

CRUD-UD, Template Language

권중훈

UD를 어떻게 하죠?

- ??? : 수정하고, 지우면 됩니다.



세션을 시작하기 전에..



대부분의 문제는 오타에서 발생합니다

- Django Template Language의 익숙하지 않음
 - {% url을 열고 안 닫아준다든지
 - {{ 을 쓰고 }만 쓴다든지
- Python 자료구조에 익숙하지 않음
 - {}의 key value를 '만 쓰고 안 닫아준다든지
 - [] (array) 타입인데 콤마(,)로 element들을 구분을 안해준다든지

- Python 문법에 익숙하지 않음
 - Indentation issue
 - 조건문에서 ==을 ===으로 쓴다든지, =로 쓴다든지
 - 반복문:for element in elements에서 element가 뭐고 elements가 무엇인지
 - class post()가 무슨 의미인지 모른다든지
 - def 함수명():이 무슨 의미인지 모른다든지
 - 함수 선언해주고 :를 안 붙였다든지
 - 함수에 전달해주는 인자를 콤마(,)로 안 구분해줬다든지(ex.모델명.objects.create(인자, 인자))
 - 파이썬 코드에 세미콜론을 붙였다든지
- 기타
 - 띄어쓰기를 안 해준다든지

오타로 인해 발생하는 대표적인 에러들

SyntaxError at /stores/

invalid syntax (views.py, line 21)

TemplateSyntaxError at /repairs/Customers/Call/

Could not parse the remainder: '['customer'].name' from 'customer[customer].name'

Request Method: GET

Page not found (404)

E

Exce

Pyth

1

Request Method: GET

Request URL: <http://127.0.0.1:8000/polls/5/polls/5/vote/>

```
'C:\\Windows\\system32\\python27.zip',
```

```
'C:\\Python27\\DLLs',
```

```
'C:\\Python27\\lib',
```

```
'C:\\Python27\\lib\\plat-win',
```

```
'C:\\Python27\\lib\\lib-tk',
```

```
'C:\\Python27',
```

```
'C:\\Python27\\lib\\site-packages']
```

Server time: Sun, 6 Apr 2014 15:37:02 +0500

(나름 어문학도의 잡소리)

- 여러분이 배우는 건 파이썬이란 언어잖아요? 자 영어 처음배울 때 생각해보세요(살다오신분 제외 부럽다) 단어도 처음 배우면 외워야할 것도 짱 많고 문법도 어렵고 학교에서 스펠링 시험 보면 뭐 여러분들이야 고대 생이시니까 100점만 맞으셔야겠지만 틀리기 마련이잖아요?? 저희가 컴퓨터입니까?? 0과 1밖에 모르는 사람들이 아니잖아요?? 하루에 100시간 이상 코딩에 투자했습니까?? (이상 문과) 그것도 아니잖아요?? 그러니까 틀리는건 너무나 자연스럽고 당연한 과정이라는걸 꼭 아셔주시고 좌절하지 말고!!!! 시간투자하셔서!!!! 교안대로 따라치기만 하셔도!!!! 코딩엔!!!! 익숙해지실수있다!!!!!!!! 라는 말을!!!! 하고 싶었어요!!!!!!!! 으아!!!!

네 아무튼 여러분은 수많은 에러를 마주하실 겁니다..

ValueError at /signup/

The User could not be created because the data didn't validate.

TemplateSyntaxError at /repairs/Customers/Call/

Could not parse the remainder: '['customer'].name' from 'customer['customer'].name'

E:
P

Page not found (404)

E:
P

OperationalError at /user/register/

no such table: django_session

Python Pa

```
path('edit/<int:post_pk>', views.edit, name="edit"),
```

IndentationError: unexpected indent

Django Version: 2.1.5

Exception Type: OperationalError

Exception Value: no such table: django_session

Server tin

Exception Location: /Users/zhaosong/Documents/WorkSpace/dev2qa.com-example-code/PythonPyC

Python Executable: /Users/zhaosong/Documents/WorkSpace/dev2qa.com-example-code/PythonPyC

Python Version: 3.6.5

Python Path: ['/Users/zhaosong/Documents/WorkSpace/dev2qa.com-example-code/PythonPyCha

개발 == 문제 해결

- 에러가 발생했으면, 고치면 됩니다!
- 치열한 디버깅을 거치고 고쳤을 때 기분이 좋다면 당신은 개발자가 될 자격이 충분합니다.
- 지금 개발이 어렵게 느껴진다면, HTML + CSS + Python + Django를 다 합쳐서 구현해야 하기 때문입니다.
- 개발자/창업가 어떤 것을 하시든 우선 교안 코드는 잘 이해하셨으면 하는 개인적 바람

잡소리를 마치며

- 추상적인 개념은 원래 익히기 어려워용! 그냥 무작정 많이 따라 치다보면 이게 무슨 소린지 알 수 있습니다.
- (현대 개발에서 중요한 건 코드의 질도 있지만 구현 속도도 중요하기 때문에 웹프레임워크의 기본 문법에 익숙해지시면 아주아주 편하실 겁니다!! 미니톤이라든가.. 미니톤이라든가.. 미니톤이라든가....)

오늘 만들 것

home.html

이 페이지는 작성한 모든 독후감을 보여줍니다

내가 읽은 책들

- [안되니까 코딩이다](#)
- [죄와 벌 - 오타의 죄](#)
- [구글치면 비로소 보이는 것들](#)
- [HTML 4대 비극](#)
- [오만과 오타](#)
- [장고와 줄리엣](#)

[글 쓰러가기](#)

오늘 만들 것

detail.html

이 페이지는 내가 작성한 특정한 독후감을 보여줍니다.

책 제목

구글치면 비로소 보이는 것들

느낀 점

갯택오버플로우 찬양합니다. 구글링의 중요성을 깨달았습니다.

[수정하기](#) [삭제하기](#) [홈으로](#)

오늘 만들 것

edit.html

이 페이지에서 작성한 독후감을 수정할겁니다

제목	구글치면 비로소 보이는 것들
내용	갓택 오버플로우 찬양합니다. 구글링의 중요성을 깨달았습니다.

수정하기

오늘 만들 것

new.html

이 페이지에서 독후감을 작성할겁니다

제목

내용

그 전에, CRUD-CR을 복습하겠습니다.



따라해봅시다.

- mkdir session9
- cd session9
- pipenv shell
- pipenv install django
- --- 가상환경 설치 ---
- django-admin startproject (프로젝트명)
- cd (프로젝트명)
- django-admin startapp (앱명)

- code . (또는 vs code로 (프로젝트명) 폴더 열기)
- ---장고 앱설치 완료---
- settings.py에 app 추가
- python manage.py runserver
- ---로컬서버에서 장고 서버 실행 완료---

로컬 개발 환경 구축 성공!

django

[View release notes for Django dev](#)



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

뭘 할지 모르겠어요, 다음은?

- 장고는 MTV 패턴을 씁니다
- 모델을 만들고 -> 템플릿을 만들고 -> 뷰를 만들어주면 됩니다.

MTV 패턴

- **모델을 만들고** -> 템플릿을 만들고 -> 뷰를 만들어줍니다.

MTV 패턴

- **모델을 만들고** -> 템플릿을 만들고 -> 뷰를 만들어줍니다.
- + url도 만들어줍니다.

models.py

```
from django.db import models

# Create your models here.

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()

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

번외) next-likelion.com/admin

Django administration

Site administration

APPLYAPP

Questions

[+ Add](#) [✎ Change](#)

Users

[+ Add](#) [✎ Change](#)

Select user to change

<div>Q <input type="text"/></div> <div>Search</div>				
Action: <div>-----</div> <div>Go</div> 0 of 100 selected				
<input type="checkbox"/>	이름	이메일(ID)	생성일	IS ADMIN
<input type="checkbox"/>	김영	10gh@naver.com	March 6, 2020, 5:09 p.m.	✖
<input type="checkbox"/>	김영	10er.com	March 3, 2020, 6:15 p.m.	✖
<input type="checkbox"/>	이재	20ver.com	March 8, 2020, 2:56 p.m.	✖
<input type="checkbox"/>	오재	50naver.com	March 2, 2020, 9:24 p.m.	✖
<input type="checkbox"/>	정재	60ver.com	March 19, 2020, 4:23 p.m.	✖
<input type="checkbox"/>	김영	70naver.com	March 19, 2020, 6:40 p.m.	✖
<input type="checkbox"/>	박재	90ver.com	March 16, 2020, 9:51 a.m.	✖
<input type="checkbox"/>	홍재	a0er.com	March 19, 2020, 12:17 a.m.	✖
<input type="checkbox"/>	전재	a0naver.com	March 18, 2020, 9:19 a.m.	✖
<input type="checkbox"/>	이재	a0orea.ac.kr	March 2, 2020, 10:10 p.m.	✖
<input type="checkbox"/>	정재	a0naver.com	March 2, 2020, 11:13 p.m.	✖
<input type="checkbox"/>	박재	a0korea.ac.kr	March 19, 2020, 12:37 a.m.	✖
<input type="checkbox"/>	이재	a0naver.com	March 16, 2020, 5 p.m.	✖
<input type="checkbox"/>	방재	b0gmail.com	March 3, 2020, 2:09 p.m.	✖
<input type="checkbox"/>	김영	b0naver.com	March 5, 2020, 3:36 p.m.	✖
<input type="checkbox"/>	이재	b027@naver.com	March 17, 2020, 5:46 p.m.	✖
<input type="checkbox"/>	조재	b0um.net	March 9, 2020, 1:43 a.m.	✖
<input type="checkbox"/>	백재	b0orea.ac.kr	March 4, 2020, 2:01 p.m.	✖
<input type="checkbox"/>	윤재	b0naver.com	March 5, 2020, 8:14 p.m.	✖
<input type="checkbox"/>	유재	b0naver.com	March 4, 2020, 10:08 p.m.	✖

Django administration

Home · Applyapp · Questions

Select question to change

Action: 0 of 80 selected

- ☐ user
- ☐ cjs
- ☐ gl
- ☐ db
- ☐ an
- ☐ 76
- ☐ d3
- ☐ jic
- ☐ jo
- ☐ 99
- ☐ hy
- ☐ ye
- ☐ di
- ☐ tle
- ☐ ky
- ☐ ce
- ☐ av
- ☐ lin
- ☐ ed
- ☐ ko
- ☐ ap

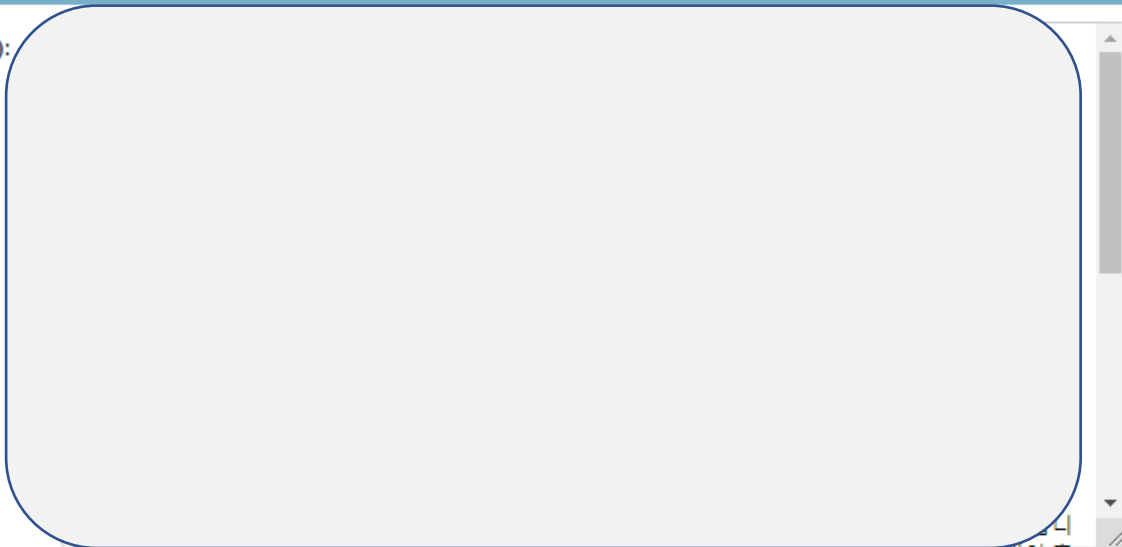
Change question

User:

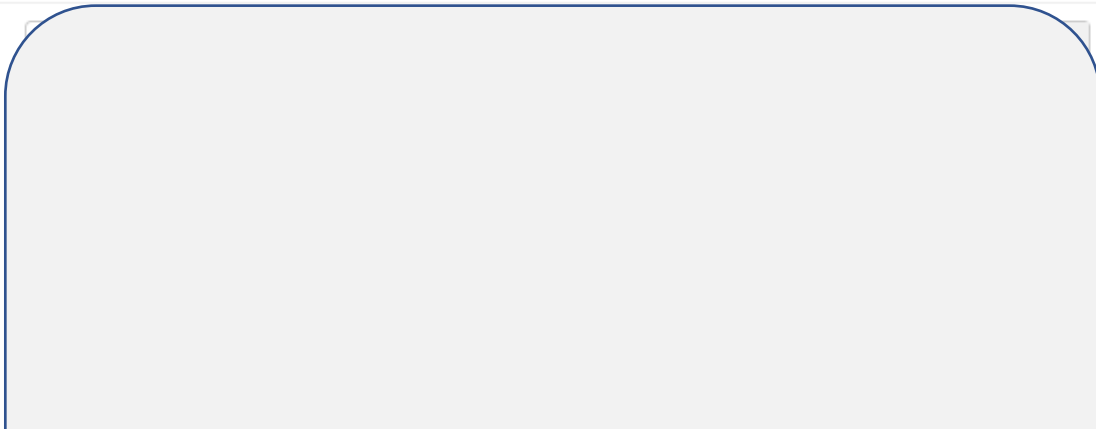


Questions

지원동기 (1000자 내외):



향후 5년 계획과, 10년 후의
나의 모습을 이야기해 주세
요. (1000자 내외):



보여드린 이유

- 장고는 /admin에서 만든 모델을 확인할 수 있다는걸 말하고 싶었어요.
- 관리자페이지를 따로 제작하거나, DBMS 사용을 안 해도 돼서 장고의 admin 기능은 편리해요.

만든 모델을 어드민에서 확인해봅시다

- localhost:8000/admin 접속
- superuser가 필요하네?
- (서버 잠시 끄(Ctrl + c))
- `python manage.py createsuperuser`

??? : 안 돼요



개발 == 문제 해결

- 문제를 고치기 전에 알아야 할 건 문제가 무엇인지 정확하게 파악하는 것!
- (가장 일반적인 방법)
- 우선 구글신의 도움을 빌려봅시다

- python manage.py migrate로 해결!
- ---admin 테이블 생성 완료---
- python manage.py createsuperuser
- Python manage.py runserver
- localhost:8000/admin
- 어드민 로그인

??? : 모델이 안 떠요



admin.py

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

# Register your models here.
admin.site.register(Post)
```

??? : 안 돼요

OperationalError at /admin/app/post/

no such table: app_post

Request Method: GET

Request URL: http://localhost:8000/admin/app/post/

Django Version: 3.0.6

Exception Type: OperationalError

Exception Value: no such table: app_post

Exception Location: C:\Users\jhkwo\virtualenvs\session-Tr9tjyrN\lib\site-packages\django\db\backends\sqlite3\base.py in execute, line 396

Python Executable: C:\Users\jhkwo\virtualenvs\session-Tr9tjyrN\Scripts\python.exe

Python Version: 3.7.2

Python Path: ['C:\\Users\\jhkwo\\desktop\\session\\crudproject',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN\\Scripts\\python37.zip',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN\\DLLs',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN\\lib',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN\\Scripts',
'c:\\Users\\jhkwo\\appdata\\local\\programs\\python\\python37-32\\Lib',
'c:\\Users\\jhkwo\\appdata\\local\\programs\\python\\python37-32\\DLLs',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN',
'C:\\Users\\jhkwo\\virtualenvs\\session-Tr9tjyrN\\lib\\site-packages']

Server time: Fri, 15 May 2020 12:52:46 +0000

중요) 모델 변경시 해줘야 하는 작업

- `python manage.py makemigrations`
- `python manage.py migrate`

- 언제 이 명령어를 쓰나요?
- 1. 새로운 모델을 생성했을 때
- 2. 모델의 필드를 추가/변경 했을때 (필드명 포함)

모델이 잘 만들어졌어요! 다음엔 뭐해야하죠?

- 장고는 MTV 패턴을 씁니다
- 모델을 만들고(끝) -> 템플릿을 만들고 -> 뷰를 만들어줍시다!

- (apps.py가 있는 곳에 templates 폴더 생성)
- home.html 파일 생성
- ---전체 post를 띄울 template 생성 완료---

우선 home.html 을 브라우저에서 확인 가능하게 만들어봅시다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <h1>독후감 블로그</h1>
</body>

</html>
```

템플릿을 만들었어요! 다음엔 뭐해야하죠?

- 장고는 MTV 패턴을 씁니다
- 모델을 만들고(끝) -> 템플릿을 만들고(끝) -> 뷰를 만들어줍시다!

views.py

```
def home(request):  
    return render(request, 'home.html')
```


MTV 패턴

- 모델을 만들고 -> 템플릿을 만들고 -> 뷰를 만들어줍니다.
- + url도 만들어줍니다.

urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home")
]
```

작성한 템플릿이 뜨면 성공!



localhost:8000

독후감 블로그

지금까지, 장고의 MTV 패턴을 익혀보았습니다

- MTV 패턴이란?
- 모델을 만들고 -> 템플릿을 만들고 -> 뷰를 만드는 패턴!

지금까지, 장고의 MTV 패턴을 익혀보았습니다

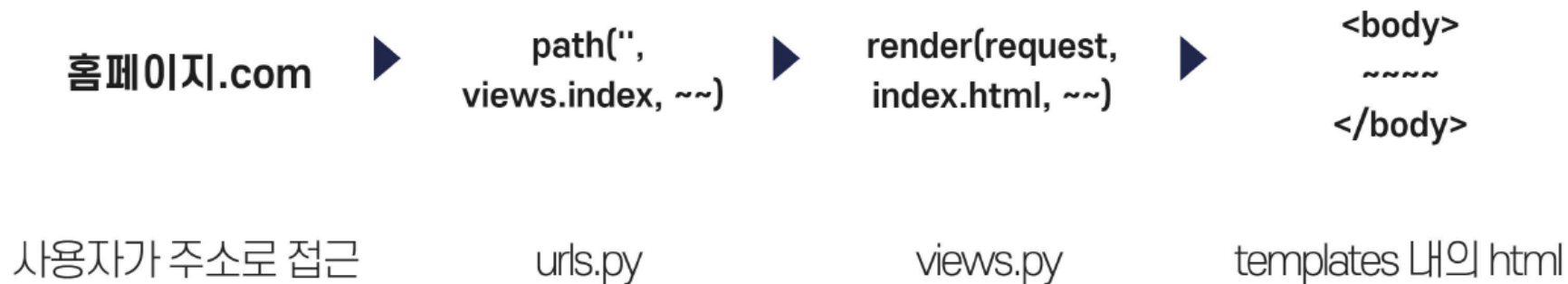
- MTV 패턴이란?
- 모델을 만들고 -> 템플릿을 만들고 -> 뷰를 만드는 패턴!
- Post란 모델을 만들고 ->
- 아이디어를 구현할 템플릿을 만들고 ->
- 템플릿을 보여줄 뷰를 만드는 것!

다시, 장고의 작동 원리 이해하기

- 유저가 /주소 로 들어오면
- urls.py가 /주소를 인식하고
- 연결된 views.함수명을 실행시켜주고
- views.py의 해당 함수가 template를 render해줍니다.

templates 페이지 만들기

이제 render 에서 지정했던 html 파일만 만들어주면 된다!



urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
]
```

- 사용자가 'localhost:8000'로 접속

urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
]
```

- urls.py:
"url이 ''인 접속을 인식했습니다"

urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
]
```

- urls.py:
"views.home을 실행시킬게요"

views.py

```
from django.shortcuts import render

# Create your views here.
def home(request):
    return render(request, 'home.html')
```

- views.py:

“안녕하세요! 제 안의 home 함수를 실행시키라는 요청을 받았어요!”

views.py

```
from django.shortcuts import render

# Create your views here.
def home(request):
    return render(request, 'home.html')
```

- views.py:

“home.html을 사용자에게 보여드릴게요!”

추가) posts를 전달해줄 때 views.py

```
from django.shortcuts import render
from .models import Post

# Create your views here.
def home(request):
    posts = Post.objects.all()

    return render(request, 'home.html', { 'posts': posts })
```

- views.py:

“안녕하세요! 제 안의 home 함수를 실행시키라는 요청을 받았어요!”

추가) posts를 전달해줄 때 views.py

```
from django.shortcuts import render
from .models import Post

# Create your views here.
def home(request):
    posts = Post.objects.all()
    return render(request, 'home.html', { 'posts': posts })
```

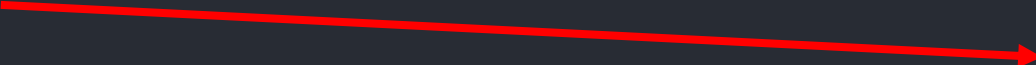
- views.py:

“Post 모델을 전부 갖고 와서 posts라는 변수에 저장합니다.”

추가) posts를 전달해줄 때 views.py

```
from django.shortcuts import render
from .models import Post

# Create your views here.
def home(request):
    posts = Post.objects.all()
    return render(request, 'home.html', { 'posts': posts })
```



- views.py:

“home.html을 보여줄건데(render), 위에서 저장한 posts 변수를 posts라는 이름으로 'home.html'에 전달해주겠습니다.”

home.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <h1>독후감 블로그</h1>
</body>

</html>
```

- home.html:
- “??? 어쩌라고 나는 파이썬 변수 못 알아먹는데”

- 전달받은 파이썬 변수를 html 파일에서 사용하기 위해선, 장고 전용 문법이 필요해요.

- 전달받은 파이썬 변수를 html 파일에서 사용하기 위해선, 장고 전용 문법이 필요해요.
- 왜?
 - HTML에서는 파이썬 변수를 인식을 할 수 없어요

- 전달받은 파이썬 변수를 html 파일에서 사용하기 위해선, 장고 전용 문법이 필요해요.

- 왜?

- HTML에서는 파이썬 변수를 인식을 할 수 없어요

=> Django Template Language가 필요해요

Django Template Language

- 장고 문법이란?
 - 장고에서 파이썬으로 전달해준 변수를 HTML에서 사용해주기 위해 쓰는 장고만의 독특한 문법

Django's template language is designed to strike a balance between power and ease. It's designed to feel comfortable to those used to working with HTML. If you have any exposure to other text-based template languages, such as Smarty or Jinja2, you should feel right at

Django Template Language

Contents

- [The Django template language](#)
 - [Templates](#)
 - [Variables](#)
 - [Filters](#)
 - [Tags](#)
 - [Comments](#)
 - [Template inheritance](#)
 - [Automatic HTML escaping](#)
 - [How to turn it off](#)
 - [For individual variables](#)
 - [For template blocks](#)
 - [Notes](#)
 - [String literals and automatic escaping](#)
 - [Accessing method calls](#)
 - [Custom tag and filter libraries](#)
 - [Custom libraries and template inheritance](#)

- <https://docs.djangoproject.com/en/3.0/ref/templates/language/>
- 구글: django template syntax 검색

Django built-in template tags and filters

Contents

- [Built-in template tags and filters](#)
 - [Built-in tag reference](#)
 - [autoescape](#)
 - [block](#)
 - [comment](#)
 - [csrf_token](#)
 - [cycle](#)
 - [debug](#)
 - [extends](#)
 - [filter](#)
 - [firstof](#)
 - [for](#)
 - [for ... empty](#)
 - [if](#)
 - [Boolean operators](#)
 - [== operator](#)
 - [!= operator](#)
 - [< operator](#)
 - [> operator](#)
 - [<= operator](#)

- <https://docs.djangoproject.com/en/3.0/ref/templates/builtins/#for>
- 구글: django template tags 검색

“너무 많아요, 다 알아야 하나요?”

Contents

- [The Django template language](#)
 - [Templates](#)
 - [Variables](#)
 - [Filters](#)
 - [Tags](#)
 - [Comments](#)
 - [Template inheritance](#)
 - [Automatic HTML escaping](#)
 - [How to turn it off](#)
 - [For individual variables](#)
 - [For template blocks](#)
 - [Notes](#)
 - [String literals and automatic escaping](#)
 - [Accessing method calls](#)
 - [Custom tag and filter libraries](#)
 - [Custom libraries and template inheritance](#)

Contents

- [Built-in template tags and filters](#)
 - [Built-in tag reference](#)
 - [autoescape](#)
 - [block](#)
 - [comment](#)
 - [csrf_token](#)
 - [cycle](#)
 - [debug](#)
 - [extends](#)
 - [filter](#)
 - [firstof](#)
 - [for](#)
 - [for ... empty](#)
 - [if](#)
 - [Boolean operators](#)
 - [== operator](#)
 - [!= operator](#)
 - [< operator](#)
 - [> operator](#)
 - [<= operator](#)

No! 외우지 말고, 필요할 때 찾아 쓰세요!

- 필요한 걸 그때그때 찾아 쓰면 돼요.
- HTML도 태그가 많았는데 자주 쓰는 태그는 몇 개 없었듯이, 장고 문법도 주로 쓰는 문법을 많이 쓰고 나머지 문법은 필요에 따라 Django documentation에서 찾아서 쓰면 돼요.
- 주로 쓰는 태그?
- 조건문, 반복문, 변수 전달 문법
- (지난 시간 교안에도 있으니 가서 확인하세요!)

기억나시나요?

HTML에서 사용할 수 있는 장고 문법 (파이썬과 비슷하다)

{% if (조건문) %} ~~	{% for ~ in ~ %} ~~	{% url 'index' %}	{{ 객체.어트리뷰트 }}
{% endif %}	{% endfor %}		

기본 문법은 {% %} 로 둘러싸고, 모델 객체 (글) / 변수 와 관련된 것은 {{ }} 로 둘러싼다!

Django Template Language 실습!

- home.html에 대표적인 Django Template Language를 실습해보겠습니다.
- 1. {{ 파이썬 변수 }}
- 2. {% for ... in ... %} (띄어쓰기 주의하세요)
- 3. {% if ... %}
- (시간이 허용하면 진행하겠습니다.)

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  {{ posts }}
</body>
</html>
```

??? : 이상한거 떠요

<QuerySet [<Post: 안되니까 코딩이다>, <Post: 죄와 벌 - 오타의 죄>, <Post: 구글치면 비로소 보이는 것들>, <Post: HTML 4대 비극>, <Post: 오만과 오타>, <Post: 장고와 줄리엣>]>

QuerySet이란?

- 쿼리셋 = 모델들의 집합

QuerySet이란?

- 쿼리셋 = 모델들의 집합
- 특징: 메소드를 가진다.
<https://docs.djangoproject.com/en/3.0/ref/models/queriesets/>

QuerySet 메소드란?

- DB에서 가져온 모델들을 사용자 입맛에 맞게 바꿔주는 일종의 함수
- Ex) id 순으로 정렬, 특정 필드값이 큰 순으로 정렬, 작성일자 순으로 정렬 등
- 참고) 메소드의 기능과 일치하는 SQL을 데이터베이스에 실행한다고 생각하셔도 됩니다.

참고) QuerySet 메소드과 SQL

in

In a given iterable; often a list, tuple, or queryset. It's not a common use case, but strings (being iterables) are accepted.

Examples:

```
Entry.objects.filter(id__in=[1, 3, 4])  
Entry.objects.filter(headline__in='abc')
```

SQL equivalents:

```
SELECT ... WHERE id IN (1, 3, 4);  
SELECT ... WHERE headline IN ('a', 'b', 'c');
```


QuerySet 메소드 documentation

- <https://docs.djangoproject.com/en/3.0/ref/models/queries/>

◦ QuerySet API

▪ Methods that return new QuerySets

- `filter()`
- `exclude()`
- `annotate()`
- `order_by()`
- `reverse()`
- `distinct()`
- `values()`
- `values_list()`
- `dates()`
- `datetimes()`
- `none()`
- `all()`
- `union()`
- `intersection()`
- `difference()`
- `select_related()`

▪ Operators that return new QuerySets

- `AND (&)`
- `OR (|)`

▪ Methods that do not return QuerySets

- `get()`
- `create()`
- `get_or_create()`
- `update_or_create()`
- `bulk_create()`
- `bulk_update()`
- `count()`
- `in_bulk()`
- `iterator()`
 - With server-side cursors
 - Without server-side cursors
- `latest()`
- `earliest()`
- `first()`

- `explain()`

▪ Field lookups

- `exact`
- `iexact`
- `contains`
- `icontains`
- `in`
- `gt`
- `gte`
- `lt`
- `lte`
- `startswith`
- `istartswith`
- `endswith`
- `iendswith`
- `range`
- `date`

주의!

- 외우실 필요 전혀 없습니다.
- 장고는 모델을 template에 보여줄 때 QuerySet이란 걸 사용하는구나, 는 정도만 아셔도 충분합니다.
- 교안에서 기본적인 CRUD를 구현할 땐 get(), filter(), create(), update(), delete() 메소드만 사용하니 너무 부담 가지시지 않으셔도 됩니다.

home.html

```
<body>
  {% for post in posts %}
    <div>
      <h2>{{ post.title }}</h2>
      <p>{{ post.content }}</p>
    </div>
  {% endfor %}
</body>
```

- <div>태그, <h2>태그 등은 자유롭게 만드셔도 됩니다
- 각 요소들이 한 줄 한 줄 띄어서 나오게 하기 위해서 이렇게 만든거예요

CRUD - Create를 구현해보자

- MTV 패턴 - 모델은 이미 만들었습니다
- 그렇다면? 템플릿을 만들면 됩니다
- 모델을 만들고 -> **템플릿을 만들고** -> 뷰를 만듭니다

오타 조심하세요!



new.html

```
<body>
  <form method="POST">
    {% csrf_token %}
    <input type="text" name="title" placeholder="제목을 입력해주세요">
    <input type="text" name="content" placeholder="내용을 입력해주세요">
    <button type="submit">작성하기</button>
  </form>
</body>
```

home.html

```
<body>
  {% for post in posts %}
    <div>
      <h2>{{ post.title }}</h2>
      <p>{{ post.content }}</p>
    </div>
  {% endfor %}
  <a href="{% url 'new' %}">글 쓰러가기</a>
</body>
```

장고의 MTV 패턴

- 모델을 만들고 -> 템플릿을 만들고(끝) -> 뷰를 만듭니다

views.py

```
def new(request):  
    if request.method == 'POST':  
        Post.objects.create(  
            title = request.POST['title'],  
            content = request.POST['content']  
        )  
    return render(request, 'new.html')
```

urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
]
```

localhost:8000/new 접속

- 제목과 내용을 입력하고 form을 제출해보세요
- /admin에서 모델이 잘 만들어졌는지 확인해봅시다
- 여러분이 만드신 모델이 Post에 들어있다면 성공!
- ---CRUD - Create 구현 완료---

CRUD - Read를 구현해보자

- MTV 패턴 - 모델은 이미 만들었습니다
- 그렇다면? 템플릿을 만들면 됩니다
- 모델을 만들고 -> **템플릿을 만들고** -> 뷰를 만듭니다

home.html

```
<body>
  {% for post in posts %}
    <div>
      <h2>{{ post.title }}</h2>
      <p>{{ post.content }}</p>
      <a href="{% url 'detail' post.pk %}">보기</a>
    </div>
  {% endfor %}
  <a href="{% url 'new' %}">글 쓰러가기</a>
</body>
```

detail.html

```
<body>
  <div>
    <h2>책제목</h2>
    <span>{{ post.title }}</span>
  </div>
  <div>
    <h2>느낀점</h2>
    <span>{{ post.content }}</span>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
</body>
```

장고의 MTV 패턴

- 모델을 만들고 -> 템플릿을 만들고(끝) -> 뷰를 만듭니다

views.py

```
def detail(request, post_pk):  
    post = Post.objects.get(pk=post_pk)  
  
    return render(request, 'detail.html', {'post': post})
```


views.py

```
def new(request):
    if request.method == 'POST':
        new_post = Post.objects.create(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('detail', new_post.pk)
    return render(request, 'new.html')

def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    return render(request, 'detail.html', {'post': post})
```

urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
    path('detail/<int:post_pk>', views.detail, name="detail"),
]
```

오타 조심하세요!

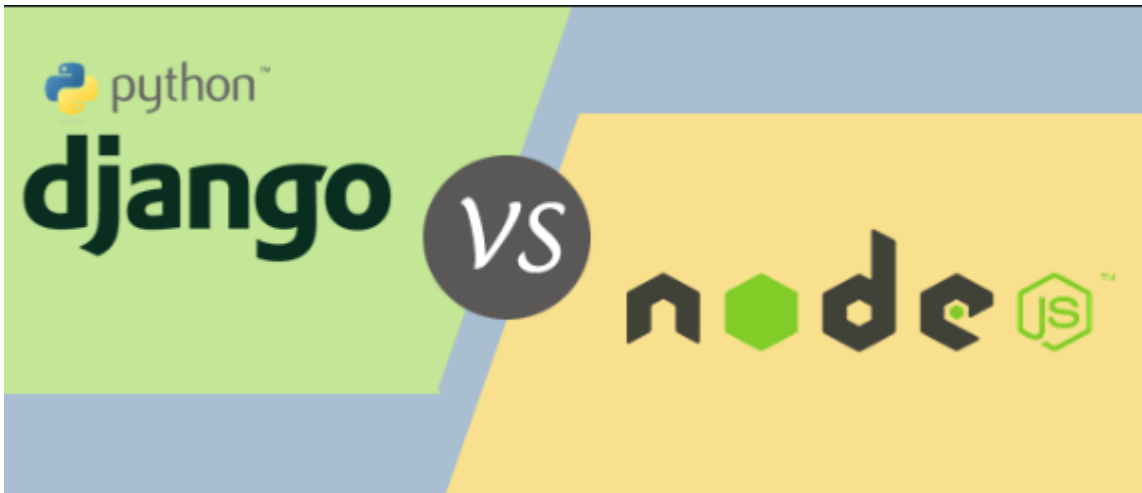


localhost:8000/detail/(id)

- id에 해당하는 pk를 가진 Post가 보이신다면 성공!
- 짹짹!
- ---CRUD - Read 구현 완료---

여기까지 지난 시간 복습 끝!

- 쉬어가기 > django vs node.js



- <https://www.youtube.com/watch?v=PnhmeFakkXg>

그래서 오늘 할 것은?

- update랑 delete

이 QuerySet 메소드만 알면 돼요

- 모델명.objects.update()
- 모델명.objects.delete()

CRUD - Update를 구현해보자

- MTV 패턴 - 모델은 이미 만들었습니다
- 그렇다면? 템플릿을 만들면 됩니다
- 모델을 만들고 -> **템플릿을 만들고** -> 뷰를 만듭니다

detail.html

```
<body>
  <div>
    <h2>책제목</h2>
    <span>{{ post.title }}</span>
  </div>
  <div>
    <h2>느낀점</h2>
    <span>{{ post.content }}</span>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
  <a href="{% url 'edit' post.pk %}">수정하기</a>
</body>
```

edit.html

```
<form method="POST">
  {% csrf_token %}
  <div>
    <label>제목</label>
    <input type="text" name="title" value="{{ post.title }}" placeholder="제목을 입력해주세요">
  </div>
  <div>
    <label>내용</label>
    <textarea name="content" cols="30" rows="10">{{ post.content }}</textarea>
  </div>
  <button type="submit">수정하기</button>
</form>
```

장고의 MTV 패턴

- 모델을 만들고 -> 템플릿을 만들고(끝) -> 뷰를 만듭니다

views.py

```
def edit(request, post_pk):  
    post = Post.objects.get(pk=post_pk)  
  
    if request.method == 'POST':  
        Post.objects.filter(pk=post_pk).update(  
            title = request.POST['title'],  
            content = request.POST['content']  
        )  
        return redirect('detail', post_pk)  
  
    return render(request, 'edit.html', {'post': post})
```

views.py (부연설명 1)

```
def edit(request, post_pk):  
    post = Post.objects.get(pk=post_pk)  
  
    if request.method == 'POST':  
        # update시, update할 모델을 get 메소드가 아니라 filter 메소드로 가져옴에 유의  
        Post.objects.filter(pk=post_pk).update(  
            title = request.POST['title'],  
            content = request.POST['content']  
        )  
        return redirect('detail', post_pk)  
  
    return render(request, 'edit.html', {'post': post})
```

views.py (부연설명 2)

```
def edit(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    if request.method == 'POST':
        # 이렇게 적어도 수정은 되지만, 유의해야할 점이 있습니다
        updated_post = Post.objects.filter(pk=post_pk).update(
            title = request.POST['title'],
            content = request.POST['content']
        )

        # update 메소드는 수정한 post를 return 하지 않고, update한 post의 개수를 return 합니다.
        # 따라서 여기서 updated_post의 값은 1입니다. post 모델이 아니에요
        # 그러므로 redirect()에 전달하는 pk 값으로 updated_post.pk가 아니라
        # edit()의 인자로 받아온 post_pk를 사용해주어야 합니다
        return redirect('detail', post_pk)

    return render(request, 'edit.html', {'post': post})
```

urls.py

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', views.home, name="home"),  
    path('new/', views.new, name="new"),  
    path('detail/<int:post_pk>', views.detail, name="detail"),  
    path('edit/<int:post_pk>', views.edit, name="edit")  
]
```

CRUD - Update 구현 확인

- 수정하기를 누르고, pk를 통해 받아온 모델을 수정해봅시다.
- 잘 수정되었다면 Update 구현 완료!
- ---CRUD - Update 구현 완료---

CRUD - Delete를 구현해보자

- MTV 패턴 - 모델은 이미 만들었습니다
- 그렇다면? 템플릿을 만들면 됩니다
- (주의, delete는 따로 삭제 페이지를 필요로 하지는 않습니다)
- 모델을 만들고 -> **템플릿을 만들고** -> 뷰를 만듭니다

home.html

```
<body>
  <div>
    <h2>책제목</h2>
    <span>{{ post.title }}</span>
  </div>
  <div>
    <h2>느낀점</h2>
    <span>{{ post.content }}</span>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
  <a href="{% url 'edit' post.pk %}">수정하기</a>
  <a href="{% url 'delete' post.pk %}">삭제하기</a>
</body>
```

장고의 MTV 패턴

- 모델을 만들고 -> 템플릿을 만들고(끝) -> 뷰를 만듭니다

views.py

```
def delete(request, post_pk):  
    post = Post.objects.get(pk=post_pk)  
    post.delete()  
    return redirect('home')
```

CRUD - Delete 구현 확인

- 삭제하기를 누르고, pk를 통해 받아온 모델을 삭제해봅시다.
- 잘 삭제되었다면 Delete 구현 완료!
- ---CRUD - Delete 구현 완료---

UD 핵심 원리

- views.py에서 수정, 삭제를 담당하는 함수에 수정 또는 삭제하길 원하는 모델의 pk를 인자값으로 넘겨서, 수정을 원하는 모델 테이블에서 pk로 특정 모델을 조회하여 수정/삭제해준다.

마지막 정리

- CRUD란?
 - Create, read, update, delete를 말하며
 - 모델을 생성하고, 모델을 조회하고, 모델을 수정하고, 모델을 삭제하는 것을 말한다.
 - 특정 모델 조회는 primary key(pk)를 이용해서 한다.

과제 (5/21 까지)

- To-do-list 게시판 만들기
- 할 일들을 전부 볼 수 있고(제목 or 제목+내용 or 내용),
- 할 일을 생성하고,
- 각 할 일들의 제목, 세부사항, 마감 기한을 볼 수 있고,
- 각 할 일들을 수정하고, 삭제할 수 있는 홈페이지를 만들어주세요.
- 필수 요구사항)
- 1. 마감기한이 적게 남은 순으로 할 일이 보이게 정렬해주세요.
- 2. CSS로 꾸며주세요.
- home.html, detail.html, edit.html, new.html 이면 충분합니다.
- 선택 사항)
- - 마감기한으로부터 오늘까지 남은 날짜를 할 일 옆에 보여주세요.
- ** 오늘 만든 CRUD 장고 프로젝트가 아니라 새로운 장고 프로젝트로 생성해주셔야 합니다

부록: 실습 코드 - home.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  {% for post in posts %}
    <div>
      <h2>{{ post.title }}</h2>
      <p>{{ post.content }}</p>
      <a href="{% url 'detail' post.pk %}">보기</a>
    </div>
  {% endfor %}
  <a href="{% url 'new' %}">글 쓰러가기</a>
</body>

</html>
```

부록: 실습 코드 - detail.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <div>
    <h2>책제목</h2>
    <span>{{ post.title }}</span>
  </div>
  <div>
    <h2>느낀점</h2>
    <span>{{ post.content }}</span>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
  <a href="{% url 'edit' post.pk %}">수정하기</a>
  <a href="{% url 'delete' post.pk %}">삭제하기</a>
</body>
```

부록: 실습 코드 - edit.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>

  <form method="POST">
    {% csrf_token %}
    <div>
      <label>제목</label>
      <input type="text" name="title" value="{{ post.title }}" placeholder="제목을 입력해주세요">
    </div>
    <div>
      <label>내용</label>
      <textarea name="content" cols="30" rows="10">{{ post.content }}</textarea>
    </div>
    <button type="submit">수정하기</button>
  </form>

</body>

</html>
```

부록: 실습 코드 - new.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <form method="POST">
    {% csrf_token %}
    <div>
      <label>제목</label>
      <input type="text" name="title" placeholder="제목을 입력해주세요">
    </div>
    <div>
      <label>내용</label>
      <textarea name="content" cols="30" rows="10">{{ post.content }}</textarea>
    </div>
    <button type="submit">작성하기</button>
  </form>
</body>

</html>
```

부록: 실습 코드 - views.py

```
from django.shortcuts import render, redirect
from .models import Post

# Create your views here.
def home(request):
    posts = Post.objects.all()

    return render(request, 'home.html', { 'posts': posts })

def new(request):
    if request.method == 'POST':
        new_post = Post.objects.create(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('detail', new_post.pk)
    return render(request, 'new.html')

def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    return render(request, 'detail.html', {'post': post})
```

```
def edit(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    if request.method == 'POST':
        Post.objects.filter(pk=post_pk).update(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('detail', post_pk)

    return render(request, 'edit.html', {'post': post})

def delete(request, post_pk):
    post = Post.objects.get(pk=post_pk)
    post.delete()
    return redirect('home')
```

부록: 실습 코드 - urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
    path('detail/<int:post_pk>', views.detail, name="detail"),
    path('edit/<int:post_pk>', views.edit, name="edit"),
    path('delete/<int:post_pk>', views.delete, name="delete")
]
```