HttpRequest와 HttpResponse

장고 서버는 request와 response 객체로 서버와 클라이언트가 정보를 주고받으며 이를 위해 장고 서버는 django.http 모듈에서 HttpRequest와 HttpResponse API를 제공한다.

서버-클라이언트 통신 시 아래와 같은 절차로 데이터가 오고 간다.

1) 특정 페이지가 요청되면, 장고 서버는 요청 시 메타데이터(여러 다양한 정보)를 포함하는 HttpRequest 객체를 생성한다.
2) 장고 서버는 urls.py에 정의한 뷰 함수에 첫 번째 인자로 생성된 HttpRequest 객체를 전달한다.
3) 해당 뷰는 처리 결과값을 HttpResponse 혹은 JsonResponse 객체에 담아 응답한다.

- HttpRequest 객체

1) 주요 속성(Attribute)

HttpRequest.body # request의 body 객체
HttpRequest.headers # request의 headers 객체
HttpRequest.COOKIES # 모든 쿠키를 담고 있는 딕셔너리 객체
HttpRequest.method # 요청 메서드 정보
HttpRequest.GET # GET 파라미터를 담고 있는 딕셔너리 같은 객체
HTTpRequest.POST # POST 파라미터를 담고 있는 딕셔너리 같은 객체

[사용 사례 : request.method]

```
if request.method == 'GET':
    do_something()
elif request.method == 'POST':
    do_something_else()
```

2) 주요 메소드(Methods)

HttpRequest.read()
HttpRequest.get_host()
HttpRequest.get_port()

- HttpResponse

HttpResponse(data, content_type)
response를 반환하는 가장 기본적인 함수로서 주로 html를 반환
string 전달하기

```
HttpResponse("Here's the text of the Web page.")
```

html 태그 전달하기

```
HttpResponse("<h1>Here's the text of the Web page.</h1>")
```

- HttpRedirect

HttpResponseRedirect(url)

url에 지정된 페이지로 redicrect(재요청)를 한다.

첫 번째 인자로 url를 반드시 지정해야 하며, 경로는 절대경로 혹은 상대경로 모두 이용할 수 있다.

- Render

render(request객체(필수), template_name(필수), context=None)

render()는 httpRespose 객체를 반환하는 함수로 template을 context와 엮어 HttpResponse로 쉽게 반환해 주는 함수이다.

template_name에는 불러오고 싶은 템플릿명을 작성한다.

context에는 뷰에서 생성된 변수(dictionary 자료형)를 html 템플릿에서 전달하는 역할을 한다.

```
# views.py
def myview(request):
    context = { 'name':name, 'address':address }
                return render(request, 'exam5.html', context)
```

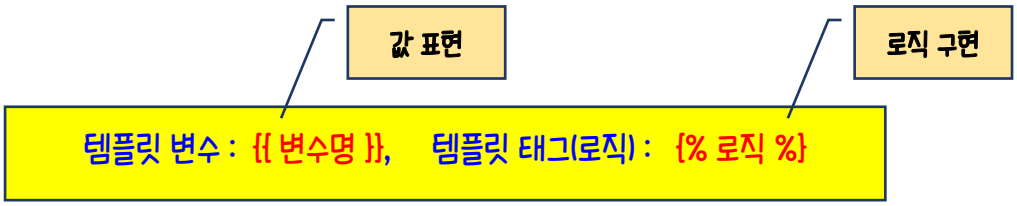
- JsonResponse

HTML 대신 JSON 형식으로 응답하려는 경우에 사용한다.

```
        jsonContent = {"pid": pid, "count": 1}
        return JsonResponse(jsonContent)

        return JsonResponse({ "time" : datetime.now().strftime('%H시 %M분 %S초') },
            json_dumps_params={'ensure_ascii':False})
```

Django Template



Django의 템플릿 언어(template language)는 강력함과 편리함 사이의 균형을 잡고자 설계되었다. 템플릿 언어를 사용하면 HTML 작업을 훨씬 수월하게 할 수 있다. 변수, 필터, 태그, 주석 등 4가지 기능을 제공한다.

1. 템플릿 변수

템플릿변수를 사용하면 뷰에서 템플릿으로 객체를 전달할 수 있다. {{ 변수 }}와 같이 사용한다.

점(.) 은 변수의 속성에 접근할 때 사용한다. {{ section.title }} 뷰에서 보내온 section객체의 title 속성을 출력한다.

변수명으로 데이터 값 추출이 안되는 경우 공백으로 처리된다.

2. 템플릿 필터

템플릿필터는 변수의 값을 특정 형식으로 변환할 때 사용한다. 변수 다음에 파이프(|)를 넣은 다음 적용하고자 하는 필터를 명시한다.

여러 개의 필터를 연속적으로 사용할 수 있다. {{ textlescapellinebreaks }}는 텍스트 컨텐츠를 이스케이프한 다음, 행 바꿈을 <p> 태그로 바꾸기 위해 종종 사용되곤 한다.

몇몇 필터는 : 문자를 통해 인자를 취한다.

필터 인자는 {{bio|truncatewords:30 }}과 같이 사용하는데, 이것은 bio 변수의 처음 30 단어를 보여준다.

필터 인자에 공백이 포함된 경우에는 반드시 따옴표로 묶는다.

장고 서버는 30개 정도의 내장 템플릿 필터를 제공하는데, 자주 사용되는 템플릿 필터를 다음과 같다.

- default

변수가 false 또는 비어 있는 경우, 지정된 default를 사용한다.

{{ value|default:"nothing" }}

value가 제공되지 않았거나 비어 있는 경우, 위에서는 “nothing“을 출력한다.

- length

값의 길이를 반환한다. 문자열과 목록에 대하여 사용할 수 있다.

{{ value|length }}

value가 ['a', 'b', 'c', 'd']라면, 결과는 4가 된다.

- upper

{{ story.headline|upper }}

'story.headline'의 값을 대문자 형식으로 변환한다.

3. 템플릿 태그

{% tag %} 또는 {% tag %} ... tag contents ... {% endtag %}

HTML 자체는 프로그래밍 로직을 구현할 수 없지만, 템플릿 태그를 사용하면 if문, for문처럼 흐름을 제어할 수 있다.
{% tag %} 로 구성한다. {% extends %}와 같이 단독으로 사용할 수 있는 템플릿 태그들도 있지만, {% if %} 처럼 뒤에 {% endif %} 템플릿 태그를 반드시 닫아주어야 하는 것들도 있다.
장고에는 20개가 넘는 템플릿 태그가 내장되어 있으며 내장 태그 레퍼런스에서 읽어볼 수 있다.

- **{% csrf_token %}**

> CSRF 란
>
> 사이트 간 요청 위조(또는 크로스 사이트 요청 위조, 영어: Cross-site request forgery, CSRF, XSRF)는 웹사이트 취약점 공격의 하나로, 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 공격을 말한다. 예를 들자면 아래와 같은 공격이다.
>
> 사용자가 xxx.xxx.xxx에 login을 한다. sessionid등의 쿠키값을 정상적으로 발급받아 login된다.
>
> 공격자가 mail이나 게시판등을 이용해 악의적인 http request의 주소를 사용자쪽으로 전달한다. 예를 든다면 http://xxx.xxx.xxx/changepassword?password=abc와 같은 request를 서버로 전송하게끔.
>
> 사용자가 해당 주소를 실행하여 원하지 않는 request를 전송하게 된다. 서버입장에선 로그인을 거친 정상적인 client가 request를 수행한 것이니 해당 프로세스를 수행한다. 위 문제를 해결 하기 위해선 changpassword 페이지의 form전달값에 특정한 값을 추가하면 된다. 예를 들어 사용자로 부터 captcha를 값을 입력받게 하여, 정상적인 페이지에서의 요청인지 확인하면 되는 것이다. django 에서는 간단히 crsf token을 발행하여 처리하게 한다.

- for

배열의 각 원소에 대하여 반복처리

```
<ul>
{% for student in student_list %}
    <li>{{ student.name }}</li>
{% endfor %}
</ul>
```

- if / else

변수가 true이면 블록의 컨텐츠를 표시. if 태그 내에 템플릿 필터 및 각종 연산자를 사용할 수 있다.

```
{% if student_list %}
   총 학생 수 : {{ student_listllength }}
{% else %}
   학생이 없어요!
{% endif %}
```

```
{% if athlete_list %}
   Number of athletes: {{ athlete_listllength }}
{% elif athlete_in_locker_room_list %}
   Athletes should be out of the locker room soon!
{% else %}
   No athletes.
{% endif %}
```

- block 및 extentds

중복되는 html 파일 내용을 반복해서 작성해야 하는 번거로움을 줄여준다.

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
    <h1>{{ section.title }}</h1>

    {% for story in story_list %}
      <h2>
      <a href="{{ story.get_absolute_url }}">
     {{ story.headlinelupper }}
      </a>
      </h2>
      <p>{{ story.teaseltruncatewords:"100" }}</p>
    {% endfor %}
{% endblock %}
```

4. 템플릿 코멘트

HTML 문서 상에서 주석이 필요할 때 사용하며 장고에서는 두 가지 형식의 코멘트 형식을 제공한다.

한 줄

```
{# 주석 내용 #}
```

개행 허용되지 않음.

여러 줄

```
{% comment %}
주석 내용
{% endcomment %}
```

[정적인 자원들 저장 폴더 설정]

settings.py 파일의 119 행 정도에 보면

STATIC_URL = 'static/'

이 이미 존재한다. 아래에 다음 내용을 추가한다.

```
STATICFILES_DIRS = [
  BASE_DIR / 'staticfiles',
]
```

[전역 template 디렉토리 설정]

settings.py 파일의 58 행 정도에 보면 다음 내용이 있다.

'DIRS': [],

다음과 같이 수정한다.

'DIRS': [BASE_DIR / 'basetemplates'],