aiodownload Documentation

Michael Copeland

Dec 12, 2018

Contents

Asynchronous Requests and Downloads Without Thinking About It	

CHAPTER 1

Basic Usage

```
>>> import aiodownload
>>> urls = ['https://httpbin.org/links/{}'.format(i) for i in range(0, 5)]
>>> bundles = aiodownload.swarm(urls)
>>>
>>> import pprint
>>> pprint.pprint(dict((b.url, b.file_path, ) for b in bundles))
{'https://httpbin.org/links/0': 'C:\\\\httpbin.org\\links\\0',
 'https://httpbin.org/links/1': 'C:\\\\httpbin.org\\links\\1',
 'https://httpbin.org/links/2': 'C:\\\\httpbin.org\\links\\2',
 'https://httpbin.org/links/3': 'C:\\\\httpbin.org\\links\\3',
 'https://httpbin.org/links/4': 'C:\\\\httpbin.org\\links\\4'}
```

Default Request Strategy (Lenient)

- two concurrent requests with 0.25 s delay between requests
- automatically retry unsuccessful requests up to 4 more times with 60 s between attempts
- response statuses greater than 400 are considered unsuccessful requests
- 404s are not retried (if they tell us it's not found, let's believe them)

Default Download Strategy

- read and write 65536 byte chunks at a time
- uses the current working directory as home path to write files
- relative path and filename are a transformation of the url path segments (the last segment being the filename)

Customizable Strategies

- Want *aiodownload* to behave differently? Configure the underlying classes to create your own strategies.

CHAPTER 2

Installation

```
$ pip install aiodownload
```

CHAPTER 3

Examples

See the *example* package for more basic usages and different ways to configure base objects.

CHAPTER 4

Documentation

4.1 API

Public API Functions

This module contains the public API functions. The goal is to gain the benefits of asynchronous HTTP requests while dropping one of these functions into a synchronous code flow. For basic usage, no prior knowledge of asynchronous programming or asyncio is required. Simply import aiodownload and call the functions with some URLs.

`aiodownload.api.`**`one`**(*url_or_bundle*, *download=None*)
Make one HTTP request and download it

Parameters

- **`url_or_bundle`** (str or `AioDownloadBundle`) – a URL string or bundle
- **`download`** (`AioDownload`) – (optional) your own customized download object

Returns a bundle

Return type `AioDownloadBundle`

`aiodownload.api.`**`swarm`**(*iterable*, *download=None*)
Make a swarm of requests and download them

Parameters

- **`iterable`** (*`iterable object`*) – an iterable object (ex. list of URL strings)
- **`download`** (`AioDownload`) – (optional) your own customized download object

Returns a list of bundles

Return type list

`aiodownload.api.`**`each`**(*iterable*, *url_map=None*, *download=None*)
For each iterable object, map it to a URL and request asynchronously

Parameters

- **iterable** (*iterable object*) – an iterable object (ex. list of objects)
- **url_map** (*callable object*) – (optional) callable object mapping an object to a url or bundle
- **download** (AioDownload) – (optional) your own customized download object

Returns generator

aiodownload API

---

class aiodownload.**AioDownload**(*client=None*, *download_strategy=None*, *request_strategy=None*)

The core class responsible for the coordination of requests and downloads

Parameters

- **client** (aiohttp.ClientSession) – (optional) client session, a default is instantiated if not provided
- **download_strategy** (aiodownload.DownloadStrategy) – (optional) download strategy, a default is instantiated if not provided
- **request_strategy** (aiodownload.RequestStrategy) – (optional) request strategy, a Lenient strategy is instantiated if not provided

main(*bundle*)

Main entry point for task creation with an asyncio event loop.

The number of concurrent requests is throttled using this async method. Depending on the download strategy used, the method will call the request_and_download async method or immediately return the bundle indicating that the file came from cache as the file existed.

Parameters **bundle** (aiodownload.AioDownloadBundle) – bundle (generally one that has just been instantiated)
Returns bundle with updated properties reflecting it's final state
Rtype bundle aiodownload.AioDownloadBundle

request_and_download(*bundle*)

Make an HTTP request and write it to disk. Use the download and request strategies of the instance to implement how this is achieved.

Parameters **bundle** (aiodownload.AioDownloadBundle) – bundle with it's url and file_path set
Returns bundle with updated properties reflecting success or failure
Rtype bundle aiodownload.AioDownloadBundle

Strategies

This module contains base class and sensible default class implementations for the request and download strategies used by AioDownload

---

class aiodownload.strategy.**DownloadStrategy**(*chunk_size=65536*, *home=None*, *skip_cached=False*)

DownloadStrategy is an injection class for AioDownload. The purpose is to control download options for AioDownload.

Parameters

- **chunk_size** (*(optional) int*) – the incremental chunk size to read from the response
- **home** – the base file path to use for writing response content to file
- **skip_cached** (*bool*) – indicates whether existing written files should be skipped

get_file_path(*bundle*)
Get the file path for the bundle
Parameters **bundle** (AioDownloadBundle) – bundle (generally, it's file_path should be None)
Returns full file_path for the bundle (via the default URL transformation)
Return type str

on_fail(*bundle*)
Write an empty file
Parameters **bundle** (AioDownloadBundle) – bundle (file_path should exist)
Returns None

on_success(*response*, *bundle*)
Write the response to the file path indicated in the bundle
Parameters
- **response** (aiohttp.ClientResponse) – successful response from an HTTP response
- **bundle** (AioDownloadBundle) – bundle (file_path should exist)

Returns None

class aiodownload.strategy.**RequestStrategy**(*concurrent=2*, *max_attempts=0*, *timeout=60*)
RequestStrategy is an injection class for AioDownload. The purpose is to control how AioDownload performs requests and retries requests.

Parameters
- **concurrent** (*(optional) int*) – the number of concurrent asynchronous HTTP requests to maintain
- **max_attempts** (*int*) – maximum number of attempts before giving up
- **time_out** (*int*) – timeout for the client session

assert_response(*response*)
Assertion for the response
Parameters **response** (aiohttp.ClientResponse) – response from an HTTP response
Returns None or AssertionError

get_sleep_time(*bundle*)
Returns how much time the bundle should sleep based on it's properties
Parameters **bundle** (AioDownloadBundle) – bundle
Returns sleep time
Return type int

retry(*response*)
Retry the HTTP request based on the response
Parameters **response** (aiohttp.ClientResponse) – response from an HTTP response
Returns
Return type bool

class aiodownload.strategy.**Lenient**(*max_attempts=5*)
Lenient request strategy designed for an average web server. Try five times with a minute between each retry.

get_sleep_time(*bundle*)
Returns how much time the bundle should sleep based on it's properties
Parameters **bundle** (AioDownloadBundle) – bundle
Returns sleep time
Return type int

retry(*response*)
Retry any unsuccessful HTTP response except a 404 (if they say it's not there, let's believe them)

class aiodownload.strategy.**BackOff**(*max_attempts=10*)
Back Off request strategy designed for APIs and web servers that can be hammered a little harder. Exponentially back away if a failed response is returned.

get_sleep_time(*bundle*)
Returns how much time the bundle should sleep based on it's properties
Parameters **bundle** (AioDownloadBundle) – bundle
Returns sleep time
Return type int

4.2 CHANGELOG

4.2.1 v.0.2.5 - 2017-07-13

- Added an example package to illustrate usage
- Added documentation
- Added tests
- Added util module (holds some utility functions for some of the default implementations)

4.2.2 v.0.2.4 - 2017-05-27

- DownloadStrategy class introduced to break up configuration between the "download" and the "request"
- BoundedSemaphore implemented to control number of concurrent requests, removed PriorityQueue implementation
- **AioDownload class reworked to feature two async methods:**
  - main (entry point for task creation)
  - request_and_download (coordinates the logic implemented in RequestStrategy and DownloadStrategy)
- UrlBundle changed to AioDownloadBundle, some property changes to object
- added short hand functions one, swarm, and each to the API

4.2.3 v.0.1.1 - 2016-07-04

- initial working code release
- Python 3.5
- built on top of aiohttp, no other 3rd party dependencies
- core classes / concepts: AioDownload, UrlBundle, RequestStrategy
- **two request strategies:**
  - Lenient (try two times with two two seconds between requests)
  - Backoff (exponential backoff up to 1 min.)
- largely undocumented

CHAPTER 5

Indices and Tables

- genindex
- modindex
- search

CHAPTER 6

Acknowledgements

This library leverages to make requests and manage the HTTP session. *aiodownload* is a lean wrapper designed to abstract the asynchronous programming paradigm and to cut down on the coding of repetitive request functions.

The public function API for this project was adapted from which utilizes and . The motivation for reimplementation was to use the native event loop introduced in Python 3. No monkey patching required.

Python Module Index

a

Index