Justo García de Paredes Haba

Bot de Telegram (Iniciación)

Actualizado a fecha de: 25 de Octubre del 2018

ENTORNO DE TRABAJO

Para esta documentación, se ha tenido en cuenta un entorno de trabajo que consta de:

- SO: Linux/MacOS

INSTALACIÓN

PIP/PIP3

PIP es un gestor de paquetes para Python.

- **Linux**
    - Si existe **Python 2.7.9+** or **Python 3.4+**
        - No hace falta, *PIP* ya está instalado.
    - **Python 2**
        - ***sudo apt-get install python-pip***
    - **Python 3**
        - ***sudo apt-get install python3-pip***
- ***MacOS***
    - Si se instala *Python3*, *PIP* viene instalado por defecto.
        - ***brew install python3***

PYTHON-TELEGRAM-BOT

Instalación de la librería ***Python-Telegram-Bot***.

- ***pip/pip3 install python-telegram-bot***

CREACIÓN DE UN BOT BÁSICO

1. Crear el bot con **BotFather** (***@BotFather***)
    a. Ejecutar el comando ***/newbot***
    b. Darle un ***nombre al bot***. Este será el nombre que tendrá el bot.
    c. Darle un ***nombre identificador***. Éste será el nombre identificativo por el que se podrá buscar el bot.
    d. Utilizar el ***TOKEN*** devuelto por el bot para su implementación en Python.
2. Crear un proyecto python o la estructura donde se van a almacenar los ficheros ***.py***.
3. Crear un fichero *.py* que será el script inicial.
    a. Importar la librería instalada.
    import telegram
    b. Iniciar el objeto *bot*, que contiene la representación del bot en una clase Python.
    bot = telegram.Bot(token='<TOKEN>')
    c. Imprimir en pantalla el bot para comprobar que se ha realizado la asignación correctamente.
    print(bot.get_me())
    d. Instanciar la clase Updater, encargada de recibir las actualizaciones de la API a través del bot.
    from telegram.ext import Updater
    updater = Updater(token='<TOKEN>')
    e. Guardar en una variable, el objeto *Dispatcher*, creado por el *Updater*. Sobre este objeto se registrarán los handlers encargados de actuar cuando llega una petición.
    dispatcher = updater.dispatcher
    f. Es posible establecer un *Logger* para controlar los eventos que suceden y llevar un registro.
    import logging
    logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)

4. En otro archivo *.py*, establecer los métodos que controlarán los eventos lanzados. Éstos serán los handlers.
   def start(bot, update):
   bot.send_message(chat_id=update.message.chat_id, text="I'm a bot, please talk to me!")

5. Importar el fichero contenedor de los handlers en el principal, para poder hacer uso de estas funciones. Ahora, se realizan el binding entre el método a ejecutar y el comando que tiene que seleccionar el usuario para invocarlo. Para hacerlo, se instancia la clase *CommandHandler*, pasándole como parámetros el comando a invocar y el nombre del procedimiento a ejecutar. Dicha instancia, se le pasa al *Dispatcher* mediante la función *add_handler*.
   from telegram.ext import CommandHandler
   start_handler = CommandHandler('start', start)
   dispatcher.add_handler(start_handler)

6. Una vez añadidos los handler, se puede iniciar el bot mediante la ejecución de: ***updater.start_polling()***

7. Para comprobar el correcto funcionamiento del bot, se puede ejecutar desde la conversación en Telegram, el comando ***/start***.

8. El desarrollador puede personalizar su bot, accediendo a BotFather.

   a. ***/mybots***. Aquí se mostrarán una serie de opciones donde se pueden editar el nombre, la descripción la imagen de perfil del bot, etc. Una opción interesante, es la de añadir comandos, con ella, se establecen una serie de comandos que contienen su descripción. De esta forma, el usuario puede ver los comandos que puede ejecutar y no tener la necesidad de escribirlo, puede únicamente seleccionarlos para ver su ejecución.

9. Existen numerosas funciones más añadidas en la librería con las que se puede lograr que el Bot haga cosas más complejas, se deja los enlaces a documentación de referencia que se pueden leer para ver ejemplos de ésto. A continuación, se listan algunos de ellas.

   a. MessageHandler, con éste objeto el bot puede responder ante el estímulo de un simple mensaje. El usuario no tiene que ejecutar ningún comando concreto.

   b. A los handlers, se les puede pasar argumentos, de tal forma que se puedan tratar. El usuario simplemente los escribe a continuación del comando a ejecutar.

c. Jobs. El bot puede llevar a cabo tareas planificadas cada día, cada mes, cada hora, etc.

d. ETC.

DOCUMENTACIÓN DE REFERENCIA

- Documentación de Python Telegram Bot
    - https://python-telegram-bot.readthedocs.io/en/stable/
- Tutorial “Tu primer bot”
    - https://github.com/python-telegram-bot/python-telegram-bot/wiki/Extensions-%E2%80%93-Your-first-Bot
- Introducción a la API
    - https://github.com/python-telegram-bot/python-telegram-bot/wiki/Introduction-to-the-API
- Bot de ejemplo desarrollado por Víctor Julián García Granado para una charla de ExtrePython.
    - https://github.com/vjgarciag96/TelegramBotExtrePython