
edeposit.amqp.downloader

Release 0.3.0

February 06, 2015

1	API	3
1.1	downloader package	3
1.2	downloader.downloader module	6
1.3	downloader.structures package	6
2	AMQP communication	9
2.1	downloader.structures.requests module	9
2.2	downloader.structures.responses module	10
3	Installation	13
4	Source code	15
5	Indices and tables	17
	Python Module Index	19

This module is basically just simple worker, which listens at AMQP queue and asynchronously downloads data from internet. Downloaded data are then returned as AMQP structure.

1.1 downloader package

1.1.1 Subpackages

downloader.structures package

Submodules

downloader.structures.requests module

class `downloader.structures.requests.Download`

Bases: `downloader.structures.requests.Download`

Download data from *url*.

url

str

URL of the internet resource.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.ProgressDownload`

Bases: `downloader.structures.requests.ProgressDownload`

Download data from *url* and report back the progress.

url

str

URL of the internet resource.

steps

int

Number of steps used to track progress.

Progress is reported using `Progress` structure.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.CheckExistence`
Bases: `downloader.structures.requests.CheckExistence`
Send HEAD request to given *url* and check it's existence.

url
str
URL of the internet resource.

Returns `Exists`.

Return type `obj`

downloader.structures.responses module

class `downloader.structures.responses.DownloadedFile`
Bases: `downloader.structures.responses.DownloadedFile`
Response to `Download` and `ProgressDownload`.

url
str
URL of the internet resource.

b64_data
str
Downloaded data encoded as base64 string.

class `downloader.structures.responses.Progress`
Bases: `downloader.structures.responses.Progress`
Response to `ProgressDownload`.

url
str
URL of the internet resource.

step
int
Number of current step.

downloaded
int
How many bytes was downloaded to this step.

content_length
int
How big is the whole file (in bytes).

class `downloader.structures.responses.Exists`
Bases: `downloader.structures.responses.Exists`
Response to `CheckExistence`.

url
str
URL of the internet resource.

result

bool

True if the file exists, False if not.

headers

dict

HTTP headers returned to this requests.

Module contents

1.1.2 Submodules

downloader.downloader module

`downloader.downloader.head_request(url)`

Send HEAD request to given *url*.

Parameters *url* (*str*) – URL of the internet resource.

Returns Dictionary with *headers*. Important headers: *content-length* and *content-type*.

Return type *dict*

`downloader.downloader.download(url)`

Download resource at *url*.

Parameters *url* (*str*) – URL of the internet resource.

Returns Content of the downloaded resource.

Return type *str*

`downloader.downloader.progress_download(url, steps, callback)`

Download resource at *url* and call *callback* after each step. The number of steps is defined by *steps* parameter.

Parameters

- *url* (*str*) – URL of the internet resource.
- *steps* (*int*) – Number of steps (how many times will be *callback* called).
- *callback* (*fn reference*) – Reference to function expecting three arguments: *step* (number of step), *downloaded* (number of downloaded bytes), *content_len* (size of downloaded resource).

1.1.3 Module contents

`downloader.get_progress_reporter(send_back)`

Construct progress reporter callback from *send_back* function.

Parameters *send_back* (*fn reference*) – Reference to function for sending messages back using AMQP.

Returns *fn reference*: Function taking 3 parameters as is required by `progress_download()`.

`downloader.react_to_amqp_message(message, send_back)`

React to given (AMQP) message. *message* is usually expected to be `collections.namedtuple()` structure filled with all necessary data.

Parameters

- **message** (**Request class*) – only `ConversionRequest` class is supported right now
- **send_back** (*fn reference*) – Reference to function for responding. This is useful for progress monitoring for example. Function takes one parameter, which may be response structure/namedtuple, or string or whatever would be normally returned.

Returns response filled with data about conversion and converted file.

Return type `ConversionResponse`

Raises `ValueError` – if bad type of *message* structure is given.

1.2 downloader.downloader module

`downloader.downloader.head_request(url)`

Send HEAD request to given *url*.

Parameters *url* (*str*) – URL of the internet resource.

Returns Dictionary with *headers*. Important headers: *content-length* and *content-type*.

Return type `dict`

`downloader.downloader.download(url)`

Download resource at *url*.

Parameters *url* (*str*) – URL of the internet resource.

Returns Content of the downloaded resource.

Return type `str`

`downloader.downloader.progress_download(url, steps, callback)`

Download resource at *url* and call *callback* after each step. The number of steps is defined by *steps* parameter.

Parameters

- **url** (*str*) – URL of the internet resource.
- **steps** (*int*) – Number of steps (how many times will be *callback* called).
- **callback** (*fn reference*) – Reference to function expecting three arguments: *step* (number of step), *downloaded* (number of downloaded bytes), *content_len* (size of downloaded resource).

1.3 downloader.structures package

1.3.1 Submodules

downloader.structures.requests module

class `downloader.structures.requests.Download`

Bases: `downloader.structures.requests.Download`

Download data from *url*.

url

str

URL of the internet resource.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.ProgressDownload`

Bases: `downloader.structures.requests.ProgressDownload`

Download data from *url* and report back the progress.

url

str

URL of the internet resource.

steps

int

Number of steps used to track progress.

Progress is reported using `Progress` structure.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.CheckExistence`

Bases: `downloader.structures.requests.CheckExistence`

Send HEAD request to given *url* and check it's existence.

url

str

URL of the internet resource.

Returns `Exists`.

Return type `obj`

downloader.structures.responses module

class `downloader.structures.responses.DownloadedFile`

Bases: `downloader.structures.responses.DownloadedFile`

Response to `Download` and `ProgressDownload`.

url

str

URL of the internet resource.

b64_data

str

Downloaded data encoded as base64 string.

```
class downloader.structures.responses.Progress
    Bases: downloader.structures.responses.Progress
    Response to ProgressDownload.

    url
        str
        URL of the internet resource.

    step
        int
        Number of current step.

    downloaded
        int
        How many bytes was downloaded to this step.

    content_length
        int
        How big is the whole file (in bytes).

class downloader.structures.responses.Exists
    Bases: downloader.structures.responses.Exists
    Response to CheckExistence.

    url
        str
        URL of the internet resource.

    result
        bool
        True if the file exists, False if not.

    headers
        dict
        HTTP headers returned to this requests.
```

1.3.2 Module contents

AMQP communication

2.1 `downloader.structures.requests` module

class `downloader.structures.requests.Download`

Bases: `downloader.structures.requests.Download`

Download data from *url*.

url

str

URL of the internet resource.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.ProgressDownload`

Bases: `downloader.structures.requests.ProgressDownload`

Download data from *url* and report back the progress.

url

str

URL of the internet resource.

steps

int

Number of steps used to track progress.

Progress is reported using `Progress` structure.

Returns `DownloadedFile`.

Return type `obj`

class `downloader.structures.requests.CheckExistence`

Bases: `downloader.structures.requests.CheckExistence`

Send HEAD request to given *url* and check it's existence.

url

str

URL of the internet resource.

Returns `Exists`.

Return type `obj`

2.2 downloader.structures.responses module

class `downloader.structures.responses.DownloadedFile`
 Bases: `downloader.structures.responses.DownloadedFile`
 Response to `Download` and `ProgressDownload`.

url
str
 URL of the internet resource.

b64_data
str
 Downloaded data encoded as base64 string.

class `downloader.structures.responses.Progress`
 Bases: `downloader.structures.responses.Progress`
 Response to `ProgressDownload`.

url
str
 URL of the internet resource.

step
int
 Number of current step.

downloaded
int
 How many bytes was downloaded to this step.

content_length
int
 How big is the whole file (in bytes).

class `downloader.structures.responses.Exists`
 Bases: `downloader.structures.responses.Exists`
 Response to `CheckExistence`.

url
str
 URL of the internet resource.

result
bool
 True if the file exists, False if not.

headers

dict

HTTP headers returned to this requests.

Installation

Installation at debian systems is really easy:

```
pip install edeposit.amqp.downloader
```

Source code

This project is released as opensource (GPL) and source codes can be found at GitHub:

- <https://github.com/edeposit/edeposit.amqp.downloader>

Indices and tables

- *genindex*
- *modindex*
- *search*

d

- `downloader`, [5](#)
- `downloader.downloader`, [6](#)
- `downloader.structures`, [8](#)
- `downloader.structures.requests`, [9](#)
- `downloader.structures.responses`, [10](#)

B

b64_data (downloader.structures.responses.DownloadedFile attribute), 4, 7, 10

C

CheckExistence (class in downloader.structures.requests), 3, 7, 9

content_length (downloader.structures.responses.Progress attribute), 4, 8, 10

D

Download (class in downloader.structures.requests), 3, 6, 9

download() (in module downloader.downloader), 5, 6

downloaded (downloader.structures.responses.Progress attribute), 4, 8, 10

DownloadedFile (class in downloader.structures.responses), 4, 7, 10

downloader (module), 5

downloader.downloader (module), 5, 6

downloader.structures (module), 5, 8

downloader.structures.requests (module), 3, 6, 9

downloader.structures.responses (module), 4, 7, 10

E

Exists (class in downloader.structures.responses), 4, 8, 10

G

get_progress_reporter() (in module downloader), 5

H

head_request() (in module downloader.downloader), 5, 6

headers (downloader.structures.responses.Exists attribute), 5, 8, 10

P

Progress (class in downloader.structures.responses), 4, 7, 10

progress_download() (in module downloader.downloader), 5, 6

ProgressDownload (class in downloader.structures.requests), 3, 7, 9

R

reactToAMQPMessage() (in module downloader), 5

result (downloader.structures.responses.Exists attribute), 4, 8, 10

S

step (downloader.structures.responses.Progress attribute), 4, 8, 10

steps (downloader.structures.requests.ProgressDownload attribute), 3, 7, 9

U

url (downloader.structures.requests.CheckExistence attribute), 4, 7, 9

url (downloader.structures.requests.Download attribute), 3, 6, 9

url (downloader.structures.requests.ProgressDownload attribute), 3, 7, 9

url (downloader.structures.responses.DownloadedFile attribute), 4, 7, 10

url (downloader.structures.responses.Exists attribute), 4, 8, 10

url (downloader.structures.responses.Progress attribute), 4, 8, 10