Web アプリ「植物観察ノート」設計概要

Plant Watching Notes

本設計概要は、全体の構成が分かるレベルに留めコードの詳細は省略した。

株式会社アイティネット

2023 年 9 月 1 日

Web アプリ植物観察ノートは、Python のフレームワーク Django を使用して作成する。

Python、Anaconda Navigator、Django などのインストールについては本書では述べない。

１．準備

（1） プロジェクトをつくる

（ｉ）Anaconda Navigator の Environment を選び、その画面の「create」ボタンを操作して仮想環境 myspace をつくる。

（ii）myspace をクリックして OpenTerminal（コマンドモード）を開く。

（iii）OpenTerminal で、作成するプロジェクトのフォルダを収めるフォルダに移動する。

```
cd　フォルダ名　　＊1
```

※ このフォルダ配下にプロジェクトのフォルダを置く。

（iv）続いて、

```
django-admin startproject プロジェクト名
```

を実行すると、プロジェクト名をもつフォルダが、上記＊1 のフォルダ配下に作られる。

（ｖ）Anaconda Navigator で Home を選び、そこに表示されている Spyder を立ち上げ、project メニューを選択し、既存のプロジェクト（前項のプロジェクト名）を選択し、作成ボタンを使ってプロジェクトを作成する。

この結果、プロジェクトのフォルダ内の諸ファイルに必要な情報が設定される。

（2） アプリをつくる

OpenTerminal で、次のコマンドを使ってアプリケーションをつくる。

```
python manage.py  startapp  pwn
```

※pwn はアプリ（Plant Watching Notes）の名前

（3） アプリ登録 settings.py

```
INSTALLED_APPS = [
        ・・・
    'pwn',
    ]
```

2．データ設計

（1）データベースを設計

ユーザアカウント

Djangoの DB 管理機能のユーザアカウント管理を使用する。

（2）モデルを作成　models.py

```
from django.db import models
class Cell(models.Model):
    name = models.CharField(max_length=20) # 名前
    url0 = models.CharField(max_length=100)  # 画像への URL
    date0 = models.DateField()  # 日付
    caption0 = models.CharField(max_length=20) # キャプション
    url1 = models.CharField(max_length=100)
    date1 = models.DateField()
    caption1 = models.CharField(max_length=20)
    url2 = models.CharField(max_length=100)
    date2 = models.DateField()
    caption2 = models.CharField(max_length=20)
    url3 = models.CharField(max_length=100)
    date3 = models.DateField()
    caption3 = models.CharField(max_length=20)
    url4 = models.CharField(max_length=100)
    date4 = models.DateField()
    caption4 = models.CharField(max_length=20)
    url5 = models.CharField(max_length=100)
    date5 = models.DateField()
    caption5 = models.CharField(max_length=20)
    url6 = models.CharField(max_length=100)
    date6 = models.DateField()
    caption6 = models.CharField(max_length=20)
    url7 = models.CharField(max_length=100)
    date7 = models.DateField()
    caption7 = models.CharField(max_length=20)
    url8 = models.CharField(max_length=100)
    date8 = models.DateField()
    caption8 = models.CharField(max_length=20)
    url9 = models.CharField(max_length=100)
```

```
    date9 = models.DateField()
    caption9 = models.CharField(max_length=20)

    flowershape = models.CharField(max_length=10) # 花の形
    inflorescense = models.CharField(max_length=10) # 花序
    leafshape = models.CharField(max_length=10) # 葉の形
    leafleaf = models.CharField(max_length=10) # 葉序
    serration = models.CharField(max_length=10) # 鋸歯
    bark = models.CharField(max_length=10) # 樹皮
    family = models.CharField(max_length=20) # 科
    genus = models.CharField(max_length=20) # 属
    distribution = models.CharField(max_length=100) # 分布
    description = models.CharField(max_length=1000) # 説明
```

（3）マイグレーションする（コマンドプロンプト）

```
python manage.py makemigrations   pwn
python manage.py migrate
```

（4）admin.py に pwn のモデル（テーブル）を登録する

管理ツールでモデルを利用できるようにしておく　＊P155 参照

```
from  django.cotrib import admin
from  .models  import Cell
admin.site.register(Cell)
```

コマンドプロンプトで

```
python manage.py runserver
```

を実行させておいて、

ブラウザで

```
https://localhost:8000/admin/
```

のページを開き、id とパスワードを入力

```
id       xxxxxxx
pswd     xxxxxxx
```

＊「Python Django 超入門」の P155

（5）フォーム　forms.py

```
from django import forms

class CellForm(forms.Form):
    data1 = [('指定なし', '指定なし'),
             ('鐘形', '鐘形'),
             ('ラッパ形', 'ラッパ形'),
             ('舌形', '舌形'),
             ('蝶形', '蝶形'),
             ('杯形', '杯形'),
             ('兜形', '兜形'),
             ('壺形', '壺形'),
             ('唇形', '唇形'),
             ('漏斗形', '漏斗形'),
             ('距のあるもの', '距のあるもの')]

    data2 = [('指定なし', '指定なし'),
             ('総状花序', '総状花序'),
             ('穂状花序', '穂状花序'),
             ('散房花序', '散房花序'),
             ('散形花序', '散形花序'),
             ('複総状花序', '複総状花序'),
             ('複散形花序', '複散形花序'),
             ('頭状花序', '頭状花序'),
```

```
            ('円錐状花序', '円錐状花序')]

    data3 = [('指定なし', '指定なし'),
            ('単葉：楕円形', '単葉：楕円形'),
            ('単葉：卵形', '単葉：卵形'),
            ('単葉：ヘラ形', '単葉：ヘラ形'),
            ('単葉：腎臓形', '単葉：腎臓形'),
            ('単葉：ハート形', '単葉：ハート形'),
            ('単葉：披針形', '単葉：披針形'),
            ('単葉：倒披針形', '単葉：倒披針形'),
            ('単葉：線形', '単葉：線形'),
            ('単葉：切れ込みあり', '単葉：切れ込みあり'),
            ('奇数羽状複葉', '奇数羽状複葉'),
            ('偶数羽状複葉', '偶数羽状複葉'),
            ('２回偶数羽状複葉', '２回偶数羽状複葉'),
            ('３回奇数羽状複葉', '３回奇数羽状複葉'),
            ('掌状複葉', '掌状複葉'),
            ('３出複葉', '３出複葉'),
            ('針葉', '針葉'),
            ('鱗片葉', '鱗片葉')]

    data4 = [('指定なし', '指定なし'),
            ('互生', '互生'),
```

```
            ('対生', '対生'),
            ('輪生', '輪生'),
            ('束生', '束生'),
            ('根生', '根生'),
            ('叢生', '叢生'),
            ('茎に流れる', '茎に流れる'),
            ('茎を抱く', '茎を抱く'),
            ('つき抜き', 'つき抜き'),
            ('葉鞘のある', '葉鞘のある')]

data5 = [('指定なし', '指定なし'),
            ('単鋸歯', '単鋸歯'),
            ('重鋸歯', '重鋸歯'),
            ('波状', '波状'),
            ('円鋸歯', '円鋸歯'),
            ('歯状', '歯状'),
            ('全縁', '全縁')]

data6 = [('指定なし', '指定なし'),
            ('縦模様', '縦模様'),
            ('横模様', '横模様'),
            ('なめらか', 'なめらか'),
            ('深・浅裂', '深・浅裂'),
```

```
            ('はがれ', 'はがれ'),
            ('まだら', 'まだら')]

    name = forms.CharField(max_length=20, label='名前',\
                           initial='指定なし')
    flowershape = forms.ChoiceField(label='花の形', choices=data1)
    inflorescense = forms.ChoiceField(label='花序', choices=data2)
    leafshape = forms.ChoiceField(label='葉の形', choices=data3)
    leafleaf = forms.ChoiceField(label='葉序', choices=data4)
    serration = forms.ChoiceField(label='鋸歯', choices=data5)
    bark = forms.ChoiceField(label='樹皮', choices=data6)
    family = forms.CharField(max_length=20, label='科', \
                           initial='指定なし')
    genus = forms.CharField(max_length=20, label='属', \
                           initial='指定なし')
    distribution = forms.CharField(max_length=20, label='分布', \
                           initial='指定なし')
    description = forms.CharField(max_length=1000, label='説明',\
                initial='指定なし', \
                widget=forms.Textarea(attrs={'cols':'80', \
                                             'rows':'12'}))
class UploadForm(forms.Form):
    file = forms.ImageField(label='画像')
    caption = forms.CharField(max_length=20, label='キャプション',\
```

```
                                    initial='指定なし')

### 以下は検索用のフォーム
class NmForm(forms.Form):
    name = forms.CharField(max_length=20, label='名前')

class DtForm(forms.Form):
    fromdate=forms.DateField(label='From', \
             widget=forms.SelectDateWidget(years=[x for x in \
             range(2018, 2030)]))
    todate=forms.DateField(label='To  ', \
             widget=forms.SelectDateWidget(years=[x for x in \
             range(2018, 2030)]))

class PlForm(forms.Form):
    place = forms.CharField(max_length=20, label='場所')

class FsForm(forms.Form):
    data1 = [('鐘形', '鐘形'),
             ('ラッパ形', 'ラッパ形'),
             ('舌形', '舌形'),
             ('蝶形', '蝶形'),
             ('杯形', '杯形'),
             ('兜形', '兜形'),
             ('壺形', '壺形'),
             ('唇形', '唇形'),
             ('漏斗形', '漏斗形'),
             ('距のあるもの', '距のあるもの')]
    flowershape = forms.ChoiceField(label='花の形', choices=data1)
```

```
class IfForm(forms.Form):
    data2 = [('総状花序', '総状花序'),
             ('穂状花序', '穂状花序'),
             ('散房花序', '散房花序'),
             ('散形花序', '散形花序'),
             ('複総状花序', '複総状花序'),
             ('複散形花序', '複散形花序'),
             ('頭状花序', '頭状花序'),
             ('円錐状花序', '円錐状花序')]
    inflorescense = forms.ChoiceField(label='花序', choices=data2)

class LsForm(forms.Form):
    data3 = [('単葉：楕円形', '単葉：楕円形'),
             ('単葉：卵形', '単葉：卵形'),
             ('単葉：ヘラ形', '単葉：ヘラ形'),
             ('単葉：腎臓形', '単葉：腎臓形'),
             ('単葉：ハート形', '単葉：ハート形'),
             ('単葉：披針形', '単葉：披針形'),
             ('単葉：倒披針形', '単葉：倒披針形'),
             ('単葉：線形', '単葉：線形'),
             ('単葉：切れ込みあり', '単葉：切れ込みあり'),
             ('奇数羽状複葉', '奇数羽状複葉'),
             ('偶数羽状複葉', '偶数羽状複葉'),
             ('２回偶数羽状複葉', '２回偶数羽状複葉'),
```

```
            ('３回奇数羽状複葉', '３回奇数羽状複葉'),
            ('掌状複葉', '掌状複葉'),
            ('３出複葉', '３出複葉'),
            ('針葉', '針葉'),
            ('鱗片葉', '鱗片葉')]
    leafshape = forms.ChoiceField(label='葉の形', choices=data3)

class LlForm(forms.Form):
    data4 = [('互生', '互生'),
            ('対生', '対生'),
            ('輪生', '輪生'),
            ('束生', '束生'),
            ('根生', '根生'),
            ('叢生', '叢生'),
            ('茎に流れる', '茎に流れる'),
            ('茎を抱く', '茎を抱く'),
            ('つき抜き', 'つき抜き'),
            ('葉鞘のある', '葉鞘のある')]
    leafleaf = forms.ChoiceField(label='葉序', choices=data4)

class SrForm(forms.Form):
    data5 = [('単鋸歯', '単鋸歯'),
            ('重鋸歯', '重鋸歯'),
            ('波状', '波状'),
```

```
            ('円鋸歯', '円鋸歯'),
            ('歯状', '歯状'),
            ('全縁', '全縁')]
    serration = forms.ChoiceField(label='鋸歯', choices=data5)

class BkForm(forms.Form):
    data6 = [('縦模様', '縦模様'),
            ('横模様', '横模様'),
            ('なめらか', 'なめらか'),
            ('深・浅裂', '秦・浅裂'),
            ('はがれ', 'はがれ'),
            ('まだら', 'まだら')]
    bark = forms.ChoiceField(label='樹皮', choices=data6)

class FmForm(forms.Form):
    family = forms.CharField(max_length=20, label='科名')

class GeForm(forms.Form):
    genus = forms.CharField(max_length=20, label='属名')

class DiForm(forms.Form):
    distribution = forms.CharField(max_length=20, label='分布')

class DeForm(forms.Form):
    description = forms.CharField(max_length=20, label='説明')

class SearchkdForm(forms.Form):
    data=[
        (1, '名前'),
```

```
        (2, '日付'),
        (3, '場所'),
        (4, '花の形'),
        (5, '花序'),
        (6, '葉の形'),
        (7, '葉序'),
        (8, '鋸歯'),
        (9, '樹皮'),
        (10, '科'),
        (11, '属'),
        (12, '分布'),
        (13, '説明')
        ]
    choice = forms.ChoiceField(label='検索の種類選択', choices=data)
```

（6）テンプレートフォルダの作成

・pwn フォルダ内に、templates フォルダをつくる。

・この templates フォルダ内に、pwn フォルダをつくる。

・その pwn フォルダ内に、次のテンプレート・ファイルを置く。

■ メインページ

index.html

■ セル表示ページ

cell.html

■ イメージ表示ページ

cellimage.html

■ セルの作成・編集ページ

cellcreate.html
celledit.html

■ 画像の登録・編集ページ

cupload.html
eupload.html

■ 検索種類の選択ページ

searchkd.html

■ 検索ページ　　１３種のフォームに対する共通ページ

search.html

（7）static フォルダの作成

・pwn フォルダ内に、static フォルダをつくる。

・この sutatic フォルダ内に、pwn フォルダをつくる。

・その pwn フォルダ内に、css フォルダをつくる。

その css フォルダ内に、style.css ファイルを置く。

（8）画像処理ライブラリ（pillow）をインストール

（これは　get_datetime のため）

Anaconda Navigator の Environment の Myspace のオープンターミナルで、

コマンド　pip install Pillow　を実行。

views.py ファイル内に、次のコードを設定する。

```
import os
from PIL import Image
from PIL.ExifTags import TAGS
 # TAGS には、Exif 情報が辞書形式で保存されている。
```

（9）ログインを必須にする

views.py ファイル内に、次のコードを設定する。

```
from Django.contrib.auth.decorators import login_required
```

ユーザ認証したい関数 or クラスの前に、

```
@login_required(login_url='/admin/login/')
```

を設定する。

django のプロジェクトには、アプリケーション：

```
django.contrib.auth
```

が組み込まれている。これは settings.py ファイルの、

INSTALLED_APPS のリスト内に存在する。

（10）settings.py ファイル

「（9）ログインを必須にする」に加えて、settings.py ファイルには次を設定する。

```
import os
from pathlib import Path

TIME_ZONE = 'Asia/Tokyo'
MEDIA_ROOT = os.path.join(BASE_DIR,'media')
MEDIA_URL = '/media/'
```

media フォルダはプロジェクトフォルダ直下につくる。

アップロードした画像ファイルは media フォルダに置かれる。

（１１）プロジェクトの urls.py ファイル

```
from django.contrib import admin
from django.urls import path, include

from . import settings
from django.contrib.staticfiles.urls import static
from django.contrib.staticfiles.urls import staticfiles_urlpatterns

urlpatterns = [
    path('admin/', admin.site.urls),
    path('pwn/', include('pwn.urls')),
    ]
urlpatterns += staticfiles_urlpatterns()
urlpatterns += static(settings.MEDIA_URL, \
                      document_root=settings.MEDIA_ROOT)
```

アプリケーションの urlpatterns をプロジェクトの urlpattern に include する。

下の２つの urlpatterns への追加は、開発中のサーバで media フォルダ内の画像ファイルのアクセスを適切にできるようにするための設定である。本番環境では Web サーバが適切に処理するため必要がない。

(11) クラス MainView のクラス変数

cellist=[] # クラス QuerySet のインスタンスを辞書のリスト、つまり、

[{…},{…},…]

の形式に変換して納める変数

この値が更新されるのは次の場合

・アプリ PWN を立ち上げたとき

・メインページで「最新のセル表示」ボタンをクリックしたとき（all()メソッド）

・検索結果を表示するとき

start=0 # インスタンス celllist から最大 10 個ずつ取り出す際の、最初の位置

number=0 # 現在のインスタンス celllist の要素数

cellplist=[] # メインページに表示する画像情報（下記の辞書）のリスト（最大

10 個のセル）

[{ posque # celllist 内のセルレコードの位置
  name # セルの名前
  url0 # セルがもつ先頭の画像の url
  date0 # url0 の画像の撮影日時
  caption0 # url0 の画像に関連するキャプション
  pictype # 画像の横型（0）、縦型（1） } , ・・・]

※ url0 の 0 は表示する各セルの先頭画像を意味する

piclist=[] # 現在処理中のセルに含まれる画像情報のリスト

画像情報は次の通り

[{url : xx, date:xx, caption:xx, pictype:x, num:xx}, ・・・]

次の２つは QuerySet 取込の種類を文字列で示し、メインページに表示するために使用する。

```
disptype = ""
curstete={
    'disptype':'',
    'total':'',
    'startp':''
    }
```

(12) メインページ（index.html）へのセルの読み込み

DB からデータをメインページに表示する機会は、

- アプリ PWN のを立ち上げ時
- メインページの「最新のセル表示」ボタンのクリック時
- 検索結果の表示時
- メインページの First、Prev、Next、Last の各ボタンのクリック時

の 4 パターンである。

これらの操作は次の考え方に基づいて行なう。

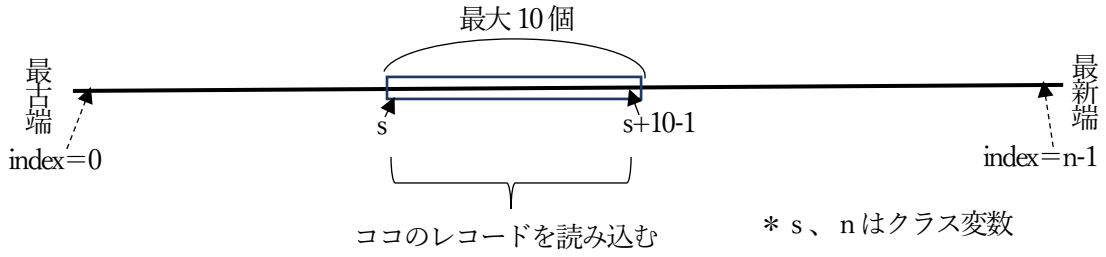
考え方

s：DB（モデル）の id=1（最古）から id=n（最新）までの間で、これからアクセスしようとするレコード集合（最大 10 個）の先頭の位置を示す。

n：DB（モデル）から読み込んだ QuerySet のインスタンス celllist のレコードの数を示す。これは次のコードで求められる。

```
n=Cell.objects.count ()
```

※Cell はモデルの名前。n は QuerySet 内のレコードの数。



メインページの各ボタンと読み込み範囲

① 「最新のセル表示」および「Last」ボタンのクリック時

s＝n－10　ただし　s＜0→s=0　とする。

そして、

※ n-s≧10　→　[s, s+10-1]

n-s≦0　→　[ ]

n-s<10　→　[s, n-1]

で読み込む。　　注）リストの要素位置は 1～ではなく、

0～で計算する。

② 「First」ボタン（最古）をクリック時、

s=0

そして　※を使用。

③　「Prev」ボタンをクリック時、

s-＝10　ただし　s<0　→　s=0

そして　※を使用。

④　「Next」ボタン（最新）をクリック時、

n-s>10 のとき、s+=10　ただし　s>n　→　s=n

そして　※を使用。

アクション（メインページの「最新のセル表示」、「First」、「Prev」、「Next」、「Last」をクリック）ごとに、先ず、クラス変数 cellplist=[]とし、上の考え方で抽出された

各セルについて、

posque　（セルの celllist 内の位置）

name　（セルの名前項目）

url0　（セルの先頭の画像）

date0　（セルの先頭画像の撮影日）

caption0　（セルの先頭画像のキャプション）

pictype　（セルの先頭画像の型：横型（0)、縦型（1））

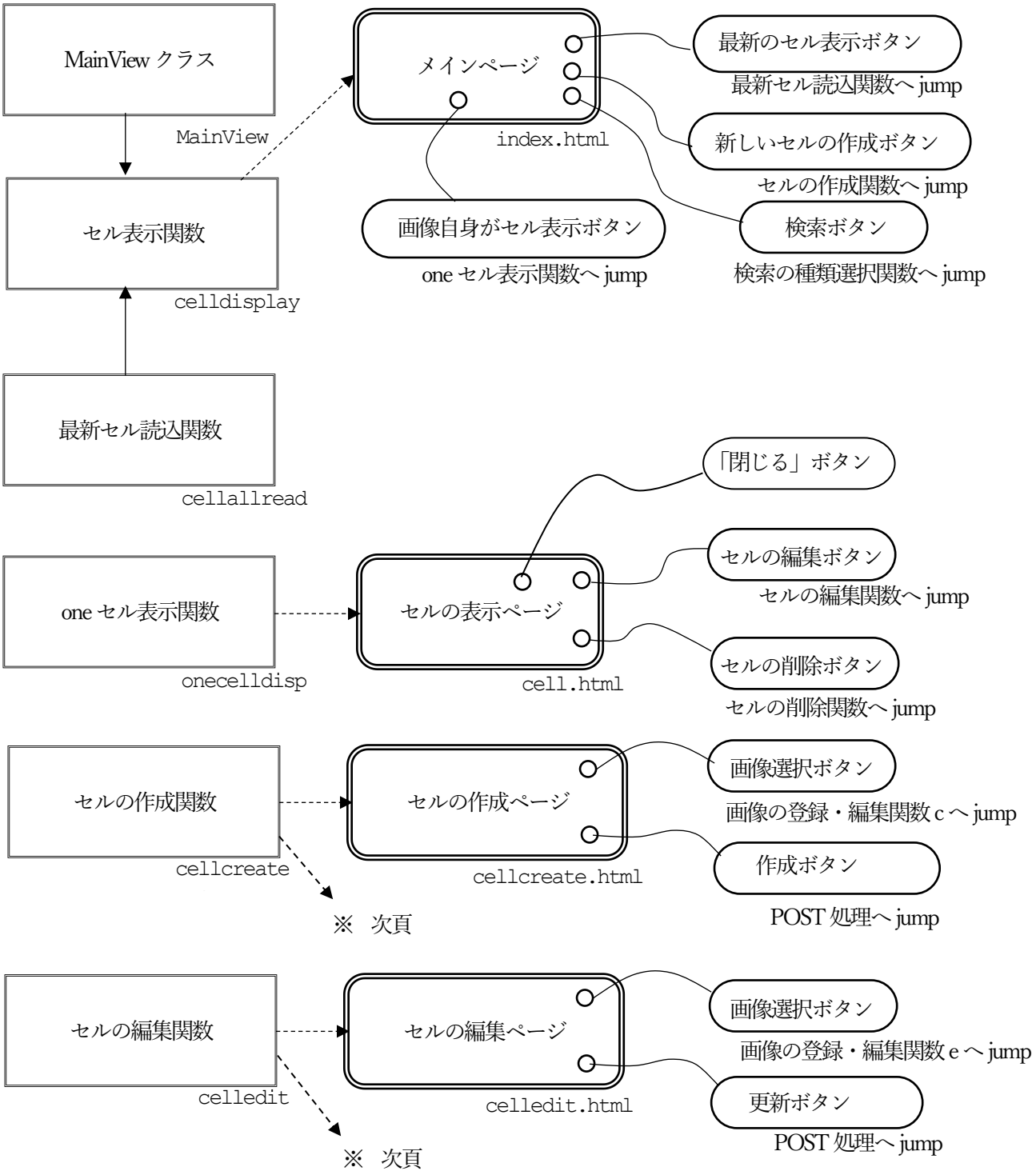
を辞書形式にして、クラス変数 cellplist に収める。

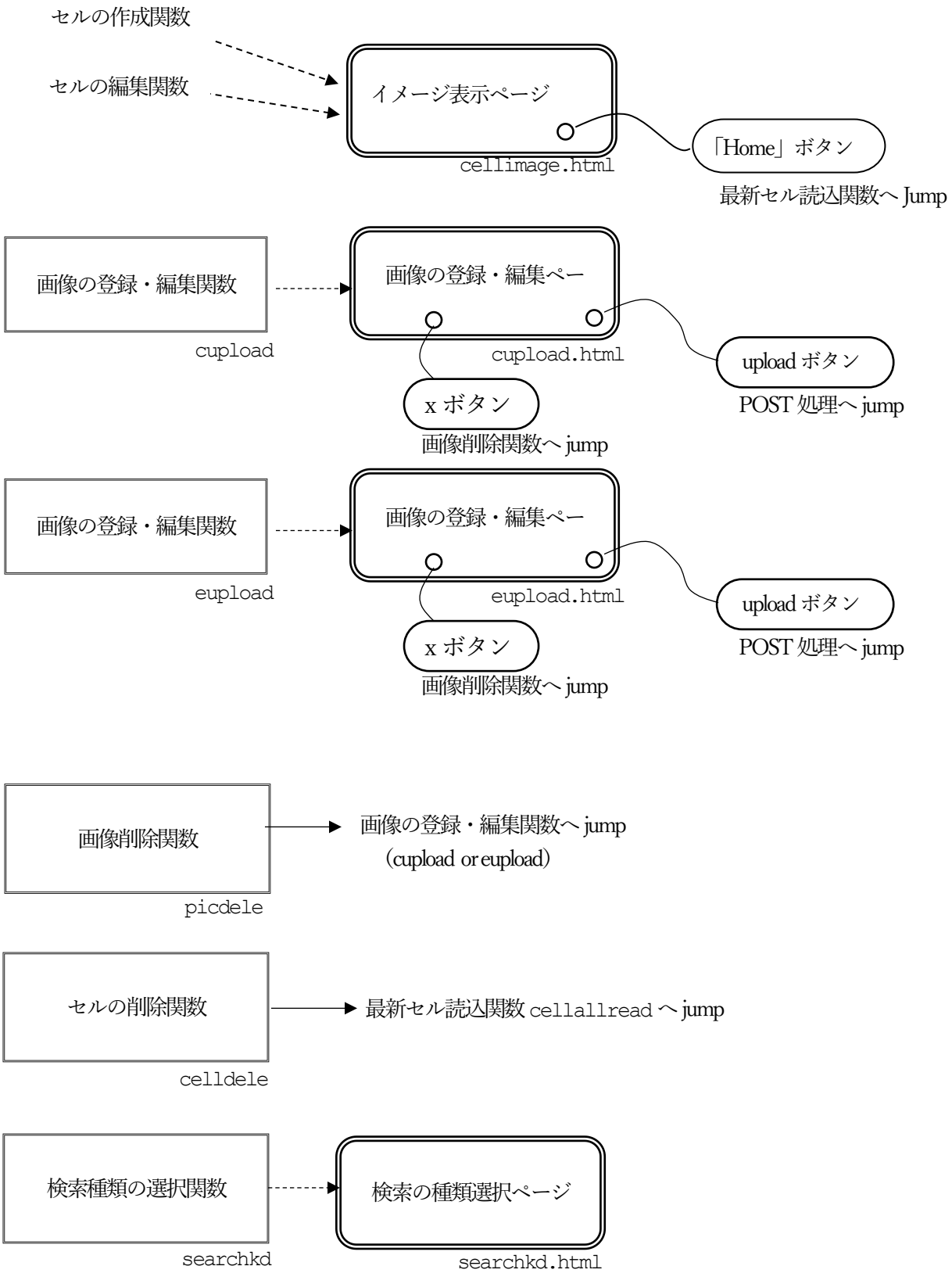
　こうして得られた cellplist のデータがメインページにレンダリングされる。表示されるのは name、url0 の画像、date0、caption0 だけである。

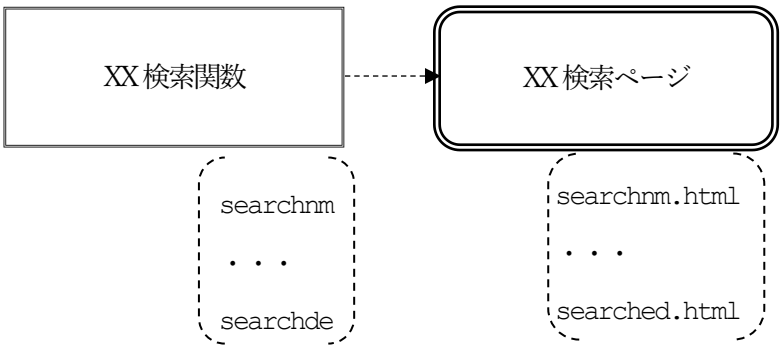
3．処理部

※　点線矢印は関数からテンプレートの呼出しを意味する

（1）操作の流れ




セルの作成関数
セルの編集関数
イメージ表示ページ
cellimage.html
「Home」ボタン
最新セル読込関数へ Jump
画像の登録・編集関数
cupload
画像の登録・編集ページ
cupload.html
upload ボタン
POST 処理へ jump
x ボタン
画像削除関数へ jump
画像の登録・編集関数
eupload
画像の登録・編集ページ
eupload.html
upload ボタン
POST 処理へ jump
x ボタン
画像削除関数へ jump
画像削除関数
picdele
画像の登録・編集関数へ jump
(cupload or eupload)
セルの削除関数
celldele
最新セル読込関数 cellallread へ jump
検索種類の選択関数
searchkd
検索の種類選択ページ
searchkd.html


XX検索関数
XX検索ページ
searchnm
・・・
searchde
searchnm.html
・・・
searched.html

（2）アドレステーブル　urls.py

Django の処理部の url のリスト（テーブル）を

```
urlpatterns = [path( , ,…), …, path( , , ),]+\
  static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

に設定する。

1）　メインクラス

```
path(r'', MainView.as_view(), name='index'),
```

2）　セル表示関数

```
path('celldisplay', views.celldisplay, name='celldisplay'),
```

3）　最新セル読込関数

```
path('cellallread', views.cellallread, name='cellallread'),
```

4）　one セル表示関数

```
path('onecelldisp/<int:pos>',views.onecelldisp \
   name='onecelldisp'),
```

5）　セル作成関数

```
path('cellcreate/<int:num>', views.cellcreate, name='cellcreate'),
```

6）　セル編集関数

```
path('celledit/<int:num>/<int:pos>', views.celledit,\
                                          name='celledit'),
```

7）　セル削除関数

```
path('celldele/<int:pos>', views.celldele, name='celldele'),
```

8）　画像の登録関数　cellcreate 用

```
path('cupload/<int:num>', views.cupload, name='cupload'),
```

9）　画像の登録関数 celledit 用

```
path('eupload/<int:num>', views.eupload, name='eupload'),
```

１０）　画像削除関数

```
path('picdele/<int:num>/<str:ret>', views.picdele, name='picdele'),
```

１１）　検索種類の選択関数

```
path('searchkd', views.searchkd, name='searchkd'),
```

１２）　名前検索

```
    path('searchnm', views.searchnm, name='searchnm'),
```

１３） 日付検索

```
    path('searchdt', views.searchdt, name='searchdt'),
```

１４） 場所検索

```
   path('searchpl', views.searchpl, name='searchpl'),
```

１５） 花の形検索

```
    path('searchfs', views.searchfs, name='searchfs'),
```

１６） 花序検索

```
    path('searchif', views.searchif, name='searchif'),
```

１７） 葉の形検索

```
    path('searchls', views.searchls, name='searchls'),
```

１８） 葉序検索

```
    path('searchll', views.searchll, name='searchll'),
```

１９） 鋸歯検索

```
    path('searchsr', views.searchsr, name='searchsr'),
```

２０） 樹皮検索

```
    path('searchbk', views.searchbk, name='searchbk'),
```

２１） 科検索

```
    path('searchfm', views.searchfm, name='searchfm'),
```

２２） 属検索

```
    path('searchge', views.searchge, name='searchge'),
```

２３） 分布検索

```
    path('searchdi', views.searchdi, name='searchdi'),
```

２４） 説明検索

```
    path('searchde', views.searchde, name='searchde'),
```

２５） First 関数

```
    path('first', views.first, name='first'),
```

２６） Prev 関数

```
    path('prev', views.prev, name='prev'),
```

２７） Next 関数

```
    path('next', views.next, name='next'),
```

２８） Last 関数

```
    path('last', views.last, name='last'),
```

（3）関数とクラス　　views.py

以下は本モジュールで引用するクラスである。

```
from django.shortcuts import render, redirect
from django.views.generic import TemplateView
from .forms import CellForm, SearchkdForm, NmForm, DtForm, PlForm,
                   FsForm, IfForm, LsForm, LlForm, SrForm, BkForm,
                   FmForm, GeForm, DiForm, DeForm, UploadForm
from .models import Cell
from django.conf import settings
from django.core.files.storage import FileSystemStorage
import os
from datetime import datetime

# 以下の２つは get_datetime()のため
from PIL import Image
from PIL.ExifTags import TAGS

# 以下はログインを必須にするため
from django.contrib.auth.models import User
from django.contrib.auth.decorators import login_required
# ユーザ認証したい関数 or クラスの前に次を設定
# @login-required(login_url='/admin/login/')

# global 変数　 cellcreate()関数, celledit()関数で利用
result = Cell()  # モデル Cell のインスタンス
celli = {}  # MainView.celllist の i 番目の要素（レコード）
```

■class MainView(TemplateView):

```
    # 以下はクラス変数
    cellist=[] # クラス QuerySet のインスタンスを納める変数
      #    この値が更新されるのは次の場合
      #    --アプリ PWN を立ち上げたとき
      #    --メインページで「最新のセル表示」ボタンをクリックしたとき
      #    --検索結果を表示するとき
    start=0
      #    QuerySet から最大 10 個ずつ組みにして取り出す際の各組の最初の位置
    number=0 # 現在の QuerySet の要素数
    cellplist=[]
      # メインページに表示するセル抜粋情報の辞書のリスト（最大 10 個）
      # 各セル抜粋情報辞書は次の要素からなる
      # posque ： QuerySet 内のセルレコードの位置
      # name ：セルの名前
      # url0：セルがもつ先頭の画像の url
      # date0：url0 の画像の撮影日時
      # caption0：url0 の画像に関連するキャプション
    piclist=[]        # 現在処理中のセルに含まれる画像情報辞書のリスト
      # 画像情報は次の通り
      # [{url : xx, date:xx, caption:xx, num:xx}, ・・・]
      # QuerySet 取込の種類を文字列で示し、メインページに表示する
    disptype = ""
    curstete={
        'disptype':'',
```

```
        'total':'',
        'startp':''
        }
    def get(self,request):
        # モデル Cell の全レコードを取り出す
        queryset = Cell.objects.all()
        MainView.celllist = tocelllist(queryset)
        #  Cell のレコード数を取り出す
        MainView.number = Cell.objects.count()
        MainView.start = MainView.number -10
        if MainView.start <= 0 :
            MainView.start = 0
        MainView.disptype = '最新のセル'
        MainView.curstate = {'disptype':MainView.disptype,
                             'total':str(MainView.number),
                             'startp':str(MainView.start)
                             }
        return redirect('celldisplay')
```

■def celldisplay(request): #セル表示関数

```
e = MainView.start +10
if (e  >= MainView.number):
        e = MainView.number - 1
```

この時、(MainView.start, e)はcelllistのサブセット（最大１０個）を示

#している。

```
# セル抜粋情報の辞書を作成する
MainView.cellplist = []

for i in range(MainView.start, e+1):
    celli = MainView.celllist[i]  # i はMainView.celllist内の位置
    data = {'posque' :  i  ,
            'name' : celli['name'],
            'url' :  celli['url0'],
            'date' : celli['date0'],
            'caption' : celli['caption0'],
            'pictype' : getpictype(celli['url0'])
            }
    MainView.cellplist.append(data)
# パラメータを作成して次のrenderに渡す
params = {'disptype' : MainView.curstate['disptype'] ,
          'total' : MainView.curstate['total'] ,
          'startp' : MainView.curstate['startp'] ,
          'obj' : MainView.cellplist
          }
return render(request, 'pwn/index.html', params)
```

■def cellallread(request): #　新しく全セルを読込む関数

```
    # モデル Cell の全レコードを取り出す
    queryset = Cell.objects.all()
    MainView.celllist = tocelllist(queryset)
    print(MainView.celllist)
    # Cell のレコード数を取り出す
    MainView.number = Cell.objects.count()
    MainView.start = MainView.number -10
    if MainView.start <= 0 :
        MainView.start = 0
    MainView.disptype = '最新のセル'
    MainView.curstate = {'disptype':MainView.disptype,
                         'total':str(MainView.number),
                         'startp':str(MainView.start)
                         }
    return redirect('celldisplay')
```

■def tocelllist(querysetp):

```
  # querysetp（QuerySet インスタンス）をレコード（辞書）のリストに変換
    return list(querysetp.values(
        'id', 'name', 'url0', 'date0', 'caption0',
        'url1', 'date1', 'caption1', 'url2', 'date2', 'caption2',
        'url3', 'date3', 'caption3', 'url4', 'date4', 'caption4',
        'url5', 'date5', 'caption5', 'url6', 'date6', 'caption6',
        'url7', 'date7', 'caption7', 'url8', 'date8', 'caption8',
        'url9', 'date9', 'caption9',
        'flowershape', 'inflorescense', 'leafshape', 'leafleaf',
        'serration', 'bark', 'family', 'genus', 'distribution',
        'description'))
```

■def onecelldisp(request, pos): # セル表示関数

```
    # この関数はメインページに表示されているセルの画像をクリックすることで呼び
    # 出される
    i = pos
    #  i は表示したセル情報の、リスト celllist 内の位置
```

######　テンプレート cell.html に渡す引数の準備　######

  ###　　先ず、MainView.piclist を用意する　###

《以下省略》

■def cellcreate(request, num): #セル作成関数

```
#  この関数はメインページ（index.html）の「新しいセル作成」ボタンをクリックす
#  ることにより呼び出される。

    global result # モデル Cell のインスタンス変数

# num=1：メインページからの呼出し。通常の GET、POST 処理を実行する。
# num=0：upload.html の「登録終了」ボタンのクリックによる呼出し。セルの作成ペ
#        ージに画像を表示する。

    if num==0 :
        params = {'form' : CellForm(),
                  'obj' : MainView.piclist
                  }
        return  render(request, 'pwn/cellcreate.html', params)
```

《以下省略》

■def celledit(request, num, pos): #セル編集関数

この関数はセルの表示ページ（cell.html）の「セルの編集」ボタンをクリックすることにより呼び出される。

```
    global result # モデル Cell のインスタンス変数
    global celli  # MainView.celllist の i 番目の要素（レコード）
# num=1：セルの表示ページからの呼出し
# num=0：eupload.html の「登録終了」ボタンクリックによる呼出し
#       {% url 'celledit' 0 %}
# pos は編集対象のセルの celllist 内の位置を示す
    if num==0 : # 編集ページの再表示
#    以下は num=1 のときの処理
```

《以下省略》》

■def cupload(request, num): # 画像登録・編集関数 cellcreate 用

```
#  画像情報のアップロードと、画像ファイルの url、日付、キャプションを取出す
#  from django.core.files.storage import FileSystemStorage
    if num == 0 :
        params = {'form':UploadForm(),
                  'obj':MainView.piclist
                  }
        return render(request, 'pwn/cupload.html', params)
    if request.method ==  'POST'  and   request.FILES['file'] :
```

《以下省略》》

■def eupload(request, num):　# 画像登録・編集関数 celledit 用

```
#  画像情報のアップロードと、画像ファイルの url、日付、キャプションを取出す
#  from django.core.files.storage import FileSystemStorage

    if num == 0 :
        params = {'form':UploadForm(),
                  'obj':MainView.piclist
                  }
        return render(request, 'pwn/eupload.html', params)
```

《《以下省略》》

■def getdate(file_url):

```
    #  画像ファイルから日付を取出し、呼び出し元へ返却する関数

    # URL を '/' で分割し、最後の要素を取得
    filename = file_url.split('/')[-1]
    # 必要に応じて、クエリパラメータやフラグメントを削除
    filename = filename.split('?')[0].split('#')[0]
    file_path = os.path.join(settings.MEDIA_ROOT, filename)
    im = Image.open(file_path)
    exif = im._getexif()
    #  画像ファイルの exi f データを辞書として取り出す

    《《以下省略》》
```

■def getpictype(file_url):

```
    #  画像ファイルから幅、高さを取出し、

    #      0 :  幅  >  高さ

    #      1 :  幅  <  高さ

    # をリターンする
```

■def picdele(request, num, ret): # 画像削除関数

```
    #  upload.html 内から呼び出される。

    i = num #  i は削除対象の画像の、MainView.piclist 内の位置
    item = MainView.piclist[i]
    file_url = item['url']

    # URL を '/' で分割し、最後の要素を取得
    filename = file_url.split('/')[-1]
    # 必要に応じて、クエリパラメータやフラグメントを削除
    filename = filename.split('?')[0].split('#')[0]
    file_path = os.path.join(settings.MEDIA_ROOT, filename)

    os.remove(file_path)     #  画像ファイルを削除
    del MainView.piclist[i]     #  i 番目の要素を削除
```

《《以下省略》》

■def celldele(request, pos): # 現在表示中のセルを削除

```
    #  セル内のすべての画像ファイルを削除し、現在のセルを DB から削除する
    #  この関数は cell.html の中から呼び出される
    i = pos
    #  i は MainView.celllist 内の当該セルの位置
    #  MainView.piclist には現在のセルの cell.html に表示されている画像情報\
    # が収容されている

    #    画像ファイルの削除
```

《《以下省略》》

■def searchkd(request):　#　検索の種類選択

```
    #　検索の種類は、名前、日付、場所、花の形、花序、葉の形、葉序、鋸歯、樹皮、
    # 科、属、分布、説明　の１３種である。これらのうちから１つを選ぶために
    # formchoice フィールドを定義しておく。
    #　種類を選択すると、フォームから対応する値（1, 2, 3, …, 13）が返される。
    #　この値を使って種類対応に予め用意しておいた関数呼び出しを行う。
    if  request.method == 'POST' :
        num = request.POST['choice']
        if num == 1 :  #　名前
            return redirect('searchnm')
```

《《以下省略》》

■def searchnm(request):　#　名前検索

```
    if  request.method == 'POST' :
        nm = request.POST['name']    #　'name'は NmForm 内のフィールド名
```

《《以下省略》》

■def searchdt(request):　#　日付検索

```
    if  request.method == 'POST' :
        form = DtForm(request.POST)
```

《《以下省略》》

■def searchpl(request): # 場所検索

```
    if  request.method == 'POST' :
        pl = request.POST['place']    # 'place'は PlForm 内のフィールド名
```

《《以下省略》》

■def searchfs(request): # 花の形検索

```
    if  request.method == 'POST' :
        fs = request.POST['flowershape']
            # 'flowershape'は FsForm 内のフィールド名
```

《《以下省略》》

■def searchif(request): # 花序検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchls(request): # 葉の形検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchll(request): # 葉序検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchsr(request): # 鋸歯検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchbk(request): # 樹皮検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchfm(request): # 科検索

```
    #  from  .forms  import
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchge(request): # 属検索

```
    if  request.method == 'POST' :
        ge = request.POST['genus']   #  'genus'はGeForm内のフィールド名
```

《《以下省略》》

■def searchdi(request): # 分布検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def searchde(request): # 説明検索

```
    if  request.method == 'POST' :
```

《《以下省略》》

■def first(request): # First 関数

```
    MainView.start = 0
    MainView.disptype = MainView.curstate['disptype']
    setstate2()
    return redirect('celldisplay')
```

■def prev(request): # Prev 関数

```
    MainView.start  -=  10
    if  MainView.start  <  0 :
        MainView.start  =  0
    MainView.disptype = MainView.curstate['disptype']
    setstate2()
    return redirect('celldisplay')
```

■def next(request): # Next 関数

```
    d = MainView.number - MainView.start
    if d > 10 :
        MainView.start  +=  10
        if MainView.start  >  MainView.number :
            MainView.start  =  MainView.number
    MainView.disptype = MainView.curstate['disptype']
    setstate2()
    return redirect('celldisplay')
```

■def last(request): # Last 関数

```
    MainView.start  =  MainView.number  -  10
    if MainView.start  <  0  :
        MainView.start  =  0
    MainView.disptype = MainView.curstate['disptype']
    setstate2()
    return redirect('celldisplay')
```

■def setstate(disptype, im):

```
#  メインページの現在状況設定。　何か所にもある同じコードを減らす目的の関数
   MainView.disptype = disptype
   MainView.curstate = {'disptype':MainView.disptype + im,
                        'total':str(MainView.number),
                        'startp':str(MainView.start)
                        }
```

■def setstate2():

```
#  メインページの現在状況設定。　何か所にもある同じコードを減らす目的の関数
   MainView.curstate = {'disptype':MainView.disptype,
                        'total':str(MainView.number),
                        'startp':str(MainView.start)
                        }
```

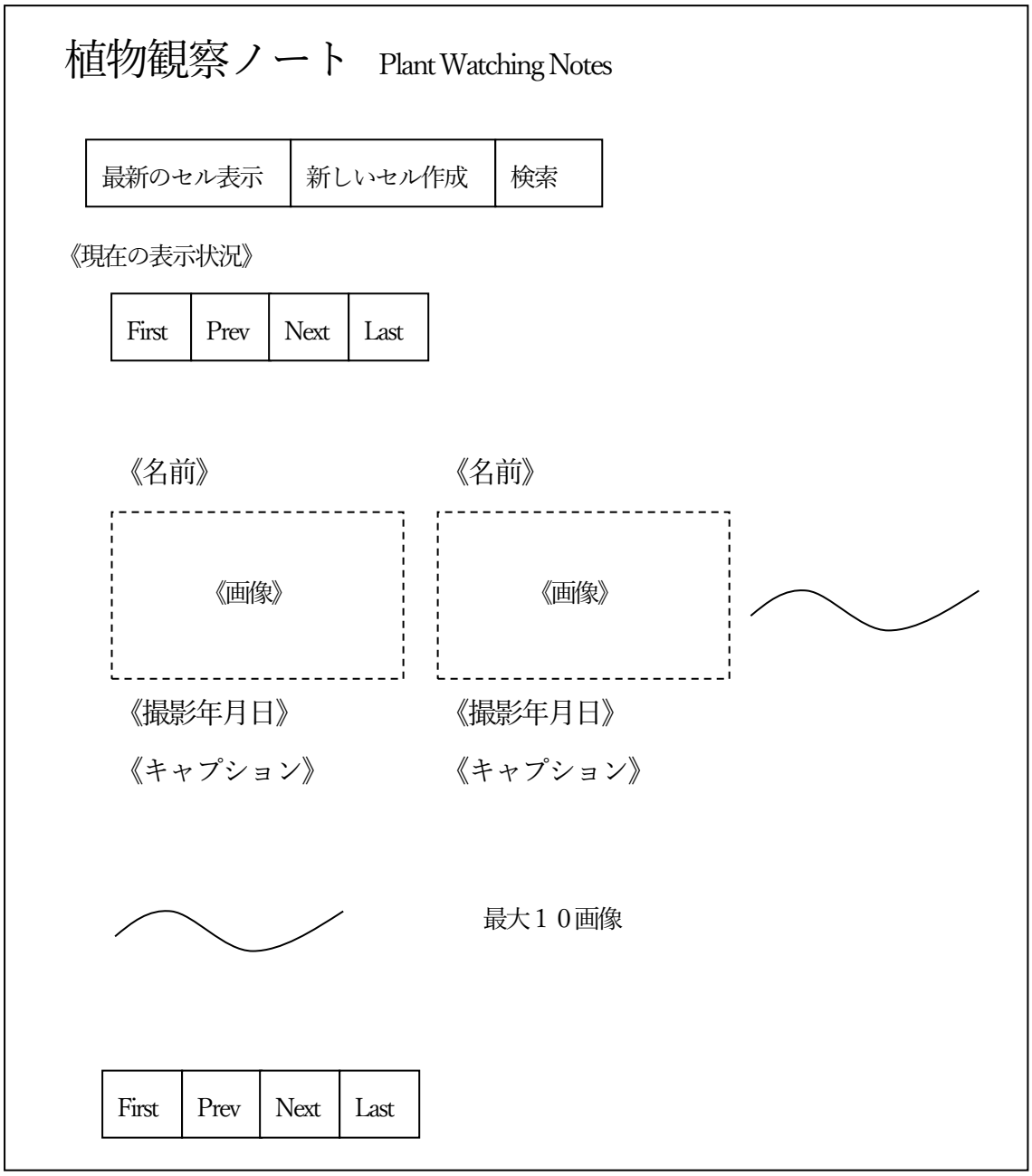
（4） テンプレート

次のページを用意する。

① メインページ	index.html
② セルの表示ページ	cell.html
③ イメージ表示ページ	cellimage.html
④ セルの作成・編集ページ	cellcreate.html celledit.html
⑤ 画像の登録・編集ページ	cupload.html eupload.html
⑥ 検索の選択ページ	searchkd.html
⑦ 検索ページ	search.html

■メインページ

index.html



✧ メニューの「最新のセル表示」をクリックすると、DB からすべてのレコードを読込んで最近登録したセルを最大１０個表示する。

✧ メニューの「新しいセル作成」をクリックすると、新たにセルを作成するためのページに移動する。

✧　メニューの「検索」をクリックすると、検索のためのページに移動する。

✧　メニュー

✧　《現在の表示状況》

【現在の表示状況】　《最新のセル》
セルの総数：xx
表示開始位置：xx

・最新のセルや検索結果などの状態を示す用語

— 名前検索：《用語》

— 日付検索：《MM.dd,YYY－MM.dd,YYYY》

— 場所検索：《用語》

— 花の形検索：《花の形》

— 花序検索：《花序》

— 葉の形検索：《葉の形》

— 葉序検索：《葉序》

— 鋸歯検索：《鋸歯》

— 樹皮検索：《樹皮》

— 科検索：《科》

— 属検索：《属》

— 分布検索：《用語》

— 説明検索：《用語》

✧　最大１０個の選択されたセルの代表的画像（各セル内の最初の画像）と《名前》、撮影年月日、キャプションが連続的に表示される。

✧　画像をクリックするとセルの内容が表示される。

✧ HTMLテキスト

```
{% load static %}
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="utf-8">
    <title>Plant Watching Notes</title>
    <link rel="stylesheet" type="text/css" href="{% static
'pwn/css/style.css' %}">
</head>
<body>
  <div id="container">
    <tr>
      <span style="color:green;font-size:32px;">植物観察ノート</span>
      <span style="color:black;font-family:century;">Plant Watching
Notes</span>
    </tr>
    <nav>
        <ul>
         <li class="first-menu"><a href="{% url 'cellallread' %}">最新
のセル表示</a></li>
         <li><a href="{% url 'cellcreate' 1 %}">新しいセル作成</a></li>
         <li><a href="{% url 'searchkd' %}">検索</a></li>
        </ul>
    </nav>
    <div style="clear:left;">
    <table>
    <tr><td>【現在の表示状況】</td><td>{{disptype}}</td></tr>
    <tr><td> </td><td>セルの総数：{{total}}</td>
    <tr><td> </td><td>表示開始位置：{{startp}}</td></tr>
    </table>
    </div>
    <div class="secmenu">
        <ul>
         <li class="first-menusec">
             <a href="{% url 'first' %}">First</a></li>
```

```
          <li><a href="{% url 'prev' %}">Prev</a></li>
          <li><a href="{% url 'next' %}">Next</a></li>
          <li><a href="{% url 'last' %}">Last</a></li>
         </ul>
    </div>
  <p style="clear:left;">  </p>
    <section>
    <article>
    <div class="matrix-container">
    {% for item in obj %}
       <div class="matrix-img">
       <ul style="list-style: none; padding-left: 0;">
          <li>{{item.name}}</li>
          <li><a href="{% url 'onecelldisp' item.posque %}">
           {% if item.pictype == 0 %}
               <img src="{{ item.url }}" width="180" height="121" />
           {% else %}
               <img src="{{ item.url }}" width="121" height="180" />
           {% endif %}
              </a>
          </li>
          <li>{{item.date}}</li>
          <li>{{item.caption}}</li>
       </ul>
       </div>
    {% endfor %}
    </div> <!-- end of matrix-container -->
   </article>
   </section>
   <div class="secmenu">
       <ul>
         <li class="first-menusec"><a href="{% url
'first' %}">First</a></li>
          <li><a href="{% url 'prev' %}">Prev</a></li>
          <li><a href="{% url 'next' %}">Next</a></li>
          <li><a href="{% url 'last' %}">Last</a></li>
       </ul>
   </div>
 </div>
</body>
</html>
```

《《その他のテンプレートについては省略》》

（5） スタイルシート　　style.css

```
#container {
  width:905px;
  margin:auto;
}
#main {
  width:705px;
  float:left;
  margin:20px 0;
}
#side {
  width:200px;
  float:right;
  margin:20px 0;
}
#footer {
  clear:both;
}

nav li {
    float : left ;
    width : 200px ;
    line-height : 30px ;
    list-style-type : none ;
    font-family : Verdana, Hevetica, sans-serif ;
    }
```

《《以下省略》》