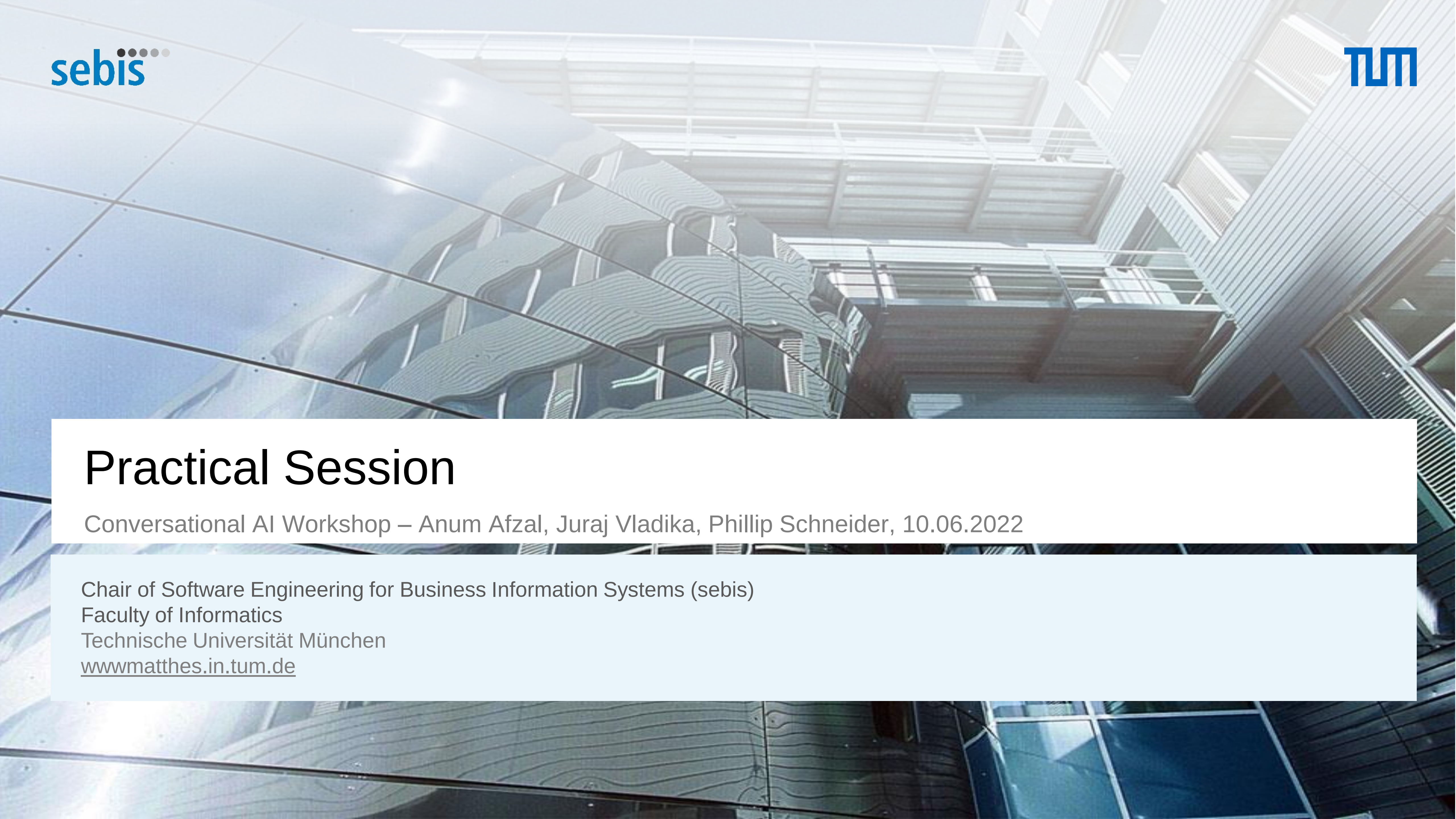


Practical Session

Conversational AI Workshop – Anum Afzal, Juraj Vladika, Phillip Schneider, 10.06.2022

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

Install the Requirements

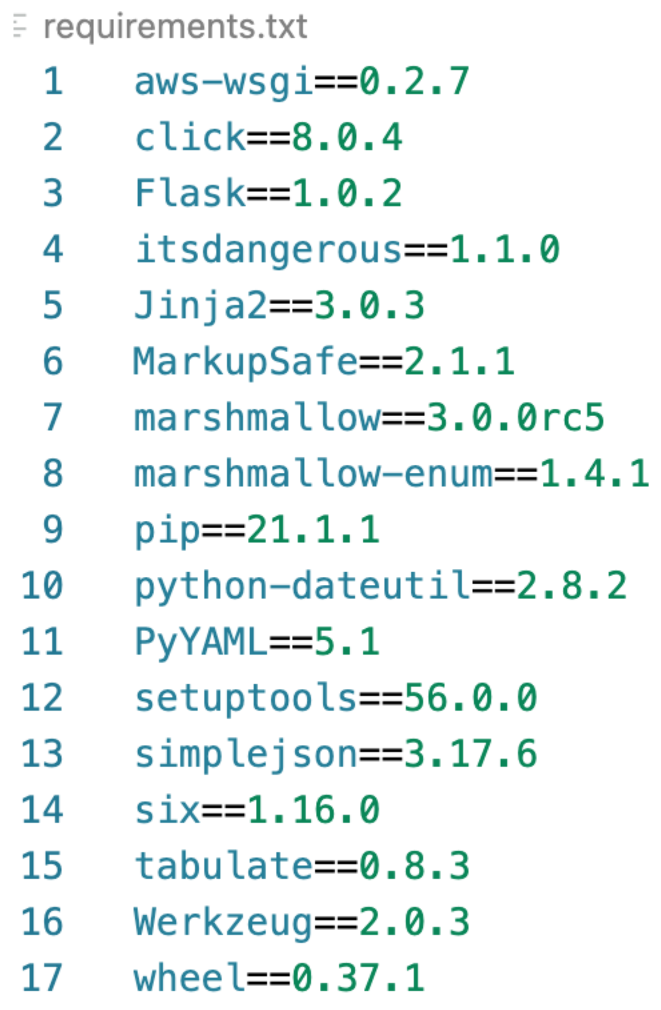
- Please <u>download the zip file from the e-mail</u> and unzip it
- Open a terminal and position yourself in the unzipped folder
- Then run following commands in the terminal

- **Unix/macOS**:

1. *python3 -m pip install --user virtualenv*
2. *python3 -m venv dialogflow-env*
3. *source dialogflow-env/bin/activate*
4. *python3 -m pip install requirements*
5. *pip install git+https://github.com/ONSEIGmbH/flask-dialogflow.git*

- **Windows**:

1. *py -m pip install --user virtualenv*
2. *py -m venv dialogflow-env*
3. *.\dialogflow-env\Scripts\activate*
4. *py -m pip install requirements*
5. *pip install git+https://github.com/ONSEIGmbH/flask-dialogflow.git*



requirements.txt

```
aws-wsgi==0.2.7
click==8.0.4
Flask==1.0.2
itsdangerous==1.1.0
Jinja2==3.0.3
MarkupSafe==2.1.1
marshmallow==3.0.0rc5
marshmallow-enum==1.4.1
pip==21.1.1
python-dateutil==2.8.2
PyYAML==5.1
setuptools==56.0.0
simplejson==3.17.6
six==1.16.0
tabulate==0.8.3
Werkzeug==2.0.3
wheel==0.37.1
```

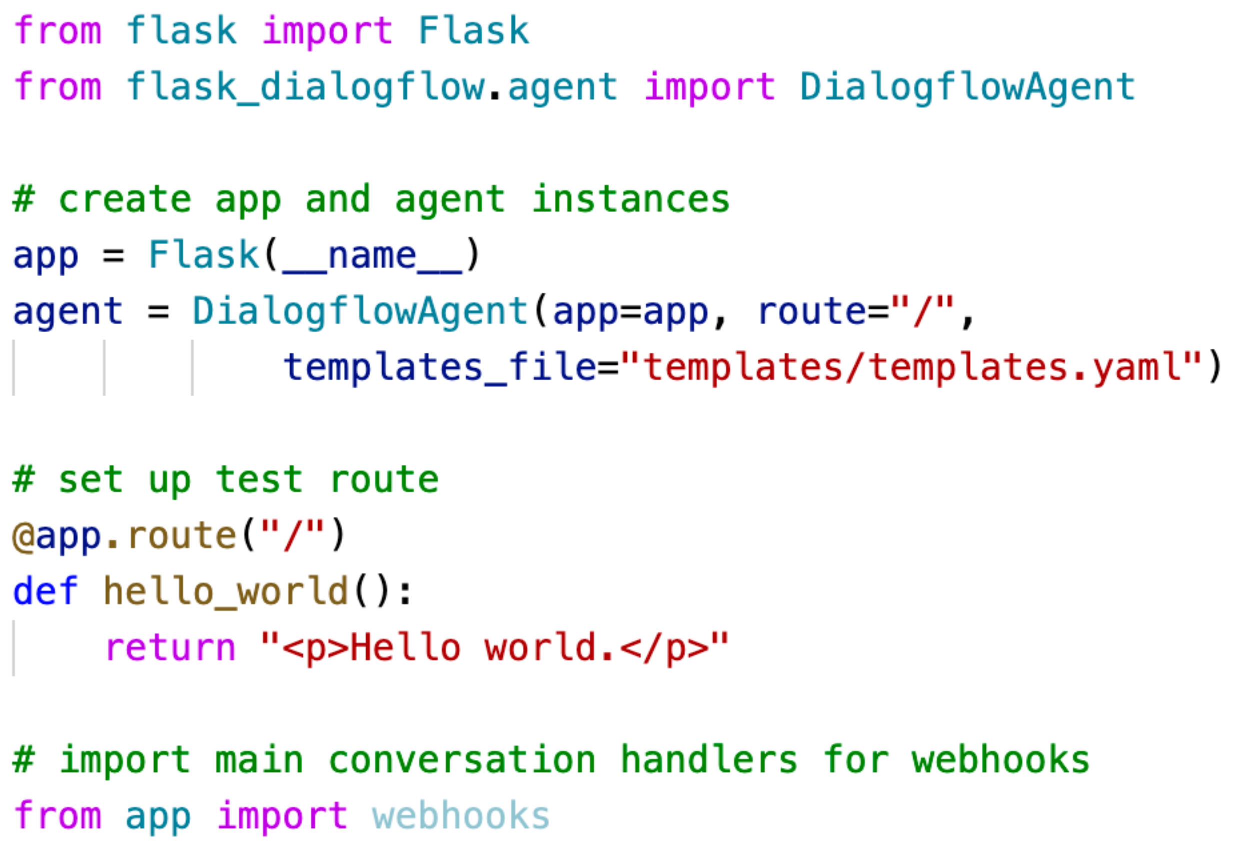
Creating the App

- This file is called __*init*__*.py* in the root folder

- Importing the Flask library
- Importing the Flask-Dialogflow library

- Creating an instance of the Flask app
- Creating an instance of an agent
  - Use the YAML file for templates

- Test route to home page (“/”)



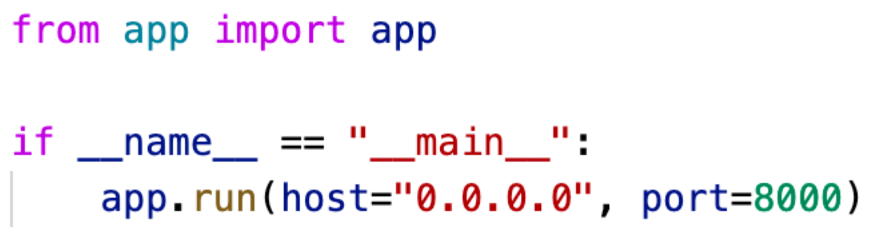
```python
from flask import Flask
from flask_dialogflow.agent import DialogflowAgent

# create app and agent instances
app = Flask(__name__)
agent = DialogflowAgent(app=app, route="/",
                        templates_file="templates/templates.yaml")

# set up test route
@app.route("/")
def hello_world():
    return "<p>Hello world.</p>"

# import main conversation handlers for webhooks
from app import webhooks
```

Running the App

- Run on **localhost** by using the *0.0.0.0* IP address and an available port (e.g. 8000)

- **Run in terminal with: *python run.py***
- You should get a message:
  - *Running on https://<localhost>:8000/*

- This file is called *run.py* in the root folder



```
from app import app

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

Ngrok: Installation

- You will need the package **ngrok** to create a public URL for your localhost server

- MacOS:
  - *brew install ngrok/ngrok/ngrok*

- Linux:
  - *curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | \
    sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null && \
    echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | \
    sudo tee /etc/apt/sources.list.d/ngrok.list && \
    sudo apt update && sudo apt install ngrok*

- Windows:
  - *choco install ngrok*

Check if it is working with:

- ***ngrok -h***

Ngrok: Running

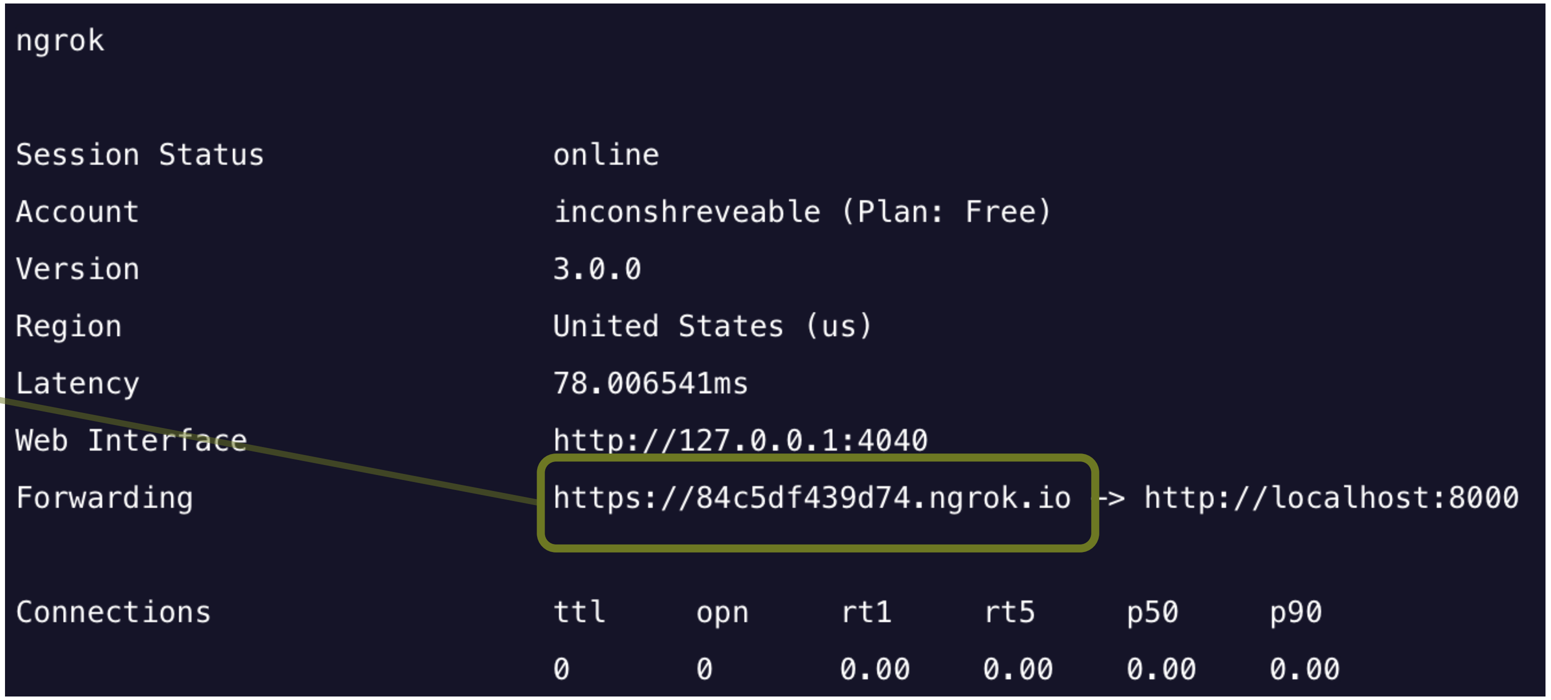
- Register for ngrok on: **https://dashboard.ngrok.com/signup**
- Then find your authentication token on: https://dashboard.ngrok.com/get-started/your-authtoken
- Copy the token and run this command:
  - *ngrok config add-authtoken [TOKEN]*

- Make sure your server (Flask app) is running on localhost and port 8000
- Then run this command:
  - *ngrok http 8000*

- Screen should look like this:

- Copy this URL

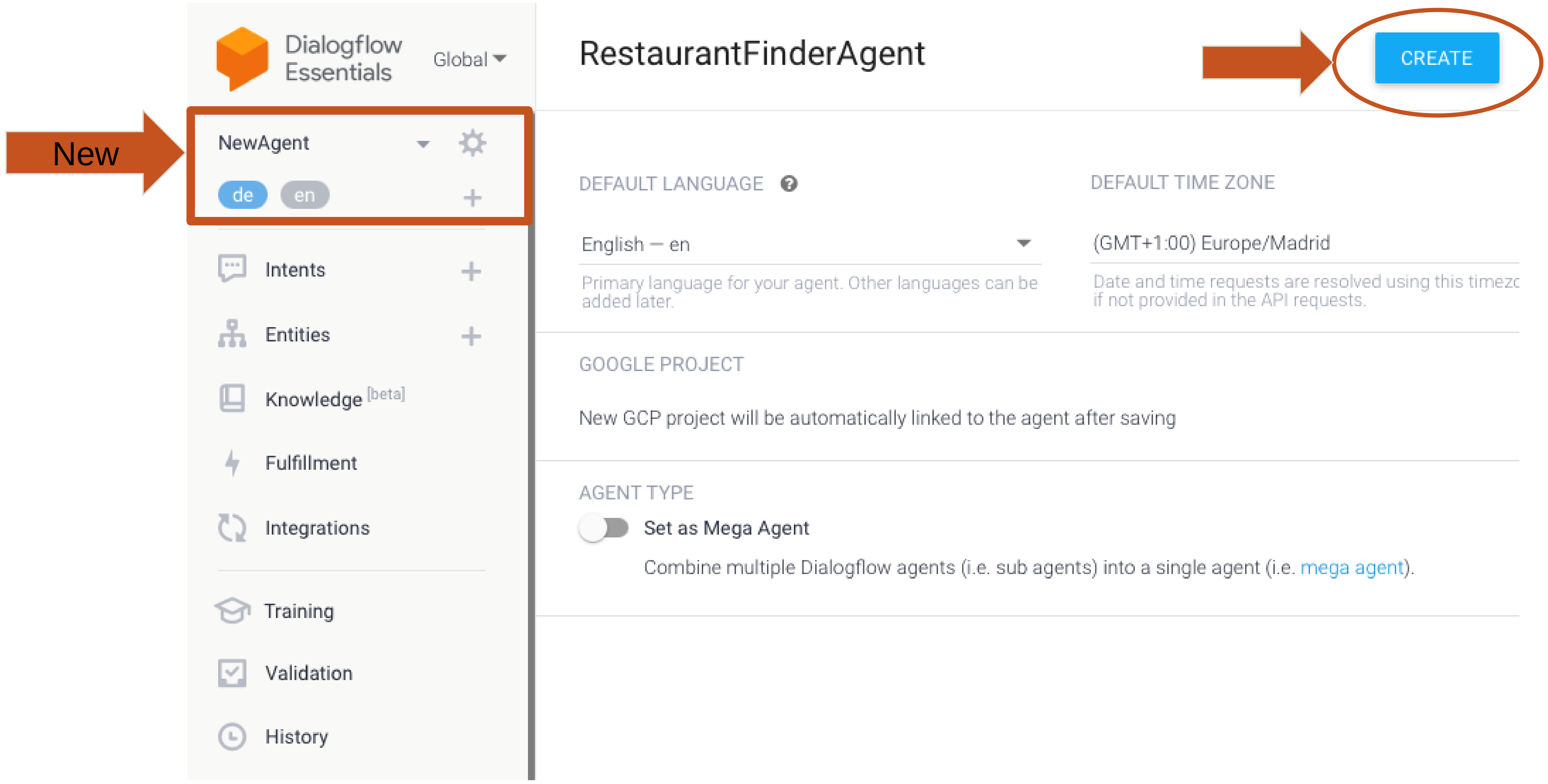
- Go to Dialogflow and paste to: *Dialogflow → Fulfilment → Webhook → URL*



```
ngrok

Session Status                online
Account                       inconshreveable (Plan: Free)
Version                       3.0.0
Region                        United States (us)
Latency                       78.006541ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://84c5df439d74.ngrok.io -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Outline

Dialogflow API Layout



RestaurantFinderAgent

DEFAULT LANGUAGE

English — en

Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE

(GMT+1:00) Europe/Madrid

Date and time requests are resolved using this timezc if not provided in the API requests.

GOOGLE PROJECT

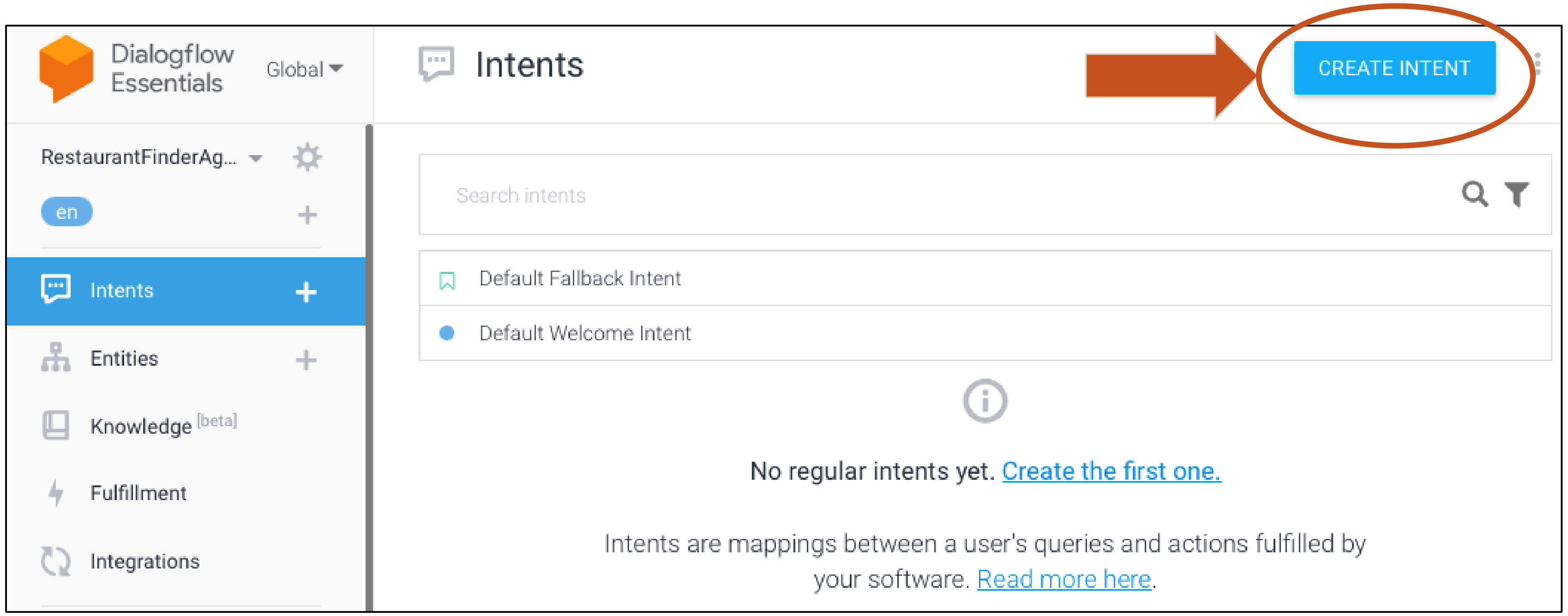
New GCP project will be automatically linked to the agent after saving

AGENT TYPE

Set as Mega Agent

Combine multiple Dialogflow agents (i.e. sub agents) into a single agent (i.e. mega agent).

Create a New Intent



Default Fallback Intent

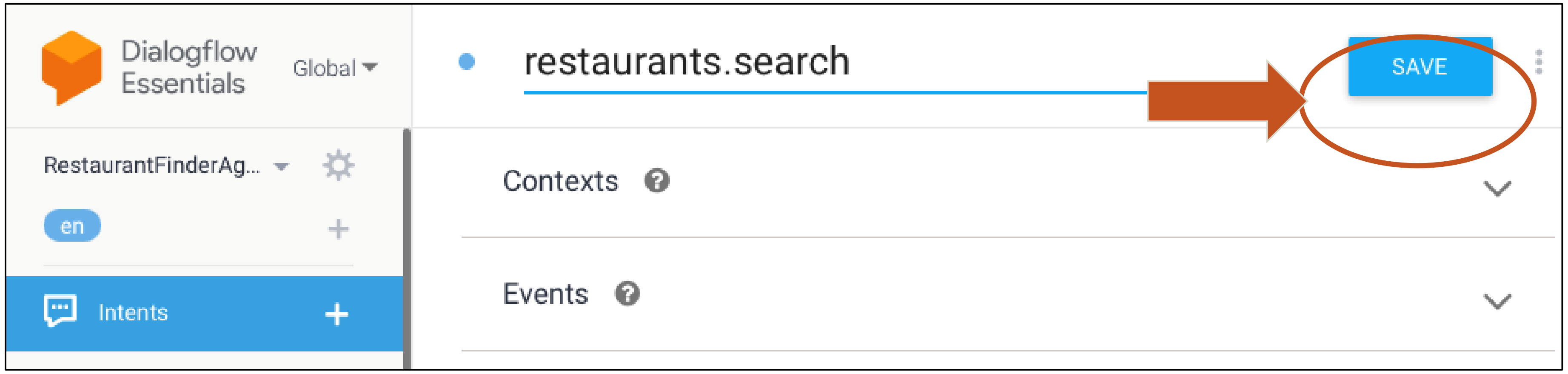
Default Welcome Intent

No regular intents yet. Create the first one.

Intents are mappings between a user's queries and actions fulfilled by your software. Read more here.

Note

A good practice for creating intents is to follow a consistent naming convention which indicates the main subject of the intent, e.g., *restaurants.search*



Intent: Components

Contexts ?

Connect intents

Contexts represent the current context of a user's request. This is helpful for differentiating phrases which may be vague or have different meanings depending on the user's preferences, geographic location, the current page in an app, or the topic of conversation. Contexts can be used to structure non-linear conversations. Learn more

ADD CONTEXT

Events ?

Trigger the intent from non-verbal signals

Events allow you to invoke an intent based on a non-verbal signal, such as a button click, or the start of a new conversation with a bot. Events can be used by external services to trigger Dialogflow intents, for example the Google Assistant's built-in intents. Learn more

ADD EVENT

Training phrases ?

When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

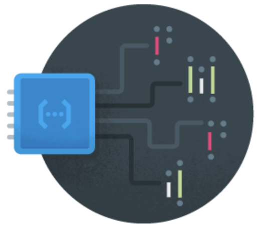
Train the intent with what your users will say

Provide examples of how users will express their intent in natural language. Adding numerous phrases with different variations and parameters will improve the accuracy of intent matching. Learn more

ADD TRAINING PHRASES

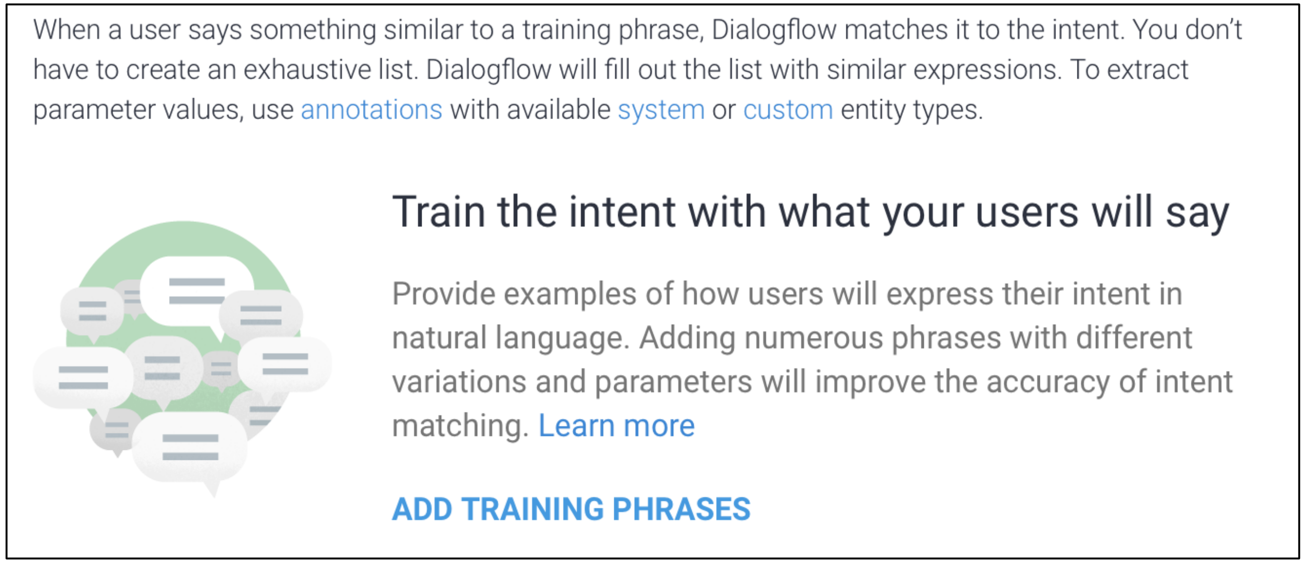
Action and parameters

Extract the action and parameters



Parameters are specific values extracted from a user's request when entities are matched. The values captured by parameters can be used in fulfillment, or in building a response. If you mark parameters as required, Dialogflow will prompt the user if their values were not extracted from their initial request. Learn more

Intent: Training Phrases

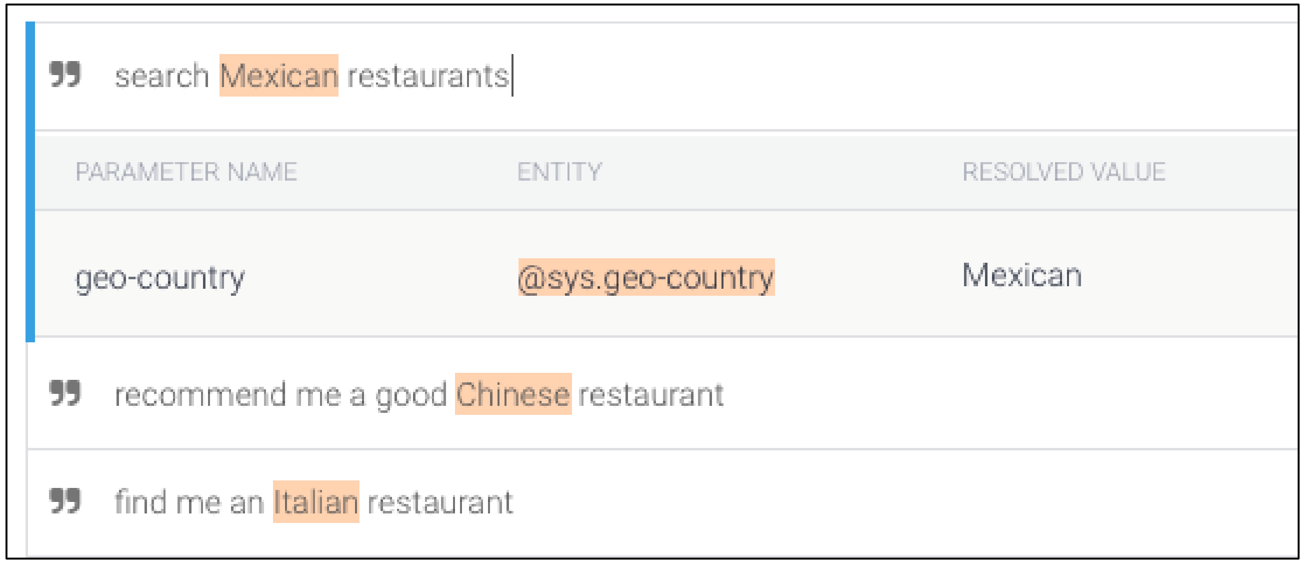


When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

Train the intent with what your users will say

Provide examples of how users will express their intent in natural language. Adding numerous phrases with different variations and parameters will improve the accuracy of intent matching. Learn more

ADD TRAINING PHRASES

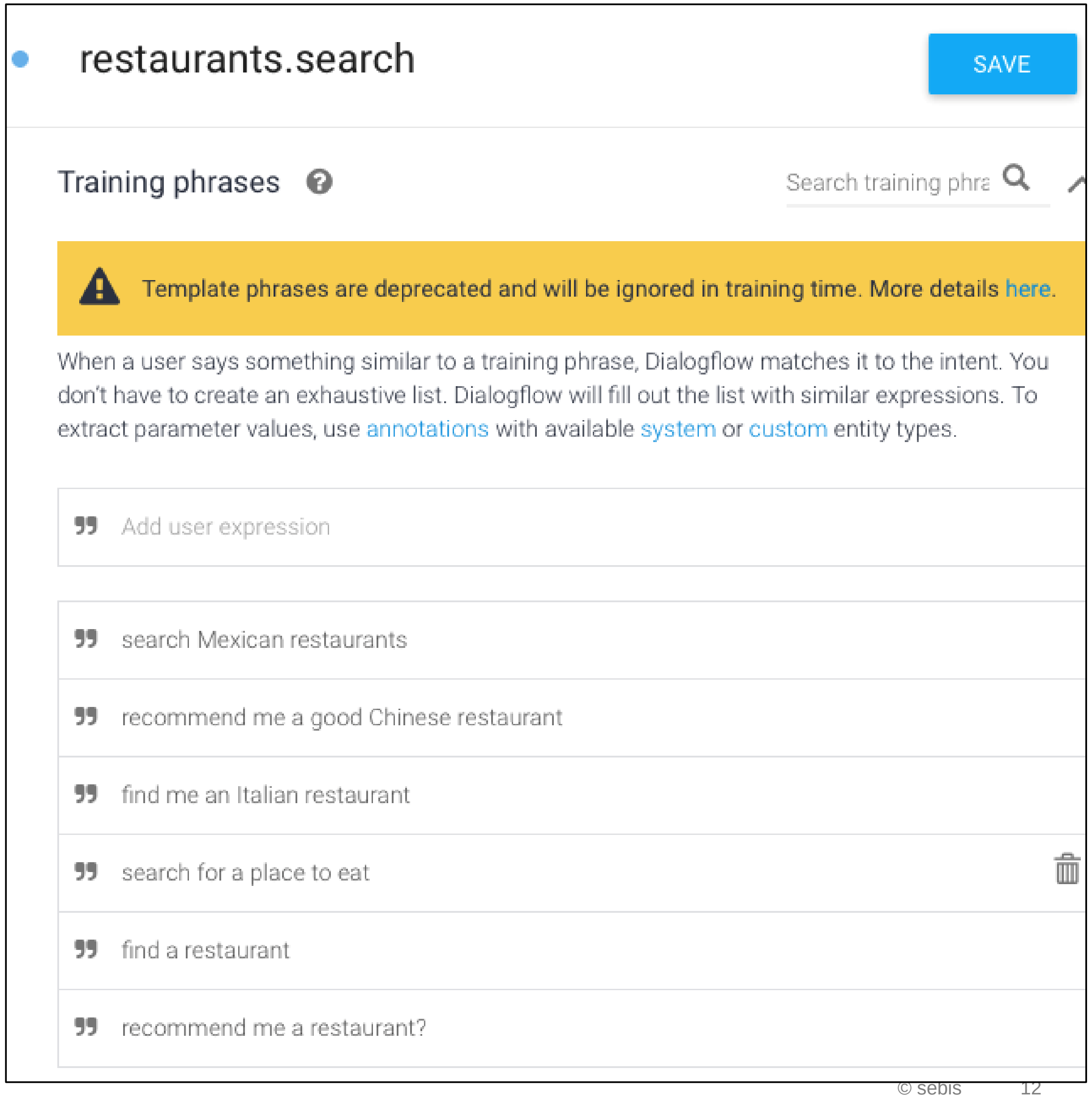


search Mexican restaurants

PARAMETER NAME	ENTITY	RESOLVED VALUE
geo-country	@sys.geo-country	Mexican

recommend me a good Chinese restaurant

find me an Italian restaurant



restaurants.search

SAVE

Training phrases

Search training phra

Template phrases are deprecated and will be ignored in training time. More details here.

When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

Add user expression

search Mexican restaurants

recommend me a good Chinese restaurant

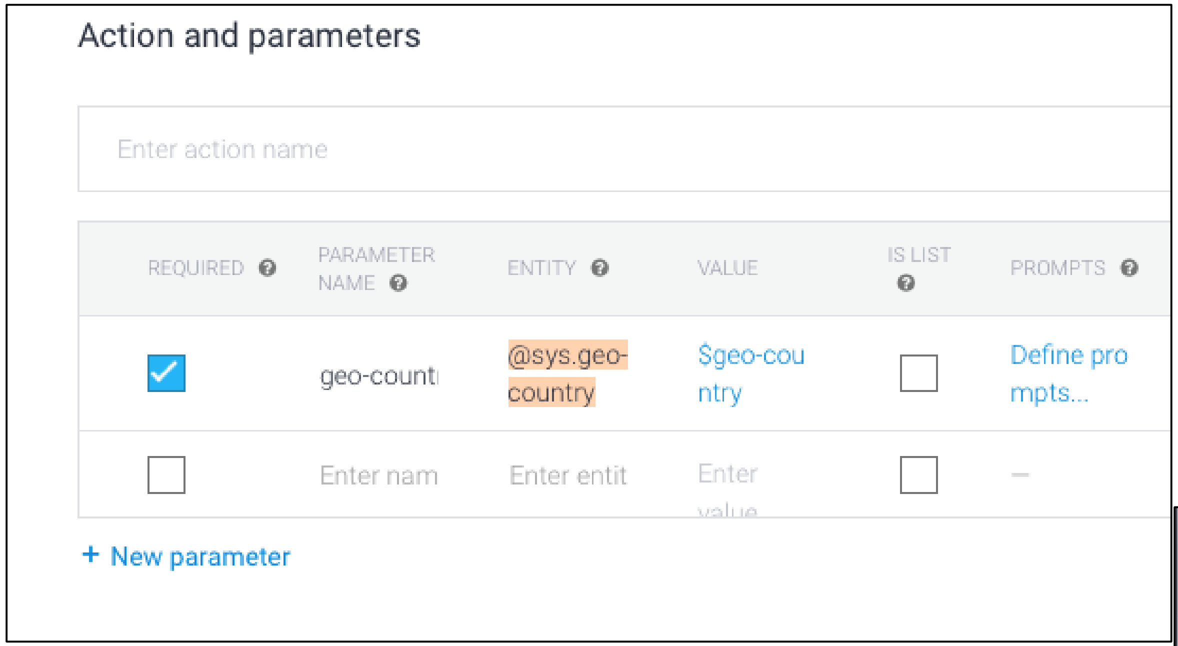
find me an Italian restaurant

search for a place to eat

find a restaurant

recommend me a restaurant?

Intent: Actions and Parameters

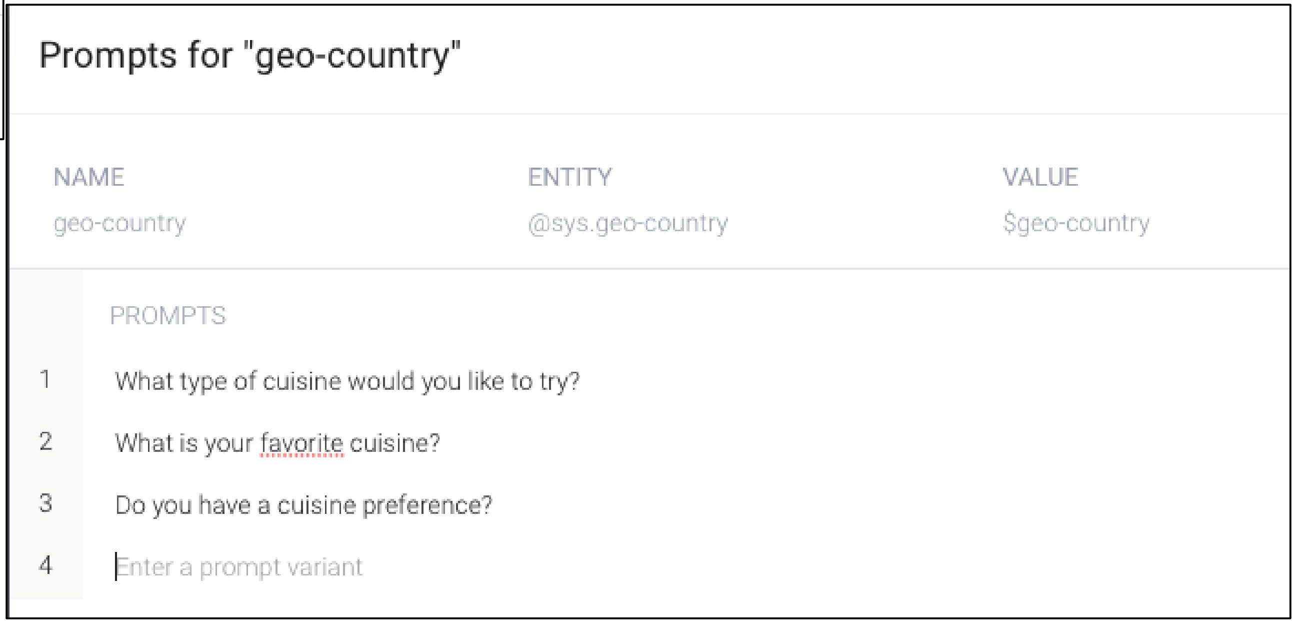


Action and parameters

Enter action name

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
✓	geo-count	@sys.geo-country	$geo-country	☐	Define prompts...
☐	Enter nam	Enter entit	Enter value	☐	–

+ New parameter



Prompts for "geo-country"

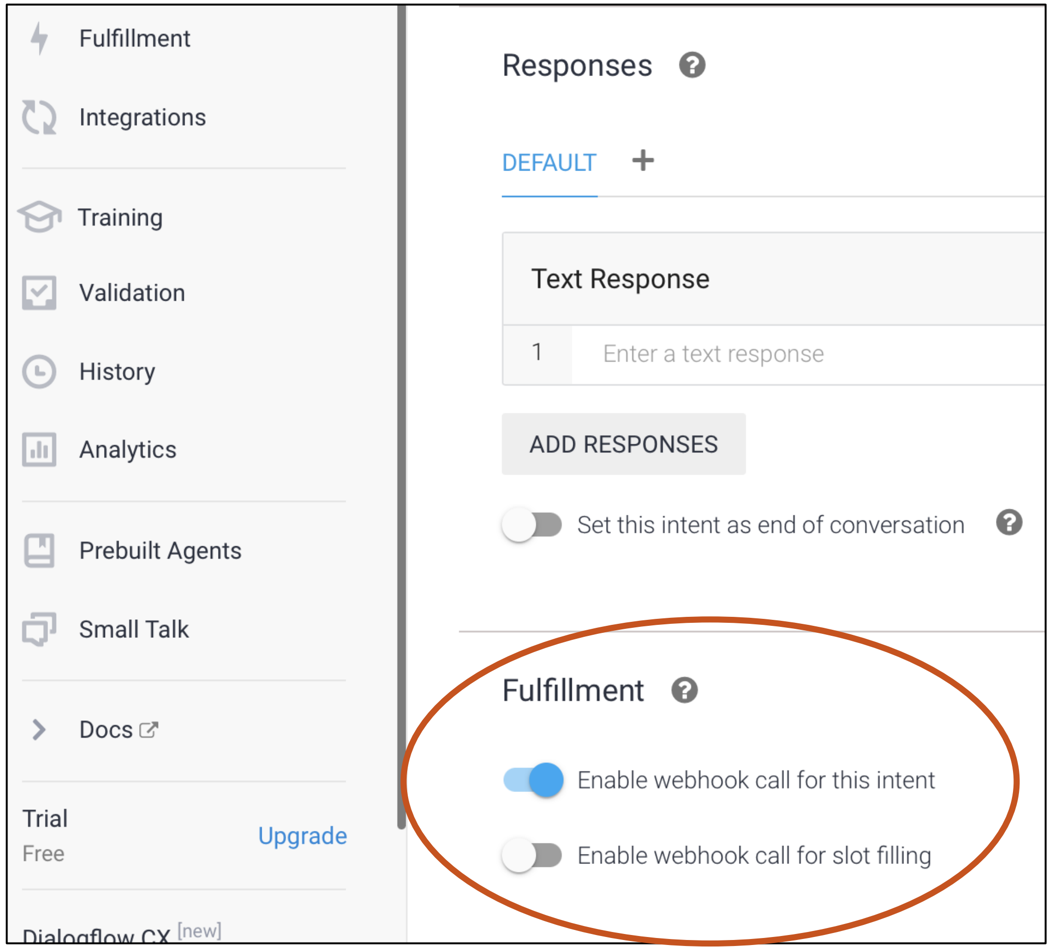
NAME	ENTITY	VALUE
geo-country	@sys.geo-country	$geo-country

PROMPTS

1. What type of cuisine would you like to try?
2. What is your favorite cuisine?
3. Do you have a cuisine preference?
4. Enter a prompt variant

Fulfillment

- In the settings of an Intent:



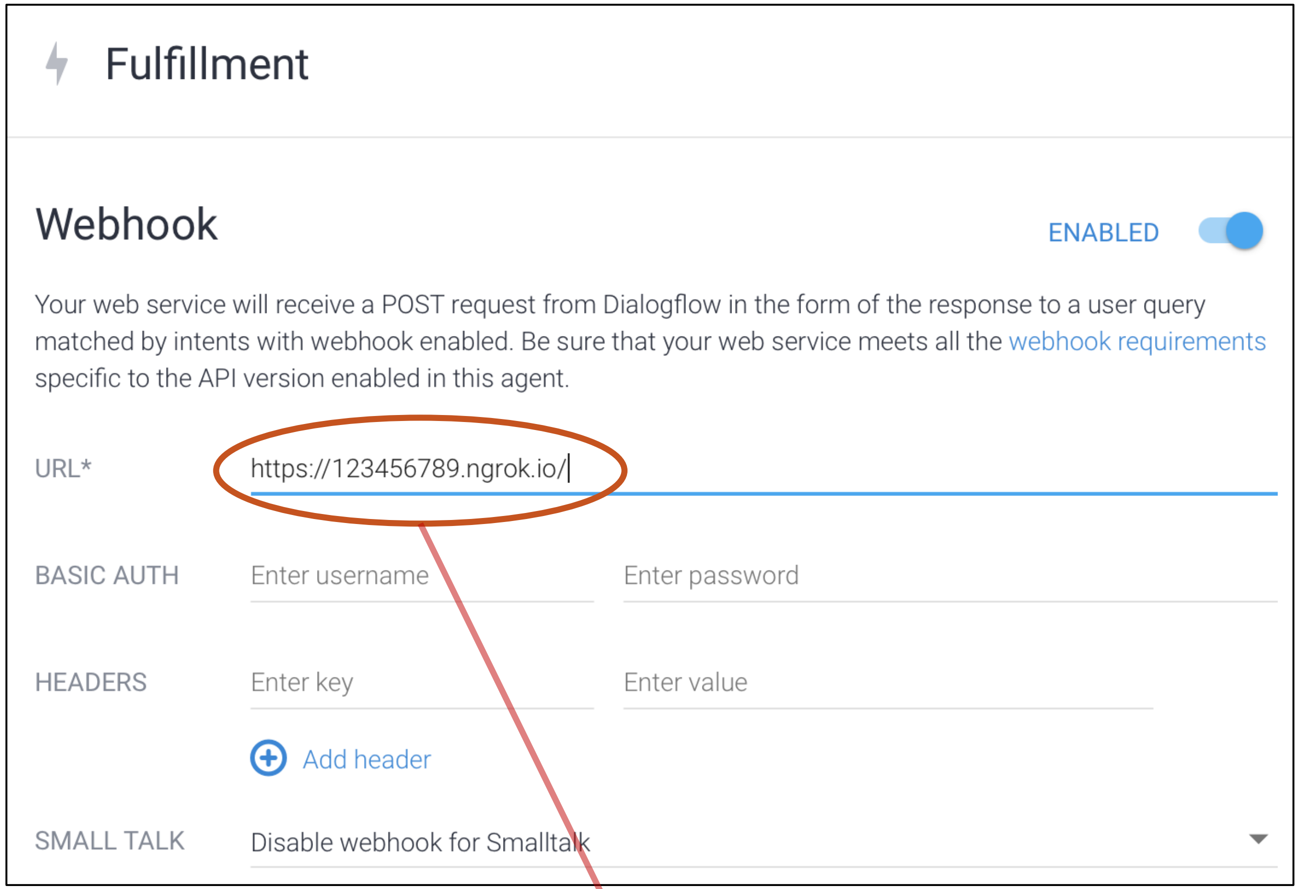
Set this intent as end of conversation

Fulfillment

Enable webhook call for this intent

Enable webhook call for slot filling

- In the Fulfillment section:



Fulfillment

Webhook ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements specific to the API version enabled in this agent.

URL* https://123456789.ngrok.io/

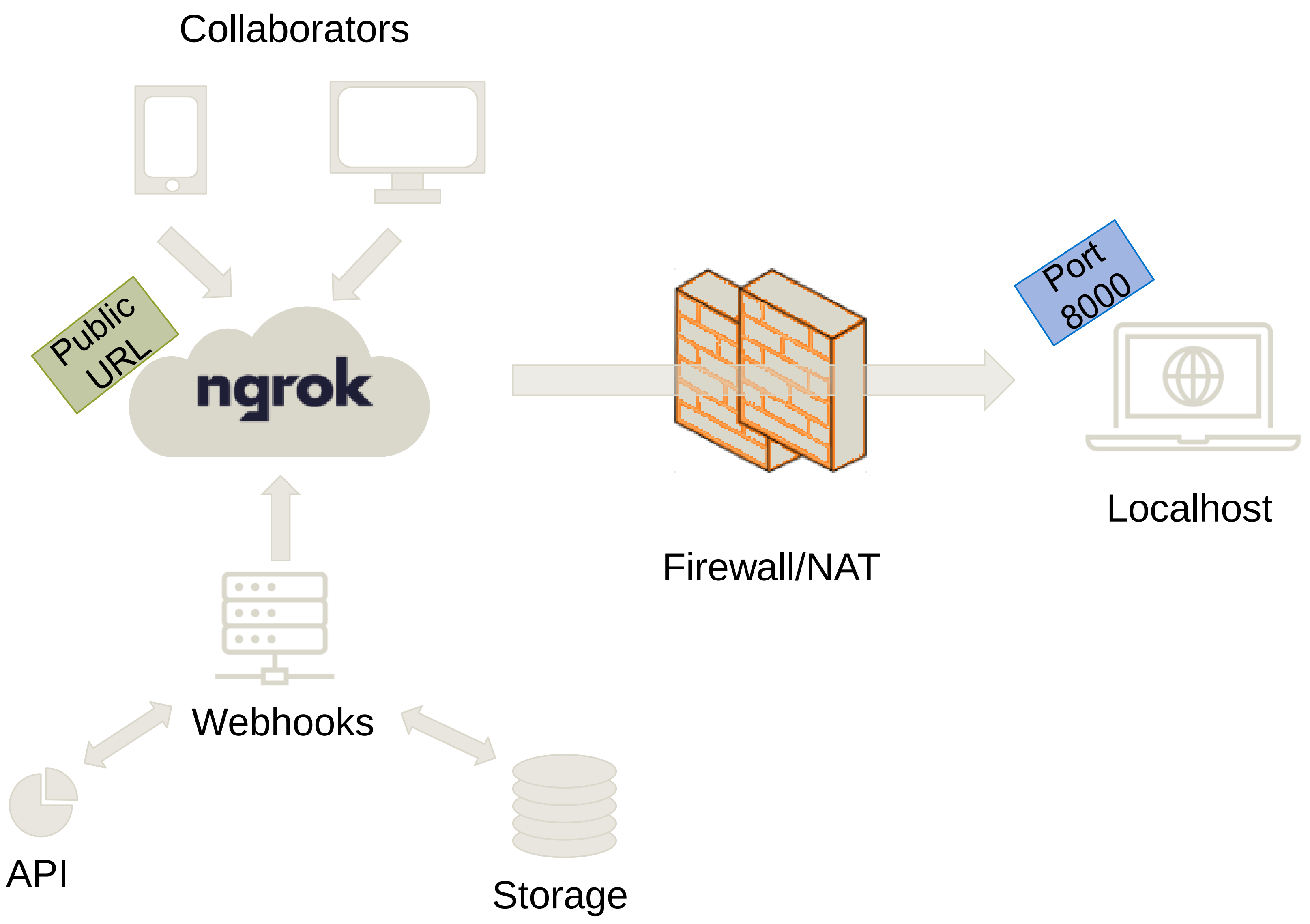
BASIC AUTH Enter username Enter password

HEADERS Enter key Enter value

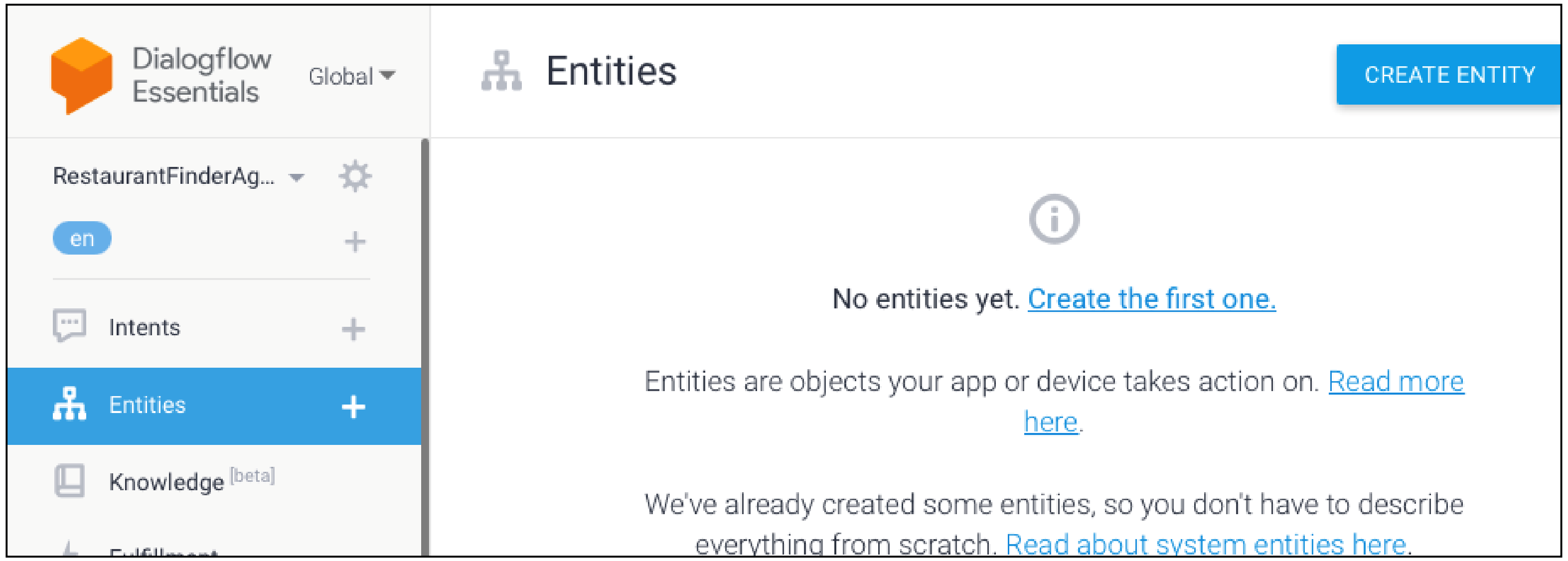
SMALL TALK Disable webhook for Smalltalk

- Copied from ngrok terminal (slide 7)

Fulfillment



Custom Entities

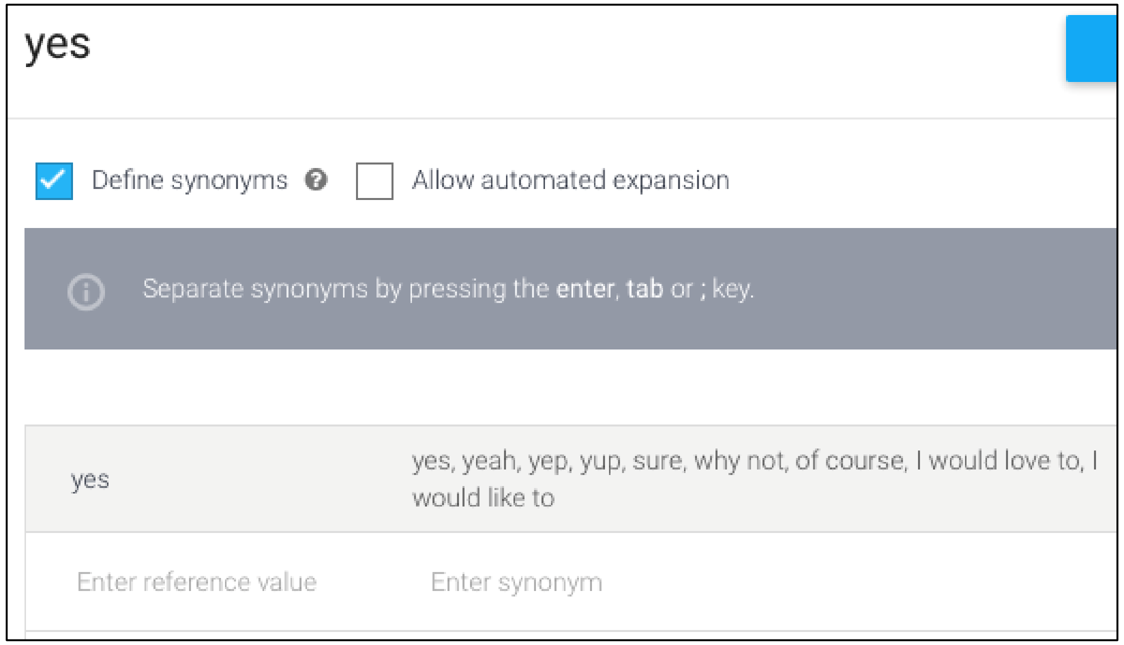


Entities

No entities yet. Create the first one.

Entities are objects your app or device takes action on. Read more here.

We've already created some entities, so you don't have to describe everything from scratch. Read about system entities here.

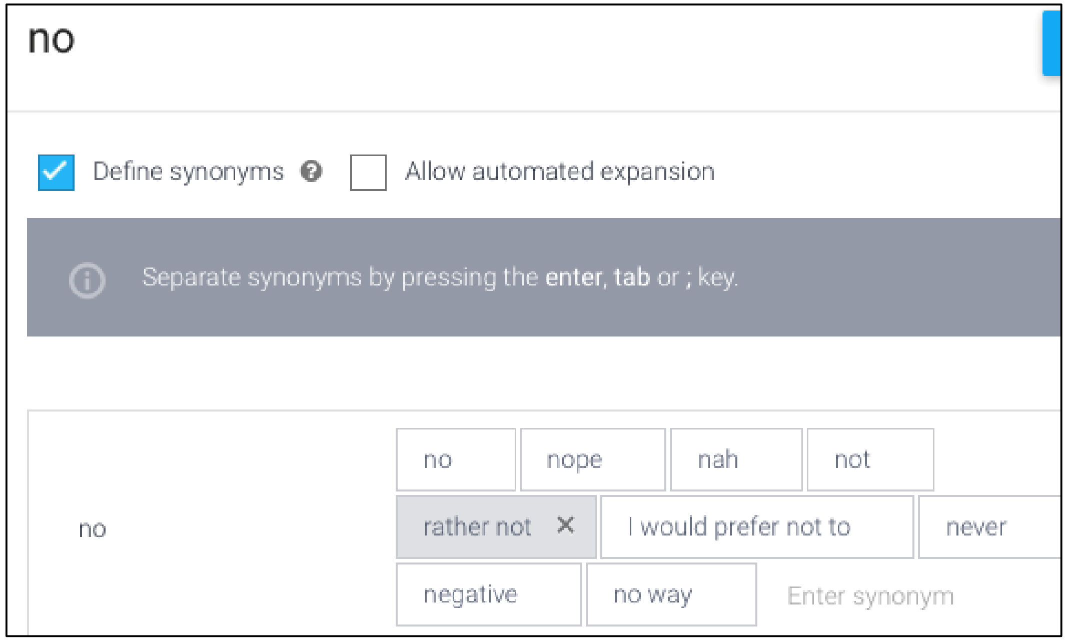


yes

☑ Define synonyms ☐ Allow automated expansion

Separate synonyms by pressing the enter, tab or ; key.

yes	yes, yeah, yep, yup, sure, why not, of course, I would love to, I would like to
Enter reference value	Enter synonym



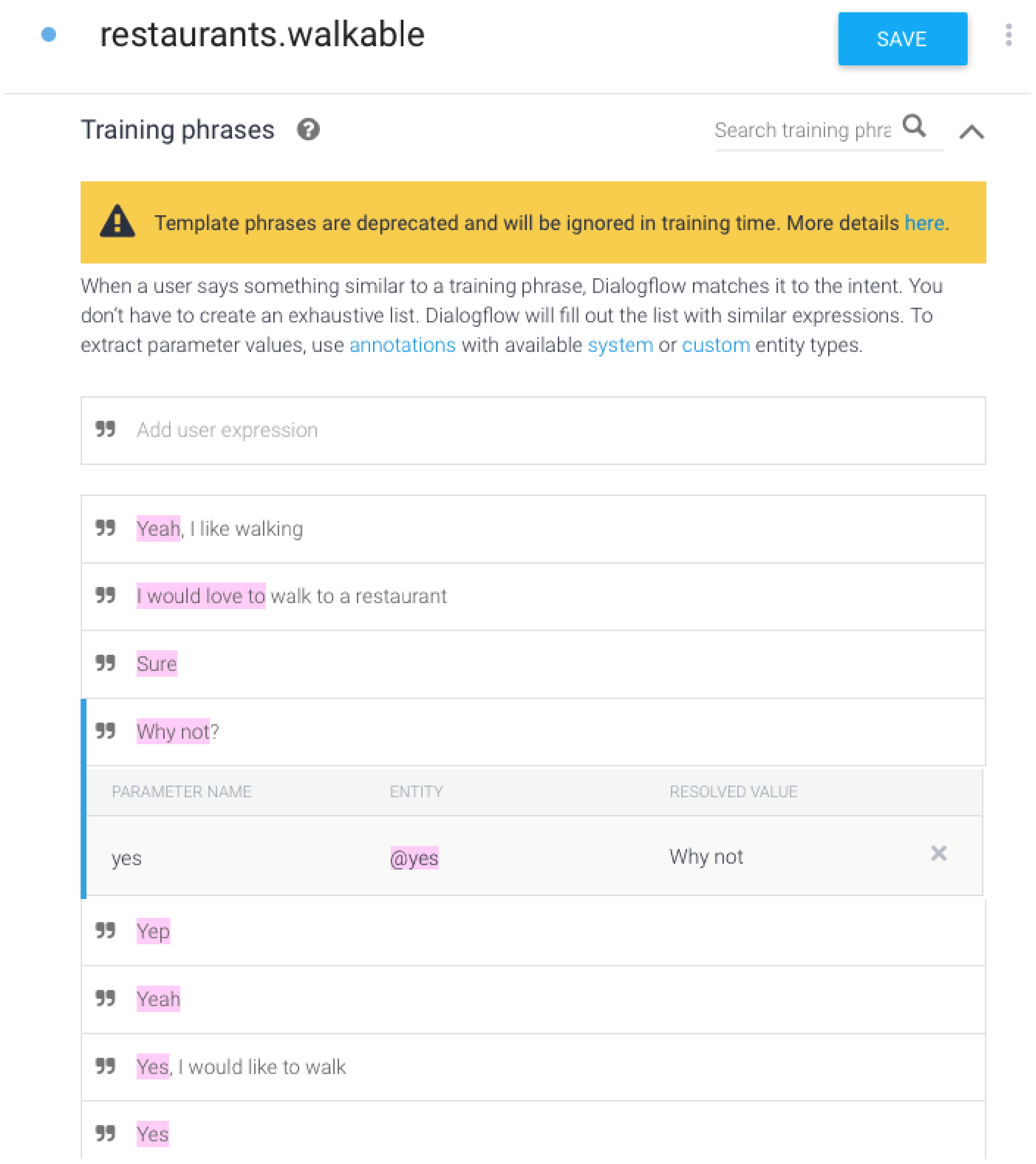
no

☑ Define synonyms ☐ Allow automated expansion

Separate synonyms by pressing the enter, tab or ; key.

no	no, nope, nah, not, rather not, I would prefer not to, never, negative, no way, Enter synonym

Using Custom Entities



restaurants.walkable

SAVE

Training phrases

Search training phra

Template phrases are deprecated and will be ignored in training time. More details here.

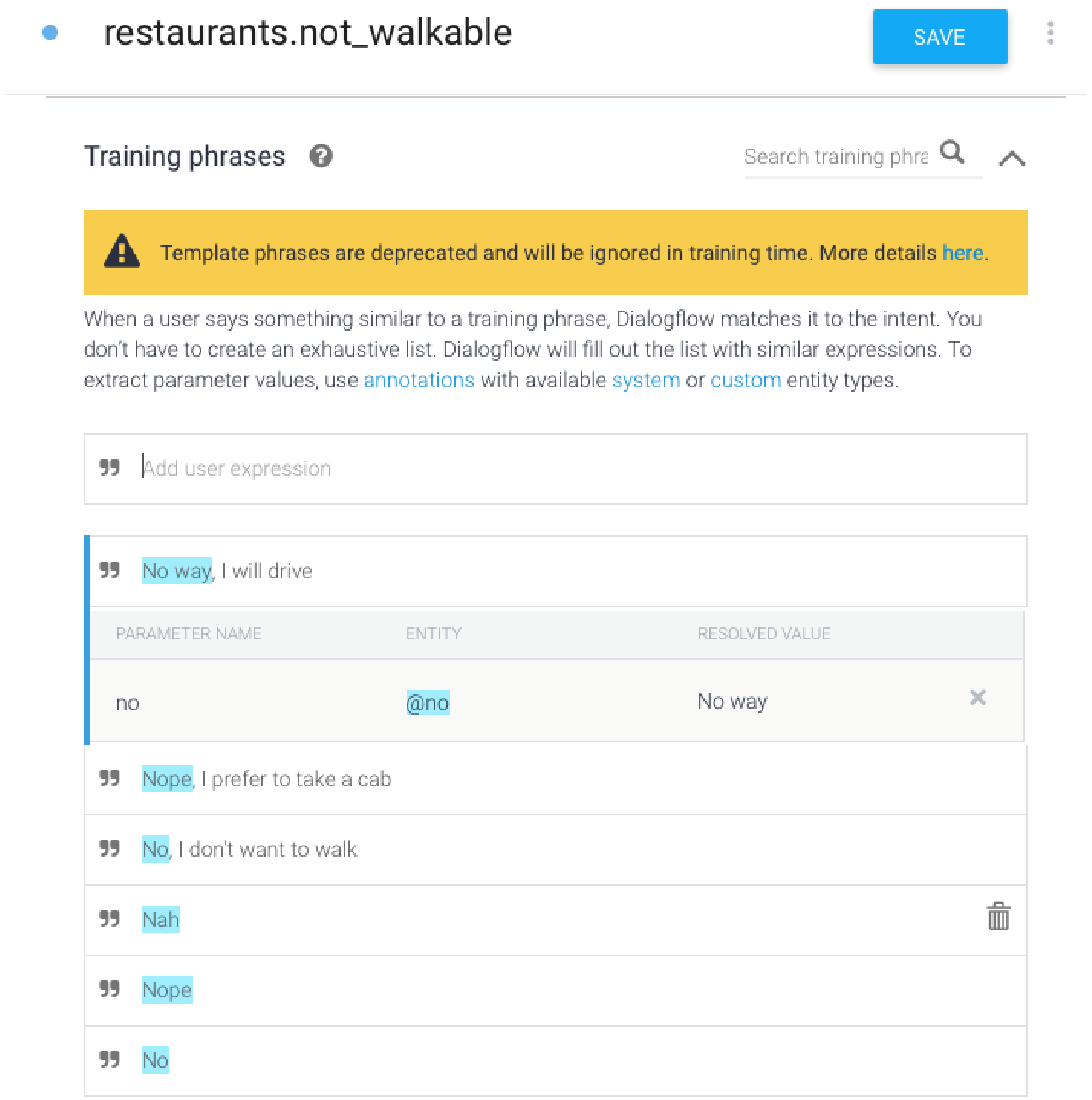
When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

Add user expression

- Yeah, I like walking
- I would love to walk to a restaurant
- Sure
- Why not?

PARAMETER NAME	ENTITY	RESOLVED VALUE
yes	@yes	Why not

- Yep
- Yeah
- Yes, I would like to walk
- Yes



restaurants.not_walkable

SAVE

Training phrases

Search training phra

Template phrases are deprecated and will be ignored in training time. More details here.

When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

Add user expression

- No way, I will drive

PARAMETER NAME	ENTITY	RESOLVED VALUE
no	@no	No way

- Nope, I prefer to take a cab
- No, I don't want to walk
- Nah
- Nope
- No

Outline

Setting up the Environment

- Installing the Requirements
- Running the App
- Ngrok Setup

Dialogflow API

- Creating an Agent
- New Intents
- Actions and Parameters
- Fulfillment
- Custom Entities

Backend with Flask-Dialogflow

- Introduction to the Library
- Webhooks
- Handlers
- Templates

Conversations in Flask-Dialogflow

- *Conversation* objects are the core idea of the library
- One conversation object = one turn of the conversation with the user
- Default type for conversation objects is **V2beta1DialogflowConversation**

- Conversation object initialized from the **WebhookRequest** behind the scenes
- Afterwards the conv. object handed over to a handler function

- The **handler** function:
  1. Does some backend job
  2. Renders the conversation object to a **WebhookResponse**
  3. Serializes the response to JSON and sends back to Dialogflow

- Attributes of the conversation objects: conv.intent (name of the intent), conv.parameters (request’s parameters), conv.session (the session ID)

Handling the Conversations

- Conversation **handlers** implement the core business logic of the agent
- Workflow of a handler:
  1. Accept the conversation object
  2. Inspect its request attributes
  3. Perform necessary business logic
  4. Build the response
  5. Return the conversation object

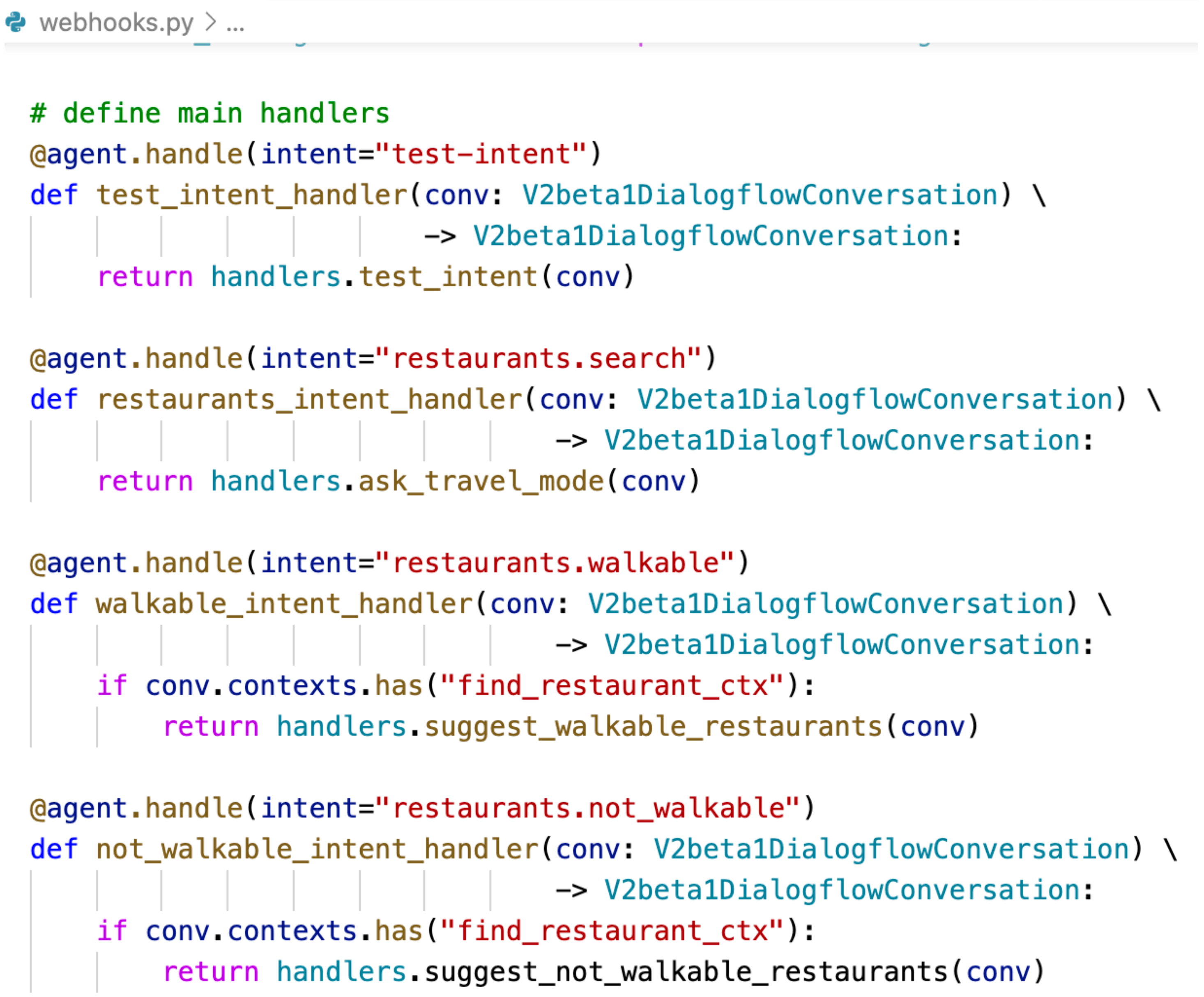
- The only condition is to accept the conversation object as its first argument
- They can be as long and complex as desired
  - Better to break down big handlers into sub-handlers (sub-functions)

Example: Long Handler with Sub-Handlers

```python
@agent.handle('SelectDate)
def choose_date_handler(conv):
    # Entry point for conversations for the SelectDate intent
    date = parse(conv.parameters['selected_date'])
    if date >= datetime.datetime.now():
        conv = valid_date(conv)
    else:
        conv = invalid_date(conv)
    return conv

def valid_date(conv):
    ... # Business logic
    conv.tell('Date was chosen!')
    return conv

def invalid_date(conv):
    ... # Business logic
    conv.tell('Date is invalid:(')
    return conv
```

Setting Up the Webhooks

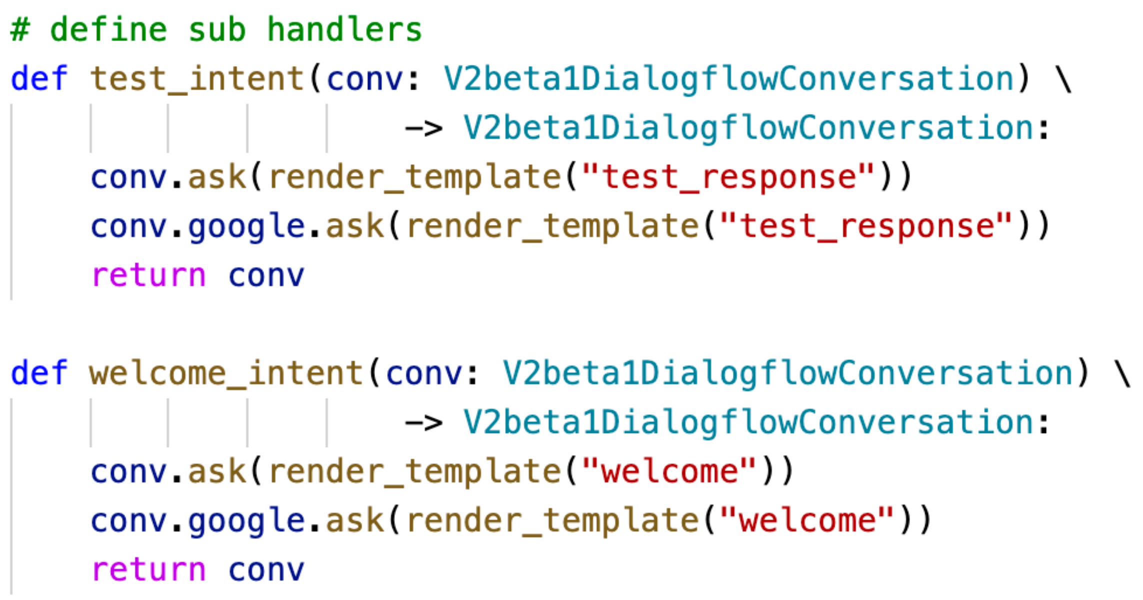
- Create the *webhook.py* file in the root folder
- Import the *agent* and *handlers*
- Return the appropriate handler for every intent



```python
# define main handlers
@agent.handle(intent="test-intent")
def test_intent_handler(conv: V2beta1DialogflowConversation) \
                                -> V2beta1DialogflowConversation:
    return handlers.test_intent(conv)

@agent.handle(intent="restaurants.search")
def restaurants_intent_handler(conv: V2beta1DialogflowConversation) \
                                        -> V2beta1DialogflowConversation:
    return handlers.ask_travel_mode(conv)

@agent.handle(intent="restaurants.walkable")
def walkable_intent_handler(conv: V2beta1DialogflowConversation) \
                                        -> V2beta1DialogflowConversation:
    if conv.contexts.has("find_restaurant_ctx"):
        return handlers.suggest_walkable_restaurants(conv)

@agent.handle(intent="restaurants.not_walkable")
def not_walkable_intent_handler(conv: V2beta1DialogflowConversation) \
                                        -> V2beta1DialogflowConversation:
    if conv.contexts.has("find_restaurant_ctx"):
        return handlers.suggest_not_walkable_restaurants(conv)
```

Rendering Responses from Templates

- Static responses to intents should be put in: *app/templates/**templates.yaml***
- This route then given as the template route while initializing the agent
- Templates should be rendered by *render_template* (imported from the Flask library)
- Reponses generated by rendering responses and sending them to *conv.ask( )* function:

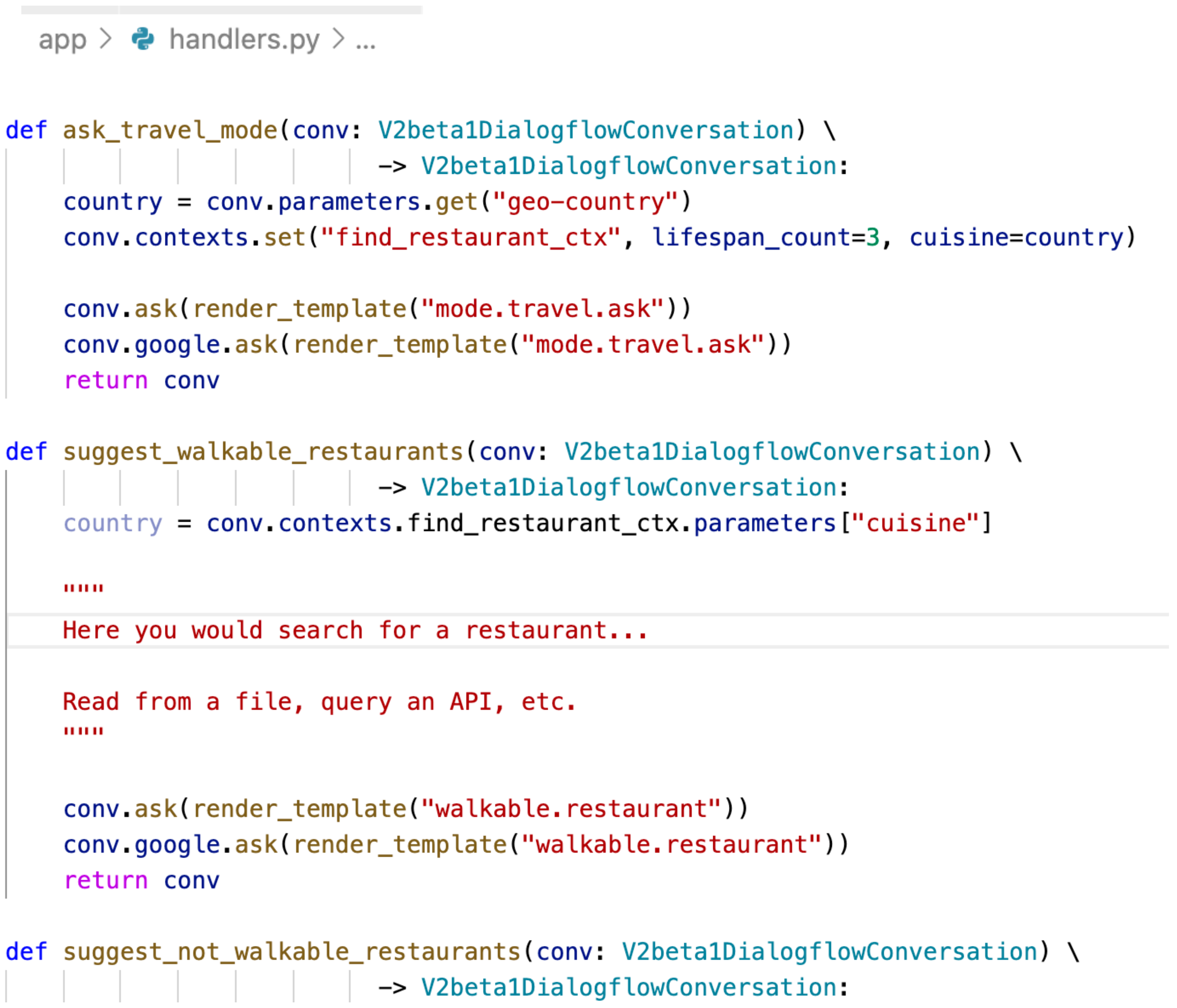


```python
# define sub handlers
def test_intent(conv: V2beta1DialogflowConversation) \
                    -> V2beta1DialogflowConversation:
    conv.ask(render_template("test_response"))
    conv.google.ask(render_template("test_response"))
    return conv

def welcome_intent(conv: V2beta1DialogflowConversation) \
                    -> V2beta1DialogflowConversation:
    conv.ask(render_template("welcome"))
    conv.google.ask(render_template("welcome"))
    return conv
```

Setting up the Handlers

- Names of handler functions should be same as those returned in the *webhooks.py* file

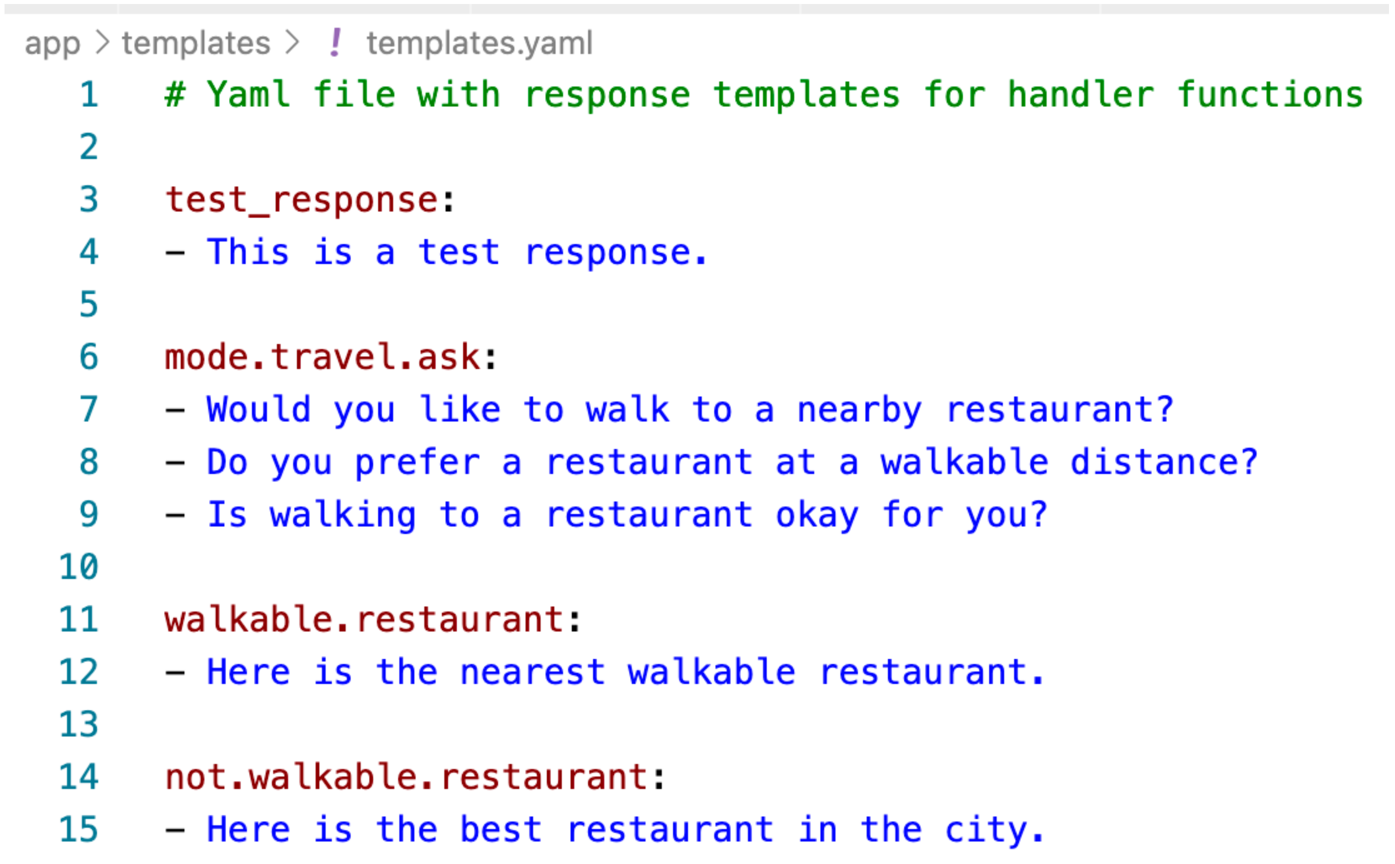
- Use them to do any kind of business logic before crafting a final response to the user


app > handlers.py > ...

```
def ask_travel_mode(conv: V2beta1DialogflowConversation) \
                                    -> V2beta1DialogflowConversation:
    country = conv.parameters.get("geo-country")
    conv.contexts.set("find_restaurant_ctx", lifespan_count=3, cuisine=country)

    conv.ask(render_template("mode.travel.ask"))
    conv.google.ask(render_template("mode.travel.ask"))
    return conv

def suggest_walkable_restaurants(conv: V2beta1DialogflowConversation) \
                                    -> V2beta1DialogflowConversation:
    country = conv.contexts.find_restaurant_ctx.parameters["cuisine"]

    """
    Here you would search for a restaurant...

    Read from a file, query an API, etc.
    """

    conv.ask(render_template("walkable.restaurant"))
    conv.google.ask(render_template("walkable.restaurant"))
    return conv

def suggest_not_walkable_restaurants(conv: V2beta1DialogflowConversation) \
                                    -> V2beta1DialogflowConversation:
```

Static Responses in Templates.yaml

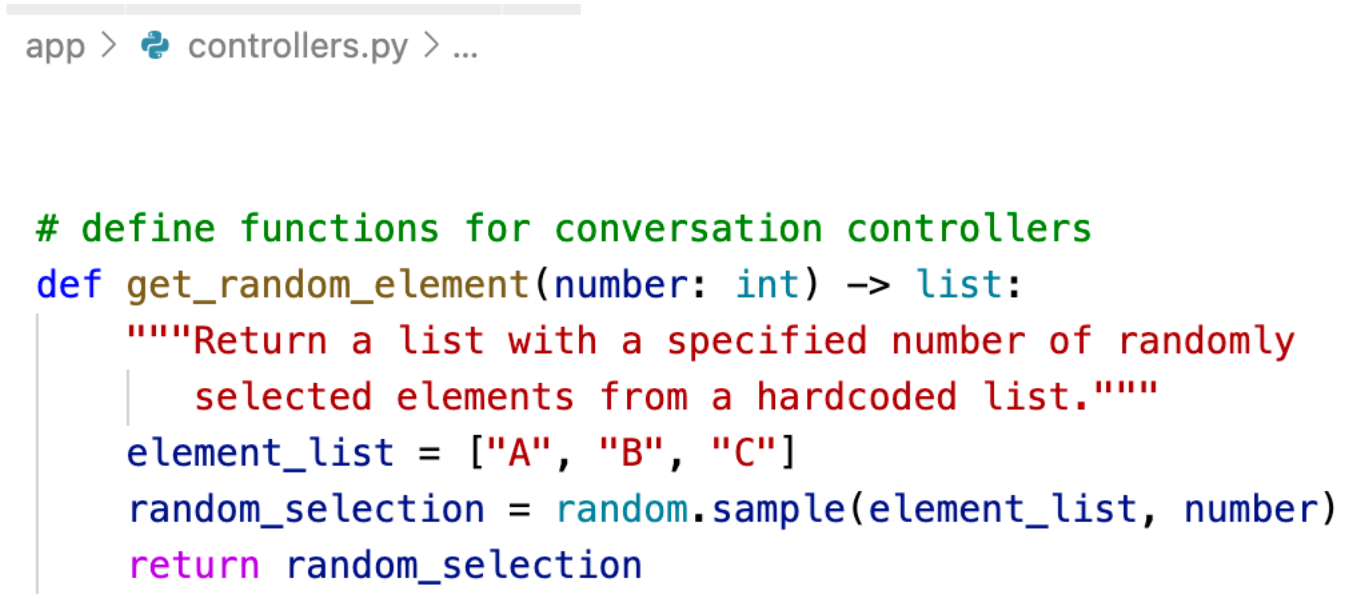
- **YAML** is a human-readable data-serialiazation language (alternative to XML or JSON)
- Intent name at the line beginning, child nodes (replies) writen with a dash before them
- Multiple responses → agent randomly selects which one to send
- *Weighted response → increase or decrease chances of a response being selected (default weight:* ***1****)*



```yaml
# Yaml file with response templates for handler functions

test_response:
- This is a test response.

mode.travel.ask:
- Would you like to walk to a nearby restaurant?
- Do you prefer a restaurant at a walkable distance?
- Is walking to a restaurant okay for you?

walkable.restaurant:
- Here is the nearest walkable restaurant.

not.walkable.restaurant:
- Here is the best restaurant in the city.
```

Setting up the Controllers

- Define additional utility functions in the *controllers.py* file in the root folder
- Example: generate random elements, read a text file, query an external API, etc.



```python
# define functions for conversation controllers
def get_random_element(number: int) -> list:
    """Return a list with a specified number of randomly
       selected elements from a hardcoded list."""
    element_list = ["A", "B", "C"]
    random_selection = random.sample(element_list, number)
    return random_selection
```

Flask-Dialogflow Documentation

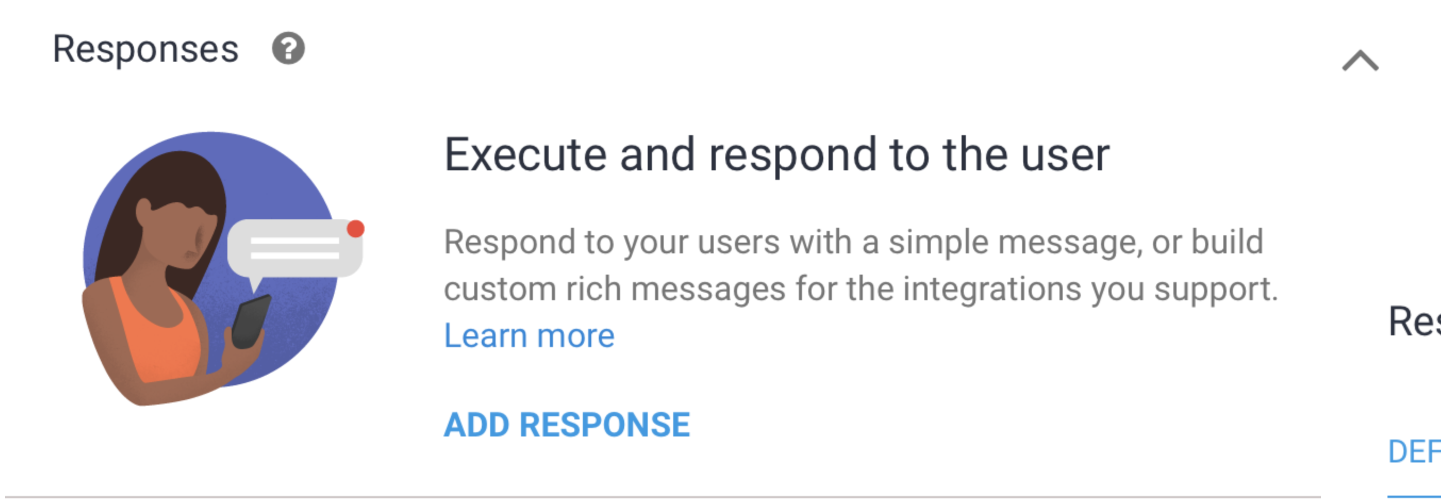
- For further explanations and additional options present in the Flask-Dialogflow library, check their official documentation: **https://flask-dialogflow.readthedocs.io/**

- Information on:
  - Installation and Setup
  - Google APIs
  - Conversations and handlers
  - Templating
  - Contexts
  - Integrations
  - Actions on Google
  - Testing

List of sources and recommended literature

- https://cloud.google.com/dialogflow/es/docs/concepts
- https://cloud.google.com/dialogflow/es/docs/quick/build-agent
- **https://flask-dialogflow.readthedocs.io/**
- https://www.manning.com/books/conversational-ai
- https://arxiv.org/abs/1801.03604
- https://www.morganclaypool.com/doi/abs/10.2200/S01060ED1V01Y202010HLT048
- https://towardsdatascience.com/how-to-create-a-conversational-agent-with-dialogflow-17bfa90aa02d
- https://medium.com/google-cloud/deconstructing-chatbots-getting-started-with-dialogflow-4f91deb32135

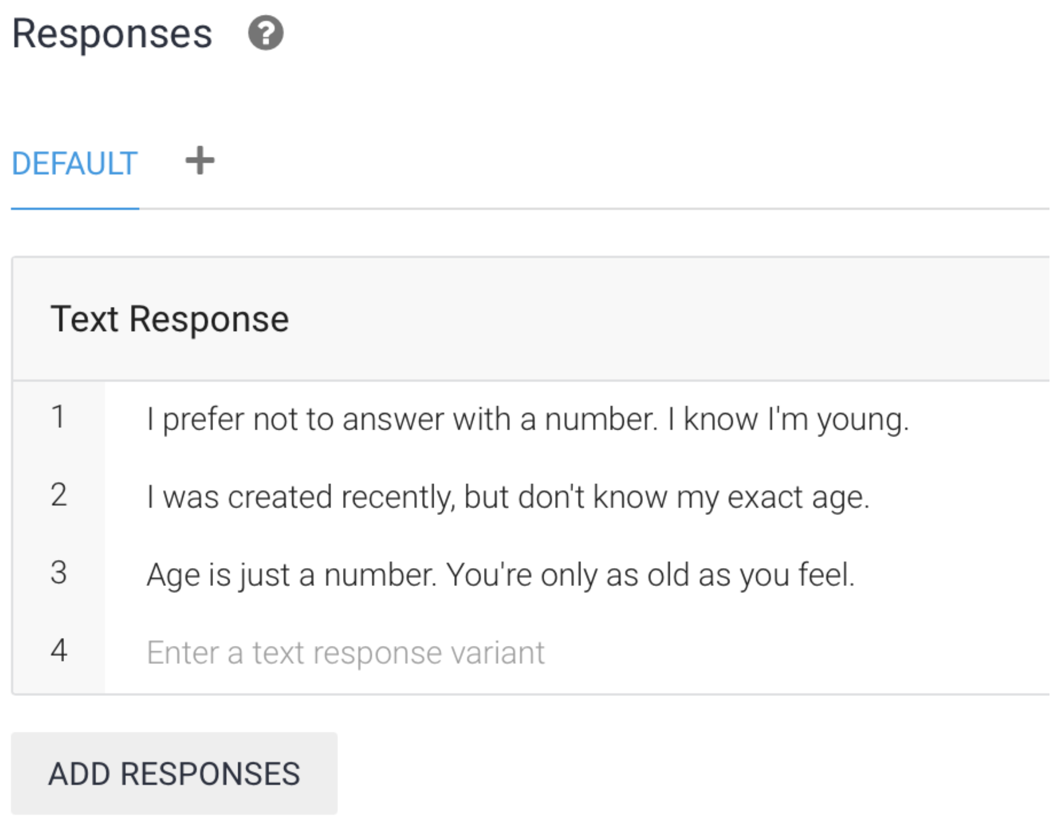
Intent: Responses


Responses

Execute and respond to the user

Respond to your users with a simple message, or build custom rich messages for the integrations you support. Learn more

ADD RESPONSE


Responses

DEFAULT +

Text Response

1 I prefer not to answer with a number. I know I'm young.

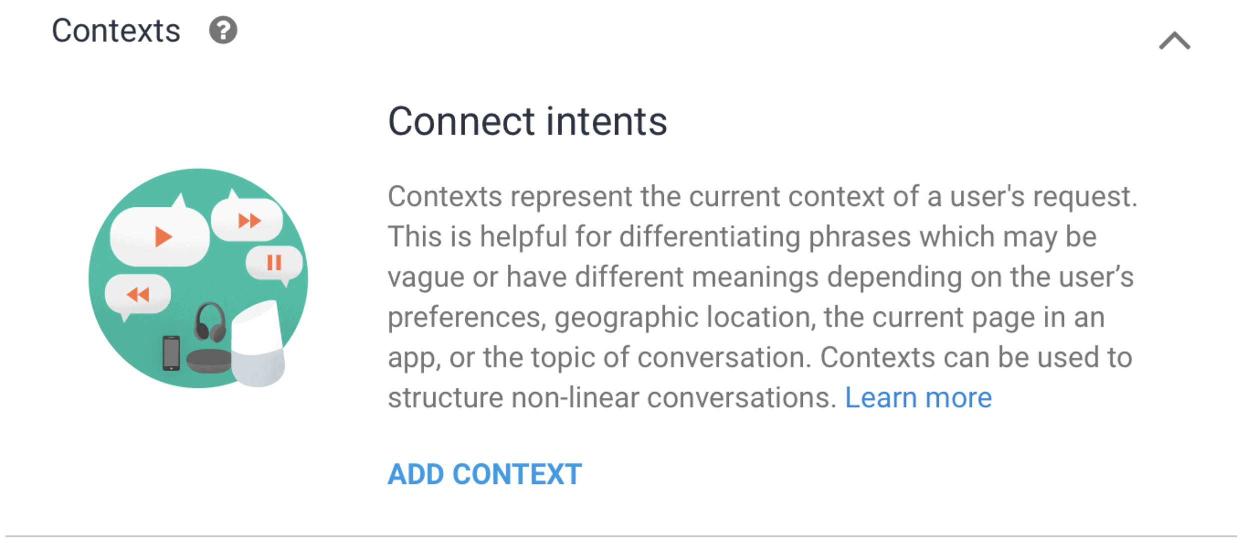
2 I was created recently, but don't know my exact age.

3 Age is just a number. You're only as old as you feel.

4 Enter a text response variant

ADD RESPONSES

Intent: Context

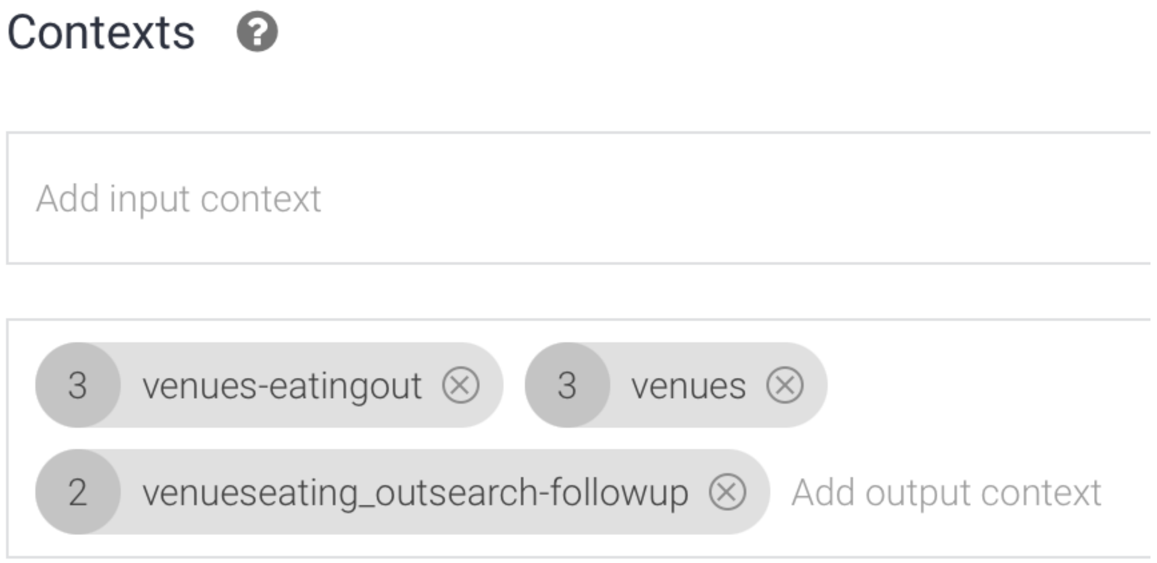


Contexts

Connect intents

Contexts represent the current context of a user's request. This is helpful for differentiating phrases which may be vague or have different meanings depending on the user's preferences, geographic location, the current page in an app, or the topic of conversation. Contexts can be used to structure non-linear conversations. Learn more

ADD CONTEXT



Contexts

Add input context

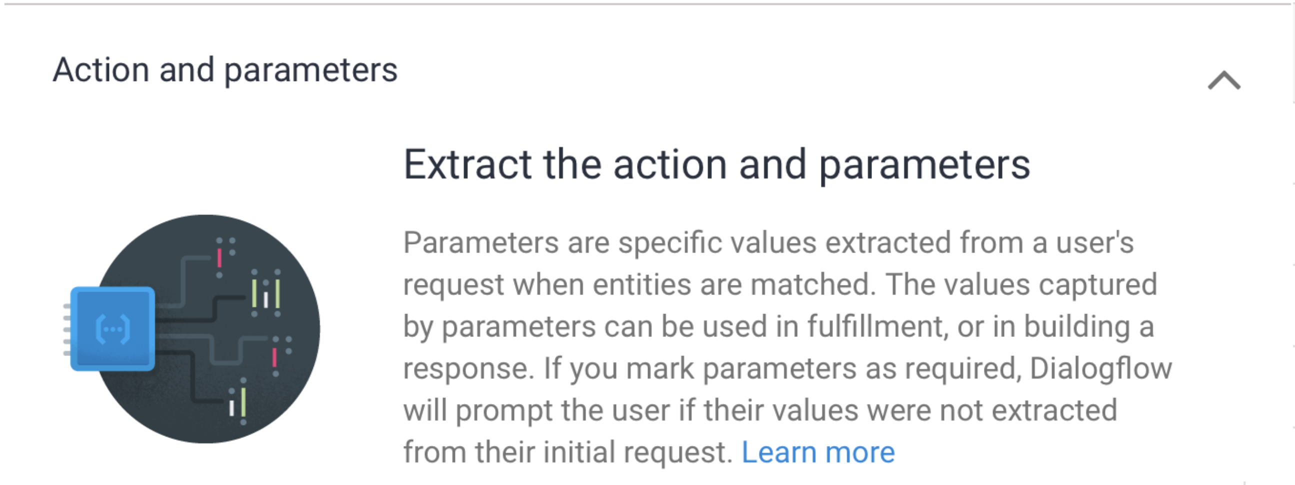
3 venues-eatingout

3 venues

2 venueseating_outsearch-followup

Add output context

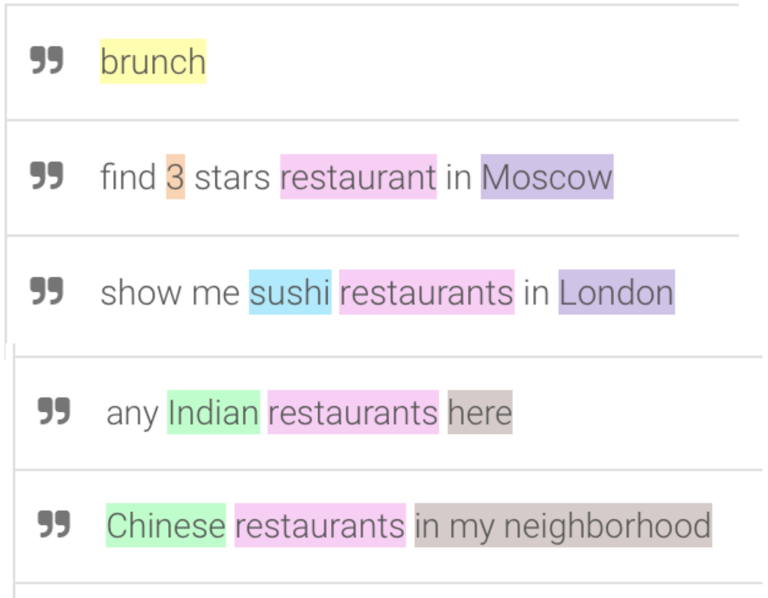
Intent: Parameters



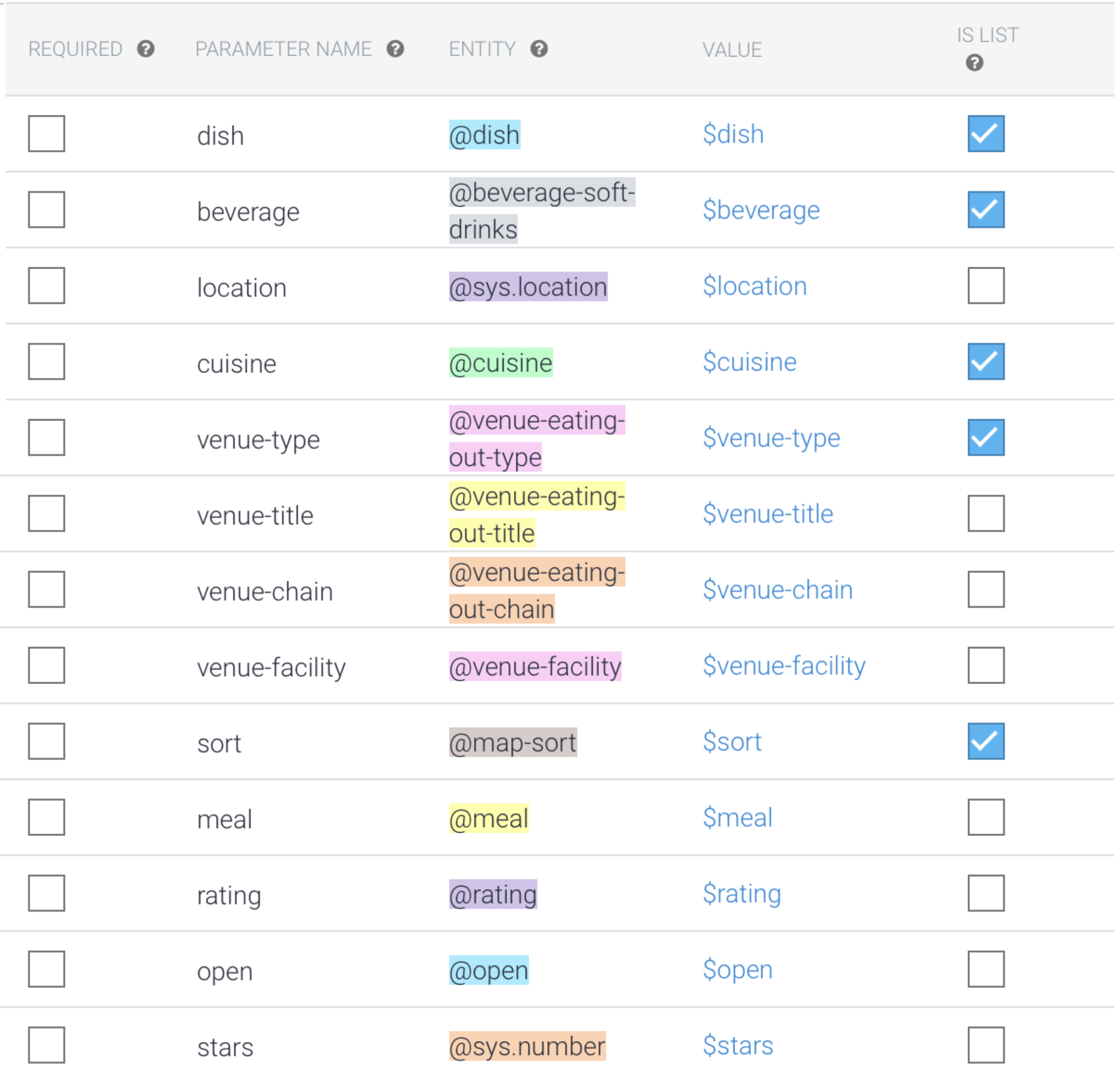
Action and parameters

Extract the action and parameters

Parameters are specific values extracted from a user's request when entities are matched. The values captured by parameters can be used in fulfillment, or in building a response. If you mark parameters as required, Dialogflow will prompt the user if their values were not extracted from their initial request. Learn more



- brunch
- find 3 stars restaurant in Moscow
- show me sushi restaurants in London
- any Indian restaurants here
- Chinese restaurants in my neighborhood



REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	dish	@dish	$dish	☑
☐	beverage	@beverage-soft-drinks	$beverage	☑
☐	location	@sys.location	$location	☐
☐	cuisine	@cuisine	$cuisine	☑
☐	venue-type	@venue-eating-out-type	$venue-type	☑
☐	venue-title	@venue-eating-out-title	$venue-title	☐
☐	venue-chain	@venue-eating-out-chain	$venue-chain	☐
☐	venue-facility	@venue-facility	$venue-facility	☐
☐	sort	@map-sort	$sort	☑
☐	meal	@meal	$meal	☐
☐	rating	@rating	$rating	☐
☐	open	@open	$open	☐
☐	stars	@sys.number	$stars	☐