api$_{a}i$*Documentation*

Release 0.1.0

Fulfil.IO Inc.

July 29, 2016

Contents

Contents:

CHAPTER 1

api_ai

API.ai Python client

- Free software: ISC license
- Documentation: https://apiai.readthedocs.org.

1.1 Features

- Make queries
- Create Entities
- Create and update Intents

1.2 Installation

```
pip install api.ai
```

1.3 Quickstart

```
from api.ai import Agent

agent = Agent(
      '<subscription-key>',
      '<client-access-token>',
      '<developer-access-token>',
)
response = agent.query("Hello there")
```

1.4 Credits

- Fulfil.IO Inc

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

At the command line:

```
$ easy_install api.ai
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv api_ai
$ pip install api.ai
```

CHAPTER 3

Usage

To use api_ai in a project:

```
from api.ai import Agent
```

Configure the agent:

```
agent = Agent(
    '<subscription-key>',
    '<client-access-token>',
    '<developer-access-token>',
)
```

Query:

```
agent.query("hello there")
```

3.1 Entities

List all entities:

```
agent.entities.all()
```

Create a new Entity:

```
agent.entities.create(
    'Languages', [
        {'value': 'Python', 'synonyms': ['python', 'py']},
        {'value': 'Golang', 'synonyms': ['go', 'google go language']},
    ]
)
```

Update an entity with new entries:

```
agent.entities.update(
    'entity-id-uuid',
    [
        {'value': 'Python', 'synonyms': ['python', 'py']},
        {'value': 'Golang', 'synonyms': ['go', 'google go language']},
    ],
)
```

3.2 Intents

List all intents:

```
agent.intents.all()
```

Create a new Intent:

```
agent.intents.create(
    'name',
    'templates': [
        "Who am I?"
    ]
)
```

3.3 API Reference

class `api.ai.`**`Agent`**(*sub_key*, *client_token*, *dev_token*, *session=None*, *timezone=None*)
Agent interface for an API.AI agent

`entities`
Return an entities handler object

`intents`
Return an intents handler object

`query`(*query*, *confidence=None*, *lang='en'*, *contexts=None*, *resetContexts=False*, *entities=None*, *timezone=None*, *session=None*)
Use when you want to make a single query.

For multiple queries use multi_query (TODO)

Parameters

- **`query`** – The natural language text to be processed.
- **`confidence`** – The confidence of the corresponding query parameter having been correctly recognized by a speech recognition system. 0 represents no confidence and 1 represents the highest confidence.

class `api.ai.`**`Entities`**(*client*)
An entity is a data type that contains mappings between a set of synonyms

`all`()
return all the records in the endpoint

`create`(*name*, *entries*)
Create a new entity with the name

Parameters

- **`name`** – Name of the entity
- **`entries`** – List of dictionaries

```
        agent.entities.create(
            'EntityName',
            [{
                "value": "Coffee Maker",
                "synonyms": ["coffee maker", "coffee machine"]
```

```
            }, {
                "value": "Thermostat",
                "synonyms": ["Thermostat", "heat", "air conditioning"]
            }, {
                "value": "Lights",
                "synonyms": ["lights", "light", "lamps"]
            }, {
                "value": "Garage door",
                "synonyms": ["garage door", "garage"]
            }]
        )
```

update(*entity_id*, *entries*, *name=None*)
Update an existing entity

class api.ai.**Intents**(*client*)
Intents convert a number of user expressions or patterns into an action.

all()
return all the records in the endpoint

create(*name*, *contexts=None*, *templates=None*, *responses=None*)
For explanation of the params read: https://docs.api.ai/docs/intents#intent-object

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/fulfilio/api_ai/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

api_ai could always use more documentation, whether as part of the official api_ai docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/fulfilio/api_ai/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *api_ai* for local development.

1. Fork the *api_ai* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/api_ai.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv api_ai
$ cd api_ai/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 api_ai tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/fulfilio/api_ai/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_api_ai
```

CHAPTER 5

Credits

5.1 Development Lead

- Fulfil.IO Inc. <hello@fulfil.io>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2016-1-22)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

a

A

C

E

I

Q

U