Apartado 17:
Programación de un skill de Alexa
(parte 2)

Autor:
Miguel Ángel García Cumbreras

Recordamos...

- En nuestro Front-End, llamado VUI (Voice User Interface), tendremos en el panel izquierdo los intents siguientes:
  - predefinidos: AMAZON.CancelIntent, AMAZON.HelpIntent, AMAZON.StopIntent y AMAZON.NavigateHomeIntent.

    Mirad la gran cantidad de intents que ofrece

  - agregamos el intent de nuestra skill ListItemsIntent.

Recordamos...

Por ejemplo:

si queremos que nuestra skill ofrezca al usuario la posibilidad de escuchar de nuevo lo que Alexa acaba de decir se utiliza AMAZON.RepeatIntent en conjunto con AMAZON.YesIntent y AMAZON.NoIntent, si queremos darle la posibilidad al usuario de responder sí o no. Estas respuestas implican otras solicitudes o transacciones en nuestra skill.

Creando un skill con Python (16)

Desarrollando la función Lambda con Python

- Necesitamos instalar para Python el paquete python-lambda-local
- También necesitaremos instalar el paquete ASK-SDK para Python, desarrollado por el equipo de Alexa (pip install ask-sdk)
- Creamos el fichero principal “index.py”, que será la función lambda

Creando un skill con Python (17)

¿Qué espera Alexa Voice Service?

- Un JSON con una cierta estructura, y nos llegará una respuesta que será otro JSON con una cierta estructura
- podemos implementar nuestra skill utilizando clases (classes) o decoradores(decorators).
- Es recomendable de hecho utilizar un solo patrón para toda la skill.

Creando un skill con Python (18)

Clases

- Cada una de las clases que vamos a implementar será capaz de responder a un intent. Por ejemplo, tendremos una clase llamada ListItemsIntent.
- De forma más abstracta, cada solicitud (Request) que se haga a la skill debe tener un handler (clase o decorador), una forma de responder a ello.
- Si esa solicitud es desconocida debe responder apropiadamente (excepciones).

Creando un skill con Python (19)

Clases

- En nuestras clases vamos a derivar de las siguientes dos clases que pertenecen al paquete ask_sdk_core.dispatch_components:
  - AbstractRequestHandler: Esta clase nos permite gestionar solicitudes y responder apropiadamente a estas.
  - AbstractExceptionHandler: Esta clase nos permite gestionar las excepciones y responder apropiadamente a estas.

Creando un skill con Python (20)

Clases

- Cada una de las clases que vamos a implementar en nuestra skill implementarán dos métodos:
  - can_handle: devuelve true o false indicando si esta clase en particular es capaz de responder a la solicitud que está recibiendo.
  - handle: se encarga de preparar la respuesta (el JSON) que requiere Alexa. Es la implementación de lo que queremos que haga nuestra skill cuando reciba una solicitud en particular.

Creando un skill con Python (21)

Clases que vamos a desarrollar

- LaunchRequest
- ListItemsIntent
- HelpIntent
- CancelIntent
- StopIntent
- SessionEndedRequest
- AllExceptions

Creando un skill con Python (22)

LaunchRequest

Esta solicitud es enviada la skill por el usuario sin una intención específica. Por ejemplo, se ejecuta cuando el usuario dice “Alexa, abre exploradores fantásticos”. En esta solicitud el usuario no especifica una intención solo pide a Alexa inicie la skill.

Creando un skill con Python (23)

SessionEndedRequest

Es la solicitud que la skill recibe notificando que la sesión ha sido cerrada por alguna de las siguientes razones:

- El usuario dice “salir”.
- El usuario no responde o dice algo que no coincide que ninguna de las intenciones definidas en la VUI mientras el dispositivo estaba escuchando (Voice User Interface, Front-End).
- Ha habido un error.

Creando un skill con Python (24)

Intent: AMAZON.HelpIntent

Se envía cuando el usuario dice, por ejemplo: “ayuda”, “ayúdame”, “puedes ayudarme”.

Intent: AMAZON.CancelIntent

Se envía cuando el usuario dice, por ejemplo: “cancela”, “olvídalo”, “cancela eso”. Podemos dejar que el usuario cancele una acción manteniéndose en la skill o que el usuario salga completamente.

Creando un skill con Python (25)

Intent: AMAZON.StopIntent

Se envía cuando el usuario dice, por ejemplo: “para”, “apagar”, “cállate”. Podemos dejar que el usuario detenga una acción manteniéndose en la skill o que el usuario salga completamente.

Creando un skill con Python (26)

La programación en Python

Podéis encontrar un ejemplo completo de este código en:
https://github.com/frivas/alexa-skill-exploradoresfantasticos

Probando el skill

Lo podemos probar de 3 formas:

- Pruebas locales utilizando python-lambda-local
- Utilizando la opción Test del servicio Lambda
- En el simulador de la consola de desarrolladores de Alexa

Probando el skill

Pruebas locales utilizando python-lambda-local

1. Vamos a la consola de AWS, al servicio Lambda
2. Seleccionamos Configure test events
3. Desplegamos las opciones y seleccionamos Amazon Alexa Start Session. Ponemos un nombre a la sesión.
4. Copiamos todo el JSON y guardamos este evento
5. Nos vamos al directorio de nuestro index.py, y creamos el fichero tests/LaunchRequest.json. Escribimos dentro el JSON
6. Probamos:
   python-lambda-local index.py -f handler tests/LaunchRequest.json

Probando el skill

Pruebas con el simulador

1. Debemos tener nuestro modelo construido. En la Consola de desarrolladores de ASK click en Build y vemos la opción Build Model
2. Luego hacemos click en Test, vemos que las pruebas para nuestra skill están desactivadas, simplemente las activamos y utilizando el pequeño icono de un micrófono, debemos dejarlo pulsado al dar la orden, como un walkie-talkie.

En el panel de la izquierda aparece nuestro enunciado y luego la respuesta de Alexa, que corresponde con lo que esperamos.

EJERCICIO (opcional)

MySkill: FCM

MTV: Crear un sencillo skill de Alexa

- Sigue los pasos descritos en las dispositivas para crear un skill de Alexa.
- Genera un documento que muestre el funcionamiento del skill, y súbelo a la actividad de ILIAS, así como el código fuente generado.