

# 目次

---

- 目次
- インスタグラムAPIを使用してフォトブックを作成しよう
  - APIリクエストの基本
    - インスタグラムAPIの概要
    - APIリクエストを使用して、インスタグラムからデータを取得
    - レートリミットの管理
- 演習
  - 演習1: フロントエンド (Webページの作成)
  - 演習2: バックエンド (pythonを使ったAPIの呼び出し)
  - 演習3: フルスタック(フォトブックの作成)

# Instagram APIを使用してフォトブックを作成しよう

---

## APIリクエストの基本

### Instagram APIの概要

Instagram APIを使用すると、Instagramのデータにアクセスし、独自のアプリケーションやサービスに統合することができます。

#### 主な機能

- ユーザーのプロフィールデータの取得
  - ユーザーIDを使用して、そのユーザーのプロフィール情報をリクエストします。
  - 例: GET /users/{user-id}
- 投稿された写真やビデオの取得
  - ユーザーIDまたはハッシュタグを使用して、関連するメディアをリクエストします。
  - 例: GET /users/{user-id}/media/recent
- ユーザーのアクティビティ情報の取得
  - ユーザーIDを使用して、そのユーザーの最近のアクティビティをリクエストします。
  - 例: GET /users/{user-id}/activities

詳しくは [https://developers.facebook.com/docs/instagram-api?locale=ja\\_JP](https://developers.facebook.com/docs/instagram-api?locale=ja_JP) を参照

### APIリクエストを使用して、Instagramからデータを取得

- エンドポイントに対してHTTPリクエストを行う
  - APIのエンドポイントURLに対してGETまたはPOSTリクエストを行います。
- 必要なパラメータをリクエストに含める
  - リクエストに必要なパラメータ（例: user-id）を追加。
- アクセストークンを使用してリクエストを認証する
  - リクエストヘッダーまたはクエリパラメータにアクセストークンを含める。

### レートリミットの管理

Instagram APIにはリクエストの呼び出す際の制限があります。これをレートリミットと呼びます。

- レートリミットによる制限の条件の一例
  - 特定の回数以上リクエストを発行したら制限をかける
  - 特定の時間内に特定の回数以上リクエストを発行したら制限をかける
- レートリミットによる制限の一例

- 24時間通信禁止
- トークンの利用を1週間停止
- アカウント停止

外部APIを利用してサービスを開発する際は、このレートリミットを考慮しましょう。

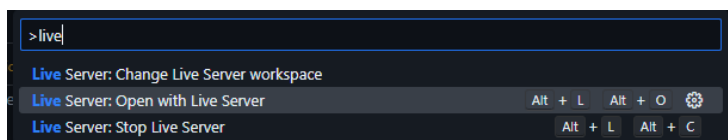
詳しい制限の内容は <https://developers.facebook.com/docs/graph-api/overview/rate-limiting/#instagram-graph-api> を参照

# 演習

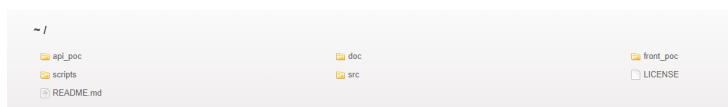
Instagram APIを使用して、簡単なアプリケーションを作成します。別紙の開発環境構築手順に沿ってgithub codespacesに接続してください。

## 演習1: フロントエンド（Webページの作成）

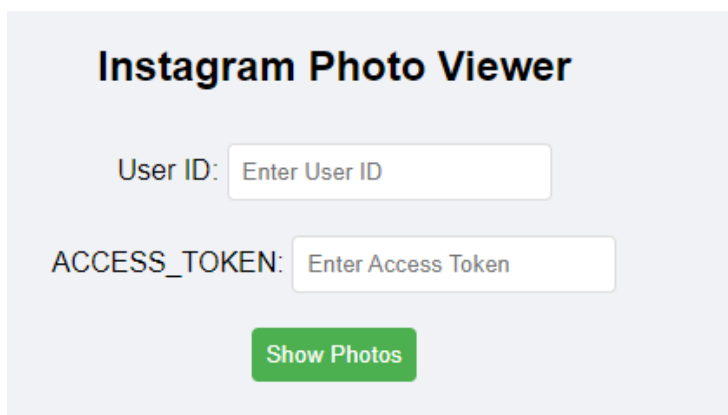
- 要件：最近の投稿を取得してウェブページ上に写真を表示してみよう
  - IDを使用して、最近の投稿をリクエスト
  - 取得した画像データを使用して、ウェブページ上に写真を表示
- 要素技術: HTML/CSS/JavaScript
- 完成形のウェブページの確認方法
  - `ctrl + shift + p` を押して, `live` と入力. `Open With Live Server` をクリック



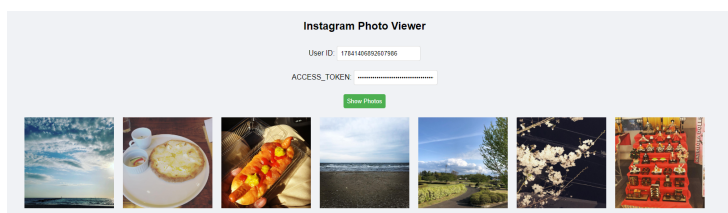
- フォルダの一覧が表示されるので, 「front\_poc > instagram\_photo\_viewer.html」 を選択



- InstagramPhotoViewerが表示される



- UserID と ACCESS\_TOKEN を入力して「Show Photos」をクリックして自分の投稿した写真が表示されればOKです.



- 課題
  - 「front\_poc/app\_exercise.js」 の課題を解こう(TODOコメントの記載箇所)

- 動作確認する場合は「front\_poc > instagram\_photo\_viewer.html」で呼び出すJSを「app.js」から「app\_exercise.js」に変更してください。
- 解答例
  - 「app.js」が解答例です。
  - 動作確認する場合は「front\_poc > instagram\_photo\_viewer.html」で呼び出すJSを「app\_exercise.js」から「app.js」に変更してください。

## 演習2: バックエンド (pythonを使ったAPIの呼び出し)

- 要件 : Pythonを使ってインスタグラムAPIを呼び出してみよう
  - トークンを利用してIDとサービス名を取得
- 要素技術: Python
- 環境構築
  - `Ctrl+@` でターミナルを呼び出す
  - `cd api_poc` -> `pip3 install -r requirements.txt` コマンドを実行
- pythonコードの実行方法
  - 「api\_poc/.env」を開いて `INSTAGRAM_ACCESS_TOKEN=HOGE` のHOGE を自分のTOKENに置き換えてください
  - python の動作確認方法
    - `python api_fetch.py` を実行します。
    - 下記のように id, name 情報が取得できます。

```
[  
$ python api_fetch.py  
{'id': '139910752530372', 'name': 'GramConnect Central'}  
$  
]
```

- 課題
  - `api_poc/api_fetch_exercise.py` の課題を解こう(TODOコメントの記載箇所)
    - 下記のように id, name 情報が取得できます。
- 解答例
  - 「api\_fetch.py」が解答例です。

## 演習3: フルスタック(フォトブックの作成)

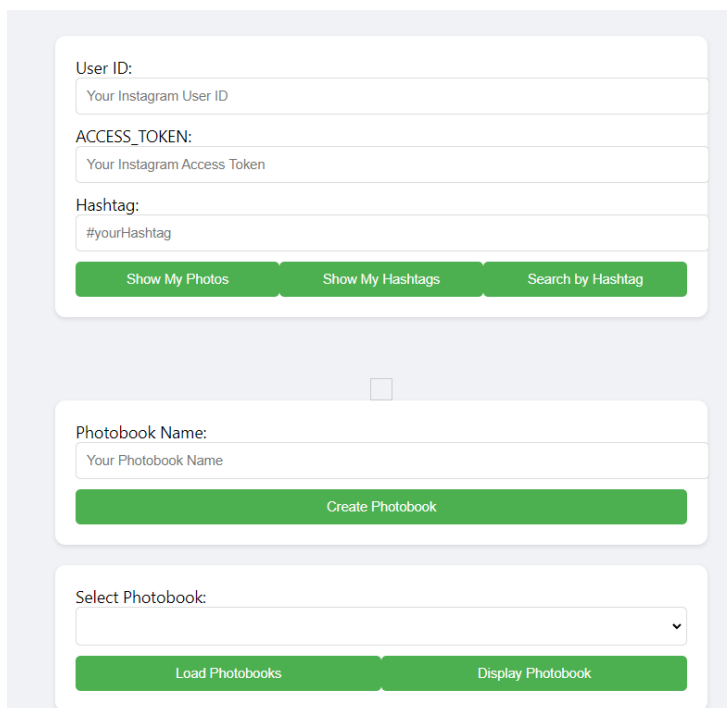
- 要件 : 演習1,演習2を組み合わせるフォトブックを作成するサービスを構築しよう。
  - IDを使用して、最近の投稿をリクエスト
  - 取得した画像データを使用して、ウェブページ上にギャラリーを表示
- 要素技術: HTML/CSS/JavaScript, Python, DB, Docker
  - フォトブックを作成するサービスの起動方法
    - ターミナルを開いて `src` ディレクトリに移動して下記のコマンドを実行してください
    - コマンドが完了するまで1分程度かかります。

```
[  
  
    docker-compose up --build  
  
]
```



```
問題 出力 デバッグ コンソール ターミナル ポート 2 GITLENS コメント  
@geliefan →/workspaces/InstagramAPISample (main) $ cd src  
@geliefan →/workspaces/InstagramAPISample/src (main) $ docker-compose up --build[]
```

- ブラウザーで開くをクリックしてサービスが起動したことを確認してください。



User ID:  
Your Instagram User ID

ACCESS\_TOKEN:  
Your Instagram Access Token

Hashtag:  
#yourHashtag

Show My Photos Show My Hashtags Search by Hashtag

Photobook Name:  
Your Photobook Name

Create Photobook

Select Photobook:  
▼

Load Photobooks Display Photobook

- `User ID`, `ACCESS_TOKEN` を入力して `Show My Photos` をクリックして写真が表示されます。

- その後 **Create Photobook** をクリックして **Photobook created successfully!** が表示されたらフォトブックが作成されます。

## ● 課題

- 「src/app/routes\_exercise.py」の課題を解こう(TODOコメントの記載箇所)
  - **User ID**, **ACCESS\_TOKEN** を入力して **Show My Photos** をクリックして写真が表示されたら課題1クリアです。
  - その後 **Create Photobook** をクリックして **Photobook created successfully!** が表示されたら課題2クリアです。
- 動作確認する場合は **run.py** で **import** しているモジュールを **routes -> routes\_exercise** に置き換えてください。

## ● 解答例

- 「routes.py」が解答例です。

- 動作確認する場合は `run.py` で `import` しているモジュールを `routes_exercise -> routes` に置き換えてください.