Persephone Web API Documentation

Release 0.1

Janis Lesinskis

Nov 25, 2018

Contents

1 Installation **3**
- 1.1 Docker . . . 3
- 1.2 Direct install . . . 4

2 API specification **7**
- 2.1 API specification . . . 7
- 2.2 API explorer . . . 7
- 2.3 Code generation . . . 7

3 Support **9**

4 Development **11**
- 4.1 Development environment automation . . . 11

5 Indices and tables **13**

The Persephone Web API provides a REST API to interact with the Persephone automatic phoneme transcription library

The Persephone Web Frontend is an example this API being used.

CHAPTER 1

Installation

We recommend that you use the Docker image for running this project because it handles the system binaries install in a repeatable way.

1.1 Docker

1.1.1 Installing Docker

First make sure that you have docker installed on your machine. Here's some steps to do that on Ubuntu:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
↪$(lsb_release -cs) stable"
sudo apt-get update
```

This will add the docker repository to your package manager lists. Make sure that the package that is the candidate to be installed is the newly added one and not the ubuntu package via this command:

```
apt-cache policy docker-ce
```

If this has worked correctly you'll see an output such as this:

```
docker-ce:
  Installed: (none)
  Candidate: 18.06.1~ce~3-0~ubuntu
  Version table:
 *** 18.06.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages
        100 /var/lib/dpkg/status
     18.06.0~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages
     18.03.1~ce-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages
```

Note that there is no package installed but the candidate for installation is from the Docker repository which is what we want.

Now install Docker with this command:

```
sudo apt-get install -y docker-ce
```

If the install has been successful then you should be able to see the Docker process running. Check with this command

```
sudo systemctl status docker
```

In a successful case the output should look like this:

```
 docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:␣
↪enabled)
   Active: active (running) since Mon 2018-10-15 16:52:50 AEDT; 25min ago
     Docs: https://docs.docker.com
 Main PID: 9841 (dockerd)
    Tasks: 41
   Memory: 52.4M
      CPU: 9.557s
   CGroup: /system.slice/docker.service
           ├─9841 /usr/bin/dockerd -H fd://
           └─9868 docker-containerd --config /var/run/docker/containerd/containerd.
↪toml
```

If you wish to not use *sudo* to invoke Docker, add your user to the Docker group with the following command:

```
sudo usermod -aG docker ${USER}
```

Note that you may have to log out and back in again for this to work

Then verify that the install has been successful via running a real container:

```
docker run hello-world
```

1.1.2 Running this container

To run this project with the docker container you will have to do the following Build the container:

```
docker build -t persephone-web-api:dev .
```

Run it:

```
docker run -p 8080:8080/tcp persephone-web-api:dev
```

If this has succeeded you should be able to access the API at the port you just specified.

1.2 Direct install

The upstream Persephone package requires some binaries to be installed and available on the path. You will need to make sure that these are installed first.

1.2.1 System packages

There are some 3rd party requirements that have to be installed in order for the upstream Persephone library to function. These packages can be found in the file "bootstrap.sh".

Required:

- sox
- ffmpeg

Optional

- Kaldi (for pitch features support)

1.2.2 The web API application

The web API itself is a python application which requires Python 3.5. We have not currently tested this package with other implementations such as PyPy.

Currently you will need to set up a virtualenvironment and install package requirements. The easiest and most reliable way to do this is as follows:

```
pipenv install
```

At this point you should have the packages required to run this API server.

(Note that the Docker image is an automated version of this direct install along with installation of system binaries)

CHAPTER 2

API specification

This project uses the OpenAPI v2 specification to define the API endpoints in an easily parsable YAML file.

2.1 API specification

The API specification is the single source of truth for how this web API will work.

This file is located at persephone_api/api_spec.yaml

The project uses the connexion library to directly create the URL routing for the underlying Flask application that powers the API service. As such the only URL routes that can be assumed to exist are defined in that API specification file.

If there are any discrepancies between what the API specification file states and what actually happens this should be treated as a bug. We would appreciate that you open an issue on the projects issue tracker including some details on the input you gave along with the exact output you got so we can more quickly identify the source of the issue.

2.2 API explorer

Since this uses OpenAPI 2.0 (formerly known as Swagger) API specification we have tooling that will help you to explore the API. This tooling creates and hosts a web frontend that shows you the various API endpoints and provides you forms to test these endpoints from your browser. Load up the API explorer page by navigating to */v0.1/ui/* (Note that the version prefix will depend on the version of the API being served).

If you find yourself needing to construct more complex web requests we would recommend you look into a tool such as Postman for ease of API testing.

2.3 Code generation

By providing a machine-parsable API interface we are aiming to make it easier for developers to consume this API.

The Persephone Web Frontend uses this API specification to generate code. This ensures that the interfaces are correct and also has reduced the time it has taken to create.

If you are interested in code-generation for this API we recommend having a look at openapi-generator

CHAPTER 3

Support

If you find an issue or bug with this code please open an issue on the issues tracker. Please use the discussion mailing list to discuss other questions regarding this project.

CHAPTER 4

Development

If you are working on the project we recommend the development environment with Vagrant as a fast way to get up and running with the same environment as the other developers.

4.1 Development environment automation

There is a Vagrantfile for automating the build and install of the development environment. This is recommended as it is likely the easiest way to get set up with a development environment as packages will be correctly installed.

To get Vagrant: https://www.vagrantup.com/

To start and provisions the vagrant environment:

```
vagrant up
```

Once that has installed you can access via ssh:

```
vagrant ssh
```

The code resides at the */vagrant* directory, set up the environment via pipenv:

```
cd /vagrant
pipenv install
pipenv shell
python transcription_api_server.py
```

If all has worked you should be able to point your browser at 127.0.0.1:8080 and you will see the page being served.

4.1.1 Development server usage

This server uses the Flask framework to service API endpoints.

Make sure you are in the active virtualenvironment and run the transcription server as follows:

```
python3 transcription_API_server.py
```

This will start up a development web server that will service the endpoints defined by the API.

Test that this server is functional by pointing your browser at the URL that pages are being served from.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search