alexa-skill Documentation

Piotr Rogulski, stanwood GmbH

Sep 28, 2018

Contents:

1 alexa_skill **3**
1.1 alexa_skill package 3

2 Installation **9**

3 Flask example usage **11**

4 Falcon example usage **13**

5 Indices and tables **15**

Python Module Index **17**

alexa-skill is flexible easy-to-use and extensible package for creating Alexa skill responses.

CHAPTER 1

alexa_skill

1.1 alexa_skill package

1.1.1 Subpackages

alexa_skill.intents package

Submodules

alexa_skill.intents.base module

class alexa_skill.intents.base.**BaseIntents**

Bases: object

Base class for creating any Intent which will be used in processor

mapper

Dictionary which map alexa intent name with intent method name.

:rtype dict

static response(**args*, ***kwargs*)

Creates alexa json response body.

Reference: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#response-format

alexa_skill.intents.buildins module

class alexa_skill.intents.buildins.**BuildInIntents**(*help_message*, *not_handled_message*, *stop_message='stop'*, *cancel_message='cancel'*)

Bases: *alexa_skill.intents.base.BaseIntents*

Buildin intents class which is handling all standard intents which are sent from Alexa.

cancel()
Handles build-in Alexa cancel intent. This method should end user session if no transactional action is made, otherwise it should end action.

Returns [Alexa voice message string, should end session bool]

Return type list

help()
Handles build-in Alexa help intent.

Returns [Alexa voice message string, should end session bool]

Return type list

mapper

not_handled()
Returns not handled message.

Returns [Alexa voice message string, should end session bool]

stop()
Handles build-in Alexa stop intent.

Returns [Alexa voice message string, should end session bool]

Return type list

Module contents

alexa_skill.tests package

Submodules

alexa_skill.tests.test_alexa_dates module

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_english_week**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_german_week**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_month**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_normal_day**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_week**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_weekend**()

alexa_skill.tests.test_alexa_dates.**test_amazon_dates_year**()

alexa_skill.tests.test_alexa_dates.**test_amazon_time**()

alexa_skill.tests.test_alexa_dates.**test_amazon_time_evening**()

alexa_skill.tests.test_alexa_dates.**test_amazon_time_morning**()

alexa_skill.tests.test_alexa_dates.**test_amazon_time_none**()

alexa_skill.tests.test_alexa_dates.**test_amazon_time_parsing_error**()

alexa_skill.tests.test_messages module

alexa_skill.tests.test_messages.**test_cards**()

alexa_skill.tests.test_messages.**test_confirm_slots_directives**()

alexa_skill.tests.test_messages.**test_reprompt_plain_text_default**()

alexa_skill.tests.test_messages.**test_reprompt_ssml**()

alexa_skill.tests.test_messages.**test_speech_output_plain_text**()

alexa_skill.tests.test_messages.**test_speech_output_ssml**()

Module contents

1.1.2 Submodules

1.1.3 alexa_skill.dates module

class alexa_skill.dates.**AmazonDateParser**

Bases: object

Amazon build-in date format parser.

Amazon Alexa API reference: https://developer.amazon.com/docs/custom-skills/slot-type-reference.html#date

WEEK_MAPPER = {'w0': 'w00', 'w1': 'w01', 'w2': 'w02', 'w3': 'w03', 'w4': 'w04', '

classmethod create_periods(*amazon_date*, *timezone=tzfile('/usr/share/zoneinfo/Europe/Berlin')*)

classmethod to_date(*amazon_date*)

Parses alexa date output string to mapped time.

Parameters **amazon_date** (*(str)*) – Amazon date string.

Returns Datetime with parsed date and date type. Otherwise (None, None) if date is not parsable.

Return type tuple

Note:

Possible date types:

- normal
- week
- weekend
- month

- year

class alexa_skill.dates.**AmazonTimeParser**
Bases: object

DAY_TIME_MAPPER = {'AV': (13, 0), 'EV': (18, 0), 'MO': (8, 30), 'NI': (21, 0)}

classmethod to_time(*amazon_time*)
Parses alexa time output string to mapped time.

Parameters **amazon_time** (*(str)*) – Amazon string time. Possible choices: [MO, AV, EV, NI].

Returns Time tuple with hour as first element and minutes as second, otherwise None when amazon_time is not parsable.

Return type tuple

1.1.4 alexa_skill.messages module

alexa_skill.messages.**cards**(*title*, *text*, *card_type='Simple'*, *image=None*)
Returns alexa cards

Parameters

- **title** (*str*) – A string containing the title of the card.
- **text** (*str*) – A string containing the text content for a Standard card (not applicable for cards of type Simple)
- **card_type** (*str*) – A string describing the type of card to render. Valid types are: * "Simple": A card that contains a title and plain text content. * "Standard": A card that contains a title, text content, and an image to display.
- **image** (*dict*) – An image object that specifies the URLs for the image to display on a Standard card. Only applicable for Standard cards. You can provide two URLs, for use on different sized screens. * smallImageUrl * largeImageUrl

Returns Dictionary with cards details

Return type dict

Alexa API Reference: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#card-object

alexa_skill.messages.**confirm_slots_directives**(*slots*)
Used to ask user for missing slots.

When confirm_slots is used, Alexa respond to a end user with message specified in *outputSpeech* element and expects the value for specified slots with the same intent as was detect in previous message.

Alexa API Reference: https://developer.amazon.com/docs/custom-skills/dialog-interface-reference.html#confirmslot

alexa_skill.messages.**create_response**(*text*, *card_title=''*, *should_end_session=True*, *reprompt=None*, *confirm_slots=False*, *speech_type='SSML'*)
Creates response for Alexa skill with answer for user intent.

Parameters

- **text** (*(str)*) – Text which should he responded to a user. Not required. Default: ‘’
- **card_title** (*(str)*) – Tile used in alexa cards
- **should_end_session** (*(bool)*) – Defines whether session should be close or not (Mike should listen for next user response or should close connection).
- **reprompt** (*(str)*) – Defines reprompt output speech text. Default: will be no reprompt.
- **confirm_slots** (*(dict)*) – Dict of slot name which require confirmation as key and updated intent dict as value.
- **speech_type** (*(str)*) – Defines type of speech which should be returned. Choices: SSML or PlainText.

Returns Dict with Alexa response format.

Alexa API Reference: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#response-format

alexa_skill.messages.**reprompt**(*text*, *speech_type='PlainText'*)

This object can only be included when sending a response to a [CanFulfillIntentRequest, LaunchRequest, IntentRequest, or InputHandlerEvent]

Parameters

- **text** (*(str)*) – Text which will be returned to the user.
- **speech_type** (*(str)*) – Type in which text is formatted. Choices: [SSML, PlainText]. Default: SSML

Returns Dictionary with output speech

Return type dict

Alexa API Reference: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#reprompt-object

alexa_skill.messages.**speech_output**(*text*, *speech_type='SSML'*)

This function is used for setting both the outputSpeech and the reprompt properties.

Parameters

- **text** (*(str)*) – Text which will be returned to the user.
- **speech_type** (*(str)*) – Type in which text is formatted. Choices: [SSML, PlainText]. Default: SSML

Returns Alexa parsable dictionary.

Return type dict

Alexa API Reference: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#outputspeech-object

1.1.5 Module contents

class alexa_skill.**Processor**(*request_body*, *buildin_intents*, *launch_message*, *session_end_message*, **intents*)

Bases: object

intent_name

intent_request()

Returns a list with: 0: message for user 1: bool: True when alexa request was handled by Backend 2: bool: if session should be ended

launch_request()

Returns a list with: 0: message for user 1: bool: True when alexa request was handled by Backend

locale
Used for setting i18n internationalization strings

request_type

session_attributes

session_end_request()

Returns a list with: 0: message for user 1: bool: True when alexa request was handled by Backend

session_hash
Return session hash of a user.

slots
Alexa slots which are defined in Alexa console.

API Reference: https://developer.amazon.com/docs/custom-skills/slot-type-reference.html

Return type dict

CHAPTER 2

Installation

Install and update using pip:

```
pip install -U alexa-skill
```

CHAPTER 3

Flask example usage

```
# The MIT License (MIT)
#
# Copyright (c) 2018 stanwood GmbH
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
import logging

from flask import Flask, request, jsonify

import alexa_skill
from alexa_skill.intents import BaseIntents
from alexa_skill.intents import BuildInIntents


app = Flask(__name__)

buildin_intents = BuildInIntents(
    help_message='Say "HI" to us',
```

(continues on next page)

(continued from previous page)

```
    not_handled_message="Sorry, I don't understand you. Could you repeat?",
    stop_message='stop',
    cancel_message='cancel'
)


class ExampleIntents(BaseIntents):
    @property
    def mapper(self):
        return {
            'EXAMPLE.hello': self.hello,
        }

    def hello(self):
        return self.response('Hello. Nice to meet you.'), True


@app.route("/v1/alexa/fulfiller", methods=['POST'])
def fulfiller():
    get_response = alexa_skill.Processor(
        request.json,
        buildin_intents,
        'Welcome to Alexa skill bot',
        'Good bye',
        ExampleIntents(),
    )
    json_response, handled = get_response()

    logging.info('Response was handled by system: {}'.format(handled))

    return jsonify(json_response)
```

CHAPTER 4

Falcon example usage

```
# The MIT License (MIT)
#
# Copyright (c) 2018 stanwood GmbH
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
import logging

import falcon

import alexa_skill
from alexa_skill.intents import BaseIntents
from alexa_skill.intents import BuildInIntents
from alexa_skill import dates


class ExampleIntents(BaseIntents):
    @property
    def mapper(self):
```

(continues on next page)

(continued from previous page)

```
        return {
            'EXAMPLE.hello': self.hello,
            'EXAMPLE.date_intent': self.date_intent,
        }

    def hello(self):
        return self.response('Hello. Nice to meet you.'), True

    def date_intent(self, slots=None):

        date, date_type = dates.AmazonDateParser.to_date(slots['dateslot']['value'])

        text = "Your date is <say-as interpret-as='date'>{}</say-as> and it is a {}".
↪format(
            date.strftime('%Y%m%d'),
            date_type
        )

        return self.response(text), True


buildin_intents = BuildInIntents(
    help_message='Say "HI" to us',
    not_handled_message="Sorry, I don't understand you. Could you repeat?",
    stop_message='stop',
    cancel_message='cancel'
)


class Fulfiller(object):
    def on_post(self, req, resp):
        get_response = alexa_skill.Processor(
            req.media,
            buildin_intents,
            'Welcome to Alexa skill bot',
            'Good bye',
            ExampleIntents(),
        )
        json_response, handled = get_response()

        logging.info('Response was handled by system: {}'.format(handled))

        resp.media = json_response


app = falcon.API(media_type=falcon.MEDIA_JSON)
app.add_route('/v1/alexa/fulfiller', Fulfiller())
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

alexa_skill, 7
alexa_skill.dates, 5
alexa_skill.intents, 4
alexa_skill.intents.base, 3
alexa_skill.intents.buildins, 4
alexa_skill.messages, 6
alexa_skill.tests, 5
alexa_skill.tests.test_alexa_dates, 4
alexa_skill.tests.test_messages, 5

Index

A

alexa_skill (module), 7
alexa_skill.dates (module), 5
alexa_skill.intents (module), 4
alexa_skill.intents.base (module), 3
alexa_skill.intents.buildins (module), 4
alexa_skill.messages (module), 6
alexa_skill.tests (module), 5
alexa_skill.tests.test_alexa_dates (module), 4
alexa_skill.tests.test_messages (module), 5
AmazonDateParser (class in alexa_skill.dates), 5
AmazonTimeParser (class in alexa_skill.dates), 6

B

BaseIntents (class in alexa_skill.intents.base), 3
BuildInIntents (class in alexa_skill.intents.buildins), 4

C

cancel() (alexa_skill.intents.buildins.BuildInIntents method), 4
cards() (in module alexa_skill.messages), 6
confirm_slots_directives() (in module alexa_skill.messages), 6
create_periods() (alexa_skill.dates.AmazonDateParser class method), 5
create_response() (in module alexa_skill.messages), 6

D

DAY_TIME_MAPPER (alexa_skill.dates.AmazonTimeParser attribute), 6

H

help() (alexa_skill.intents.buildins.BuildInIntents method), 4

I

intent_name (alexa_skill.Processor attribute), 7
intent_request() (alexa_skill.Processor method), 7

L

launch_request() (alexa_skill.Processor method), 8
locale (alexa_skill.Processor attribute), 8

M

mapper (alexa_skill.intents.base.BaseIntents attribute), 3
mapper (alexa_skill.intents.buildins.BuildInIntents attribute), 4

N

not_handled() (alexa_skill.intents.buildins.BuildInIntents method), 4

P

Processor (class in alexa_skill), 7

R

reprompt() (in module alexa_skill.messages), 7
request_type (alexa_skill.Processor attribute), 8
response() (alexa_skill.intents.base.BaseIntents static method), 3

S

session_attributes (alexa_skill.Processor attribute), 8
session_end_request() (alexa_skill.Processor method), 8
session_hash (alexa_skill.Processor attribute), 8
slots (alexa_skill.Processor attribute), 8
speech_output() (in module alexa_skill.messages), 7
stop() (alexa_skill.intents.buildins.BuildInIntents method), 4

T

test_amazon_dates_english_week() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_dates_german_week() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_dates_month() (in module alexa_skill.tests.test_alexa_dates), 4

test_amazon_dates_normal_day() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_dates_week() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_dates_weekend() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_dates_year() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_time() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_time_evening() (in module alexa_skill.tests.test_alexa_dates), 4
test_amazon_time_morning() (in module alexa_skill.tests.test_alexa_dates), 5
test_amazon_time_none() (in module alexa_skill.tests.test_alexa_dates), 5
test_amazon_time_parsing_error() (in module alexa_skill.tests.test_alexa_dates), 5
test_cards() (in module alexa_skill.tests.test_messages), 5
test_confirm_slots_directives() (in module alexa_skill.tests.test_messages), 5
test_reprompt_plain_text_default() (in module alexa_skill.tests.test_messages), 5
test_reprompt_ssml() (in module alexa_skill.tests.test_messages), 5
test_speech_output_plain_text() (in module alexa_skill.tests.test_messages), 5
test_speech_output_ssml() (in module alexa_skill.tests.test_messages), 5
to_date() (alexa_skill.dates.AmazonDateParser class method), 5
to_time() (alexa_skill.dates.AmazonTimeParser class method), 6

W

WEEK_MAPPER (alexa_skill.dates.AmazonDateParser attribute), 5