

장고 Form/ModelForm 제대로 알고 쓰기

당신의 파이썬/장고 페이스메이커가 되겠습니다. ;)

EP02. HttpRequest와 HttpResponse

HttpRequest

HttpRequest 객체

- 클라이언트로부터의 모든 요청 내용을 담고 있으며
 - 함수 기반 뷰 : 매 요청 시마다 뷰 함수의 첫번째 인자 request로 전달
 - 클래스 기반 뷰 : 매 요청 시마다 self.request를 통해 접근
- Form 처리 관련 속성들
 - .method : 요청의 종류 "GET" 또는 "POST" 로서 모두 대문자
 - .GET : GET 인자 목록 (QueryDict타입)
 - .POST : POST 인자 목록 (QueryDict타입)
 - .FILES : POST 인자 중에서 파일 목록 (MultiValueDict 타입)

MultiValueDict (1)

- dict을 상속받은 클래스
- 동일 key의 다수 value를 지원하는 사전
- URL의 QueryString은 같은 Key로서 다수 Value지정을 지원
 - ex) name=Tom&name=Steve&name=Tom

MultiValueDict (2)

동일 Key의 다수 Value를 지원하는 사전

```
>>> from django.utils.datastructures import MultiValueDict
>>> d = MultiValueDict({'name': ['Adrian', 'Simon'], 'position': ['Developer']})
>>> d['name']                                     # dict과 동일하게 동작. 단일값을 획득
'Simon'

>>> d.getlist('name')                            # 다수값 획득을 시도. 리스트를 반환
['Adrian', 'Simon']
>>> d.getlist('doesnotexist') # 없는 Key에 접근하면 빈 리스트를 반환
[]
>>> d['name'] = 'changed'
>>> d
<MultiValueDict: {'name': ['changed'], 'position': ['Developer']}>
```

<https://github.com/django/django/blob/2.1/django/utils/datastructures.py#L43>

MultiValueDict (3)

수정 불가능한 (Immutable) 특성

아래 코드는 Django Shell을 통해서 실행이 가능

```
>>> from django.http import QueryDict
>>> qd = QueryDict('name=Adrian&name=Simon&position=Developer', encoding='utf8')
>>> qd['name']
'Simon'
>>> qd.getlist('name')
['Adrian', 'Simon']

>>> qd['name'] = 'changed'
AttributeError: This QueryDict instance is immutable
```

<https://docs.djangoproject.com/en/2.1/ref/request-response/#django.http.QueryDict>

QueryDict

- 수정불가능한 MultiValueDict

```
class QueryDict(MultiValueDict):
    _mutable = True
    _encoding = None

    def __init__(self, query_string=None, mutable=False, encoding=None):
        # ...
        self._mutable = mutable

    def _assert_mutable(self):
        if not self._mutable:
            raise AttributeError("This QueryDict instance is immutable")

    def __setitem__(self, key, value):
        self._assert_mutable()
        # ...

    def __delitem__(self, key):
        self._assert_mutable()
        # ...

    # ...
```

<https://docs.djangoproject.com/en/2.1/ref/request-response/#django.http.QueryDict>

HttpResponse

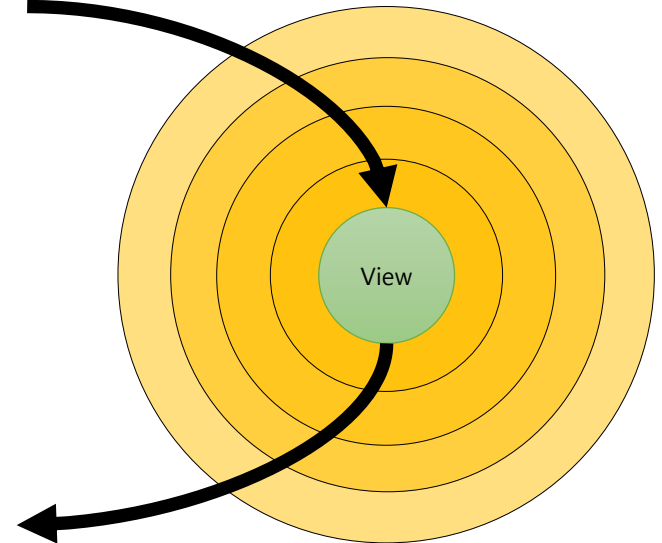
django.http.HttpResponse (1)

- 다양한 응답을 Wrapping : HTML문자열, 이미지 등등
- View에서는 반환값으로서 HttpResponse 객체를 기대
 - → Middleware에서 HttpResponse 객체를 기대

프로젝트/settings

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

HttpRequest 객체 전달



HttpResponse 객체 전달

<https://docs.djangoproject.com/en/2.1/ref/request-response/#httpresponse-objects>

django.http.HttpResponse (2)

파일-like 객체

- .write(content)
- .flush()
- .tell()

```
response = HttpResponse(  
    "<p>Here's the text of the Web page.</p>"  
    "<p>Here's another paragraph.</p>"  
)
```

```
response = HttpResponse()  
response.write("<p>Here's the text of the Web page.</p>")  
response.write("<p>Here's another paragraph.</p>")
```

django.http.HttpResponse (3)

- 사전-like 인터페이스로 응답의 커스텀 헤더 추가/삭제

```
response = HttpResponse()  
response['Age'] = 120  
del response['Age']
```

- 파일 첨부로 처리되기를 브라우저에게 알리기

```
response = HttpResponse(excel_data, content_type='application/vnd.ms-excel')  
response['Content-Disposition'] = 'attachment; filename="foo.xls"'
```

다양한 HttpResponse 서브 클래스

지정 상태코드의 응답이 필요할 때

- HttpResponse**Redirect** : 상태코드 302
- HttpResponse**PermanentRedirect** : 상태코드 301
- 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

<https://docs.djangoproject.com/en/2.1/ref/request-response/#httpresponse-subclasses>

django.http.JsonResponse

```
from django.core.serializers.json import DjangoJSONEncoder

class JsonResponse(HttpResponse):
    def __init__(self, data, encoder=DjangoJSONEncoder, safe=True,
                 json_dumps_params=None, **kwargs):
        if safe and not isinstance(data, dict):
            raise TypeError(
                'In order to allow non-dict objects to be serialized set the '
                'safe parameter to False.')
        if json_dumps_params is None:
            json_dumps_params = {}
        kwargs.setdefault('content_type', 'application/json')
        data = json.dumps(data, cls=encoder, **json_dumps_params)
        super().__init__(content=data, **kwargs)
```

```
import json

class DjangoJSONEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, datetime.datetime):
            r = o.isoformat()
            if o.microsecond:
                r = r[:23] + r[26:]
            if r.endswith('+00:00'):
                r = r[:-6] + 'Z'
            return r
        elif isinstance(o, datetime.date):
            return o.isoformat()
        elif isinstance(o, datetime.time):
            if is_aware(o):
                raise ValueError("JSON can't represent timezone-"
                                   "aware times.")
            r = o.isoformat()
            if o.microsecond:
                r = r[:12]
            return r
        elif isinstance(o, datetime.timedelta):
            return duration_iso_string(o)
        elif isinstance(o, (decimal.Decimal, uuid.UUID, Promise)):
            return str(o)
        else:
            return super().default(o)
```

<https://docs.djangoproject.com/en/2.1/ref/request-response/#jsonresponse-objects>

django.http.StreamingHttpResponse (1)

- 효율적인 큰(긴) 응답을 위함.
 - 혹은 메모리를 많이 먹는 응답 → iterator를 통한 응답
- 하지만, Django는 short-lived 요청에 맞게 디자인
 - 큰(긴) 응답 시에는 극심한 성능 저하로 이어질 수 있습니다.
- HttpResponse를 상속받지 않음.
 - 필히 iterator를 지정해야만, 제대로 동작
 - .content 속성 사용 X → .streaming_content 사용
 - .tell(), .write() 사용 X

<https://docs.djangoproject.com/en/2.1/ref/request-response/#streaminghttpresponse-objects>

django.http.StreamingHttpResponse (2)

```
class StreamingHttpResponse(HttpResponseBase):
    streaming = True

    def __init__(self, streaming_content=(), *args, **kwargs):
        super().__init__(*args, **kwargs)
        # `streaming_content` should be an iterable of bytestrings.
        # See the `streaming_content` property methods.
        self.streaming_content = streaming_content

    @property
    def content(self):
        raise AttributeError(
            "This %s instance has no `content` attribute. Use "
            "`streaming_content` instead." % self.__class__.__name__
        )

    @property
    def streaming_content(self):
        return map(self.make_bytes, self._iterator)
```

```
@streaming_content.setter
def streaming_content(self, value):
    self._set_streaming_content(value)

    def _set_streaming_content(self, value):
        # Ensure we can never iterate on "value" more than once.
        self._iterator = iter(value)
        if hasattr(value, 'close'):
            self._closable_objects.append(value)

    def __iter__(self):
        return self.streaming_content

    def getvalue(self):
        return b''.join(self.streaming_content)
```

django.http.StreamingHttpResponse (3)

Streaming large CSV files

```
import csv

from django.http import StreamingHttpResponse

class Echo:
    """An object that implements just the write method of the file-like interface."""
    def write(self, value):
        """Write the value by returning it, instead of storing in a buffer."""
        return value

def some_streaming_csv_view(request):
    """A view that streams a large CSV file."""
    # Generate a sequence of rows. The range is based on the maximum number of
    # rows that can be handled by a single sheet in most spreadsheet applications.
    rows = (["Row {}".format(idx), str(idx)] for idx in range(65536))    # Generator Expression
    pseudo_buffer = Echo()
    writer = csv.writer(pseudo_buffer)
    response = StreamingHttpResponse((writer.writerow(row) for row in rows), content_type="text/csv")
    response['Content-Disposition'] = 'attachment; filename="somefilename.csv"'
    return response
```

```
In [1]: import csv

In [2]: class Echo:
...:     def write(self, value):
...:         return value
...:

In [3]: pseudo_buffer = Echo()

In [4]: pseudo_buffer.write("hello")
Out[4]: 'hello'

In [5]: writer = csv.writer(pseudo_buffer)

In [6]: writer.writerow(["Row 0", str(0)])
Out[6]: 'Row 0,0\r\n'

In [7]: writer.writerow(["Row 1", str(1)])
Out[7]: 'Row 1,1\r\n'

In [8]: writer.writerow(["Row 2", str(2)])
Out[8]: 'Row 2,2\r\n'
```

<https://docs.djangoproject.com/en/2.1/howto/outputting-csv/#streaming-csv-files>

django.http.FileResponse

FileResponse(open_file, as_attachment=False, filename='', **kwargs)

- StreamingHttpResponse를 상속받음.
 - 파일 내용 응답에 최적화
 - Content-Length, Content-Type, Content-Disposition 헤더 자동 지정
- 인자
 - open_file : Streaming Content
 - as_attachment : Content-Disposition 헤더 지정 여부
 - filename

```
from django.http import FileResponse
```

```
response = FileResponse(open('myfile.png', 'rb'))
```

```
response = FileResponse(open('myfile.pdf', 'rb'), as_attachment=True)
```

<https://docs.djangoproject.com/en/2.1/ref/request-response/#fileresponse-objects>

shortcuts 함수

django.shortcuts.render

`render(request, template_name, context=None, content_type=None, status=None, using=None) -> HttpResponse`

- context : 템플릿 context에 추가할 값 목록 (dict)
- content_type : 응답의 MIME Type (디폴트: "text/html")
- status : 응답의 상태 코드 (디폴트: 200)
- using : 템플릿 엔진 지정

<https://docs.djangoproject.com/en/2.1/topics/http/shortcuts/#render>

django.shortcuts.redirect

redirect(**to**, permanent=False, *args, **kwargs) -> HttpResponseRedirect

- to
 1. get_absolute_url()이 구현된 모델 객체
 2. URL Reverse를 수행할 문자열
 3. 직접 지정한 URL 문자열
- permanent
 - True : HttpResponseRedirect (301 응답)
 - False : HttpResponseRedirect (302 응답)

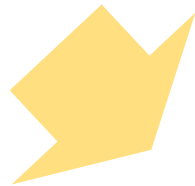
<https://docs.djangoproject.com/en/2.1/topics/http/shortcuts/#redirect>

django.shortcuts.get_object_or_404 (1)

- Model.DoesNotExist 예외 대신에 404 응답 발생

```
from django.http import Http404
```

```
def my_view(request):  
    try:  
        obj = MyModel.objects.get(pk=1)  
    except MyModel.DoesNotExist:  
        raise Http404("No MyModel matches the given query.")
```



```
from django.shortcuts import get_object_or_404
```

```
def my_view(request):  
    obj = get_object_or_404(MyModel, pk=1)
```

django.shortcuts.get_object_or_404 (2)

Model

```
get_object_or_404(Book, title__startswith='M', pk=1)
```

QuerySet

```
queryset = Book.objects.filter(title__startswith='M')  
get_object_or_404(queryset, pk=1)
```

Custom Manager

```
get_object_or_404(Book.dahl_objects, title='Matilda')
```

Related Manager

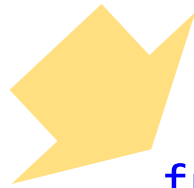
```
author = Author.objects.get(name='Roald Dahl')  
get_object_or_404(author.book_set, title='Matilda')
```

django.shortcuts.get_list_or_404

- 지정 QuerySet 조건에 대해 empty일 경우, 404 응답 발생

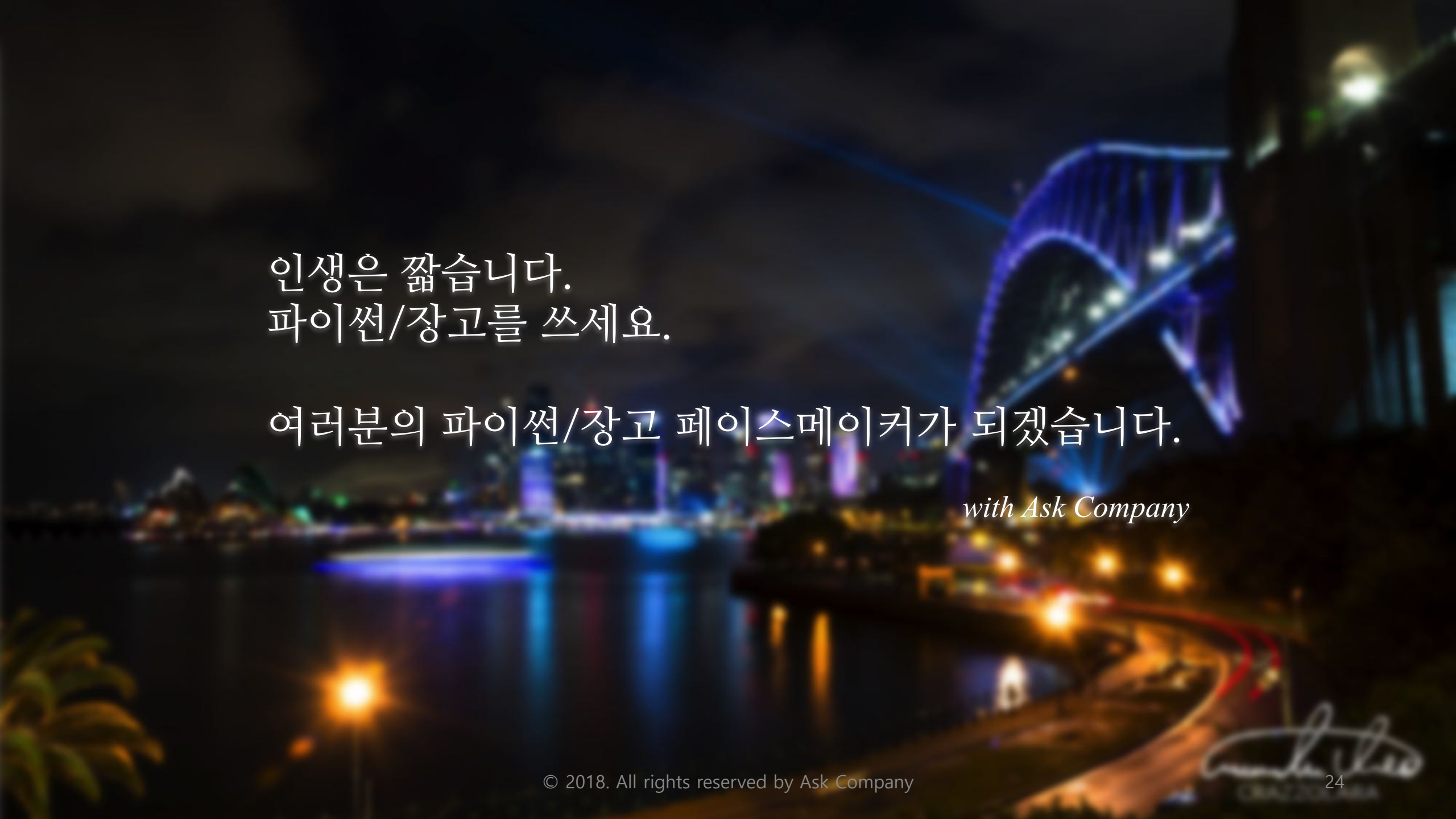
```
from django.http import Http404
```

```
def my_view(request):  
    my_objects = list(MyModel.objects.filter(published=True))  
    if not my_objects:  
        raise Http404("No MyModel matches the given query.")
```



```
from django.shortcuts import get_list_or_404
```

```
def my_view(request):  
    my_objects = get_list_or_404(MyModel, published=True)
```

A nighttime photograph of a cityscape featuring a bridge with blue and white lights. The lights are reflected in the water below. The sky is dark, and the overall scene is illuminated by the city lights.

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

with Ask Company