1. Djangoって何？

・Djangoは無料でオープンソースとして公開されているPythonを使用したWebアプリケーションフレームワークです。

1. HTTPリクエスト時には何が起きているのか？

・Webサーバーにリクエストがあると、Djangoに伝えられ、リクエストの内容を把握しようとします。　まずWebページのアドレスを調べ、リクエストに対して何をするか決めます。これはDjangoのurlresolverが行います。 Djangoは上から下にURLパターンを順に調べていきます。そこで何かがマッチすると、Djangoはビューと呼ばれる関数にリクエストを送ります。

・ビュー関数では、面白いことが行われます。私たちは、DBに情報を探しにいきます。ビューは、まずあなたにその権限があるか確認します。 次に、データを書き換えて、「完了しました！」というメッセージをあなたに送り返します。そして、ビューが反応を返して、DjangoがユーザのWebブラウザに情報を送ります。

1. Djangoのインストール
   1. 仮想環境

・仮想環境により、開発単位で環境をまとめることができます。

〇Weindows

>仮想環境の作成

Python –m venv myenv #myenvが仮想環境名

>仮想環境のアクティブ化

myenv\Scripts\activate

* 1. PIPのアップグレード

(myenv)pip install –upgrade pip

* 1. PIPのインストール

(myenv)pip install django==1.11

ダブルイコールサインを使用して、バージョンを指定。

1. プロジェクトを作成しよう

・ここからは、シンプルなブログを作っていきます。

* 1. Djangoプロジェクトの作成

・Djangoでは、ファイルやディレクトリの名前がとても重要です。プロジェクトを作成後は、ファイル名を変更したり、ディレクトリ構成を変更したりするのは避けるべきである。

(myenv)Django-admin startproject mysite .

manage.py

mysite

├ \_\_init\_\_.py

├ settings.py

├ urls.py

└ wsgi.py

manage.py は、サイト管理用のスクリプトです。 これで、他のものを一切インストールすることなく、コンピューター上でWebサーバーを動かすことができます。

　　settings.py は サイトの設定ファイルです。

　　Urls.py ファイルは、urlresolverを使ったURLのパターンのリストを含んでいます。

* 1. 設定の変更（settings.py）

・タイムゾーン等を修正します。

LANGUAGE\_CODE=’ja-JP’

TIME\_ZONE=’Asiz/Tokyo’

USE\_TZ=False

* 1. DBの設定

・mysite/settings.pyファイルに設定されている。

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

* 1. DBの作成

>python manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, contenttypes, sessions

Running migrations:

Applying contenttypes.0001\_initial... OK

Applying auth.0001\_initial... OK

Applying admin.0001\_initial... OK

Applying admin.0002\_logentry\_remove\_auto\_add... OK

Applying contenttypes.0002\_remove\_content\_type\_name... OK

Applying auth.0002\_alter\_permission\_name\_max\_length... OK

Applying auth.0003\_alter\_user\_email\_max\_length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK

Applying auth.0005\_alter\_user\_last\_login\_null... OK

Applying auth.0006\_require\_contenttypes\_0002... OK

Applying auth.0007\_alter\_validators\_add\_error\_messages... OK

Applying auth.0008\_alter\_user\_username\_max\_length... OK

Applying sessions.0001\_initial... OK

・db.sqlite3ファイルが作成される。

* 1. サーバーの起動

>python manage.py runserver

<http://127.0.0.1:8000/>にアクセスするとページが閲覧できます。

1. オブジェクト

・ブログではどのようなオブジェクトが存在するか？　まずは投稿が存在する。

投稿には以下のようなプロパティがある。

・title,text,author,created\_date,publish\_date

投稿には以下のようなメソッドがある。

・publish

* 1. Django model

・投稿のDjangoモデルを作成する。 Djangoのモデルは特別なオブジェクトで、DBに格納される。

・ここにユーザーやブログポストの情報を格納します。

* 1. アプリの作成

・いったん別のアプリを作成する。

blog

├ \_\_init\_\_.py

├ admin.py

├ apps.py

├ models.py

├ test.py

├ views.py

└ migrations

└ \_\_init\_\_.py

・アプリケーションを作ったら、Djangoにそれを使うように伝えないといけません。それはmysit/settings.pyファイルの中で行います。

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

‘blog’,

]

* 1. 投稿モデルの作成

・blog/models.pyファイルでModelsと呼ばれる全てのオブジェクトを定義します。 これがブログポストを定義する場所です。以下のように記載します。

from django.db import models

from django.utils import timezone

# Create your models here.

class Post(models.Model):

author = models.ForeignKey('auth\_User')

title = models.CharField(max\_length=200)

text = models.TextField()

create\_date = models.DateTimeField(default=timezone.now)

publish\_date = models.DateTimeField(blank=True,null=True)

def publish(self):

self.publish\_date = timezone.now()

self.save()

def \_\_str\_\_(self):

return self.title

・classはオブジェクトを定義

・Postはモデル名で、大文字で始める。

・models.ModelはこれがDjango Modelだという意味になり、DjangoがこれはDBに保存すべきものと分かるようにします。

これに続いてプロパティを定義します。

・title,text,created\_date,publishe\_date,authoになる。フィールドには肩を指定する必要があります。

・models.CharField：テキスト数を定義するフィールド

・models.TextField：制限なしの長いテキスト

・models.DateTimeField：日付と時間

・models.ForeignKey：外部キー

・def publish(self)、def \_\_str\_\_(self)はメソッドになる。

* 1. DBにモデル用のテーブルを作る。

・最後のステップは新しいモデル用のDBテーブルを追加することになる。

・まずは作ったモデルの中で少し変更があったことをDjangoに知らせます。

>python manage.py makemigrations blog

Migrations for 'blog':

blog\migrations\0001\_initial.py

- Create model Post

・DjangoがDBに入れるための移行ファイルを作ります。

>python manage.py migrate blog

Operations to perform:

Apply all migrations: blog

Running migrations:

Applying blog.0001\_initial... OK

1. Django管理画面

・作成した投稿を追加、編集、削除するためにDjango adminを利用します。

* 1. 作成したモデルを管理機能に追加する

・blog/admin.pyを編集する。

from django.contrib import admin

from .models import Post

# Register your models here.

admin.site.register(Post)

* 1. 管理ユーザの作成

・管理画面にログインするには管理ユーザを作成する必要があります。

>python manage.py createsuperuser

Username (leave blank to use 'maruy'): admin

Email address: admin@admin.com

Password:

Password (again):

Superuser created successfully.

* 1. 管理画面

・http://127.0.0.1:8000/admin/

作成した管理ユーザでログインする。

・ユーザ、グループ、Postのデータをメンテナンスできます。

1. 配備

・PaaSのHerokuを利用して公開します。

* 1. requirements.txtファイル

・このファイルは、サーバーにどのようなPythonパッケージがインストールされているかを知らせるものです。

　　・その前に、Herokuを使用するのに必要なパッケージをインストールして

いきます。

>pip install dj-database-url gunicorn whitenoise

・仮想環境に設定されているパッケージのリストを作成します。

>pip freeze > requirements.txt

・作成したrequirements.txtの最終行に、@sycopg2==2.5.4を追加する。

これはHerukuで動かすために必要な１行になります。

* 1. Procfile

・次に必要なのがProcfileです。このファイルが、どのコマンドを実行してWebサイトをスタートするかHerokuに伝えます。Procfileを作成して、以下を記載します。

web: gunicorn mysite.wsgi

Webアプリを配備しようとしていること、そしてgunicorn mysite.wsgiというコマンドを実行して配備することを意味しています。 Gunicornは、Djangoのrunserverコマンドのもっとパワフルなものと考えてください。）

* 1. runtime.txtファイル

・Herokuに使っているPythonのバージョンを伝える必要がある。そのためにこのファイルに python-3.5.2と記載する。

* 1. mysite/local\_settings.py

・コンピューター上のローカルとサーバーでは設定に違いがある。Herokuとコンピューターでは別のDBを使用すると思います。そこでlocal\_setting.pyというファイルを作成し、ローカル環境で動かすための設定を保存する。

import os

BASE\_DIR = os.path.dirname(os.path.dirname(\_\_file\_\_))

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

DEBUG = True

* 1. mysite/settings.py

・すべきことがまだあります。settings.pyファイルに変更を加える必要がある。ファイルの末尾に以下を追加する。

import dj\_database\_url

DATABASES['default'] = dj\_database\_url.config()

SECURE\_PROXY\_SSL\_HEADER = ('HTTP\_X\_FORWARDED\_PROTO', 'https')

ALLOWED\_HOSTS = ['\*']

STATIC\_ROOT = 'staticfiles'

DEBUG = False

try:

from .local\_settings import \*

except ImportError:

pass

このファイルは、Herokuに必要な構成だけでなく、mysite/local\_settings.pyファイルがある時にはローカルの設定にも重要な役割となります。

* 1. mysite/wsgi.py

・次にmysite/wsgi.pyファイルを開き、最終行に次の通り追加していく。

from whitenoise.django import DjangoWhiteNoise

application = DjangoWhiteNoise(application)

1. Heroku account

・Heroku toolbeltをインストールする。Herokuをコマンドライン操作を行うためのアプリです。インストーラは、<https://toolbelt.heroku.com/>から取得します。heroku-cli-x64.exeをインストールして、パスを通す。

・Herokuの無料アカウントを作成する。: <https://id.heroku.com/signup/www-home-top>　ID：[fwka1605@gmail.com](mailto:fwka1605@gmail.com) PASS:@mam6mck3

・コマンドプロンプトでHerokuにログインします。

>heroku login

Enter your Heroku credentials:

Email:fwka1605@gmail.com

Passwrod:@mam6mck3

Logged in as [fwka11650@gmail.com](mailto:fwka11650@gmail.com)

・SSHキーがない場合はここで自動的に作成されます。 SSHキーは、コードをHerokuにプッシュするために必要になりみあす。

1. Git

・Herokuでは、Gitのリポジトリを使って、プロジェクトのファイルを管理します。

・ディレクトリに.gitignoreファイルを作成します。

myvenv

\_\_pycache\_\_

staticfiles

local\_settings.py

db.sqlite3

\*.py[co]

・Gitのリポジトリを作成します。

>git init

Initialized empty Git repository in C:/Users/maruy/Documents/git/Study2/Django/virtual/.git/

> git config user.name "Kouji Maruyama"

> git config user.email [fwka1605@gmail.com](mailto:fwka1605@gmail.com)

Gitリポジトリを作成するのは、プロジェクトにつき最初の１度だけです。

>git add -A .

>git commit -m "My Django Girls app"

[master (root-commit) 0458813] My Django Girls app

17 files changed, 277 insertions(+)

create mode 100644 .gitignore

create mode 100644 Procfile

create mode 100644 blog/\_\_init\_\_.py

create mode 100644 blog/admin.py

create mode 100644 blog/apps.py

create mode 100644 blog/migrations/0001\_initial.py

create mode 100644 blog/migrations/\_\_init\_\_.py

create mode 100644 blog/models.py

create mode 100644 blog/tests.py

create mode 100644 blog/views.py

create mode 100644 manage.py

create mode 100644 mysite/\_\_init\_\_.py

create mode 100644 mysite/settings.py

create mode 100644 mysite/urls.py

create mode 100644 mysite/wsgi.py

create mode 100644 requirements.txt

create mode 100644 runtimi.txt

1. アプリケーション名を選択する

・アプリケーション名は、Django blog app やmysiteといったこれまでに作成したプロジェクト名等と関連している必要はありません。ただし、名前に大文字や記号は使用できません。

>heroku create djangogirlsblog

Djangogirlsblogという名前のアプリケーション名を作成します。

参考：heroku create と実行するとHerokuが自動で名前を付けます。

参考：heroku apps:rename new-name を実行する。 これでWebのURLは、new-name.herokuapp.comに代わります。