

Exercise 1: Configuring Slot's Found and Not Found

Slots are awesome. However, their stubborn nature (for required slots that specify a question) can come across as rude if we are not careful. They keep the user to the task, which is fine if the user enters something irrelevant. It's less okay if the user is asking a legitimate side question, however.

Consider the following interaction.

The image shows a chatbot interface with a dark background. At the top, a user message says "I'd like flower recommendations". Below it, a system message shows the slot "#flower_recommendations" with a blue checkmark icon. Then, the user asks "What occasion are the flowers for?" with a location pin icon. The system responds with "actually wait... How late are you open until tonight?". Below this, another system message shows the slot "#hours_info" with a blue checkmark icon. Finally, the user asks "What occasion are the flowers for?" again with a location pin icon. The interface highlights that the chatbot understood the user's digression but did not address it, instead continuing to ask for the original task's information.

That's not great. It looks like we completely ignored the user's legitimate question. Notice also that the chatbot understood what the user wanted (i.e., #hours_info was detected) but we were still somewhat rude to them by not addressing their digression.



Lab 12: Enable Digressions

Configuring Found and Not Found responses

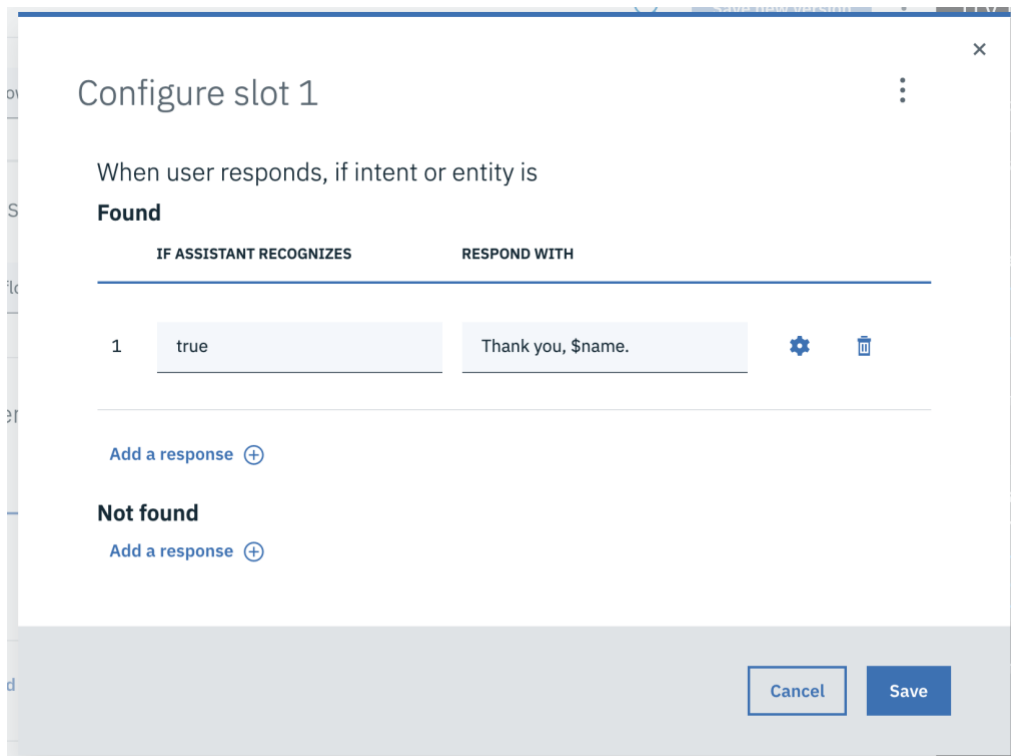
A first course of action we have is to use the Slot's *Found* and *Not Found* responses. These are issued to the user before the node's own responses (in our case the actual flower suggestions for the given occasion), so they allow us to interject a message before the node replies to the user.

Let's see them in action.

1. Select the *Flower Recommendations* node and **click on the gear icon next to *Required* in our node's slot.**

	CHECK FOR	SAVE IT AS	IF NOT PRESENT, ASK	TYPE	
1	@occasion	\$occasion	What occasion are tl	Required	 



2. Configure the slot by scrolling to the bottom of the dialog that appears, and by **clicking *Add a response* below the *Found* section.** Then **enter *true* as the condition and *Thank you, \$name.* as the response in the *Found* section.**



Configure slot 1

When user responds, if intent or entity is

Found

	IF ASSISTANT RECOGNIZES	RESPOND WITH	
1	true	Thank you, \$name.	 

Add a response (+)

Not found

Add a response (+)

Cancel Save

Lab 12: Enable Digressions

This will ensure that when the slot receives the answer that it's looking for (the occasion in this case), we will always issue a thank you. If we wanted more refined control, we could add different responses for different inputs from the user. In our case, we are happy with a standard thank you response.

3. Similarly, add a **Not Found** response with a true condition that responds with, **Sorry to ask again, \$name, but what occasion are the flowers for?** as shown in the image below.

×

Configure slot 1

⋮

Found

IF ASSISTANT RECOGNIZES

RESPOND WITH

1

true

Thank you, \$name.

⚙️

🗑️

Add a response

⊕

Not found

IF ASSISTANT RECOGNIZES

RESPOND WITH

1

true

Sorry to ask again, \$name, but v

⚙️

🗑️

Cancel

Save

Finally, click *Save*.

Now, when we try the interaction again, we get a slightly more friendly interaction as shown below.

Lab 12: Enable Digressions

What occasion are the flowers for?

actually wait... how late are you open until?

#hours_info

▼

👁

Sorry to ask again, Antonio, but what occasion are the flowers for?

birthday

Irrelevant

▼

👁

@occasion:Birthday

Thank you, Antonio.

Opt for a fun bouquet of flowers, choosing a colorful one from [our catalog](#).

That's better, maybe even good enough! We are apologizing when we have to ask again, and we are thanking the user when they provide the answer we need. Since we have their name, we sweetened the deal by including it in our messages.

Lab 12: Enable Digressions

But... we are still not answering the user's legitimate side question about hours of operation. To properly handle that, we'll need *Digressions*.

Exercise 2: Enabling Digressions

Digressions are a feature that enables nodes to pass control to a different node while they are processing a slot. What this means is that they allow the dialog to respond to a user's side question, while they are in the middle of answering a slot.

Let's see them in action.

1. The first thing we need to do is ensure that the node containing the slot allows digressing away from it. Select the *Flower Recommendations* node again and click on *Customize*, and then its *Digressions* tab.

Here you'll want to expand *Digressions cannot go away from this node* by clicking on the > next to it and **turn on the option to allow digressions away from this node**. Make sure you click *Apply*.

×

Customize "Flower Recommendations"

Customize node

Digressions

▼

Digressions can go **away from** this node

Allow digressions away while slot filling

Off ☒ On

Users can divert the conversation away from this node in the middle of processing slots.

☐ Only digress from slots to nodes that allow returns

If a user goes off topic, only nodes with digressions that allow returns will be considered.

>

Digressions can come **into** this node

Cancel

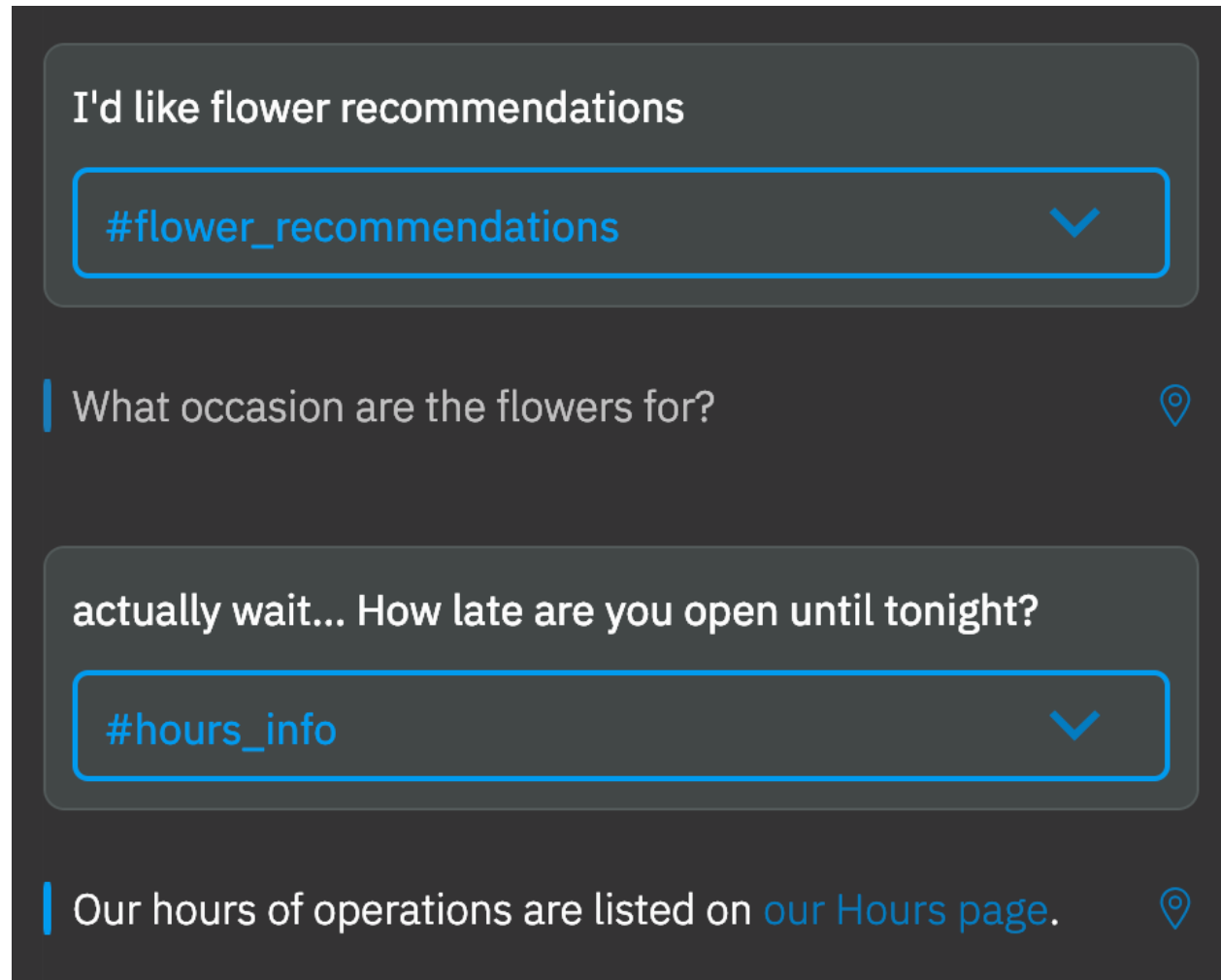
Apply

Lab 12: Enable Digressions

With the digression away enabled in our node, we'll now be able to ask other questions in the middle of answering the slot's question and get a proper response for them.

Technically, the nodes we digress into need to allow the digression, but that's the default setting so we don't have to worry about it. (Should you ever need to prevent a node from being digressed into, you can disable the *Digressions can come into this node* for that node.)

Let's see how this altered our interaction.



Awesome. We fixed our original problem. However, you might notice that we don't come back to the original question about flower recommendations.

This may or may not be what we want, depending on our chatbot. If we'd like to return, we'll need to explicitly **set *Return after digression*** in the nodes we might digress to.

Lab 12: Enable Digressions

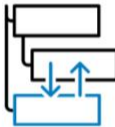
Go ahead and set the option for all three nodes with slots (i.e., *Hours of Operation*, *Location Information*, and *Flower Recommendations*) as shown below. As a reminder, you can access the *Digressions* section by clicking on *Customize* in each of these three nodes.

Customize "Flower Recommendations"

Customize node Digressions

2. This node has **edited** digressions settings ⓘ

- > Digressions can go **away from** this node ⓘ
- ✓ Digressions can come **into** this node ⓘ

3.  Allow digressions into this node Off —●— On

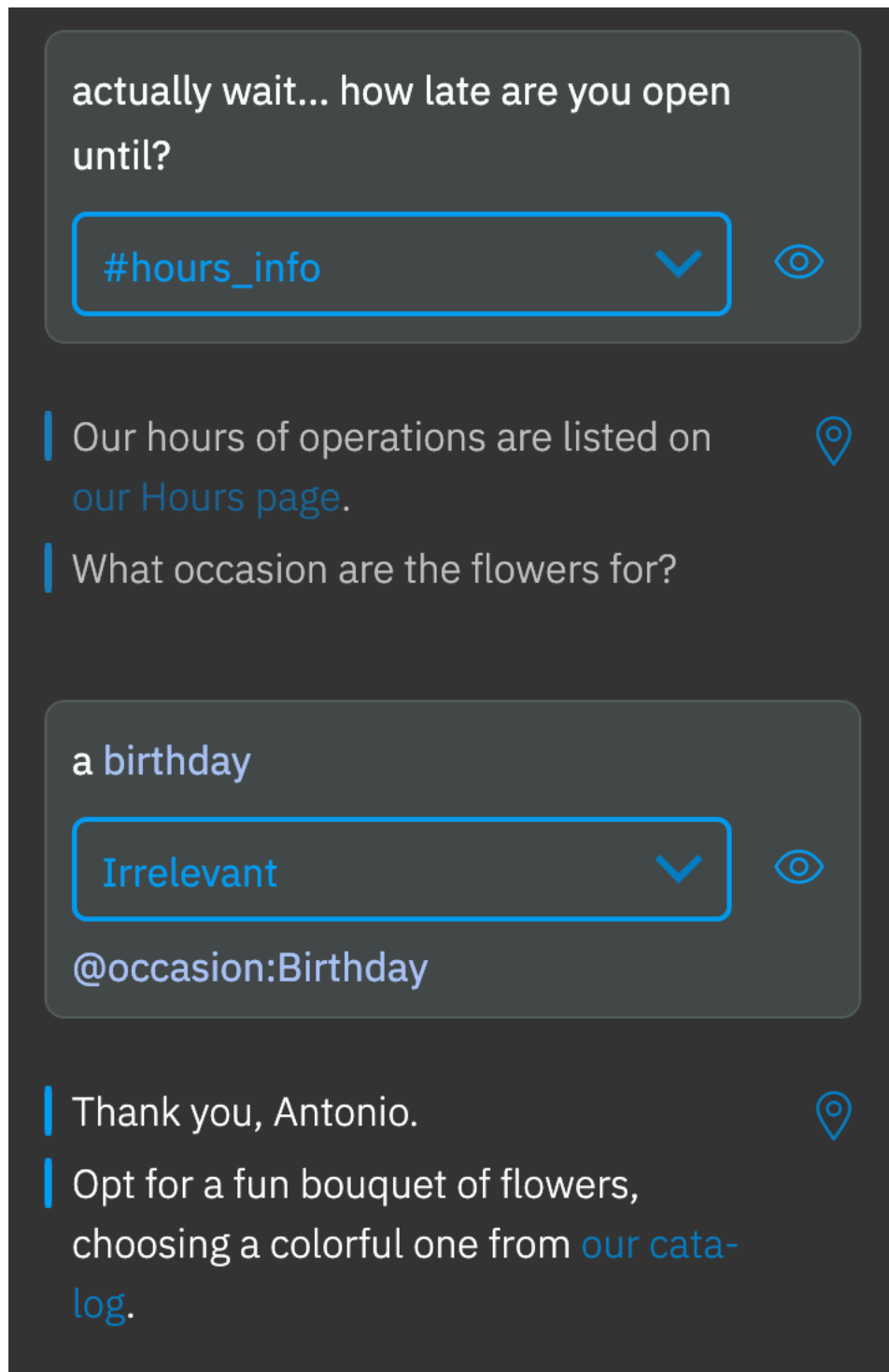
Users can digress to this node from other dialog flows.

4. ☒ Return after digressions

After this dialog flow is processed, return to the dialog flow that was previously in progress.

Cancel Apply

Don't forget to click on *Apply*, and then test the interaction again, as shown in the image below.



That's quite nice! Our chatbot interacts like a polite human would in such a conversation.

2. Technically, neither *Hours of Operation* nor *Location Information* have required slots, so we don't need digressions. But in the future, we might decide to make their slots mandatory, so as

Lab 12: Enable Digressions

an exercise, there is no harm in enabling the digression for both nodes now. Go ahead and **enable *Allow digressions away* for both nodes.**

3. Test a few conversations with your chatbot from the WordPress site you deployed to earlier in the course.

Found and *Not Found* responses, along with *Digressions*, are useful tools to add further polish to our chatbot and make it appear more human-like in its interactions with our users.

Handlers



We didn't need it in our chatbot, but sometimes we might want to execute certain actions if the user responds to a slot in a certain way. For example, if we detect that the user is trying to cancel a reservation, we want to escape/exit the slot rather than continue to pester the user for details until they get frustrated and just leave the chat.


To skip a slot when a certain condition is met, you might use *Handlers*, accessible via the *Manage Handlers* link in the slot section of your node. Handlers are evaluated even before the *Not Found* response is issued.

For example, we could have the following response in our handler. Clicking on the gear icon offers us more detailed control.

Handlers are how your bot will respond when the users answer to a prompt is not found. These handlers will be checked before trying the "Not found" responses in a slot.

If answer to any prompt is not found and

	IF ASSISTANT RECOGNIZES	RESPOND WITH	
1	#cancel_reservation	Got it \$name, we won't re	 

[Add handler](#) 

[Cancel](#) [Save](#)

This includes specifying that the handler should escape the current slot by skipping it when the handler's condition (e.g., a cancellation intent) is detected in response to the slot's question. We can skip the current slot or skip all the slots in the node jumping straight to the response.

Lab 12: Enable Digressions

In the theoretical scenario of a reservation cancelled mid-booking, we'd want to skip any further question and will therefore opt to skip straight to response.

×

Configure handler 1

Response variations are set to **sequential**. Set to [random](#)

[Learn more](#)

Add response type +

Then assistant should

Skip to response ▼

Cancel

Save

The fallback/*true* response for such a reservation node, will likely have some kind of message inviting the user to reserve again at a later time or something equally user friendly.

Again, this is not relevant to our chatbot, but I wanted you to know about the existence of *Handlers* and why they might be useful at times.

Our chatbot

Congratulations on getting so far into the course. We're almost done. Meanwhile, if you need it you can [download and import a JSON file](#) of the dialog skill we created so far.