MTVを理解しよう

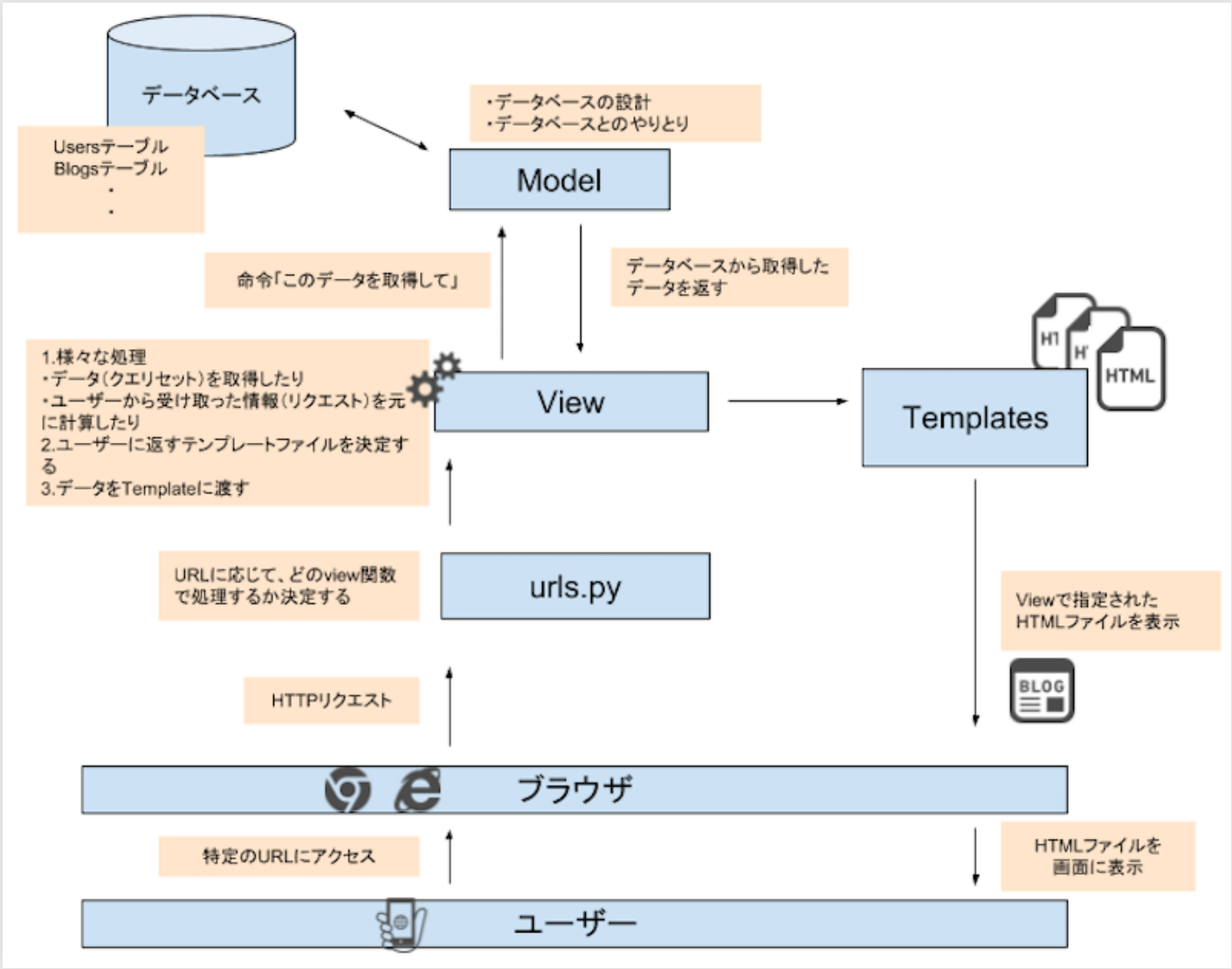
それでは、アプリのページを作成していきましょう。まずはトップページからです。

新たにページを作るときは、以下３つの作業が必要です。

1. URL設定　（urls.py）
2. View設定　(views.py)
3. Template設定 (html)

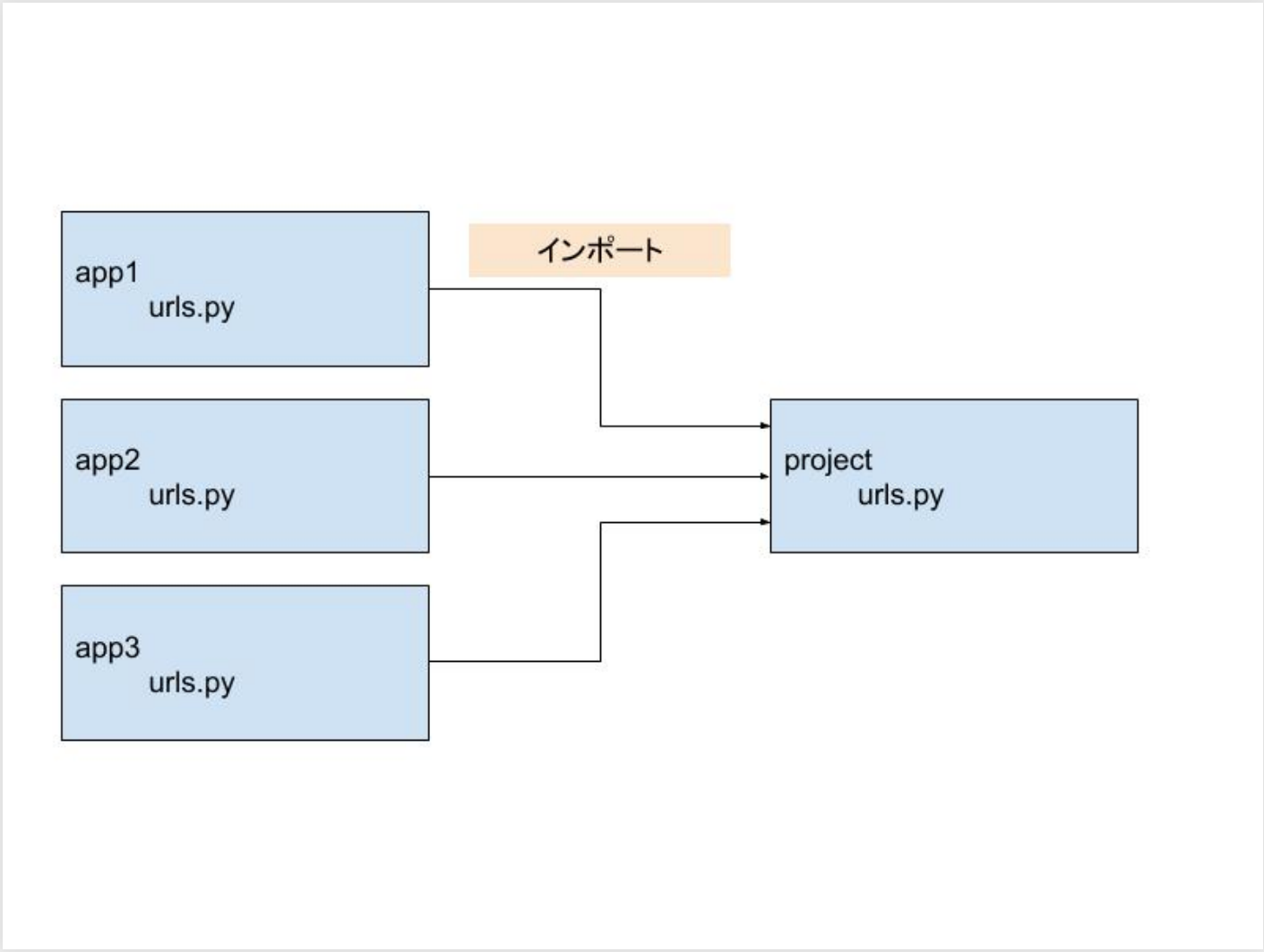
このレッスンでは、ページ作成に加えてモデルも作成します。Djangoプロジェクトは主にModel、Templates、Viewによって構成されますが、このことを**MTV**構成と呼びます。

MTVのイメージ図は以下の通りです。



URL設定

まずは、URLを設定しましょう。URLはプロジェクトディレクトリ（今回の場合は~/memo/memo）内のurls.pyで定義できます。しかし通常は、アプリディレクトリ内にもう１つurls.pyファイルを作成し、これをインポートする方法を取ることが多いです。また、複数のアプリを作成する場合はそれぞれのアプリごとにurls.pyを作成し、プロジェクトディレクトリのurls.pyでそれらをインポートします。



もちろん、urls.pyをアプリごとにに作らずに、プロジェクトディレクトリ内のurls.pyに全てのパスを記述しても正常に動作します。ただ、たくさんのパス設定が必要になる大規模なプロジェクトなどでは、１つのurls.pyファイルに全て記述するよりも、アプリごとに分割しておくことで可読性が高くなるなどのメリットがあります。

では、まずプロジェクトディレクトリ内のurls.pyファイルを編集しましょう。２行目で`include`をインポートすることを忘れないでください。

~/memo/memo/urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls')),
]
```

この記述により、`http://127.0.0.1:8000/`にアクセスされたときは、appディレクトリ内のurls.pyファイルを参照することを定義しました。

続いては、appディレクトリの中にurls.pyファイルを作成しましょう。作成したurls.pyファイルには以下のように記述してください。

~/memo/app/urls.py

```
from django.urls import path
from . import views

app_name = 'app'
urlpatterns = [
    path('', views.index, name='index')
]
```

これにより、`http://127.0.0.1:8000/`のパスで、views.pyファイル内のindex関数が実行されることになりました。当然、まだindex関数は作っていないので、今のままでアクセスしてもエラーになります。

path関数の第一引数はURLを指定しており、第二引数はそのURLに結びつけるviewを指定してます。この場合、appの中のindexというview関数に紐づけています。name引数はそれぞれのURLに名前を設定しており、他のviewから呼び出すときに利用できる名前になります。

View設定

Viewは、views.pyファイルで定義します。下の記述で、`templates/app/index.html`を表示するようになります。

~/memo/app/views.py

```
from django.shortcuts import render

def index(request):
    return render(request, 'app/index.html')
```

Template設定

最後にTemplateファイル（HTML)を作りましょう。Viewで設定したように、`templates/app/index.html`を作ります。

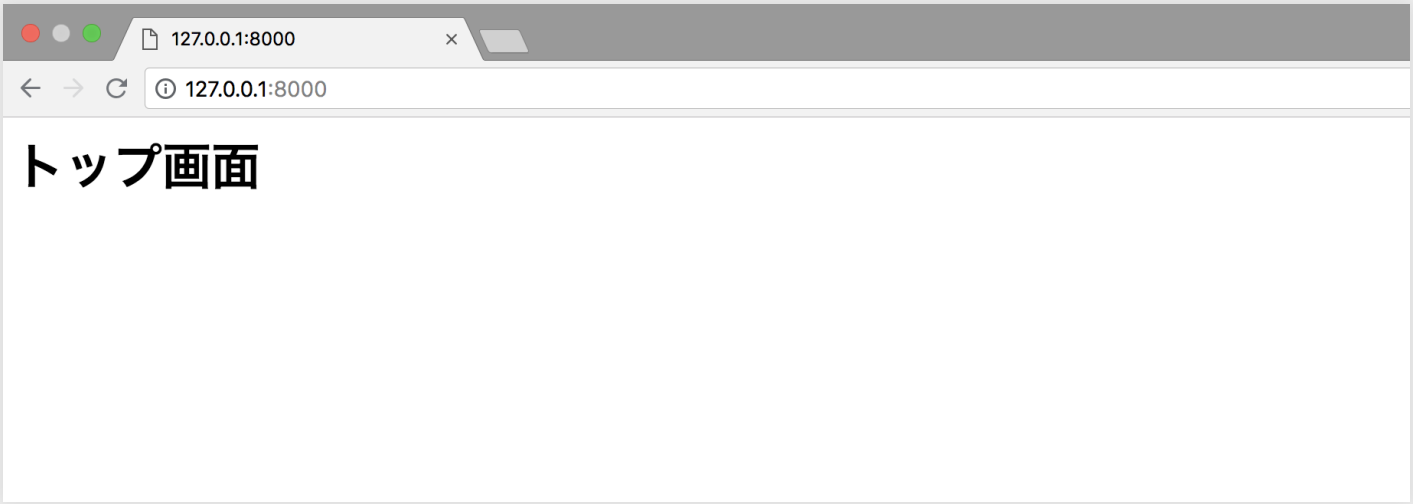
~/memo/app/

```
$ mkdir templates
$ cd templates
$ mkdir app
$ cd app
$ touch index.html
```

これで、`memo/app/templates/app/index.html`というフォルダ階層ができました。`index.html`には、ここがトップページであることがわかる適当な記述をしておきましょう。

~/memo/app/templates/app/index.html

```
<h1>トップ画面</h1>
```



どうしてtemplatesディレクトリの中にもう１つのディレクトリを作るのか？

上記では、`memo/app/templates/app/index.html`のように、templatesディレクトリ内にわざわざアプリ名と同名のディレクトリを作成して、その中にHTMLファイルを作っています。一見、`memo/app/templates/index.html`のようにしてもよさそうですが、どうしてこのようにしないのでしょうか。

views.pyファイル`render`関数の第2引数では、表示するテンプレートファイルを指定していますよね。このとき、Djangoデフォルトの設定ではアプリ内のtemplatesディレクトリを自動で参照するようになっています。よって、`'app/index.html'`という書き方で、`templates/app/index.html`を指定していることになるのです。

フォルダ構成を`memo/app/templates/index.html`のようにtemplates直下にindex.htmlファイルを置き、第2引数で`'index.html'`とする書き方もできます。

この方がシンプルで良さそうですが、この書き方だと、他のアプリ内にもindex.htmlがあった場合、意図しない結果となってしまうので望ましくありません。

例えば、以下のようにapp1とapp2という２つのアプリがあったとします。ここでは、それぞれがindex.htmlファイルという同名のファイルを保有しているとします。このとき、`render(request, 'index.html')`とすると、render関数は常にapp1内のtemplatesディレクトリを参照することになり、app2内のindex.htmlは指定できなくなります。

※仮にapp2の中にあるviews.pyで`render(request, 'index.html')`と記述してもapp1のindex.htmlを参照します。これは、テンプレートファイルを参照するときに、上のtemplatesディレクトリから探していくという決まりがあるからです。

（悪い例）

```
app1/
    templates/
        index.html
app2/
    templates/
        index.html
```

この状況を防ぐために、あえてtemplates内にディレクトリを作ることで、同名ファイルがあっても明示的に指し示せるようにします。

（良い例）

```
app1/
    templates/
        /app1
            index.html
app2/
    templates/
        /app2
            index.html
```

このようにすれば、`'app1/index.html'`や`'app2/index.html'`のように記述でき、それぞれのアプリ内に同名ファイルがあっても大丈夫になります。

ちなみにですが、app内のtemplatesを自動的に参照するのは、settings.pyファイルの`'APP_DIRS'`が **True**になっているからです。

~/memo/memo/settings.py

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

公式ドキュメントでは、こちらに簡単な説明があります。

Memoモデルを作る

最後に、モデルを作りましょう。モデルは設計図の役割を果たし、モデルがあることで同じような情報をもったインスタンスを簡単に作成することができます。

今回作るMemoモデルは、タイトル、テキスト（メモ内容）、作成日付、更新日付、の４つの情報を持たせます。

~/memo/app/models.py

```
from django.db import models

class Memo(models.Model):
    title = models.CharField(max_length=150)
    text = models.TextField(blank=True)
    created_datetime = models.DateTimeField(auto_now_add=True)
    updated_datetime = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title
```

CharFieldとTextFieldにおける、blankとnullの初期値はFalseです。つまり、投稿や保存をする際に空欄であったり、DB（データベース）に何も保存されていないという状況は許容されません。

タイトルが空欄なのはまずいので、titleフィールドでは何の変更も加えず初期値のままFalseとします。テキストに関しては空欄でも問題ないこととし、今回はblank=Trueを指定しています。

TextFieldは、balnk=Trueを指定すれば、null=Trueを指定する必要はありません。その理由は、こちらの記事に書いています。

モデルの設計ができたら、マイグレーションファイルを作ってマイグレートしましょう！

~/memo

```
# マイグレーションファイル作成
$ python manage.py makemigrations

# マイグレート（マイグレーションファイルの情報をDBに反映）
$ python manage.py migrate
```

これでDBが作成できているはずなので、この情報がAdminページで見ることができるように設定します。

~/memo/app/admin.py

```
from django.contrib import admin
from .models import Memo

class MemoAdmin(admin.ModelAdmin):
    list_display = ('id', 'title', 'created_datetime', 'updated_datetime')
    list_display_links = ('id', 'title')

admin.site.register(Memo, MemoAdmin)
```

これでこのレッスンは終了です！ここでやったことは、ページを作るたびに必要になる作業ですのでしっかりMTVの流れを押さえておきましょう！