AsyncSender Documentation

Bakhtiyor Ruziev

Feb 23, 2023

Contents

CHAPTER 1

API

class async_sender.api.**Attachment**(*filename: str = None*, *content_type: str = None*, *data=None*, *disposition: str = 'attachment'*, *headers: dict = None*)

File attachment information.

Parameters

- **filename** – filename
- **content_type** – file mimetype
- **data** – raw data
- **disposition** – content-disposition, default to be 'attachment'
- **headers** – a dictionary of headers, default to be {}

class async_sender.api.**Connection**(*mail*)

This class handles connection to the SMTP server. Instance of this class would be one context manager so that you do not have to manage connection close manually.

Parameters **mail** – one mail instance

send(*message: async_sender.api.Message*)

Send one message instance.

Parameters **message** – one message instance.

class async_sender.api.**Mail**(*hostname: str = ''*, *port: int = None*, *use_tls: bool = False*, *use_starttls: bool = False*, *username: str = None*, *password: str = None*, *from_address: str = None*, *timeout: Union[int, float] = None*, *source_address: str = None*, *validate_certs: bool = True*, *client_cert: str = None*, *client_key: str = None*, *tls_context: ssl.SSLContext = None*, *cert_bundle: str = None*)

AsyncSender Mail main class. This class is used for manage SMTP server connections and send messages.

Parameters

- **hostname** – Server name (or IP) to connect to
- **port** – Server port. Defaults to `25` if `use_tls` is `False`, `465` if `use_tls` is `True`.

- **source_address** – The hostname of the client. Defaults to the result of `socket.getfqdn()`. Note that this call blocks.
- **timeout** – Default timeout value for the connection, in seconds. Defaults to 60.
- **use_tls** – If True, make the initial connection to the server over TLS/SSL. Note that if the server supports STARTTLS only, this should be False.
- **use_starttls** – If True, make the initial connection without encrypt to the server

over TCP and upgrade plain connection to an encrypted (TLS or SSL) connection. :param validate_certs: Determines if server certificates are

validated. Defaults to True.

Parameters

- **client_cert** – Path to client side certificate, for TLS verification.
- **client_key** – Path to client side key, for TLS verification.
- **tls_context** – An existing `ssl.SSLContext`, for TLS verification. Mutually exclusive with `client_cert`/ `client_key`.
- **cert_bundle** – Path to certificate bundle, for TLS verification.

connection
Open one connection to the SMTP server.

send(**messages*)
Sends a single or multiple messages.

Parameters **messages** – Message instance.

send_message(**args*, ***kwargs*)
Shortcut for send.

class async_sender.api.**Message**(*subject: str = None*, *to: Union[str, Iterable[T_co]] = None*, *body: str = None*, *html: str = None*, *from_address: Union[str, Iterable[T_co]] = None*, *cc: Union[str, Iterable[T_co]] = None*, *bcc: Union[str, Iterable[T_co]] = None*, *attachments: Union[Attachment, Sequence[Attachment]] = None*, *reply_to: Union[str, Iterable[T_co]] = None*, *date: Optional[int] = None*, *charset: str = 'utf-8'*, *extra_headers: dict = None*, *mail_options: list = None*, *rcpt_options: list = None*)
One email message.

Parameters

- **subject** – message subject
- **to** – message recipient, should be one or a list of addresses
- **body** – plain text content body
- **html** – HTML content body
- **from_address** – message sender, can be one address or a two-element tuple
- **cc** – CC list, should be one or a list of addresses
- **bcc** – BCC list, should be one or a list of addresses
- **attachments** – a list of attachment instances
- **reply_to** – reply-to address

- **date** – message send date, seconds since the Epoch, default to be time.time()
- **charset** – message charset, default to be 'utf-8'
- **extra_headers** – a dictionary of extra headers
- **mail_options** – a list of ESMTP options used in MAIL FROM commands
- **rcpt_options** – a list of ESMTP options used in RCPT commands

as_bytes() → bytes

as_string() → str
The message string.

attach(**attachment*)
Adds one or a list of attachments to the message.

Parameters **attachment** – Attachment instance.

attach_attachment(**args*, ***kwargs*)
Shortcut for attach.

to_address

validate()
Do email message validation.

exception async_sender.api.**SenderError**

CHAPTER 2

Changelog

2.1 2.0.0

- BREAKING: update dependency aionsmtplib==2.0.1
- BREAKING: drop python 3.6, 3.7 support
- Bugfix: subject charset issue #228
- Changes: move to github action

2.2 1.4.4

- bump aiosmtplib

AsyncSender provides a simple interface to set up a SMTP connection and send email messages asynchronously.

CHAPTER 3

Installation

Install with the following command:

```
$ pip install async_sender
```

CHAPTER 4

Quickstart

AsyncSender is really easy to use. Emails are managed through a `Mail` instance:

```
from async_sender import Mail
import asyncio


async def run():
    mail = Mail()

    await mail.send_message("Hello", from_address="from@example.com",
                            to="to@example.com", body="Hello world!")


asyncio.run(run())
```

CHAPTER 5

Message

To send one message, we need to create a `Message` instance:

```
from async_sender import Message

msg = Message("demo subject", from_address="from@example.com",
              to="to@example.com")
```

Python Module Index

a

Index