Ask Company

리액트와 함께 장고 시작하기 / 장고 Views

적절한 HTTP 상태코드로 응답하기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

HTTP 상태코드

웹서버는 적절한 상태코드로서 응답해야 합니다.

각 HttpResponse 클래스마다 고유한 status_code가 할당 (코드)

REST API를 만들 때, 특히 유용

```
# django/http/response.py
class HttpResponseRedirect(HttpResponseRedirectBase):
    status_code = 302
```

```
from django.http import HttpResponse

def test_view(request):
    # Return a "created" (201) response code.
    return HttpResponse(status=201)
```

대표적인 상태 코드

200번대 : 성공

200 : 서버가 요청을 잘 처리했다. ➔ OK

201 : 작성됨. 서버가 요청을 접수하고, 새 리소스를 작성했다.

300번대 : 요청을 마치기 위해, 추가 조치가 필요하다.

301 : 영구 이동, 요청한 페이지가 새 위치로 영구적으로 이동했다.

302 : 임시 이동, 페이지가 현재 다른 위치에서 요청에 응답하고 있지만, 요청자는 향후 원래 위치를 계속 사용해야 한다.

400번대 : 클라이언트측 오류

400 : 잘못된 요청.

401 : 권한없음.

403 (Forbidden) : 필요한 권한을 가지고 있지 않아서, 요청을 거부

404 : 서버에서 요청한 리소스를 찾을 수 없다.

405 : 허용되지 않는 방법. POST 방식만을 지원하는 뷰에 GET요청을 할 경우

500번대 : 서버측 오류

500 : 서버 내부 오류 발생

200 응답하는 몇 가지 예

```python
from django.http import HttpResponse, JsonResponse
from django.shortcuts import render

def view1(request):
    return HttpResponse('Hello, Ask Company')

def view2(request):
    return render(request, 'template.html')

def view3(request):
    return JsonResponse({'hello': 'Ask Company'})
```

302 응답하는 몇 가지 예

```python
from django.http import HttpResponseRedirect
from django.shortcuts import redirect, resolve_url


def view1(request):
    return HttpResponseRedirect('/shop/')

def view2(request):
    url = resolve_url('shop:item_list')  # 후에 배울 URL Reverse 적용
    return HttpResponseRedirect(url)

def view3(request):
    # 내부적으로 resolve_url 사용
    # 인자로 지정된 문자열이 url reverse에 실패할 경우,
    # 그 문자열을 그대로 URL로 사용하여, redirect 시도
    return redirect('shop:item_list')
```

404 응답하는 몇 가지 예

```python
from django.http import Http404, HttpResponseNotFound
from django.shortcuts import get_object_or_404
from shop.models import Item

def view1(request):
    try:
        item = Item.objects.get(pk=100)
    except Item.DoesNotExist:
        raise Http404
    # 생략

def view2(request):
    item = get_object_or_404(Item, pk=100)  # 내부에서 raise Http404
    # 생략

def view3(request):
    try:
        item = Item.objects.get(pk=100)
    except Item.DoesNotExist:
        return HttpResponseNotFound()       # 잘 쓰지 않는 방법
    # 생략
```

500 응답하는 몇 가지 예

뷰에서 요청 처리 중에, 뷰에서 **미처 잡지못한 오류**가 발생했을 경우

IndexError, KeyError, django.db.models.ObjectDoesNotExist 등

```
from shop.models import Item

def view1(request):
    # IndexError
    name = ['Tom', 'Steve'][100]

    # 지정 조건의 Item 레코드가 없을 때, Item.DoesNotExist 예외
    # 지정 조건의 Item 레코드가 2개 이상 있을 때, Item.MultipleObjectsReturned 예외
    item = Item.objects.get(pk=100)
```

다양한 HttpResponse 서브 클래스

지정 상태코드의 응답이 필요할 때

HttpResponse**Redirect** : 상태코드 302

HttpResponse**PermanentRedirect** : 상태코드 301 (영구 이동)

HttpResponse**NotModified** : 상태코드 304

HttpResponse**BadRequest** : 상태코드 400

HttpResponse**NotFound** : 상태코드 404

HttpResponse**Forbidden** : 상태코드 403

HttpResponse**NotAllowed** : 상태코드 405

HttpResponse**Gone** : 상태코드 410

HttpResponse**ServerError** : 상태코드 500

Life is short.
You need Python and Django.

I will be your pacemaker.