dialogflowpy-webhook

Release 0.0.1

Vishal Balasubramanian

Apr 27, 2020

CONTENTS

A Python module for parsing and creating Requests and Responses of Dialogflow Fulfillment Library. The Dialogflow Fulfillment Library allows you to connect natural language understanding and processing to your own system. Using Fulfillment, you can surface commands and information from your services to your users through a natural conversational interface.

This dialogflowpy-webhook module is intended to help build Python Dialogflow Fulfillment for multiple integrations including Google Assistant, Facebook and Telegram. It is expected to work for Slack and Line as well.

CHAPTER

ONE

FEATURES

- Parsing Dialogflow requests and get details like Intent, Slot Values, Contexts and more!
- Supports creation of Responses like: Text Responses, Cards, Quick Replies/Suggestions and Images
- Supports creation of all Google Assistant Rich Responses

CHAPTER

TWO

A SIMPLE BOT WITH DIALOGFLOWPY-WEBHOOK AND FLASK

```
from flask import Flask,request
import dialogflowpy_webhook as dfw
app = Flask(__name__)

app.route("/fulfillment")
def fulfillment():
   response_handler = dfw.response_handler()
   response_handler.generic_response("Hello from Python!")
   return response_handler.create_final_response()
```

The snippet above will make a dialogflow bot reply “Hello from Python!” whenever invoked.

CHAPTER

THREE

RESOURCES

- Head over to the Getting Started section to create a Dialogflow bot with dialogflowpy_webhook
- Take a look at the Module Reference which contains all the modules available in dialogflowpy_webhook
- Take a look at the examples here which contain some use cases for many functions

CHAPTER

FOUR

REPORTING BUGS

In case of any problems or suggestions with this module, feel free to open a Github Issue

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 Getting Started

This Page gives instructions to create a basic bot

5.1.1 Installation

```
pip3 install dialogflowpy-webhook flask
```

Note: For this example, we are using Flask. You can use any Python web framework (like Django, Bottle etc). Installing Flask is not a must for dialogflowpy-webhook to work

5.1.2 Webhook Server Setup

Dialogflow has a few requirements for the webhook service. Your Webhook must have a public URL and must have an SSL certificate. There are many free options for this to try. You can use services like Heroku, NGROK, Pythonanywhere etc. Your webhook code must be hosted on such sites and you should note the URL

5.1.3 Basic Fulfillment code

Create a new file (bot.py and paste the code into it)

```
from flask import Flask,request
import dialogflowpy_webhook
app = Flask(__name__)

app.route("/fulfillment")
def fulfillment():
   response_handler = dialogflowpy_webhook.response_handler()
   response_handler.generic_response("Hello from Python!")
   return response_handler.create_final_response()

 if __name__ == "__main__":
   app.run()
```

This code handles our webhook. This file should be deployed on the webhook server listed above

5.1.4 Dialogflow Setup

- Sign in to Dialogflow
- Create a new agent with any name of your choice and set the timezone to your home timezone and click on "Save"
- After creating the agent, go to fulfillment and enter the URL where your app is located and click on "Save"

5.1.5 Testing

For this example, we will be testing using Actions on Google.

- In the Dialogflow Console, click on "Intergrations" and click on Google Assistant -> Test. This will take you to Actions on Google Console. There, in the simulator, type "Talk to my Test App" and Google should say "Hello from Python!"

5.1.6 Extending further

Now that we have created a basic bot, you can extend it further. You can:

- Take a look at the Module Reference which contains various functions which can be used to extend your bot
- Take a look at the examples here which contain some use cases for many functions

5.2 Module Reference

5.2.1 The `request_handler` Class

class dialogflowpy_webhook.**request_handler**(*dialogflowRequestJson*)

This Class Handles the Parsing of Dialogflow Requests and get details like Intent, Parameters, Session ID etc

Parameters **dialogflowRequestJson** – The Dialogflow Request JSON

get_intent()

Returns the Intent Dictionary which triggered the Webhook

Raises **TypeError** – This Error is Raised if the Intent Dictionary can't be retived if the Request JSON is Malformed

Returns Intent Object

Return type dict

get_intent_name()

Returns the Intent Name which triggered the Webhook

Raises **TypeError** – This Error is Raised if the Intent Name can't be retived if the Request JSON is Malformed

Returns Intent Name

Return type str

get_intent_displayName()

Returns the Intent Display Name (this is the Intent Name which you would have specified in Dialogflow) which triggered the Webhook

Raises **TypeError** – This Error is Raised if the Intent Display Name can't be retived if the Request JSON is Malformed

Returns Intent Display Name

Return type str

get_parameters()
Returns a Dictionary of filled Parameter Values

Returns Parameter Object

Return type dict

get_parameter(*param*)
Returns a Parameter Value by Parameter Name

Parameters **param** – The Parameter name to retrive the Value

Raises **KeyError** – This Error is Rasied if the Parameter is not found

Returns Parameter Value

Return type str

get_action()
Returns the Action Name Specified for the Intent

Returns Action Name

Return type str

get_session_id()
Returns the Session ID of the Dialogflow Session

Raises **TypeError** – This Error is Raised if the Session ID can't be retived if the Request JSON is Malformed

Returns Session ID

Return type str

get_context_by_name(*contextName*)
Returns a Context Dictionary by Context Name

Parameters **contextName** (*str*) – The Context Name to retrive the Context JSON

Raises **LookupError** – This Error is Raised if The Context is not found

Returns Context Object

Return type dict

get_capabilities()
Returns a list Google Assistant Capabilities for a particular surface (eg. Smart Display, Mobile Phone, Chromebook etc.) from where the bot is accessed.

Returns Capabilities List

Return type list

Note: This Feature is specific only for Google Assistant. This will return an empty list if the bot is accessed from platforms which are not Google Assistant

---

get_payload()
Returns the Platform Specific Payload from where the request originated

Returns Payload Object

Return type dict

get_source()
Returns the source where the request originated

Returns Source where the request originated

Return type str

5.2.2 The response_handler Class

class dialogflowpy_webhook.**response_handler**
The Class handles the creation of Dialogflow Responses

Note: There are 2 types of Rich Responses which can be created using this class. They are: Generic Rich Responses and Google Assistant Rich Responses. Generic Responses work on all platforms except Google Assistant. Functions that create generic responses start with 'generic'. For Google Assistant, you should use Google Assistant Rich Responses. These functions start with 'google_assistant'

add_context(*sessionID*, *contextName*, *lifespan=0*, *params={}*)
Adds/Changes a Dialogflow Context

Parameters

- **sessionID** (*str*) – The Session ID
- **contextName** (*str*) – The name of the Context to add/edit
- **lifespan** (*int, optional*) – The number of conversational turns for which the context remains active, defaults to 0
- **params** (*dict, optional*) – The Dictionary of Data to store in the context, defaults to {}

trigger_event(*event*, *params*, *langcode='en-US'*)
Triggers a Dialogflow Event

Parameters

- **event** (*str*) – The Name of the Event to Trigger
- **params** (*dict*) – The Dictionary of Parameters
- **langcode** (*str, optional*) – The Language Code of the Agent, defaults to "en-US"

Note: When the response contains event, other things are ignored (except Contexts)

simple_response(*speech*)
A Generic Text to be displayed or told to the user.

Parameters **speech** (*str*) – The Text to be displayed or said to the user

Note: `simple_response` works on all platforms including Google Assistant. However, it is recommended to use `google_assistant_response` for Google Assistant and `generic_rich_text_response` for text responses on other platforms.

generic_rich_text_response(*text*)
A Generic Rich Text Response to display to the user. Unlike `generic_response`, you can have multiple `generic_rich_text_response`

Parameters **text** (*str*) – The Text to be displayed to the user

generic_card(*title*, ***kwargs*)
A Generic Card to be displayed to the user

Parameters

- **title** (*str*) – The Title of the Card
- **subtitle** (*str, optional*) – The Subitle of the Card
- **imageURL** (*str, optional*) – The Link of the Image to be displayed on the card

generic_card_add_button(*btntitle*, *btnlink*)
Adds a button to a Generic Card. When clicked, directs to a website

Parameters

- **btntitle** (*str*) – The button's title
- **btnlink** (*str*) – The link to redirect to on click

Raises **AttributeError** – This Error is Raised if a new button is added before calling `generic_card`

generic_add_suggestions(*suggestionList*, ***kwargs*)
Adds Suggestion Chips/Quick Replies to be displayed.

Parameters

- **suggestionList** (*str, optional*) – The List of Suggestions/Quick Replies
- **title** – The title of the Suggestions

generic_image(*imageURL*, *imgalt*)
Sends an Image to the User

Parameters

- **imageURL** (*str*) – The URL of the Image
- **imgalt** (*str*) – The Alt Text for the Image

google_assistant_response(*speech*, ***kwargs*)
A Google Assistant speech to be said (and displayed) to the user

Parameters

- **speech** (*str*) – The Text to be said to the user
- **displayText** (*str, optional*) – The text to be displayed in the chat bubble while telling the speech
- **endConversation** (*bool*) – Specifies wheather this response should end the conversation or not

Note: This MUST Before any Google Assistant Rich Response. Failing to do so will result in an error in Google Assistant

google_assistant_card(*title*, ***kwargs*)
A Google Assistant Card to be displayed to the user

Parameters

- **title** (*str*) – The Title of the Card
- **subtitle** (*str, optional*) – The subtitle of the Card
- **formatted_text** (*str, optional*) – The text to be displayed along with the card
- **btnName** (*str, optional*) – The Name of the button to be displayed on the card
- **btnLink** (*str, optional*) – The link to redirect on button click
- **imageURL** (*str, optional*) – The URL of the image to be displayed on the card
- **imageAlt** (*str, optional*) – The Alt Text of the image to be displayed on the card
- **imageDisplayOption** (*str, optional*) – The Display options for the image (Click here For a list of image display options)

google_assistant_new_carousel()
Creates a New Google Assistant Carousel

google_assistant_carousel_add_item(*title*, *url*, *imageURL*, *imgalt*, *description=''*, *footer=''*)
Adds a new item to a Google Assistant Carousel

Parameters

- **title** (*str*) – The title of the carousel item
- **url** (*str*) – The URL to redirect to when the Carousel item is clicked
- **imageURL** (*str*) – The URL of the image to be displayed on the caarousel item
- **imgalt** (*str*) – The Alt text of the image to be displayed on the caarousel item
- **description** (*str, optional*) – The description to be displayed on the carousel item, defaults to “”
- **footer** (*str, optional*) – The footer to be displayed on the carousel item, defaults to “”

Raises **AttributeError** – This Error is raised if a new item is added before calling `google_assistant_new_carousel`

google_assistant_add_suggestions(*suggestionList*)
Adds Google Assistant Suggestion Chips to be displayed

Parameters **suggestionList** (*list*) – The list containing the suggestions to be displayed

google_assistant_new_table(***kwargs*)
Creates a new Google Assistant Table Card

Parameters

- **title** (*str, optional*) – The title of the Table Card
- **subtitle** (*str, optional*) – The subtitle of the Table Card

- **imageURL** (*str, optional*) – The URL of the image to be displayed on the table card
- **imageAlt** (*str, optional*) – The Alt text of the image to be displayed on the table card

google_assistant_table_add_header_row(*headerList*)
Adds a Header row to a Google Assistant Table Card

Parameters **headerList** (*list*) – The list containing the header rows to be added

Raises **AttributeError** – This Error is raised if a header row is added before calling `google_assistant_new_table`

google_assistant_table_add_row(*cellList*, *addDivider*)
Adds a new row to a Google Assistant Table Card

Parameters

- **cellList** (*list*) – The list containing the rows to be added
- **addDivider** (*bool*) – Specifies if a divider should be added after the row

Raises **AttributeError** – This Error is raised if a row is added before calling `google_assistant_new_table`

google_assistant_media_response(*mediaURL*, *description*, *displayName*, ***kwargs*)
Creates a Google Assistant Media Response to play music

Parameters

- **mediaURL** (*str*) – The URL where the music is located
- **description** (*str*) – The description of the music
- **displayName** (*str*) – The name of the music to display
- **imageURL** (*str,optional*) – The URL of the image to be displayed along with the media response
- **imgAlt** (*str,optional*) – The Alt Text of the image to be displayed along with the media response

google_assistant_ask_permisson(*speech*, *permissionList*)
Asks for permission from user in Google Assistant to get details like User's real name and address

Parameters

- **speech** (*str*) – The reason for the Permisssion Request
- **permissionList** (*list*) – The list of Permissions to get from the user

create_final_response()
Creates the Final Response JSON to be sent back to Dialogflow

Raises **AttributeError** – This error is raised if you try to insert Google Assistant Suggestions to a Google Assistant Rich Response with no items

Returns The Response JSON

Return type Dictionary

INDEX

R

S

T