



리액트와 함께 장고 시작하기 / 장고 Views

# 다양한 응답의 함수 기반 뷰

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

# View

1개의 HTTP 요청에 대해 ➔ 1개의 뷰가 호출

urls.py/urlpatterns 리스트에 매핑된 호출 가능한 객체

함수도 "호출 가능한 객체" 중의 하나

웹 클라이언트로부터의 HTTP 요청을 처리

크게 2가지 형태의 뷰

함수 기반 뷰 (Function Based View) : 장고 뷰의 기본.

호출 가능한 객체. 그 자체

클래스 기반 뷰 (Class Based View)

클래스.as\_view() 를 통해 호출가능한 객체를 생성/리턴

# View 호출 시, 인자

HttpRequest 객체 및 URL Captured Values

## 1번째 인자 : HttpRequest 객체

현재 요청에 대한 모든 내역을 담고 있습니다.

## 2번째~ 인자 : 현재 요청의 URL로부터 Capture된 문자열들

url/re\_path 를 통한 처리에서는 → 모든 인자는 str 타입으로 전달

path 를 통한 처리에서는 → 매핑된 Converter의 to\_python에 맞게 변환된 값이 인자로 전달

지난 에피소드의 4자리 년도를 위한 FourDigitYearConverter에서는 int 변환 → 뷰의 인자로 int 타입의 년도가 전달

# View 호출에 대한 리턴값

HttpResponse 객체

필히 HttpResponse 객체를 리턴해야 합니다.

장고 Middleware에서는 뷰에서 HttpResponse 객체를 리턴하기를 기대합니다. ➔ 다른 타입을 리턴하면 Middleware에서 처리 오류.

django.shortcuts.render 함수는 템플릿 응답을 위한 shortcut 함수

파일like객체 혹은 str/bytes 타입의 응답 지원

str 문자열을 직접 utf8로 인코딩할 필요가 없습니다.

장고 디폴트 설정에서 str 문자열을 utf8로 인코딩해줍니다.

response = HttpResponse( 파일like객체 또는 str객체 또는 bytes객체 )

파일 like 객체

response.write( str객체 또는 bytes객체 )

# HttpRequest와 HttpResponse 예시

```
from django.http import HttpRequest, HttpResponse
```

```
def index(request: HttpRequest) -> HttpResponse: # View 함수
    # 주요 request 속성
    request.method # 'GET', 'POST', etc.
    request.META
    request.GET, request.POST, request.FILES, request.body

    content = '''
        <html>...</html>
    ''' # 문자열 혹은 이미지, 각종 파일 등

    response = HttpResponse(content)
    response.write(content) # response -> file-like object
    response['Custom-Header'] = 'Custom Header Value'
    return response
```

# FBV의 예

Item 목록 보기

```
# myapp/views.py
```

```
from django.shortcuts import render
```

```
from shop.models import Item
```

```
def item_list(request):
```

```
    qs = Item.objects.all()
```

```
    return render(request, 'shop/item_list.html', {
```

```
        'item_list': qs,
```

```
    })
```

```
# myapp/urls.py
```

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

```
urlpatterns = [
```

```
    path('items/', item_list, name='item_list'),
```

```
]
```

# 살짝 구성해본 클래스 기반의 호출 가능한 객체

(실제 CBV와는 다른 구성)

```
from django.shortcuts import render
from shop.models import Item
```

```
class GenericListView:
    def __init__(self, model_cls):
        self.model_cls = model_cls

    def get_list_name(self):
        return '{}_list'.format(
            self.model_cls._meta.model_name)

    def get_template_name(self):
        return self.model_cls._meta.app_label + '/' + \
            self.get_list_name() + '.html'

    def get_queryset(self):
        return self.model_cls.objects.all()

    def get_context(self):
        return {
            self.get_list_name(): self.get_queryset(),
        }
```

```
def __call__(self, request):
    context = self.get_queryset()
    return render(request, self.get_template_name(),
                  self.get_context())
```

```
item_list = GenericListView(Item)
```

```
from django.urls import path
urlpatterns = [
    path('items/', item_list, name='item_list'),
]
```

# CBV의 예

## Item 목록 보기

```
from django.views.generic import ListView
from shop.models import Item
```

```
item_list = ListView.as_view(model=Item)
```

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

```
urlpatterns = [
    path('items/', item_list, name='item_list'),
]
```

```
from django.views.generic import ListView
from shop.models import Item
```

```
class ItemListView(ListView):
    model = Item
```

```
item_list = ItemListView.as_view()
```

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

```
urlpatterns = [
    path('items/', item_list, name='item_list'),
]
```



# 다양한 타입의 HttpResponse

# Excel 파일 다운로드 응답

```
from django.http import HttpResponse
from urllib.parse import quote
```

```
def response_excel(request):
```

```
    filepath = '/other/path/excel.xls'
```

```
    filename = os.path.basename(filepath)
```

파일like객체를 지정하면, 내부적으로 읽기를 시도  
구지 f.read()를 호출할 필요 X

```
    with open(filepath, 'rb') as f:
```

```
        response = HttpResponse(f, content_type='application/vnd.ms-excel')
```

```
    # 브라우저에 따라 다른 처리가 필요합니다.
```

```
    encoded_filename = quote(filename)
```

```
    response['Content-Disposition'] = "attachment; filename*=utf-8'{}".format(encoded_filename)
```

```
    return response
```

ref) [django.http.FileResponse](#) 를 통해 첨부 헤더 지원

참고 1) [프로그램적인 파일 다운로드](#)

참고 2) [\[Browser\] Content-Disposition 헤더의 filename 매개 변수를 HTTP로 인코딩하는 방법은 무엇입니까?](#)

# Pandas를 통한 CSV 응답 생성

필요한 라이브러리 : pandas

```
import pandas as pd
from io import StringIO
from django.http import HttpResponse

def response_csv(request):
    df = pd.DataFrame([
        [100, 110, 120],
        [200, 210, 220],
        [300, 310, 320],
    ])

    io = StringIO()
    df.to_csv(io)
    io.seek(0) # 끝에 있는 file cursor를 처음으로 이동

    response = HttpResponse(io, content_type='text/csv')
    response['Content-Disposition'] = "attachment; filename*=utf-8' '{}".format(encoded_filename)
    return response
```

# Pandas를 통한 엑셀 응답 생성

필요한 라이브러리 : pandas, xlwt

```
import pandas as pd
from io import BytesIO
from urllib.parse import quote
from django.http import HttpResponse

def response_excel(request):
    df = pd.DataFrame([
        [100, 110, 120],
        [200, 210, 220],
    ])

    io = BytesIO()
    df.to_excel(io)
    io.seek(0)

    encoded_filename = quote('pandas.xlsx')
    response = HttpResponse(io, content_type='application/vnd.ms-excel')
    response['Content-Disposition'] = "attachment; filename*=utf-8'{}".format(encoded_filename)
    return response
```

# Pillow를 통한 이미지 응답 생성 - 기본

필요한 라이브러리 : pillow, requests

```
import requests
from io import BytesIO
from PIL import Image, ImageDraw, ImageFont

ttf_path = 'C:/Windows/Fonts/malgun.ttf'          # 윈도우의 맑은고딕 폰트 경로, 맥에서는 애플고딕 경로 → '/Library/Fonts/AppleGothic.ttf'

image_url = 'http://www.flowermeaning.com/flower-pics/Calla-Lily-Meaning.jpg'

res = requests.get(image_url)                    # 서버로 HTTP GET 요청하여, 응답 획득
io = BytesIO(res.content)                       # 응답의 Raw Body. 메모리 파일 객체 BytesIO 인스턴스 생성
io.seek(0)                                       # 파일의 처음으로 커서를 이동

canvas = Image.open(io).convert('RGBA' )        # 이미지 파일을 열고, RGBA모드로 변환

font = ImageFont.truetype(ttf_path, 40)         # 지정 경로의 TrueType 폰트, 폰트크기 40
draw = ImageDraw.Draw(canvas)                  # canvas에 대한 ImageDraw 객체 획득

text = 'Ask Company'
left, top = 10, 10
margin = 10
width, height = font.getsize(text)
right = left + width + margin
bottom = top + height + margin

draw.rectangle((left, top, right, bottom), (255, 255, 224))
draw.text((15,15), text, font=font, fill=(20, 20, 20))

canvas.show()
```

# Pillow를 통한 이미지 응답 생성 - View

```
import requests
from io import BytesIO
from PIL import Image, ImageDraw, ImageFont

def response_pillow_image(request):
    ttf_path = 'C:/Windows/Fonts/malgun.ttf'          # 윈도우의 맑은고딕 폰트 경로, 맥에서는 애플고딕 경로 → '/Library/Fonts/AppleGothic.ttf'

    # 이미지 파일 다운로드 혹은 로컬 디스크 상의 이미지 직접 열기
    image_url = 'http://www.flowermeaning.com/flower-pics/Calla-Lily-Meaning.jpg'
    res = requests.get(image_url)                    # 서버로 HTTP GET 요청하여, 응답 획득
    io = BytesIO(res.content)                        # 응답의 Raw Body 메모리 파일 객체 BytesIO 인스턴스 생성
    io.seek(0)                                       # 파일의 처음으로 커서를 이동

    canvas = Image.open(io).convert('RGBA')          # 이미지 파일을 열고, RGBA 모드로 변환
    font = ImageFont.truetype(ttf_path, 40)          # 지정 경로의 TrueType 폰트, 폰트크기 40
    draw = ImageDraw.Draw(canvas)                   # canvas에 대한 ImageDraw 객체 획득

    text = 'Ask Company'
    left, top = 10, 10
    margin = 10
    width, height = font.getsize(text)
    right = left + width + margin
    bottom = top + height + margin
    draw.rectangle((left, top, right, bottom), (255, 255, 224))
    draw.text((15, 15), text, font=font, fill=(20, 20, 20))

    response = HttpResponse(content_type='image/png')
    canvas.save(response, format='PNG')              # HttpResponse 의 file-like 특성 활용
    return response
```

Life is short.  
You need Python and Django.

I will be your pacemaker.

