このドキュメントの目的

VScodeのデバッグ有効化を行う。
Djangoの現在の状態を整理し、プロジェクトの基礎的な部分を作成する

目次

1. VScodeデバッグの有効化

VScodeのデバッグを行うためには、デバッグしたいソースコードの環境(何の言語、フレームワークなのか等)を設定する必要がある。
まずは基盤となるファイルを作成する。\

1. 『Ctrl』+『shift』+『D』キーもしくはVScode左側メニューの『実行とデバッグ』を押下する。
2. 『F5』キーもしくは左側の『実行とデバッグをカスタマイズするにはlaunch.jsonファイルを作成します。』を押下する。＊1
3. 『環境の選択』が表示されるので、『Node.js』を選択する。＊2
4. ディレクトリ『.vscode』とその中に『launch.json』が自動生成される。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?
linkid=830387
    "version": "0.2.0",
     "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "プログラムの起動",
            "skipFiles": [
                "<node_internals>/**"
            ],
            "program": "${file}"
        }
```

```
    ]
}
```

5. 以下のように書き換える。

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?
linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Django",
            "type": "python",
            "request": "launch",
            "program":
"C:\\Users\\XXXX\\Documents\\env1\\Practice\\mybook\\manage.py",
            "args": [
                "runserver",
                "--noreload"
            ],
            "django": true
        }
    ]
}
```

＊主な編集ポイント
拡張子の通りJSON形式で記述する。
『name』や『type』：Python、Djangoを選択する。
『program』：プロジェクト直下にある、『manage.py』の絶対パスを記述する。 『django』：『true』にして有効化する

＊1
ネットの情報では開かれたタブ内に歯車アイコンがあり、そこから選択できるとあったが、
私が行った環境では表示されていなかった。

＊2
ネットでは『デバッグの構成選択時にPythonが表示されている』等の情報があったが、
私が行った環境では表示されていなかったのでNode.jsを選択して自動生成させてから編集する手段を採用した。

6. 編集を終え保存したら、『F5』キーを押下してデバッグを実行する。

7. 画面下部にある『ターミナル』に、以下が出力されれば成功。

> Performing system checks...
> 中略

> Starting development server at http://127.0.0.1:8000/
> Quit the server with CTRL-BREAK.

＊前回コマンドプロンプトからmanage.pyを実行したときと同じものが出力される。

8. デフォルトでは画面上部中央にデバッグを操作するUIが表示される。
   終了したいときはここから『停止』を押下する。

---

2. 現在の仮想環境ディレクトリ配下構造

前作業とVScodeでデバッグを有効化した時点では以下のようになっている。
ひとまずはコロン書きがあるディレクトリの位置・役割を覚えておけばよい。

```
env1
    .vscode          ：ここにlaunch.jsonがある。Djangoとは関係がないファイル。
    Include
    Lib
    LICENSE.txt
    Practice         ：自分で作成した、プロジェクト管理ディレクトリ。
    Scripts          ：ここに仮想環境操作batなどがある。
    tcl
```

---

3. 現在のプロジェクト配下構造

前回作成したmybookプロジェクトの構造を見る。
こちらもコロンがついているものをひとまず意識する。

```
mybook                   ：プロジェクトのルートディレクトリ。
    db.sqlite3           ：DjangoはデフォルトでSQLiteを利用している。
    manage.py            ：プロジェクト全体に指示を出すために呼び出すプログラム。
    mybook               ：プロジェクト名のディレクトリが内部に生成される。ややこしい。
        asgi.py
        settings.py      ：このファイルにプロジェクト単位の設定を書き込むことが多い。大事。
        urls.py
        wsgi.py
        __init__.py
        __pycache__
```

---

4. 初期準備

1. VScodeのターミナルのシェル変更

VScodeにはターミナルが備わっており、そこからコマンドを実行することができる。
デフォルトではPowershellを使用する設定になっているが、コマンドプロンプトに変更しておく。

1. 画面下部の『ターミナル』のタブを押下する。
   ない場合は上部メニューの『ターミナル』から、『新しいターミナル』
   もしくは『Ctrl』+『@』キーを押下する。

2. 今VScodeで開いているフォルダ(例えばenv1など)を初期位置として、ターミナルが起動する。
   ここからコマンドを入力・実行できる。

3. VScode下部、ターミナルの右上にあるプルダウンを押下し、『既定のシェルの選択』を押下する。

4. VScode上部中央に『優先するターミナル　シェルを選択します...』と表示されるので、
   『Command prompt』を選択する。

5. 先ほどのプルダウンの右にある『ターミナルの強制終了』を押下する。

6. 再度上部メニューの『ターミナル』から、『新しいターミナル』
   もしくは『Ctrl』+『@』キーを押下する。

7. 先ほどのプルダウンの欄に『cmd』と表示されていれば成功。

試しに仮想環境に入るコマンドを入力してみる。

```
scripts\activate
```

実行結果
コマンドプロンプトで行った時と同様に仮想環境へ入れる。

```
C:\Users\XXXX\Documents\env1>scripts\activate
(env1) C:\Users\71506192\Documents\env1>
```

2. プロジェクトのタイムゾーン変更

1. VScode左メニューの『エクスプローラー』から、以下のファイルを開く。

   ```
   ENV1 > Practice > mybook > mybook > settings.py
   ```

2. ファイルの下のほうに行くと、以下のような行が出てくる。

   ```
   LANGUAGE_CODE = 'en-us'
   TIME_ZONE = 'UTC'
   ```

3. 以下のように変更する。
   念のためデフォルトはコメントアウトで残しているが、不要なら消しても構わない。

   ```
   # default
   # LANGUAGE_CODE = 'en-us'
   LANGUAGE_CODE = 'ja'
   ```

```
# default
# TIME_ZONE = 'UTC'
TIME_ZONE = 'Asia/Tokyo'
```

3. プロジェクトのデータベース初期設定(migrate)

＊migrate(マイグレート)とはDjangoを通じてデータベースに変更を加える作業のことをいう。
変更点の差分管理を行ったり、変更と反映が独立しているシステムである。
(イメージとして近いのはGitのcommit、margeだろうか。)

1. まずmigrateを行う。

```
cd C:\Users\XXXX\Documents\env1\Practice\mybook
python manage.py migrate
```

実行結果
以下のようなメッセージが出力される。

> Operations to perform:
> Apply all migrations: admin, auth, contenttypes, sessions
> Running migrations:
> Applying contenttypes.0001_initial... OK ~ 中略 ~
> Applying sessions.0001_initial... OK

2. データベースにアクセスするためのアカウントを作成する。

```
python manage.py createsuperuser
```

3. 以下のようなメッセージが１行ずつ出力されるので入力していく。

```
Username (leave blank to use 'XXXX'):   →アカウント名。デフォルトはPCユーザ名。
Email address:                          →メールアドレス。実在しなくてもよい。
Password:                               →パスワード。
Password (again):                       →パスワード再入力。
```

今回以下のように入力したところ、ポリシーに引っかかった部分があり注意が表示された。

```
Username (leave blank to use 'XXXX'): admin
Email address: admin@exsample.com
Password:admim
Password (again):admin
```

> The password is too similar to the username.
> This password is too short. It must contain at least 8 characters.
> This password is too common.
> Bypass password validation and create user anyway? [y/N]:

構わずyを入力してよい。

> Superuser created successfully.

4. 『F5』キーなどで一度デバッグし、サーバ起動やページアクセスが正しくできるか確認する。

今回はデフォルト生成された変更内容を管理するファイル(migration fileという)を\migrateする、という流れであった。
今後、**プログラムを作成・編集** → **migrationを作成** → **migrate**という手順で開発していくので、
この流れをつかんでおくこと。

ここまでで、開発に必要な初期設定や準備は終了である。
次回から、具体的に中身を実装してゆく。