aplicación http://<myapp>.mybluemix.net

   b. Se desplegará una página con información sobre Node-RED y un botón con la leyenda **"Go to your Node-RED flow editor"** que te llevará al editor gráfico.

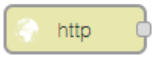


Estás listo para comenzar a crear flujos.

En el panel izquierdo encontrarás diferentes nodos para manejar entradas, salidas y funciones que puedes utilizar en tus flujos. La sección central presenta espacio en blanco para diseñar los flujos. Finalmente el panel de la derecha tiene una pestaña para desplegar información del nodo seleccionado y una pestaña para desplegar mensajes de debug.

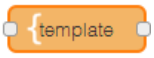
Forma HTML

Todo flujo comienza con una entrada. La sección **input** del panel de nodos contiene diferentes formas de ingresar datos a la aplicación. En este caso usaremos un browser para accederla así que nuestro primer paso será desplegar una página HTML con una forma.



1. Crea un nodo **http in** y configura las siguientes propiedades:

   Method: GET
   url: /img



2. Crea un nodo **template** con el que se construirá el contenido de la página. Por ahora sólo incluiremos el código estático de la forma HTML.

   Agrega el siguiente código en las propiedades del nodo:

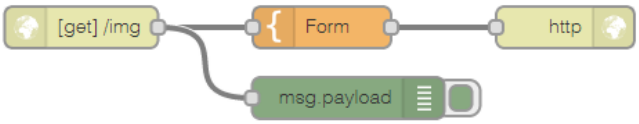
```
<form action="/img" method="GET">
  <input type="text" name="imgurl"/>
  <button type="submit">
  Analizar</button>
</form>
```

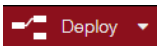
3. Crea un nodo **http response** para generar una salida tipo HTTP del flujo.

4. Agrega un nodo **debug** para escribir en la consola la propiedad `payload` del mensaje que se le envía. Conserva las propiedades por defecto del nodo:

   Output: message property
   msg.payload

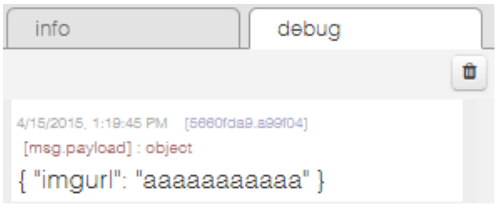
5. Liga los nodos de la siguiente manera:





6. Despliega el flujo pulsando el botón **Deploy** de la esquina superior derecha.

El flujo será invocado al acceder la dirección http://<my-app>.mybluemix.net/img en un browser. Se desplegará una forma con un campo tipo texto para ingresar la dirección de la imagen a analizar.



Por ahora la forma aún no hace más que llamar de nuevo a la página con el parámetro de la dirección de la imagen a analizar.

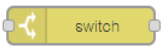
Gracias al nodo debug que agregamos podemos ver en la consola de Node-RED el mensaje que se está enviando al iniciar el flujo. La propiedad `payload` del objeto `msg` que se envía entre nodos contiene en un inicio la información de la forma con su único campo

Agreguemos una llamada al servicio de AlchemyAPI para procesar esta información.

Llamada REST

Vamos a acceder los servicios de Alchemy API mediante llamadas REST, si no estás familiarizado con esta tecnología es recomendable que leas este artículo http://www.ibm.com/developerworks/library/ws-restful

Modifiquemos el flujo para incluir la llamada al servicio de Alchemy.



1. Agrega un nodo **switch** al flujo para llamar al API sólo cuando tengamos información qué procesar.

   Incluye dos condiciones en las propiedades para la variable `payload.imgurl`:

   - `is null`
   - `is not null`

2. Agrega un nodo **function** al flujo y nómbralo `construyeMsg`. Este tipo de nodos definen funciones JavaScript para manipular los mensajes entre nodos. Este nodo tomará como parámetro de entrada el objeto `msg` del nodo que lo invoca y enviará como mensaje al nodo siguiente un objeto o arreglo de objetos.

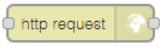
   Este nodo se asociará a la condición `is not null` del nodo `switch`.

   En las propiedades del nodo escribe la siguiente función que construirá el mensaje que se enviará como solicitud a AlchemyAPI.

   En la propiedad `apikey` utiliza tu propia clave.

```
msg.headers = {
 'Content-Type' : 'application/x-www-form-urlencoded'
}

msg.payload = {
        apikey : 'XXXXXXXXXXXXX',
        url : msg.payload.imgurl,
        outputMode : 'json'
};
return msg;
```



3. Agrega un nodo **http request** por medio del cual realizaremos la llamada REST. Configura las siguientes propiedades:

   Method: POST
   URL: http://access.alchemyapi.com/calls/url/URLGetRankedImageKeywords
   Return: a parsed JSON object

   Este es el *endpoint* para una función que devuelve palabras clave relacionadas al contenido de la imagen. El detalle de parámetros que acepta esta llamada y el formato de respuesta pueden ser consultados en la documentación del API: http://www.alchemyapi.com/api

4. Inserta un nuevo nodo **function**. Nómbralo `setHead`

   Para utilizar el objeto respuesta del nodo anterior como entrada del template debemos modificar los encabezados. Incluye el siguiente código en él cuerpo de la función:

```
msg.headers = {
  'Content-Type':'text/html',
  accept:'text/html,application/xhtml+xml'
}
return msg;
```

5. Modifica el nodo **template**. Incluye después de la forma HTML el código para desplegar la respuesta de la llamada

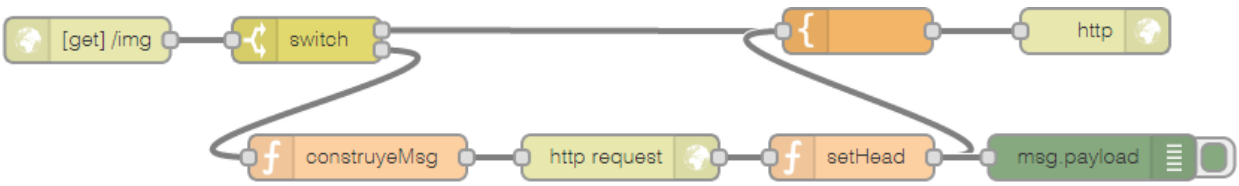
   Este nodo utiliza la notación **mustache** para construir el contenido. Si no estás familiarizado con esta notación puedes obtener más información en http://mustache.github.io/mustache.5.html

```
{{payload.status}} {{payload.statusInfo}}
<br>

<img src="{{payload.url}}"><br>
<b>Palabras Clave:</b><br>
{{#payload.imageKeywords}}
    {{score}} {{text}}<br>
{{/payload.imageKeywords}}
```

6. Opcionalmente incluye un nodo **debug** después de la llamada REST para ver el contenido completo de la respuesta.

7. Los nodos deben quedar ligados de la siguiente manera.



8. Despliega la aplicación.

Ahora, al enviar la forma con la dirección de una imagen, se desplegará una lista de palabras clave relacionadas con ésta y un valor de referencia que indica la confianza que tiene el API en esta clasificación.

¡Felicidades! Has creado una aplicación que utiliza los servicios análisis de imágenes de AlchemyAPI usando Node-RED.

¡Es tu turno!

La mejor forma de comprobar lo que has aprendido es agregando funcionalidad a la aplicación por ti mismo.

Este es el requerimiento:

Cuando envías imágenes que contienen personas, la llamada al método `GetRankedImageKeywords` incluye la palabra clave **"person"** en la respuesta.

El API de Alchemy tiene un método especial para analizar imágenes de personas, incluyendo la clasificación por género, rango de edad, e identificación de la persona en caso de algunas celebridades.

Consulta la documentación del método `RankedImageFaceTags` y despliega la información de esta llamada sólo en caso de identificar personas en la imagen.

Tip: El endpoint de este método es:

http://access.alchemyapi.com/calls/url/URLGetRankedImageFaceTags