

# Building a Conversational Agent

Grace Hopper Conference 2017

<b><u>Quick Links</u></b>	<b>2</b>
<b><u>Group Work</u></b>	<b>2</b>
<a href="#">Step 1: Create a new agent!</a>	2
<a href="#">Step 2: Change the Welcome and Fallback Intents.</a>	3
<a href="#">Step 3: Create your own Intent.</a>	6
<a href="#">Step 4: Add in your webhook.</a>	8
<a href="#">Step 5: Add some entities.</a>	12
<b><u>Self-Paced Exercises</u></b>	<b>14</b>
<a href="#">Exercise 0: Debugging.</a>	14
<a href="#">Exercise 1: Play with mandatory parameters.</a>	15
<a href="#">Exercise 2: Add a topic field.</a>	17
<a href="#">Exercise 3: Ask for a bio.</a>	19
<a href="#">Exercise 4: Use context.</a>	19
<a href="#">Exercise 5: Use Events.</a>	22
<a href="#">Exercise 6: Play time!</a>	24
<b><u>Appendix</u></b>	<b>24</b>
<a href="#">Glossary</a>	24
<a href="#">Integrating your agent with other platforms</a>	24
<a href="#">Building your own backend</a>	27
<b><u>Answers</u></b>	<b>27</b>
<a href="#">Answer to Exercise 3: Ask for a bio.</a>	27

# Quick Links

For your reference:

- [Github repo](#) Includes the backend for the webhook we've prepared
- [Quote database](#) The quote database from which our webhook pulls quotes and bios
- [List of topics](#) The full list of topics that appear in our quote database
- [List of authors](#) The full list of authors that appear in our quote database
- [API.AI docs](#) Basic documentation for getting started with API.AI

## Group Work

Welcome! This workshop will focus on building a conversational agent on the **API.AI** platform. Our agent will be able to tell us quotes and fun facts from awesome women in STEM!

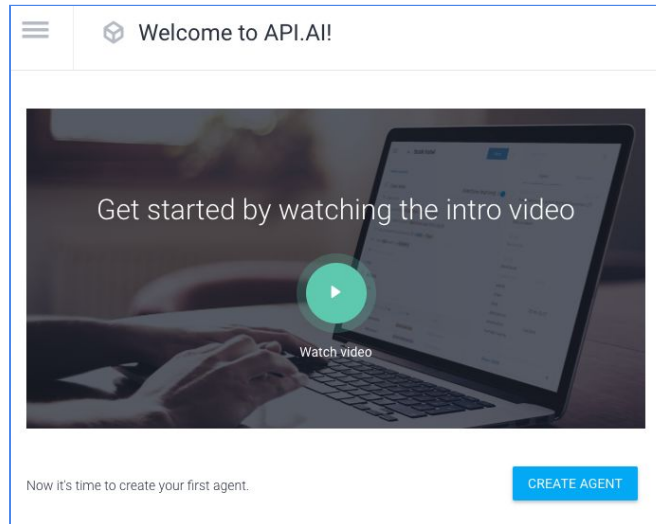
***We will go through Step 1-5 together.*** The rest of the workshop will be self-paced. Don't worry about getting through every step. You can always come back later!

**Note:** The naming of certain variables and parameters matters as we go through the course. We'll try to call out explicitly when a name is important. In general, the course will be easier to understand if you follow our naming conventions.

### Step 1: Create a new agent!

Your first task is to create a new **Agent** on **API.AI**. An **Agent** is a Natural Language Understanding module that helps users get a set of tasks done with conversation.

When you log in to the site, you'll see a blue button that says **CREATE AGENT**. Click the button to get started.



Let's start by naming our agent and adding a description. We've named our version QuoteMaster. You can name yours the same or pick something else fun!

Add a simple description:

Fetches quotes by women in STEM.

You can always come back and add more detail later.

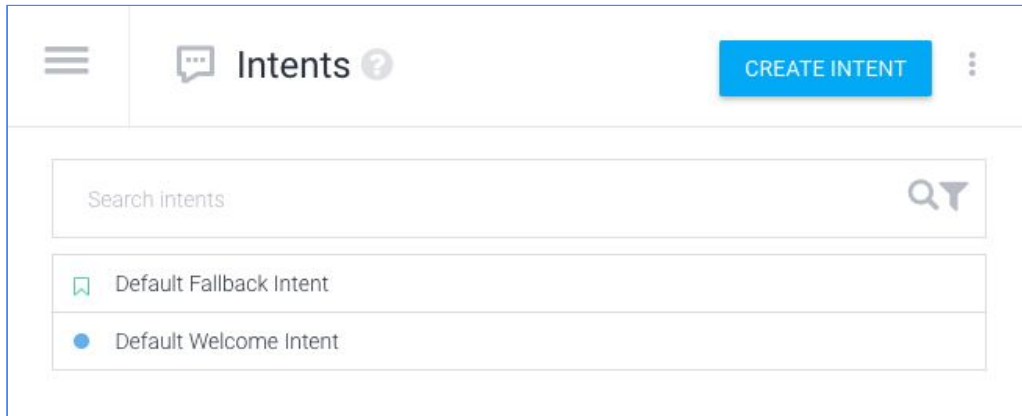
A screenshot of the API.AI agent configuration page. The header shows a hamburger menu icon, the agent name 'QuoteMaster', and a blue 'SAVE' button. Below the header, there is a 'DESCRIPTION' label and a text input field containing the text 'Fetches quotes by women in STEM'.

Hit SAVE. (Note: the **SAVE** button will become **Working...** – give it a moment.)

## Step 2: Change the Welcome and Fallback Intents.

To start, let's make sure our users know how to interact with our agent properly. Once you've created your agent, you should see a page called **Intents**.

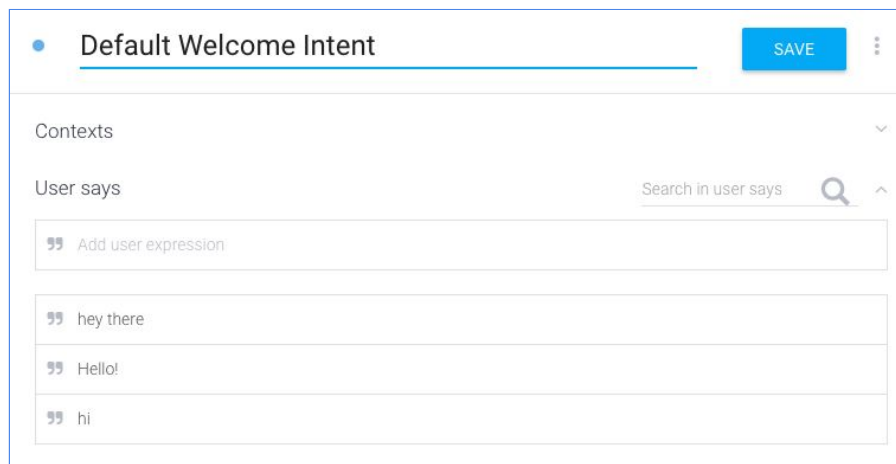
An **Intent** represents a mapping between what a user says and what action should be taken by your software.



Click on **Default Welcome Intent**. This is how we greet our users when the first start using our agent.

First, give the agent some examples of how people might start interacting with it.

Hi  
Hello!  
Hey there.



Go ahead and delete the existing **Text response** messages. Add one that will help your users understand what your agent can do. For example:

Welcome! Talk to me to hear cool quotes from awesome women in STEM. For example, you can say, "Tell me a quote by Grace Hopper."

Text response

1

Welcome! Talk to me to hear cool quotes from awesome women in STEM. For example, you can say, "Tell me a quote by Grace Hopper."

2

Enter a text response variant

If you accidentally deleted the entire **Text response** box, click on **Add message content > Text response** to create a new one:

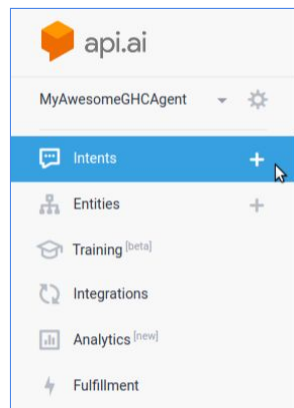
DEFAULT

+

ADD MESSAGE CONTENT

Hit **SAVE** at the top.

Now, navigate back to the Intents page by clicking **Intents** on the left hand side of the page.



This time, click on **Default Fallback Intent**. Here is where we manage what our agent will say to users when it doesn't know how to help them.

Again, go ahead and delete the existing **Text response** messages. Enter your own fallback message:

Sorry, I don't know how to help with that. Try saying, "tell me a quote"!

Text response

1

Sorry, I don't know how to help with that. Try saying, "tell me a quote"!

2

Enter a text response variant

Hit **SAVE**.

Now we can try out our agent. Try typing some things into the **Try it now** box on the righthand side of the page.

Hi  
Who is Grace Hopper?

Try it now

Agent

Domains

USER SAYS

COPY CURL

Hi!

DEFAULT RESPONSE

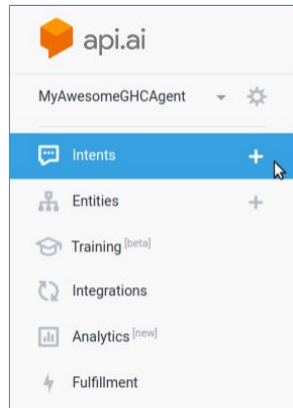
PLAY

Welcome! Talk to me to hear cool quotes from awesome women in STEM. For example, you can say, "Tell me a quote by Grace Hopper."

### Step 3: Create your own Intent.

Right now our agent can only welcome us and tell us that it can't help us. Let's add some real features!

On the left hand side of your page, you will see a tab called **Intents** with a **+** sign next to it. Click the **+** sign to create a new intent.



Let's name this intent *quote*.

A screenshot of the 'Intents' list in the Google Assistant API console. It shows a single entry with a blue dot icon and the text 'quote'.

First, we need to give our agent some examples of how people might ask for a quote. Try adding some user expressions. Don't worry about capitalization for these examples; your agent will learn to handle both uppercase and lowercase variants.

Tell me a quote  
I'd like to hear a quote  
Give me a quotation  
Quote please!




Can you think of more variations? You can add others as well.

A screenshot of the 'User says' section in the Google Assistant API console. It features a search bar at the top right labeled 'Search in user says' with a magnifying glass icon. Below the search bar is a list of user expressions, each preceded by a quote icon. The expressions are: 'Add user expression' (placeholder), 'Quote please!', 'Give me a quotation', 'I'd like to hear a quote', and 'Tell me a quote'.

Now we need to tell our agent how to respond! First, we'll just enter a placeholder response. Eventually, we will get the quote by looking it up in our quotes backend.

Add a simple **Text response** for now (we will flesh this out with real functionality later):

Grace Hopper once said, "Hi! I'm Grace!"

Text response		 
1	Grace Hopper once said, "Hi! I'm Grace!"	
2	Enter a text response variant	

Hit **SAVE**. Now you can try out your agent!

Try typing some things into the **Try it now!** box:

Tell me a quote  
I'd like to hear a quote

What happens if you ask for a slight variation of your original quote examples?

Please tell me a cool quote  
I want to hear a quote


What happens if you ask for something irrelevant?

Fly me to the moon!

Your agent should now know how to respond with the simple Grace Hopper quote to your original examples plus other similar queries, but it won't give that response if you ask for something else.

Be sure to **SAVE** your agent before proceeding to step 3.


## Step 4: Add in your webhook.

Our agent would be a lot cooler if it could tell us *real* quotes. To do this, we need to hook up our agent to our quotes backend. Click on the **Fulfillment** tab on the lefthand side. (If you don't see this, press the hamburger symbol  in the top left to reveal the menu.)

**Enable** the Webhook and copy in the URL for our backend:


<https://grace-hopper-quotes.appspot.com/quotesearch>



 Fulfillment

---

Webhook

ENABLED 

Your web service will receive a POST request from API.AI 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](#).

[Webhook example](#)

URL\*

<https://grace-hopper-quotes.appspot.com/quotesearch/>

BASIC AUTH

Enter username

Enter password


HEADERS

Enter key

Enter value

Enter key

Enter value

 Add header

DOMAINS

Enable webhook for all domains

SAVE

Under **Domains**, change the setting to **Enable webhook for all domains**.

Hit **SAVE**.

## Optional

You can play around with our webhook in the terminal to get a sense for how it works by using curl to send some json requests. The code and documentation are available at <https://github.com/juliar/quotes-agent>.

Our webhook takes a json request with a "result" object that has several fields:

```

action: mandatory.
parameters: object with the optional fields:
    author
    subject
    
```

Try a few requests with the [get\\_quote\\_response](#) action:

```

curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
-d '{"result": {"action": "get_quote_response"}}' \
-H "Content-Type: application/json"

curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
-d '{"result": {"action": "get_quote_response",
               "parameters": {"author": "Marie Curie" }}}' \
-H "Content-Type: application/json"

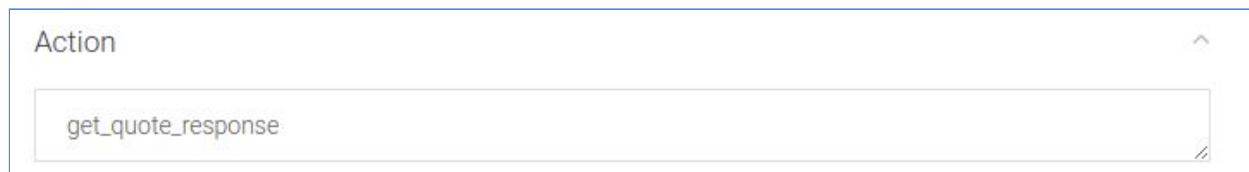
curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
-d '{"result": {"action": "get_quote_response",
               "parameters": {"subject": "technology" }}}' \
    
```

```
-H "Content-Type: application/json"

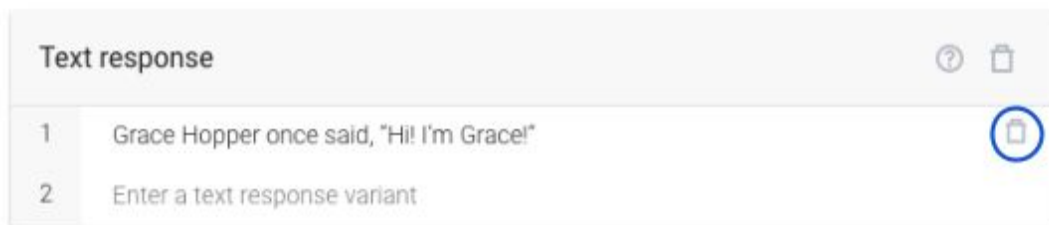
curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
-d '{"result": {"action": "get_quote_response",
                "parameters": {"author": "Marie Curie",
                              "subject": "technology" }}}' \
-H "Content-Type: application/json"
```

Now we need to update our *quotes* intent to send our queries to the webhook.

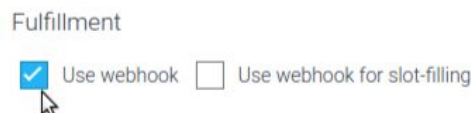
Navigate back to the *quotes* intent. Fill in the **Action** field with the text *get\_quote\_response*. An **Action** is the step your application takes when an intent is triggered. ***Make sure to copy this name exactly.*** This will tell our backend what kind of step to take.



Then, delete your text response by clicking the trashcan symbol next to the text:



Click **Fulfillment** and check **Use webhook**.



Now hit **SAVE**.

Try typing "tell me a quote" into the **Try it now** box.

**Optional**

To see the format of the request we send to our webhook and the response we get back, click the **SHOW JSON** button:

```
{
  "id": "0bde6dc5-5c5b-4dc8-b1a9-5fc75421e87a",
  "timestamp": "2017-09-06T16:46:15.053Z",
  "lang": "en",
  "result": {
    "source": "agent",
    "resolvedQuery": "tell me a quote",
    "action": "get_quote_response", # this matches the action we added to our quote intent
    "actionIncomplete": false,
    "parameters": {},
    "contexts": [],
    "metadata": {
      "intentId": "72ad9641-5621-414f-b6e2-2bdeedcf31eec",
      "webhookUsed": "true", # we are using our webhook!
      "webhookForSlotFillingUsed": "false",
      "webhookResponseTime": 74,
      "intentName": "quote" # our quote intent
    },
    "fulfillment": { # our backend sends back the text our agent will speak and display
      "speech": "Here is a quote by mary somerville: Mathematics are the natural bent of my mind.",
      "displayText": "Here is a quote by mary somerville: Mathematics are the natural bent of my mind.",
      "messages": [
        {
          "type": 0,
          "speech": "Here is a quote by mary somerville: Mathematics are the natural bent of my mind."
        }
      ]
    },
    "score": 1
  },
  "status": {
    "code": 200,
    "errorType": "success" # good!
  },
  "sessionId": "fb4093ce-ab34-4a8f-9c98-c372159c3fbd"
}
```

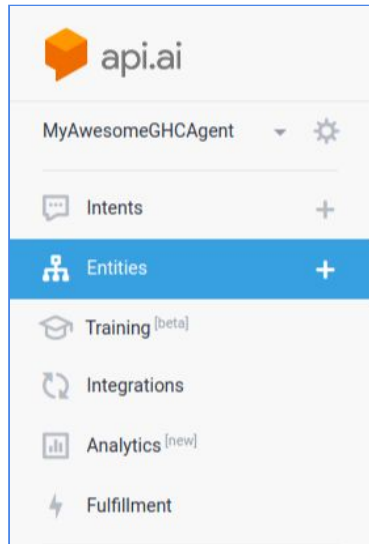
The first fields in the **result** object are actually the ones we sent to our backend. You can see that we set **action = get\_quote\_response** and sent empty **parameters** object. Later we will add some parameters to this request.

The response from our backend is the **fulfillment** object, where both **speech** and **displayText** fields have been filled out. This tells our agent what to say back to our user and what to display as a response. Later we will see how to get our backend to return structured data in place of **speech** and **displayText**.

## Step 5: Add some entities.

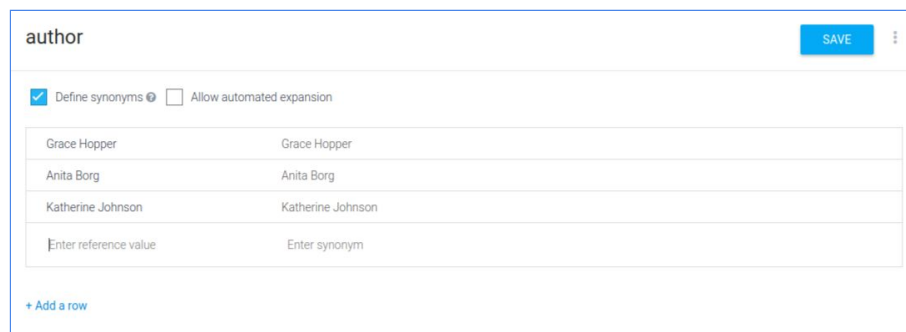
Our database has quotes from a lot of cool women in STEM. Let's update our agent so that users can ask for quotes from a *particular author*.

For this we will create an **Entity**. An **Entity** represents a category of things in the world. They are powerful tools used for extracting parameter values from natural language inputs. Our category will be women in STEM who authored interesting quotes.



Click the **+** sign by the **Entity** tab on the left. Start by adding a new **author** Entity, and add some sample values of authors. You can see our full list of authors [here](#) – make sure anyone you add is part of this list. For now, just enter a few. For the purposes of the demo, make sure to add **Grace Hopper**:

Grace Hopper  
Ada Lovelace  
Katherine Johnson  
*your choice...*

The image shows the configuration page for an entity named 'author'. At the top left is the entity name 'author', and at the top right is a blue 'SAVE' button. Below the name, there are two checkboxes: 'Define synonyms' (which is checked) and 'Allow automated expansion' (which is unchecked). Underneath these checkboxes is a table with two columns: 'Enter reference value' and 'Enter synonym'. The table contains three rows of data: 'Grace Hopper' mapped to 'Grace Hopper', 'Anita Borg' mapped to 'Anita Borg', and 'Katherine Johnson' mapped to 'Katherine Johnson'. At the bottom left of the table area, there is a link that says '+ Add a row'.

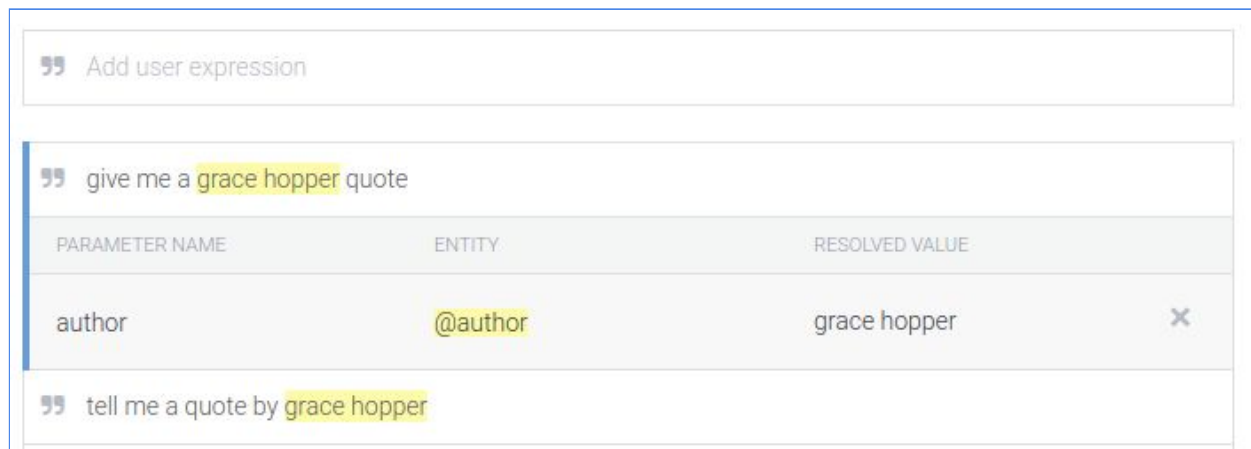
Hit **SAVE**. Now we need to give our agent some examples of users asking for quotes *by an author*.

Navigate back to **Intents > quote**. Add some more **User says** examples:

Give me a quote by Grace Hopper.

Tell me a Grace Hopper quote.

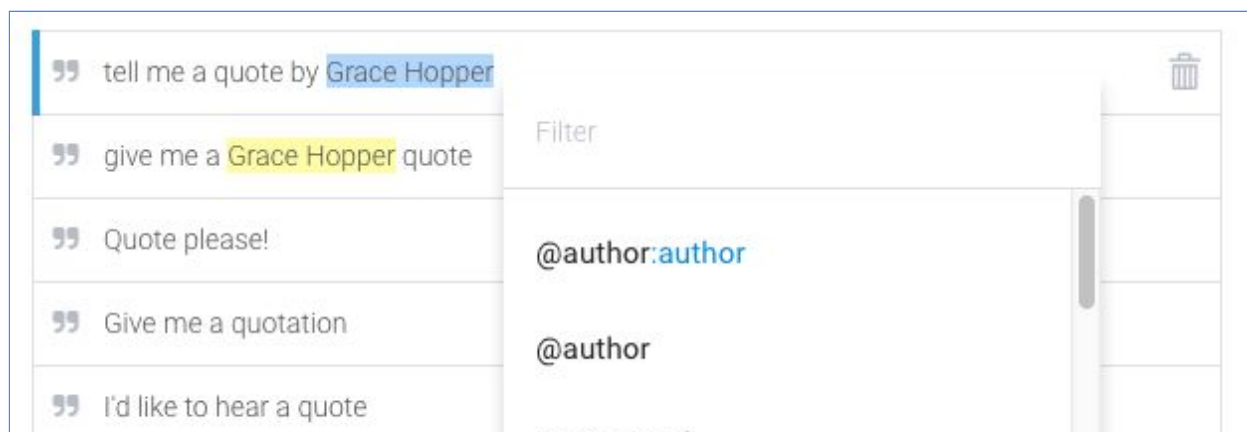
For now, we'll just give Grace Hopper examples. Notice how API.AI understands which portion of the example refers to our author.



The screenshot shows the 'Add user expression' section of the API.AI interface. It contains two user expressions: 'give me a grace hopper quote' and 'tell me a quote by grace hopper'. Below these, a table displays the resolved values for the parameters.

PARAMETER NAME	ENTITY	RESOLVED VALUE
author	@author	grace hopper

If for some reason the author's name is not automatically highlighted, you can do it yourself by selecting the text and choosing @author from the drop down:



The screenshot shows the 'Add user expression' section with a list of user expressions. The first expression, 'tell me a quote by Grace Hopper', has 'Grace Hopper' highlighted. A dropdown menu is open, showing the option '@author:author' selected, with '@author' also visible below it.

Under the Action section, we also now have a table that shows that *author* is now one of the parameters to our backend request.

REQUIRED ⓘ	PARAMETER NAME ⓘ	ENTITY ⓘ	VALUE	IS LIST ⓘ
<input type="checkbox"/>	author	@author	\$author	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

[+ New parameter](#)

Hit **SAVE** and try it out. What happens when you ask:

Give me a quote by Grace Hopper.

What about:

Give me a quote by Katherine Johnson.

Notice that we didn't have to give our agent any examples with Anita. It already knows about all our *authors* that we listed under **Entity**!

We also didn't need to change anything about the link to our backend, which is also already set up to handle the *author* parameter.

## Self-Paced Exercises

***Great work! The following exercises are for you to try on your own. You can choose which ones you want to try; some will indicate other exercises as prerequisites.***

***Get creative and try adding new functionality to your agent! Flag down a facilitator if need any help.***

### Exercise 0: Debugging.

API.AI offers some debugging output that will be useful as you make your way through the exercises. Start by typing the following into the **Try it now** box:

Give me a quote by Grace Hopper.

Below the response, you should see some additional output:

A screenshot of a debug output window with a blue border. It contains the following information:

INTENT	
quote	

ACTION	
get_quote_response	

PARAMETER	VALUE
author	Grace Hopper

At the bottom, there is a button labeled "SHOW JSON".

The output contains a few fields with the values filled:

**INTENT:** quote

**ACTION:** get\_quote\_response

**PARAMETER:** author      **VALUE:** Grace Hopper

You can use this to check what intent your agent identified, and whether it correctly identified the *author*.



For more detailed debug output, you can click **SHOW JSON** to see how all fields have been filled.

## Exercise 1: Play with mandatory parameters.

Navigate to **Intents > quote** (if you're not already there).

What happens when you make the *author* a **Required parameter** by checking the checkbox on the left hand side of the parameter?

Action


get_quote_response					
REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?	PROMPTS ?
<input checked="" type="checkbox"/>	author	@author	\$author	<input type="checkbox"/>	Define prompts...  
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>	—

+ New parameter

Click **Define prompts** on the right hand side of the table and try to come up with some good prompts for the agent to ask the user to specify who they want a quote from.

For example:

From whom would you like a quote?


Prompts for "author" 

	NAME	ENTITY	VALUE
	author	@author	\$author

PROMPTS

1

From whom would you like a quote?



2

Enter a prompt variant

Close

Click **Close**. Now, you should see your the beginning of your prompts in the rightmost prompt column:

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?	PROMPTS ?
<input checked="" type="checkbox"/>	author	@author	\$author	<input type="checkbox"/>	From whom would...
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>	—



Hit **SAVE** and try out your updated agent. What happens now if you only ask for a quote without specifying who the author should be?

Tell me a quote please!

**What do you think is better:** An interface where *author* is required, or where *author* is optional? **This is your decision to make for your agent!**

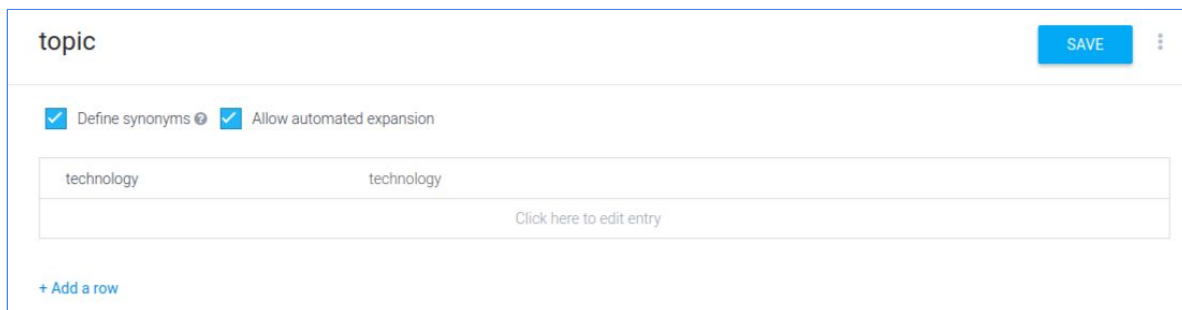
Note as well that if you ask for a quote by an author who does not exist in our database, the agent may continue to ask whose quote you would like to hear.

## Exercise 2: Add a topic field.

Our backend also has the ability to search for quotes by topic. For example, you could ask for a quote about computers, biology, or education.

Create a new **Entity**. Name this entity *topic*. (**Make sure to use this name**). Give it just one example: *technology*.

**Next, you will need to check *Allow automated expansion*.** This will let your agent learn examples of topics on the fly, instead of being restricted to the list you provide.



The screenshot shows a web interface for configuring an entity named "topic". At the top, the name "topic" is in a text field, followed by a blue "SAVE" button and a vertical ellipsis menu. Below this, there are two checked checkboxes: "Define synonyms" and "Allow automated expansion". Underneath is a table with two columns, both containing the word "technology". Below the table is a link that says "Click here to edit entry". At the bottom left, there is a link that says "+ Add a row".

Then **SAVE**. We will train our agent to understand what a *topic* is by pointing out topics within examples queries, rather than giving it a list of all subjects.

Now let's train our agent to recognize the subject parameter. Go back to the *quote* intent and start by adding an example:

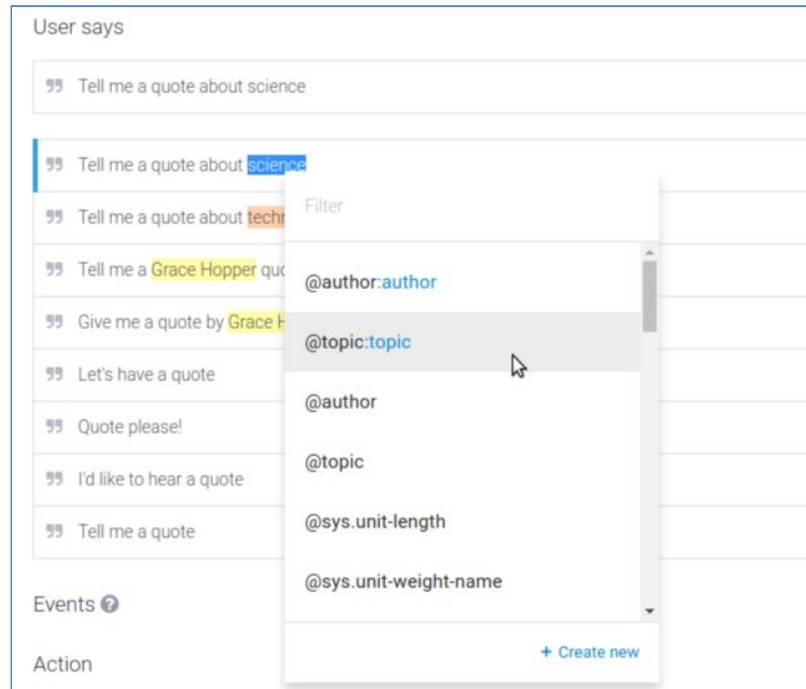
Tell me a quote about technology.

Note that *technology* is already recognized as a *topic* entity. Now we'll teach it some other words that can be topics. (The full list of topics our backend supports is [here](#).)

Give the agent a **User says** example that it might not recognize:

Tell me a quote about science.

We need to tell it that *science* is also a topic. Double click on the word *science* to highlight it, and select the `@topic:topic` entity.



Hit **SAVE**. What happens when you try the following in the **Try it now** box:

**Tell me a quote about adventure**

You should see that the *topic* parameter has the value *adventure* when you look at the debug output:

INTENT	
quote	
ACTION	
get_quote_response	
PARAMETER	VALUE
author	
topic	adventure

What are some other ways you could ask for a quote about a particular topic? Do any of these examples work? Check the debug output to see if the topic parameter was properly identified.

**Give me a quote about diversity**

**Give me a technology quote**

Give me a mathematics quote  
I want a quote on invention  
I want to hear a quote about language

Add some more training examples to make your agent more robust and get all of these examples working. Let the person next to you stress test your agent with some creative examples.

### Exercise 3: Ask for a bio.

Our webhook also has the ability to give us bios of our awesome STEM women.

#### Optional

Try out a curl command in any command line (i.e., terminal) to get a bio from our backend:

```
curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
  -d '{"result": {"action": "get_bio_response",
                  "parameters": {"author": "Marie Curie"}}}' \
  -H "Content-Type: application/json"
```

Use your knowledge from the previous steps to create a new **Intent** that can give you a bio.

**HINT:** The **Action** for this **Intent** should be called *get\_bio\_response*.

**HINT:** You may want to make the *author* parameter mandatory for this intent.

**HINT:** Make sure to enable the webhook for this intent.

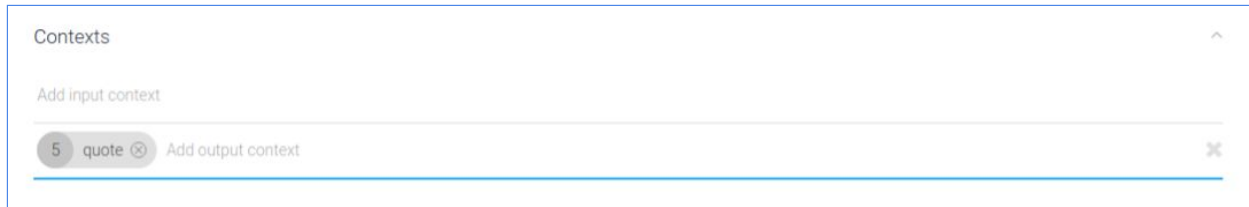
Stuck? Head to the [Answers](#) section at the end of the doc for the instructions.

### Exercise 4: Use context.

After hearing one awesome quote by a woman in STEM, our users will probably want more. Let's add functionality that lets users ask for "another one" to get another quote.

We can do this with API.AI using **Contexts**. You can read about **Contexts**, and what they mean on the **API.AI** platform, in the [API.AI docs](#).

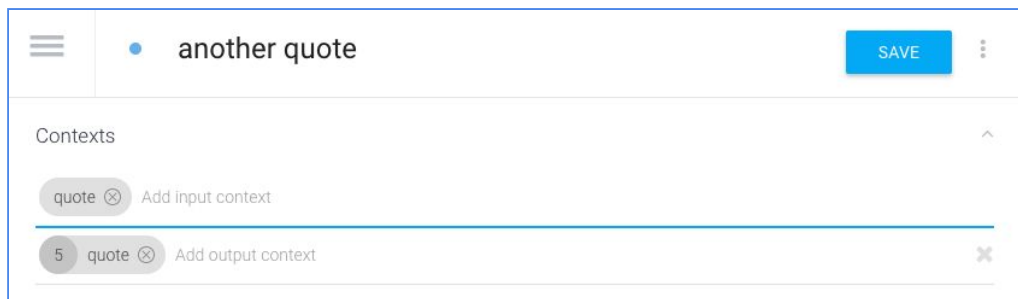
Navigate to your *quote* intent and click **Contexts**. A context is a way of recording previous intents. In **Add output context**, write *quote* and hit enter.



Note that the number “5” that automatically gets populated next to the *quote* output context.

What we’ve just done is set a context token that will be passed around for the next 5 turns the user takes in the conversation. For the next 5 things the user asks the agent, the agent will be able to access the context token *quote* to know that the user previously asked for a quote. If you want to adjust the length of the agent’s memory, you can click on the 5 to change it to another number. We will revisit this later.

Now let’s use this context. Create a new intent called *another quote*. Click **Context** and add *quote* as its **input context**. This means that the intent will *only* be triggered if the quote intent was triggered in the last 5 turns. Notice a *quote* output context automatically gets populated. You can decide later if you want this or not.



Give some examples of a user asking for another quote. For example:

another one  
another quote  
more quotes please  
i want to hear another

• another quote
SAVE

Contexts

quote
Add input context

5 quote
Add output context

User says

Search in user says

Add user expression

more quotes

more please!

another quote please

I want another

tell me another quote

tell me another one!

We still need to tell the agent what to do when this intent is triggered. We want to perform the same action as in the *quote* intent: telling the user a quote. Copy the Action from the *quote* intent: `get_quote_response`. Under **Fulfillment**, check **Use webhook**.

Finally, we can even grab our parameter values from the *quotes* intent. Add two parameters to the table below **Action** (you will need to type this in manually):

Parameter name	Entity	Value
author	@author	#quote.author
topic *	@topic	#quote.topic

\* If you completed Exercise 2 (Add a topic).

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?
<input type="checkbox"/>	author	@author	#quote.author	<input type="checkbox"/>
<input type="checkbox"/>	topic	@topic	#quote.topic	<input type="checkbox"/>

We use `#quote` to refer to parameters from the `quote` context. Any parameters that were set while the `quote` context was in action can be accessed this way.

Hit **SAVE** and try it out!

give me a quote by katherine johnson  
another one please!

Note that the `quote` context lasts for 5 turns after the user initially asks for a quote. Is this reasonable? Would you change it? You can do so by clicking on the “5”.

Note that in the current implementation, we will always give our user quotes from the same author and topic as the previous quote. You can change this by removing parameters from the `another quote` intent.

## Exercise 5: Use Events.

**Events** are a feature that allows you to invoke intents from your backend.

In our current setup, we ask for a quote and receive a blob of display text and speech output back from our backend as the reply.

We can change the settings so that we pass some structured data instead, in the form of the author and the quote itself, to a new **Intent**. This way we control how we phrase our response to the user, or even add some variation in how we respond.

### Optional

Under the quote intent, if we change the action parameter to our webhook, we can get a structured response:

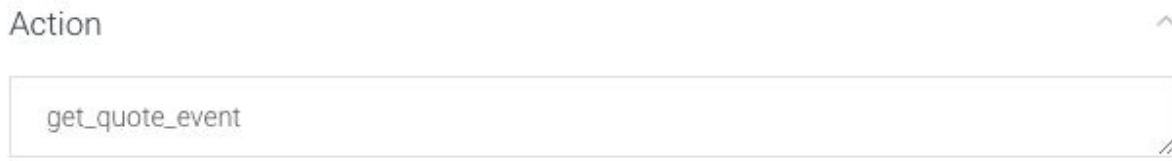
```
curl -X POST https://grace-hopper-quotes.appspot.com/quotesearch \
-d '{"result": {"action": "get_quote_event",
               "parameters": {"author": "Marie Curie"}}}' \
-H "Content-Type: application/json"
```

Backend response:

```
{"followupEvent":
  {"data":
    {"quote": "That brain of mine is something more than merely
              mortal; as time will show.",
      "author": "ada lovelace"},
    "name": "respond_with_quote" } }
```

When we change the **Action** from `get_quote_response` to `get_quote_event`, the reply from our backend takes the form of a `followupEvent` object.

We need to make a few changes to our agent to use this structured data in the follow-up event. First, navigate to your *quote* intent and change the **Action** to *get\_quote\_event*:



Action

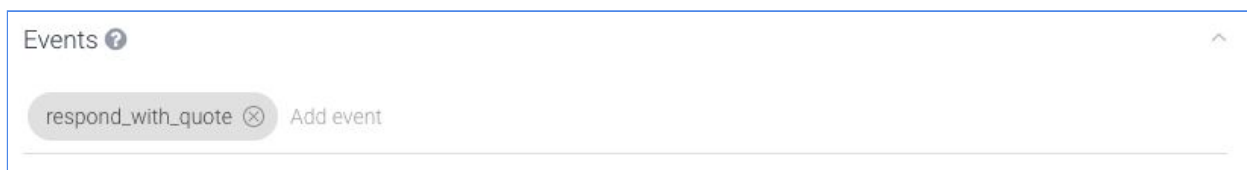
get\_quote\_event

Hit **SAVE**.

We now need to create a new intent that uses the structured data returned from our backend.

Create a new **Intent**. Call this intent *respond with quote*.

Click on the **Event** field and name this event *respond\_with\_quote*.



Events ?

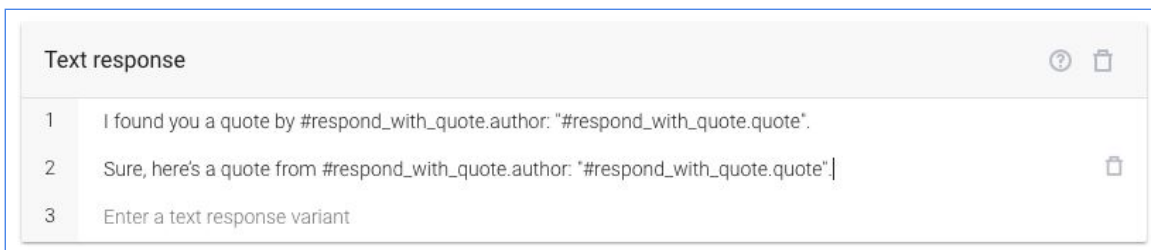
respond\_with\_quote (X) Add event

This means that our new *respond with quote* intent will be triggered whenever our backend returns a *followupEvent* object with name *respond\_with\_quote*.

You can leave the **User says** fields blank.

Now we need to create a response from our agent that uses the data from our backend. Add a couple of **Text response** messages

I found you a quote by #respond\_with\_quote.author: "#respond\_with\_quote.quote".  
Sure, here's a quote from #respond\_with\_quote.author: "#respond\_with\_quote.quote".



Text response

1	I found you a quote by #respond_with_quote.author: "#respond_with_quote.quote".
2	Sure, here's a quote from #respond_with_quote.author: "#respond_with_quote.quote".
3	Enter a text response variant

Note the use of `#respond_with_quote`. This allows us to refer to parameters from the `followupEvent` json object from our backend.

Hit **SAVE** and try it out!

Give me a quote by Ada Lovelace.

It should still work when you ask the agent:

tell me another one

## Exercise 6: Play time!

What are some other actions you could implement for your agent? Think about some things the user might want to ask. For example, “What can you do?” or “How do I hear a quote?”

Have fun. You could have your agent tell a joke, tell you how it’s doing, whatever you want!

# Appendix

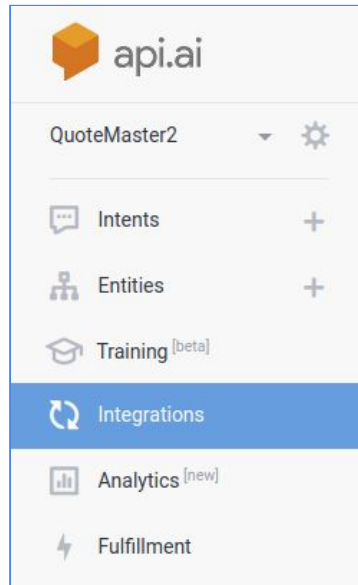
## Glossary

- Agent** Natural Language Understanding module that helps users get a set of tasks done with conversation. (Example: Panera agent)
- Intent** Mapping between what a user says and what action should be taken by the software. (Example: `add_food_to_cart` intent)
- Action** The step your application takes when an intent is triggered. (Example: adding a food item to the backend shopping cart)
- Entity** Represents a category of things in the world, and is used to express the type of a parameter for intents. (Example: `food_item` entity)

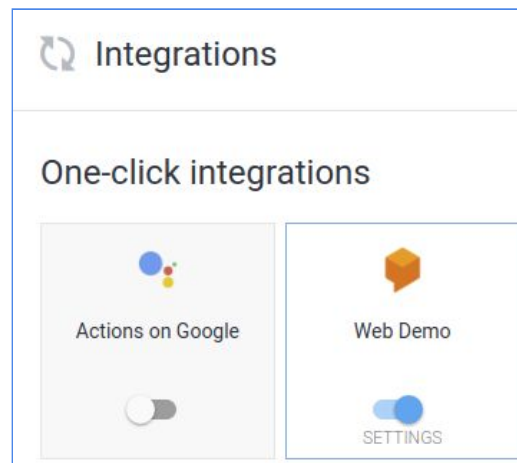
## Integrating your agent with other platforms

**API.AI** offers one-click integrations with a number of platforms. Click the **Integrations** tab on the right hand menu to see what’s available.





You'll see toggles that let you easily integrate with Actions on Google, the API.AI Web Demo, and more. A fun one to try right away is the **Web Demo**. Click the toggle to enable it:



A window will pop up with a link to a page just for your agent:



## Web Demo



Test the agent on its own page. Share the link to the page or embed the widget in other websites to get more conversations going. [More in documentation.](#)

<https://bot.api.ai/a85e56ca-b3a7-4443-8005-3a3959591135>



Seems that your agent info is not filled yet. Set icon and description for better end-user experience.

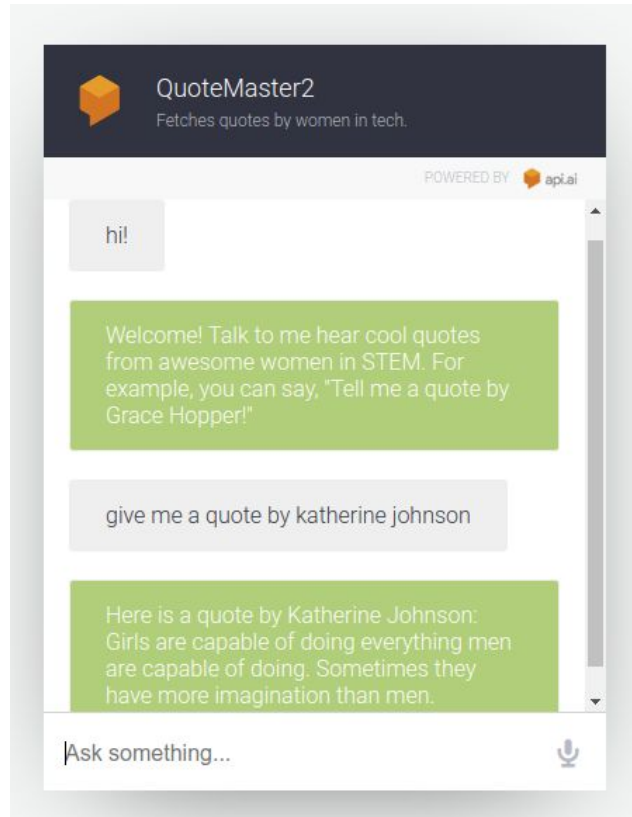


Add this agent to your website by copying the code below:

```
<iframe
  width="350"
  height="430"
  src="https://console.api.ai/api-client/demo/embedded/a85e56ca-b3a7-4443-8005-3a3959591135">
</iframe>
```



Follow the link to try out the Web Demo of your agent!



## Building your own backend

Interested in setting up your own backend? Start by reading the **API.AI** docs on [Fulfillment](#). These will step you through setting up a webhook.

You can see the code for our backend at <https://github.com/juliar/quotes-agent>.

## Answers

### Answer to Exercise 3: Ask for a bio.

Start by creating a new **Intent**. You can call this intent *bio*. Fill in the **Action** field with the value *get\_bio\_response*.

Action
get_bio_response

Now add some **User says** expressions:

Who was Grace Hopper?  
Give me Grace Hopper's bio.  
Tell me about Ada Lovelace.  
Who is Katherine Johnson?  
I want the bio of Grace Hopper.  
Tell me a bio for Ada Lovelace.

Your agent should recognize the **author** parameter in all of these statements.

” Tell me a bio for Ada Lovelace

PARAMETER NAME	ENTITY	RESOLVED VALUE	
author	@author	Ada Lovelace	✕

” I want the bio of Grace Hopper

” Who is Katherine Johnson?

” Tell me about Ada Lovelace

” Give me Grace Hopper's bio


” Who was Grace Hopper?

Now scroll down to the parameter table under Action and check the **Required** box by the **author** parameter.

Action					
get_bio_response					
REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
<input checked="" type="checkbox"/>	author	@author	\$author	<input type="checkbox"/>	<a href="#">Define prompts...</a>

Click Define prompts and add a prompt:

Whose bio do you want?

Prompts for "author" <span>×</span>		
NAME	ENTITY	VALUE
author	@author	\$author
PROMPTS		
1	Whose bio do you want?	
2	Enter a prompt variant	

Finally, at the bottom of the page, click **Fulfillment** and select **Use webhook**.

Fulfillment

☒ Use webhook
☐ Use webhook for slot-filling

Hit **SAVE**.

Now try a few expressions in the **Try it now** box:

Who was Ada Lovelace?

What is the bio for Grace Hopper?

Tell me about Katherine Johnson.

You should receive the bio for each of these women.



