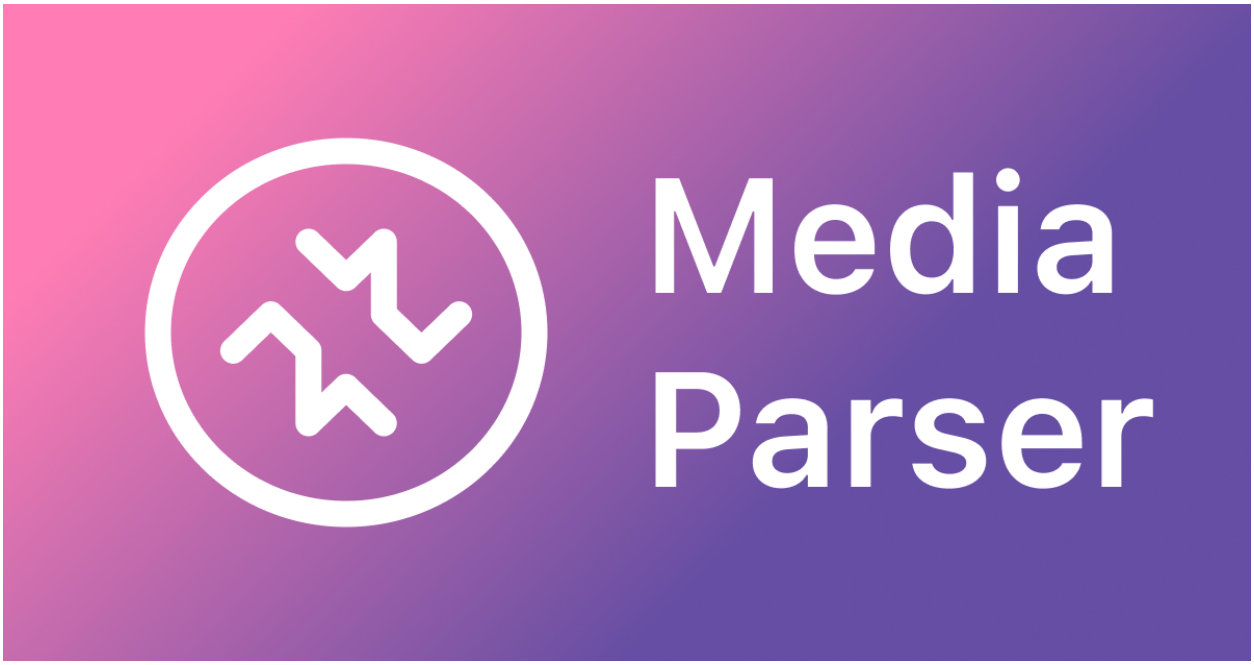
Media Parser

Release 1.1.3

Jag_k

Aug 09, 2023

CONTENTS



Server for parse Media by URL.

CHAPTER

ONE

SUPPORTED MEDIAS

- Youtube
- Tiktok
- Instagram
- Twitter
- Reddit
- Pinterest

CHAPTER

TWO

INSTALLATION AND CONFIGURATION SERVER

Use the `docker-compose.yml` file to run the server.

```yaml
version: "3.8"

service:
    media-parser:
        image: ghcr.io/jag-k/media-parser:latest
        ports:
            - 8000:8000
        environment:
            # Sentry integration (optional)
            SENTRY_DSN: "https://abcabc@sentry.io/2"
            SENTRY_ENVIRONMENT: "dev"

            # Enable sentry user feedback (optional)
            SENTRY_ORGANISATION_SLUG: "sentry"
            SENTRY_PROJECT_SLUG: "media-parser"
            SENTRY_AUTH_TOKEN: "..."  # with scope project:write
            SENTRY_API_HOST: "https://api.sentry.io/"

            # Database
            MONGO_URL: "mongodb://mongodb:27017"
            MONGO_DATABASE: "test"

        volumes:
            - ./config:/config

    mongodb:
        image: mongo:latest
        volumes:
            - ./data:/data/db
```

2.1 Parsers Configuration

All configs for parsers stored in `config/parsers.json`. JSON Schema for this: schemas/parser_schema.json.

To enable parser, you need to add config for this parser. If parser hasn't config, like `tiktok` set an empty object (`{}`) to enable it.

Example:

```
// config/parsers.json
{
    "$schema": "https://raw.github.com/jag-k/media-parsers/blob/main/schemas/parser_
↪schema.json",
    "instagram": {
        // Optional
        "lamadava_saas_token": "asdasd"
    },
    "reddit": {
        "client_id": "",
        "client_secret": "",
        // Optional
        "user_agent": "video downloader (by u/Jag_k)"
    },
    "tiktok": {},
    "twitter": {
        "twitter_bearer_token": "asdasd"
    },
    "youtube": {}
}
```

Or you can use YAML file like `config/parsers.yaml` or `config/parsers.yml`:

```
# config/parsers.yml
$schema: "https://raw.github.com/jag-k/media-parsers/blob/main/schemas/parser_schema.json
↪"
instagram:
    # Optional
    lamadava_saas_token: "asdasd"
reddit:
    client_id: ""
    client_secret: ""
    # Optional
    user_agent: "video downloader (by u/Jag_k)"
tiktok: {}
twitter:
    twitter_bearer_token: "asdasd"
youtube: {}
```

CHAPTER

THREE

USAGE

API documentation available on `/docs` endpoint.

CHAPTER

FOUR

CLIENTS

4.1 Installation

```
poetry add media-parser  # or pip install media-parser
```

4.2 Usage

```python
from media_parser import Client, FeedbackTypes

client = Client(url="http://localhost:8000")


async def main():
    # Get all media
    media = await client.parse("https://www.youtube.com/watch?v=9bZkp7q19f0", user="jag-k")
    print(media)

    # If media is incorrect, you can send feedback
    await client.send_feedback(media, "jag-k", FeedbackTypes.wrong_media)


if __name__ == '__main__':
    import asyncio

    asyncio.run(main())
```

CHAPTER

FIVE

LICENSE

MIT

5.1 Usage

API documentation available on `/docs` endpoint.

5.1.1 Content

Client Usage

Installation

The easiest way to install the package:

```
pip install media-parser
```

```
pipx install media-parser
```

```
poetry add media-parser
```

Code examples

```python
from media_parser import Client, FeedbackTypes

client = Client(url="http://localhost:8000")


async def main():
    # Get all media
    media = await client.parse("https://youtu.be/dQw4w9WgXcQ", user="jag-k")
    print(media)

    # If media is incorrect, you can send feedback
    await client.send_feedback(media, "jag-k", FeedbackTypes.wrong_media)

```

(continues on next page)

(continued from previous page)

```python

if __name__ == '__main__':
    import asyncio

    asyncio.run(main())
```

For more info check *media_parser.client.Client*

Server Usage

Installation

You can start server choosing one of the following options:

Requirements

- Docker
- Docker Compose

This is example of `docker-compose.yml` file:

```yaml
version: '3.8'

services:
  app:
    image: "ghcr.io/jag-k/media-parser:latest"
    container_name: "media_parser"
    ports:
      - "8000:8000"
    environment:
      # Sentry integration (optional)
      SENTRY_DSN: "https://asdasda@sentry.io/2"
      SENTRY_ENVIRONMENT: "dev"

      # Enable sentry user feedback (optional)
      SENTRY_ORGANISATION_SLUG: "sentry"
      SENTRY_PROJECT_SLUG: "media-parser"
      SENTRY_AUTH_TOKEN: "asdasdasd"  # with scope project:write
      SENTRY_API_HOST: "https://sentry.io/"

      # Database
      MONGO_URL: "mongodb://user:password@mongodb:27017"
      MONGO_DATABASE: "media-parser"

    depends_on:
      - mongodb

    volumes:
      - ./config:/config
```

(continues on next page)

(continued from previous page)

```
  mongodb:
    image: mongo:latest
    container_name: mongodb
    environment:
      MONGO_INITDB_ROOT_USERNAME: "user"
      MONGO_INITDB_ROOT_PASSWORD: "password"
      MONGO_INITDB_DATABASE: "media-parser"
    volumes:
      - ./config/mongo:/data/db

    ports:
      - "27017:27017"
```

Requirements

- Docker
- MongoDB - You can also use MongoDB Docker image

```
docker run -d \
  --name media_parser \
  -p 8000:8000 \
  -e SENTRY_DSN="https://asdasda@sentry.io/2" \  # optional
  -e SENTRY_ENVIRONMENT="dev" \  # optional
  -e SENTRY_ORGANISATION_SLUG="sentry" \  # optional
  -e SENTRY_PROJECT_SLUG="media-parser" \  # optional
  -e SENTRY_AUTH_TOKEN="asdasdasd" \  # optional
  -e SENTRY_API_HOST="https://sentry.io/" \  # optional
  -e MONGO_URL="mongodb://user:password@mongodb:27017" \
  -e MONGO_DATABASE="media-parser" \
  -v ./config:/config \
  ghcr.io/jag-k/media-parser:latest
```

Example of MongoDB Docker image:

```
docker run -d \
  --name mongodb \
  -e MONGO_INITDB_ROOT_USERNAME="user" \
  -e MONGO_INITDB_ROOT_PASSWORD="password" \
  -e MONGO_INITDB_DATABASE="media-parser" \
  -v ./config/mongo:/data/db \
  -p 27017:27017 \
  mongo:latest
```

Requirements

- Python 3.11
- Poetry - For installing dependencies
- MongoDB - You can also use MongoDB Docker image

Clone the repository:

```
git clone https://github.com/jag-k/media-parser.git
```

Install dependencies:

```
poetry install --no-root
```

Run server:

```
cd media-parser && poetry run uvicorn media_parser.main:app
```

After that, you can open http://localhost:8000/docs in your browser.

Configuration

For run server you need to set environment variables and parser config files.

Env Variables

Name	Description	Required
`MONGO_URL`	MongoDB URL	`True`
`MONGO_DATABASE`	MongoDB database name	`True`
`SENTRY_DSN`	Sentry DSN	`False`
`SENTRY_ENVIRONMENT`	Sentry environment	`False`
`SENTRY_ORGANISATION_SLUG`	Sentry organization slug	`False`
`SENTRY_PROJECT_SLUG`	Sentry project slug	`False`
`SENTRY_AUTH_TOKEN`	Sentry auth token	`False`
`SENTRY_API_HOST`	Sentry API host	`False`

Parsers Config

To configure parsers, you need to create a config file. JSON Schema for config file: schemas/parser_schema.json.

To enable parser, you need to add config for this parser. If parser hasn't config, like `tiktok` set an empty object (`{}`) to enable it.

Example of config file:

`config/parsers.yaml` or `config/parsers.yml`

```yaml
# config/parsers.yml
$schema: "https://raw.github.com/jag-k/media-parsers/blob/main/schemas/parser_schema.json
↪"
instagram:
    # Optional
    lamadava_saas_token: "asdasd"
reddit:
    client_id: ""
    client_secret: ""
    # Optional
```

(continues on next page)

(continued from previous page)

```
    user_agent: "video downloader (by u/Jag_k)"
tiktok: {}
twitter:
    twitter_bearer_token: "asdasd"
youtube: {}
```

config/parsers.json

```
{
    "$schema": "https://raw.github.com/jag-k/media-parsers/blob/main/schemas/parser_
↪schema.json",
    "instagram": {
        // Optional
        "lamadava_saas_token": "asdasd"
    },
    "reddit": {
        "client_id": "",
        "client_secret": "",
        // Optional
        "user_agent": "video downloader (by u/Jag_k)"
    },
    "tiktok": {},
    "twitter": {
        "twitter_bearer_token": "asdasd"
    },
    "youtube": {}
}
```

You can find more information about parsers props config in *parsers config docs*.

Parsers Configuration

Instagram

Parser for Instagram

Key: `instagram`

Properties:

`lamadava_saas_token`

Set this for enable lamadava proxy

type: `str`

`user_agent`

Set this to change user agent

type: `str`

default: `'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36'`

Reddit

Parser for Reddit

Key: `reddit`

Properties:

`user_agent`

User agent for Reddit API

type: `str`

default: `'video downloader (by u/Jag_k)'`

`client_id`

This field is required to enable `reddit` parser!

Client ID for Reddit API

type: `str`

`client_secret`

This field is required to enable `reddit` parser!

Client secret for Reddit API

type: `str`

TikTok

Parser for TikTok

Key: `tiktok`

Properties:

Does not have any properties.

Twitter

Parser for Twitter

Key: `twitter`

Properties:

`twitter_bearer_token`

This field is required to enable `twitter` parser!

Bearer token for Twitter API

type: `str`

YouTube

Parser for YouTube

Key: `youtube`

Properties:

Does not have any properties.

5.2 API Reference

This page contains auto-generated API reference documentation[1].

5.2.1 media_parser.models

Submodules

media_parser.models.medias

Module Contents

Classes

ParserType	BaseParser types. Using for identify parsers and .
Media	Base class for all medias.
Video	Video media.
Image	Image media.
Audio	Audio media.
GroupedMedia	

API

class media_parser.models.medias.**ParserType**

Bases: `str`, `enum.Enum`

BaseParser types. Using for identify parsers and .

[1] Created with sphinx-autodoc2

Initialization

Initialize self. See help(type(self)) for accurate signature.

TIKTOK

'TikTok'

TWITTER

'Twitter'

YOUTUBE

'YouTube'

REDDIT

'Reddit'

INSTAGRAM

'Instagram'

class media_parser.models.medias.**Media**

Bases: pydantic.BaseModel

Base class for all medias.

Parameters

- **type** – Type source of media (TikTok, Twitter, YouTube, Reddit, Instagram)
- **original_url** – Original URL of media
- **caption** – Caption of media
- **thumbnail_url** – URL to thumbnail
- **author** – Author of media
- **extra_description** – Extra description
- **language** – Language of media
- **mime_type** – MIME type of media

Example extra_description
"Video from YouTube"

Example language
"en"

Example mime_type
"video/mp4"

type: *media_parser.models.medias.ParserType*

None

original_url: **str**

None

caption: **str | None**

None

thumbnail_url: **str | None**

None

author: str | None

None

extra_description: str = <Multiline-String>

language: str | None

None

mime_type: str | None

None

class media_parser.models.medias.**Video**

Bases: *media_parser.models.medias.Media*

Video media.

Parameters

- **url** – URL to video
- **max_quality_url** – URL to max quality video
- **audio_url** – URL to audio from video
- **height** – Height of video
- **width** – Width of video
- **duration** – Duration of video

Info max_quality_url
If max quality video is not available, max_quality_url is equal to url

Info audio_url
Is it necessary?

url: str = <Multiline-String>

max_quality_url: str | None

None

audio_url: str | None

None

mime_type: str

'video/mp4'

height: int | None

None

width: int | None

None

duration: int | None

None

class media_parser.models.medias.**Image**

Bases: *media_parser.models.medias.Media*

Image media.

Parameters

- **url** – URL to image
- **max_quality_url** – URL to max quality image
- **height** – Height of image
- **width** – Width of image

Info max_quality_url
If max quality image is not available, max_quality_url is equal to url

url: str
None

max_quality_url: str | None
None

mime_type: str
'image/jpeg'

height: int | None
None

width: int | None
None

class media_parser.models.medias.**Audio**
Bases: *media_parser.models.medias.Media*

Audio media.

Parameters

- **url** – URL to audio
- **mime_type** – MIME type of audio

url: str = <Multiline-String>

mime_type: str
'audio/mpeg'

class media_parser.models.medias.**GroupedMedia**
Bases: pydantic.BaseModel

audios: list[*media_parser.models.medias.Audio*]
None

images: list[*media_parser.models.medias.Image*]
None

videos: list[*media_parser.models.medias.Video*]
None

classmethod from_medias(*medias: list[*media_parser.models.medias.Media*]*) → Self
Generate GroupedMedia from list of Media.

Parameters
medias – list of Media

Returns
GroupedMedia

flat() → list[*media_parser.models.medias.Media*]

Makes list of Media from GroupedMedia.

media_parser.models.server

Module Contents

Classes

StatusResponse
ParserStatus
ParserStatusResponse
ParseRequest
FeedbackTypes
FeedbackRequest
FeedbackResponse

API

class media_parser.models.server.**StatusResponse**

Bases: pydantic.BaseModel

status: str

'ok'

class media_parser.models.server.**ParserStatus**

Bases: pydantic.BaseModel

parser_type: *media_parser.models.medias.ParserType*

None

enabled: bool

False

class media_parser.models.server.**ParserStatusResponse**

Bases: pydantic.BaseModel

statuses: list[*media_parser.models.server.ParserStatus*]

None

class media_parser.models.server.**ParseRequest**

Bases: pydantic.BaseModel

url: pydantic.HttpUrl

None

use_cache: bool

None

class media_parser.models.server.**FeedbackTypes**

Bases: str, enum.Enum

not_found

'not_found'

wrong_media

'wrong_media'

other

'other'

class media_parser.models.server.**FeedbackRequest**

Bases: pydantic.BaseModel

event_id: str

None

username: str

None

feedback_type: *media_parser.models.server.FeedbackTypes*

None

request_url: str | None

None

class media_parser.models.server.**FeedbackResponse**

Bases: pydantic.BaseModel

status: Literal[ok, error]

'ok'

sentry_status: bool

None

5.2.2 media_parser.client

Module Contents

Classes

Client	Client for media-parser

API

class media_parser.client.**Client**

Bases: pydantic.BaseModel

Client for media-parser

Parameters

- **url** – URL to media-parser
- **service** – Service name (used for feedback)

url: `pydantic.HttpUrl`

None

service: `str | None`

None

property client

async parse(*url: str*, *use_cache: bool = True*, *user: str | None = None*) → *models.medias.GroupedMedia*

Parse media from url

Parameters

- **url** – URL to parse
- **use_cache** – Use cache on server
- **user** – Username (used for feedback)

Returns

GroupedMedia

async send_feedback(*media:* models.medias.GroupedMedia, *user: str*, *feedback_type: models.FeedbackTypes*)

Send feedback to media-parser

Its work only if sentry enabled

Parameters

- **media** – Media
- **user** – Username
- **feedback_type** – Feedback type

PYTHON MODULE INDEX

m

INDEX