**Creating a Python Widget and Alexa Skill**

We will create an Alexa Hosted Python skill and a gadget connected to this.

This is based on the Alexa example at:

<https://github.com/alexa-samples/skill-sample-plant-care-widget>

See my videos:

This video: <https://youtu.be/ebU2fi1WU0I>

Plant care skill in action: <https://youtu.be/DNv0sozL-kE>

Plant care widget in action: <https://youtu.be/Ooj_GGOLfo4>

Code available at:

<https://github.com/jallwork/AlexaWidgets>

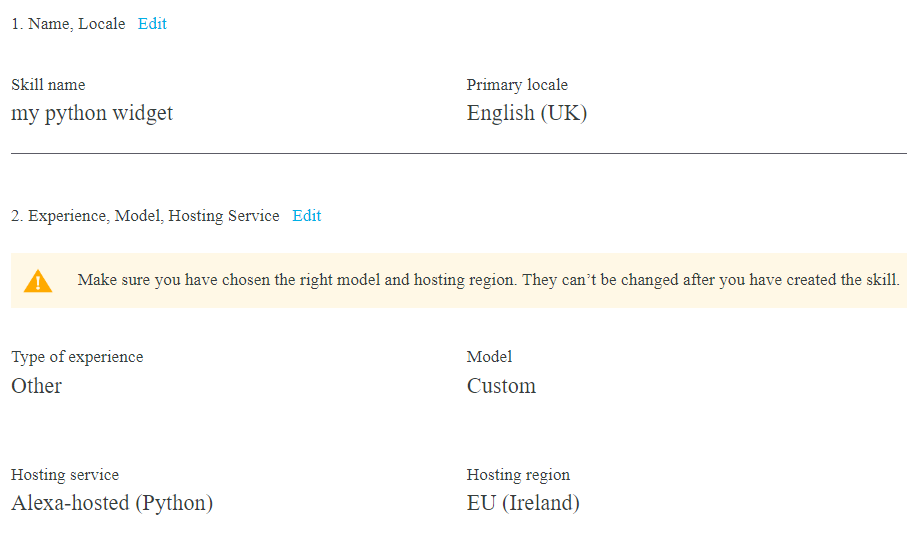
You can create a widget on its own, but it has to be connected to a skill, so we’ll start by creating a skill.

1. In the developer console, click Create Skill.

I’ve used the following options:

* Skill name: My Python widget
* Primary locale: English (UK) - (I’m in the UK)

Then: Other, Custom, Alexa Hosted Python, and your nearest hosting region (EU (Ireland)) and Start from Scratch and Create Skill. You should have something similar to this:



1. Select the interfaces.

Once the skill has been created, select Build > Interfaces, and enable the

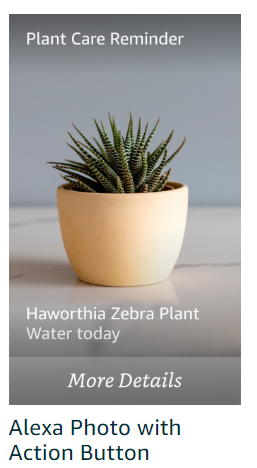
APL, Data Store and Data Store Packages.

Then rebuild and deploy the Skill

1. Add the gadget to the skill

Select Multimodal Responses and Widget. Select Create Widget.

From the suggestions, choose Alexa Photo with Action Button. We’ll over-write this code.

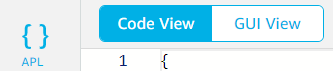


Before we do that, you can preview the widget on your device, click Install, choose your device and Send to Device.



1. Change the widget code.

Select APL > Code View



Copy and paste the code at the end for: the mainTemplate APL, and also the Data, Manifest, Document and Settings. Get these from <https://github.com/jallwork/AlexaWidgets/tree/main/dataStorePackages/plant-care>

If you’re not sure which is which, the mainTemplate.json (about 133 lines) starts with:

{

    "type": "APL",

    "version": "2023.1",

The data.json starts with:

{

    "alexaPhotoData": {

The manifest.json starts with:

{

    "manifest": {

        "id": "plant-care",

The document.json starts the same as mainTemplate, but is only 17 lines long.

And the settings.json starts with:

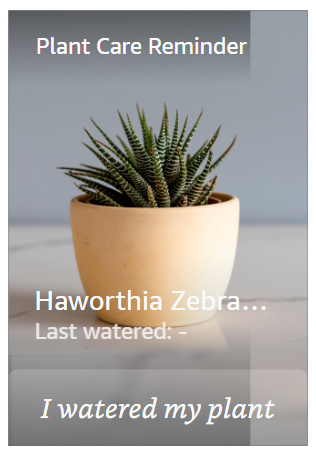
{

    "DataStore": {

        "dataBindings": [

Check that you have copied the five files

Your widget should display as follows:



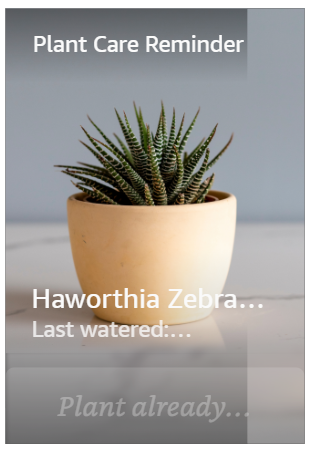
Save the widget as **plant-care**. When you do this, the name becomes the widget ID.

In this example, you can click the ‘I watered my plant’ button’ to show that you’ve watered your plant, or click ‘Plant Care Reminder’ to launch the Plant care skill.

You can test the widget by clicking the Arrow symbol in the top right-hand corner:



Click the ‘I watered my plant’ button and see it in action and ‘Plant already watered’ appear.



When your widget is installed and you click on the button it sends and event which is picked up by our code:

Either:

"type": "SendEvent",

"arguments": [

    "plantWateredWidget",

    "${currentDate}"

],

or:

"onPress": [

    {

        "type": "SendEvent",

        "arguments": [

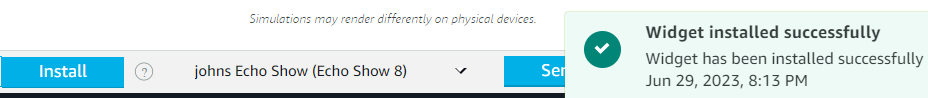
            "openSkill"

        ],

We’ll see this picked up in the code later.

1. Check the widget on your device

To install the widget on your device, click ‘Install’, select your device and ‘Send to Device’



If you get an error here, have you saved your widget? Is your device on line?

Note that we don’t have any APL for our skill in this example. That’s included in the code.

Save the widget and return to the Build tab

1. Update the Interaction model

Now we can deal with our skill code

Firstly, check your invocation is as expected.

Then open Interaction model > Json editor and copy the code from:

<https://github.com/jallwork/AlexaWidgets/blob/main/interactionModels/custom/en-GB.json>

This adds two intents to the skill: a PlantCareIntent and I’ve added a ‘unwaterplant’ intent.

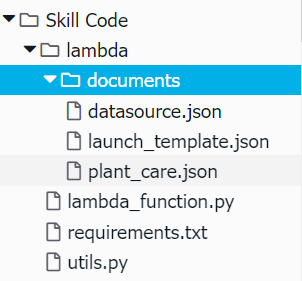
If you are in a different locale, or have more than one locale, remember to check or change those as well.

Save and build the skill.

1. The Code

Click the code tab.

First, create a folder structure as follows:



The datasource.json, launch\_template.json and plant\_care.json are the json APL used for the plant care skill. They are included in the addDirective call in the code.

* Copy and paste these files from:

<https://github.com/jallwork/AlexaWidgets/tree/main/lambda/documents>

* Change (or copy from github) **requirements.txt** to add the persistence adapter:

boto3==1.9.216

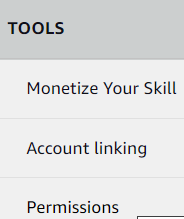
ask-sdk-core==1.11.0

**ask-sdk-dynamodb-persistence-adapter==1.15.0**

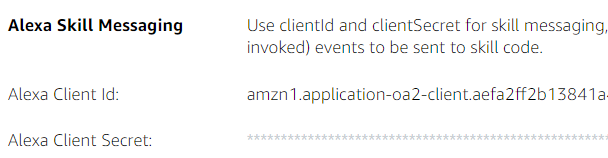
* Copy lambda\_function.py from:

<https://github.com/jallwork/AlexaWidgets/blob/main/lambda/lambda_function.py>

You will have to find and replace your Alexa ID and secret. Find Build > Tools >permissions



And scroll down to Skill Messaging



If they aren’t there, you’ve most probably forgotten to select data store / packages

Copy those and replace in the lines (#37 and #38)

**ALEXACLIENTID = "your ID here (in the quotes)"**

**ALEXACLIENTSECRET = "your Alexa secret here (in the quotes)"**

That’s all the files copied. Check you have done them all.

1. Code discussion

Code has

* getAccessToken() – gets an access token so that send messages to your widget, e.g. install, remove and update.
* RemoveWidgetRequestHandler – used when widget is removed
* InstallWidgetRequestHandler – used when widget is installed
* UpdateWidgetRequestHandler - used e.g when updating last watered date
* WidgetInstallationErrorHandler – error handling
* updateDatastore – sends REQUEST POST to update the widget’s datastore items
* APLEventHandler – used to deal with the touch events for 'openSkill', 'plantWateredWidget' or plantWateredSkill
* PlantCareIntentHandler – runs when user says ‘water my plant’ to the skill
* unwaterIntentHandler – I’ve added to reset the watered date to null

For more information on requests, see:

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-datastore-package-manager-interface.html#requests>

The commands install, remove, update, unwater, etc. send a token, command and target to the update datastore function.

These are done with a request POST (axios is used in the Alexa NodeJS example)

If the target is type “USER” with a userId, then **all** the user’s devices are targeted and widgets updated. Individual devices can be targeted, but not used in this code.

My code also keeps a record of how many widgets are installed on a device (I think!)

1. Try your skill

Note that after pressing the button in the skill it goes back to the skill’s front page

Check that the widget/s also sees the updated watering.

1. Summary

We have created a skill and a widget using APL.

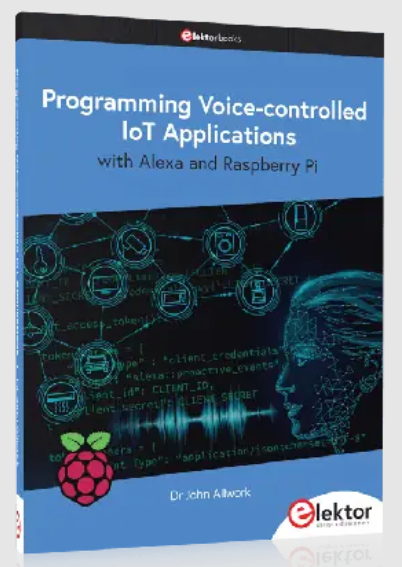
We saw how to create, preview and install a widget on an Echo device.

We wrote (or at least copied) the skill code and saw that it needed install, remove and update widget code; code to update the widget datastore using *requests.post* ; an APL event handler to deal with the skill and APL events, as well as normal skill intent code.

Feel free to use this code, but if it all goes wrong, I shall deny ever writing it.

If you like this, then there’s MUCH MORE in my book:

<https://www.elektor.com/programming-voice-controlled-iot-applications-with-alexa-and-raspberry-pi>



Or watch more videos at:

<https://www.youtube.com/channel/UCTV3U6BgaM5hdmjF-5Jyg8w>

And my GitHub at:

<https://github.com/jallwork>

References:

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/about-widgets-and-apl.html>