Start Lab 02:00:00

Exploring the Dialogflow API

2 hours Free

Overview

This lab shows you how to build a simple Dialogflow agent, walking you through the most important features of Dialogflow.

You'll learn how to: Create a Dialogflow account and your first Dialogflow agent, which lets you define a natural language understanding model.

Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Task 1. Login to Dialogflow

This section describes how to create and log in to a Dialogflow account

Create your Dialogflow account

Now that you're signed into your student account in an incognito (private browser) window, you can sign into Dialogflow by following these steps:

1. Click dialogflow.cloud.google.com to open DialogFlow in a new browser.
2. Click **Sign in with Google** and sign in with your lab account
3. Dialogflow will need access to your Google account.

To explore permissions by role, refer to Access control, Roles section.

Next steps

Next, you'll create your first Dialogflow agent and test it out.

Task 2. Create and query your first agent

This section describes how to create and try out your first Dialogflow agent.

Create your first Dialogflow agent

To create a Dialogflow agent:

1. Click **Create agent** in the left menu.
2. Enter your agent's name:

```
NewAgent
```

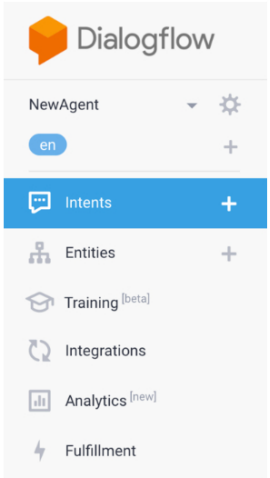
3. Accept the default language, and default time zone.

4. Click on the **GOOGLE PROJECT** drop-down arrow button and select the project ID **qwiklabs-gcp-xxxxxxxxx**.

5. Click the **Create** button.

The Dialogflow console

You should now see the Dialogflow console and the menu panel on the left.

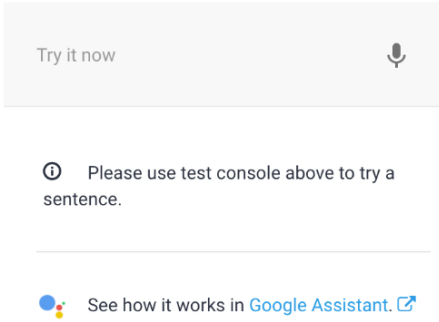


> **Note:** If you're working on a smaller screen and the menu is hidden, click on the menu button in the upper left corner. The settings button takes you to the current agent's settings.

The middle of the page will show the list of intents for the agent.

> **Note:** By default, Dialogflow agents start with two intents:
>
> - Your agent matches the **Default Fallback Intent** when it doesn't understand what your users say.
> - The **Default Welcome Intent** greets your users. These can be altered to customize the experience.

Notice the Dialogflow simulator on the right side of the screen.



The simulator lets you try out your agent by speaking or typing messages.

Query your agent

Agents are best described as **NLU (Natural Language Understanding)** modules. These can be included in your app, product, or service and transform natural user requests into actionable data.

Time to try out your agent!

1. Locate the **Dialogflow simulator** on the right.

2. Click into the text field that says **Try it now**.

3. Type **hi**, and press enter.

You just spoke to your Dialogflow agent! You may notice your agent understood you.

Since your input matched to the **Default Welcome Intent**, you received one of the default replies inside the welcome intent.

In the case that your agent doesn't understand you, the **Default Fallback Intent** is matched and you receive one of the default replies inside that intent.

The Default Fallback Intent reply prompts the user to reframe their query in terms that can be matched. You can change the responses within the Default Fallback Intent to provide example queries and guide the user to make requests that can match an intent.

Task 3. Create your first intent

Task 3. Create your first intent

Dialogflow uses intents to categorize a user's intentions. Intents have **Training Phrases**, which are examples of what a user might say to your agent. For example, someone wanting to know the name of your agent might ask, "What is your name?", "Do you have a name?", or just say "name". All of these queries are unique but have the same intention: to get the name of your agent.

To cover this query, create a "name" intent:

1. Click on the **+** icon next to **Intents** in the left menu.
2. In the **Intent name** text field enter the following:

```
Name
```

3. In the **Training Phrases** section, click **Add Training Phrases**.
4. Enter the first query, pressing enter after the entry:

```
What is your name?
```

5. Enter a second query, pressing enter after the entry:

```
Do you have a name?
```

6. Enter a third query, pressing enter after the entry:

```
name
```

7. In the **Responses** section, click **Add Response** enter the following response:

```
My name is Dialogflow!
```

8. Click the **Save** button.

Now try asking your agent for its name.

In the simulator on the right, type "What's your name?" and press enter.

> **Note:** Dialogflow uses training phrases as examples for an AI Platform model to match users' queries to the correct intent.
>
> The AI Platform model checks the query against every intent in the agent, gives every intent a score, and the highest-scoring intent is matched.
>
> If the highest scoring intent has a very low score, the fallback intent is matched.

Your agent now responds to the query correctly. Notice that even though your query was a little different from the training phrase (**"What's your name?"** versus **"What is your name?"**). Dialogflow still matched the query to the right intent.

Task 4. Extract data with entities

This section describes how to extract data from a user's query.

Add parameters to your intents

Parameters are important and relevant words or phrases in a user's query that are extracted so your agent can provide a proper response. You'll create a new intent with parameters for spoken and programming languages to explore how these can match specific intents and be included in your responses.

1. Create a new intent by clicking on the **+** icon next to **Intents** in the left menu.
2. Name the intent "Languages" at the top of the intent page.
3. Add the following as Training phrase:

```
I know English
```

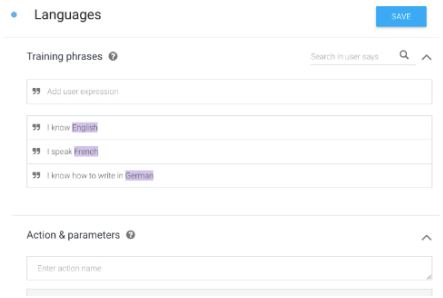
4. Add the following as Training phrase:

```
I speak French
```

5. Add the following as Training phrase:

```
I know how to write in German
```

Once the training phrases are entered it should resemble the image below:



REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	language	@sys.language	$language	☐

Dialogflow automatically detects known parameters in your Training phrases and creates them for you.

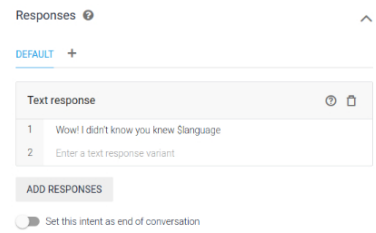
Below the **Training phrases** section, Dialogflow fills out the parameter table with the information it gathered:

- The parameter is optional (not required)
- named language
- corresponds to the system entity type @sys.language
- has the value of $language
- is not a list

> **Note:** If entities aren't automatically detected, you can highlight the text in the Training phrase and manually annotate the entity.

Use parameter data

The value of a parameter can be used in your responses.



In this case, you can use $language in your responses and it will be replaced with the language specified in the query to your agent.

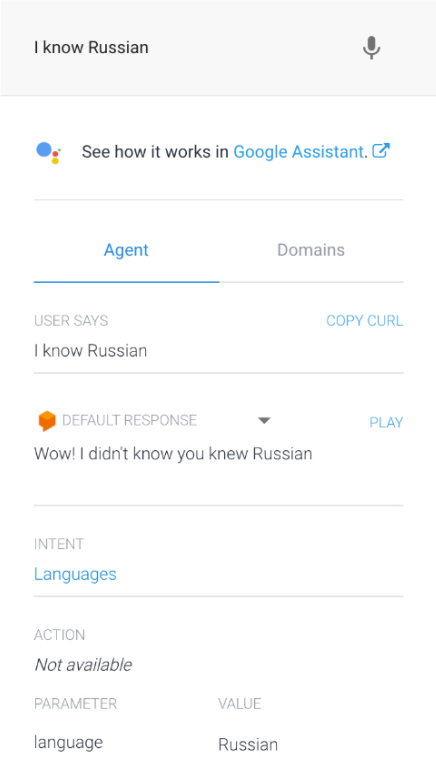
1. In the **Responses** section, add the following response

```
Wow! I didn't know you knew $language
```

2. Click the **Save** button.

Try it out!

- Now, query your agent with "I know Russian" in the simulator in the right panel.



You can see in the bottom of the simulator output that Dialogflow correctly extracted the language parameter with the value "Russian" from the query. In the response, you can see "Russian" was correctly inserted where the parameter value was used.

Task 5. Create your own entities

You can also create your own entities, which function similarly to Dialogflow's system entities. To create an entity:

1. Click on the **+** icon next to **Entities** in the left menu.

2. Enter "ProgrammingLanguage" for the name of the entity.

3. Click on the text field and add the following entry:

```
JavaScript
```

4. Add the following JavaScript synonym:

```
JavaScript, js
```

5. Click on the text field and add the following entries:

```
Java
```

6. Add the following Java synonym:

```
Java
```

7. Click on the text field and add the following entries:

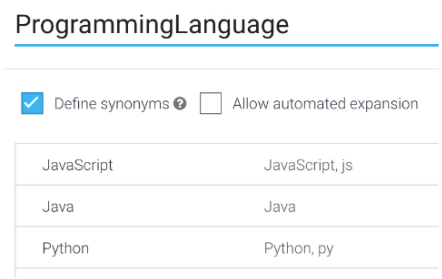
```
Python
```

8. Add the following Python synonym:

```
Python, py
```

9. Click the **Save** button.

Your entities should look similar to below:



Each entity type has to have the following:

- a name to define the category (ProgrammingLanguage)
- one or more entries (JavaScript)
- one or more synonyms (js, JavaScript)

Dialogflow can handle simple things like plurality and capitalization, but make sure to add all possible synonyms for your entries. The more you add, the better your agent can determine your entities.

Add your new entities

Now that we've defined our entity for programming languages, add Training Phrases to the "Languages" intent:

1. Click on the **+** icon next to **Intents** in the left menu.

2. In the **Intent name** text field enter the following:

```
ProgrammingLanguage
```

3. Add the following Training phrase:

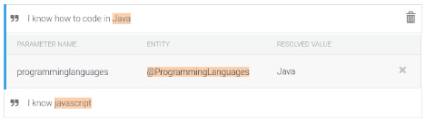
```
I know javascript
```

4. Add the following Training phrase:

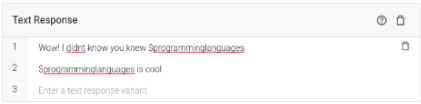
```
I know how to code in Java
```

5. You should see the programming languages automatically annotated in the Training phrases you entered.

This adds the ProgrammingLanguage parameter to the table, which is below the **Training phrases** section.



6. In the **Responses** section, add "$ProgrammingLanguage is cool"



7. Click the **Save** button.

Try it out!

- Now, query your agent with "I know how to code in py" in the simulator in the right panel.

Task 6. Manage state with contexts

This section describes how to track conversational states with follow-up intents and contexts.

Add contexts to conversational state

1. Click on **Intents** in the left menu, and then click on the "Languages" intent.

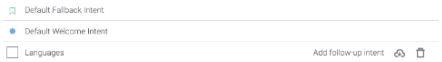
2. Extend one of the original Text response in the **Response** section to the following:

```
Wow! I didn't know you knew $language. How long have you known
$language?
```

3. Click the **Save** button.

4. Click on **Intents** in the left menu.

5. Hover over the "Languages" intent to show the menu.



6. Click on **Add follow-up intent > Custom**.

Dialogflow automatically names the follow-up intent "Languages - custom" and lists it in Languages.

Intent matching with follow-up intents

Follow-up intents are only matched after the parent intent has been matched. Since this intent is only matched after the "Languages" intent, we can assume that the user has just been asked the question "How long have you known $language?". You'll now add Training Phrases indicating users' likely answers to that question.

1. Click on **Intents** in the left menu and then click on the "Languages - custom" intent.

2. Add the following Training Phrase:

```
3 years
```

4. Add the following Training Phrase:

```
about 4 days
```

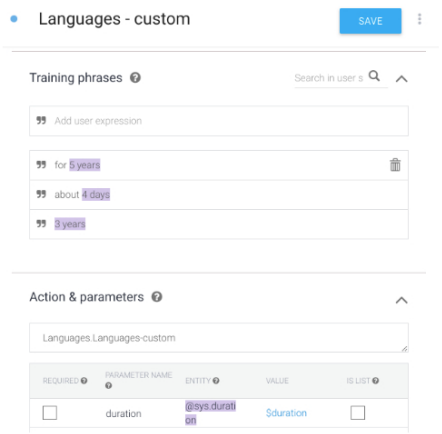
5. Add the following Training Phrase:

```
for 5 years
```

> **Note:** Ensure each Training phrase entity is set as **@sys.duration**. You can change the entity type by clicking on the entity.

6. Click the **Save** button.

You screen should now look similar to the image below:



Try it out

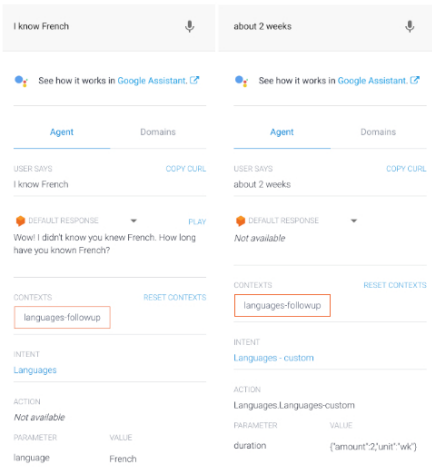
Try this out in the **Dialogflow simulator** on the right.

1. First, match the "Languages" intent by entering the query:

```
I know French.
```

2. Then, answer the Dialogflow follow up question "How long have you known $language"?:

```
about 2 weeks.
```



Despite there being no response for the second query ("about 2 weeks"), we can see our query is matched to the correct intent ("Languages - custom") and the duration parameter is correctly parsed ("2 weeks").

Task 7. Intents and contexts

Now that your follow-up intent is being matched correctly, you need to add a response. In "Languages - custom" you've only asked for the duration the user has known the language, and not the referenced language itself.

To respond with a parameter gathered from the "Languages" intent, you need to know how follow-up intents work.

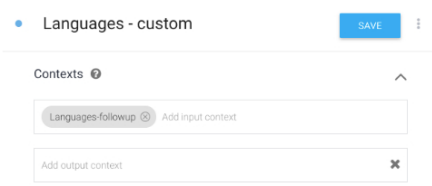
Follow-up intents use contexts to keep track of if a parent intent has been triggered. If you inspect the "Languages" intent, you'll see "Languages-followup" listed as an **Output context**, prefaced by the number 2:

After the "Languages" intent is matched, the context "Languages-followup" is attached to the conversation for two turns. Therefore, when the user responds to the question, "How long have you known $language?", the context "Languages-followup" is active.

Any intents that have the same **Input context** are heavily favored when Dialogflow matches intents.

1. Click on **Intents** in the left navigation
2. Click on the **"Languages - custom"** intent.

You can see that the intent has the same input context ("Languages-followup") as the output context of "Languages".



Because of this, "Languages - custom" is much more likely to be matched after the "Languages" intent is matched.

Contexts and parameters

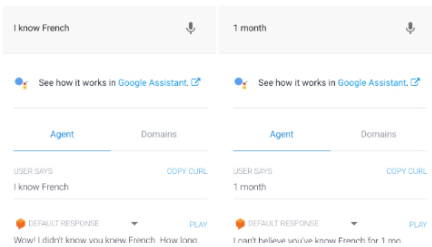
Contexts store parameter values, which means you can access the values of parameters defined in the "Languages" intent in other intents like "Languages - custom".

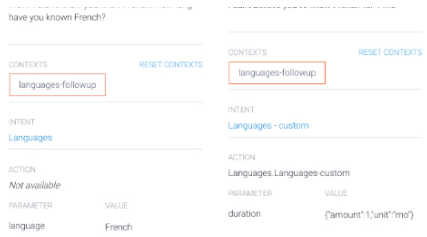
1. Add the following response to the "Languages - custom" intent and click the **Save** button:

```
I can't believe you've known #languages-followup.language for
$duration!
```

2. **Save** the changes.

Now you can query your agent again and get the proper response. First enter "I know French", and then respond to the question with "1 month".





You should see that the language parameter value is retrieved from the context.

Next steps

If you have any questions or thoughts, let us know on the Dialogflow Google Plus Community. We'd love to hear from you!

Now that you've completed your first agent, you can extend your response logic with fulfillment and consider which additional platforms you want to support via Dialogflow's one-click integrations.

Fulfillment allows you to provide programmatic logic behind your agent for gathering third-party data or accessing user-based information.

- Fulfillment
- How to get started with fulfillment
- Integrate your service with fulfillment
- Integrate your service with Actions on Google

Dialogflow's integrations make your agent available on popular platforms like Facebook Messenger, Slack and Twitter.

- Integrations Overview
- Facebook Messenger
- Slack
- Twitter

You might also want to check out:

- Contexts
- Dialogflow and Actions on Google

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Manual Last Updated: September 27, 2022

Lab Last Tested: May 23, 2022