

Python Django

3강 Python Django 기초

Python Django 를 하기 전에

이번에 나오는 중요 python 문법!

1. 정규 표현식

메타 문자	기능	설명
^	처음	문자열이나 행의 처음
\$	끝	문자열이나 행의 끝
.	문자	1개의 문자와 일치, 단일행 모드시 새줄 문자는 제외
[]	문자 클래스	‘[’와 ‘]’ 사이 문자 중 하나를 선택 ex) [abc]d = ad, bd, cd “-” 기호로 범위 지정 가능 ex) [az] = a~z 중 하나, [1-9] = 1~9 중 하나

Python Django 를 하기 전에

메타 문자	기능	설명
()	하위식	여러 식을 하나로 묶을 수 있다. abc adc 와 a(b d)c는 같은 의미를 가진다.
\Wn	일치하는 n번째 패턴	일치하는 패턴들 중 n번째를 선택하며, 여기에서 n은 1에서 9 중 하나가 올 수 있다
*	0회 이상	0개 이상의 문자를 포함한다. "a*b"는 "b", "ab", "aab", "aaab"를 포함한다.
?	0 또는 1회	"a? b"는 "b", "ab"를 포함한다
+	1회 이상	"a+b"는 "ab", "aab", "aaab"를 포함하지만 "b"는 포함 하지 않는다.
\W\W+	단어	하나 또는 여러 단어와 문자들.
	선택	여러 식 중에서 하나를 선택한다. "abc adc"는 abc 와 adc를 모두 포함한다.

Python Django 를 하기 전에

2. Class

Class : 객체를 만드는 틀. 객체에 대한 구조(변수)와 기능(메서드)을 담고 있음

Method : 해당 클래스나 객체가 가지고 있는 기능 -> 함수

Object : 클래스라는 틀을 이용해서 만들어진, 실제의 객체 = Instance

- Class는 메서드, 속성, 클래스 변수, 초기자(initializer), 소멸자(destructor) 등 여러 멤버를 가질 수 있음!
- 다른 언어에서는 각 멤버들에 대해 public, private, protected와 같은 접근 제한자를 가지지만, Python은 접근 제한자를 가지지 않고 기본적으로 모든 멤버가 public
=> Private로 만들고 싶을 경우에는 특정 변수명이나 메서드 앞에 '_'를 붙여주면 된다.

뒤에서 계속해서 설명을 보면 다음과 같다!

Python Django 를 하기 전에

2. Class

Class 예시

```
class Rectangle :  
    count = 0 #클래스 변수  
    #초기자  
    def __init__(self,width,height):  
        #self : 인스턴스변수  
        self.width = width  
        self.height = height  
        Rectangle.count +=1  
    #메서드  
    def calcArea(self):  
        area = self.width * self.height  
        return area  
rec1 =Rectangle(4,7)    #인스턴스 = object
```

#클래스 변수 : 클래스 정의에서 메서드 밖에 존재하는 변수

#초기자 : 클래스로부터 객체를 생성할 때마다 실행되는 특별한 메서드

#인스턴스 변수 : 클래스 정의에서 메서드 안에서 사용되면서

“self.변수명”처럼 사용되는 변수

⇒ self.를 사용하여 클래스 내부의 변수에서 액세스한다

⇒ 각 객체별로 서로 다른 값을 가지게 된다

#메서드 : 클래스의 행위 -> 클래스 내의 함수

=> 항상 첫번째 파라미터로 해당 클래스 객체를 나타내는 self를 가짐

Python Django 를 하기 전에

3. 오류 예외처리 기법 try except / raise

오류 예외처리 기법 : 코드를 짜면서 예외를 쉽게 처리하기 위해 제공되는 예약어를 사용하여 예외를 처리하는 것

try, except문 기본 구조

```
try:                                <- 오류가 발생하면~
    ...(예외가 발생할 가능성이 있는 구문)
except [발생 오류[as 오류 메시지 변수]]: <- except 블록 수행
    ...(예외 발생시 수행 구문)
```

raise : 해당 문이 실행되도록 하는 예약어
-> raise 에러종류(에러 발생시 나타낼 텍스트)

예시)

```
try :
    num = int(input("Insert Num:"))
    raise ValueError("smaller 0 ", "0", "Bigger 0")
except ValueError as e :
    if num < 0 :
        print(e.args[0])
    elif num == 0 :
        print(e.args[1])
    elif num > 0 :
        print(e.args[2])
```

Python Django

1. 데이터베이스 API

API(Application Program Interface)

: 프로그램 또는 애플리케이션이 운영 체제에 어떤 처리를 위해서 호출 할 수 있는 서브루틴 또는 함수의 집합

-> DB API를 통해서 DB에 데이터를 넣거나 표시하거나 지우는 등의 실행을 할 수 있음

CMD 창에서 프로젝트 파일 위치 >

`python manage.py shell`

```
C:\Users\강민주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>python manage.py shell
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

Python Django

1. 데이터베이스 API

이후에 작성하게 될 코드의 문법은 Python!

Python 코드로 DB에 데이터를 추가, 표시, 제거를 할 수 있다!

```
# Create your models here.  
class Video(models.Model):  
    category=models.CharField(max_length=100)  
  
class movie(models.Model):  
    category = models.ForeignKey(Video, on_delete=models.CASCADE)  
    movie_title=models.CharField(max_length=200)  
    director=models.CharField(max_length=250)  
    genre=models.CharField(max_length=100)  
    file_type = models.CharField(max_length=10)
```

>>> From video.models import Video -> 내가 사용할 DB Model import

>>> Video.objects.all() -> 그 model의 모든 object 표시

>>> a = Video(category = 'Movie') -> object 생성

>>> a.save() -> Shell에서 작성된 객체를 실제 DB에 저장 => 이걸 하지 않으면 DB에 적용되지 않음

>>> a.category -> 해당 a 객체의 category 출력

'Movie'

>>> b = Video() -> object 생성

>>> b.category = 'Drama'

>>> b.save()

>>> Video.objects.all()

[<Video: Video object>, <Video: Video object>]

```
(InteractiveConsole)  
>>> from video.models import Video  
>>> Video.objects.all()  
>>> a = Video(category = 'Movie')  
>>> a.save()  
>>> a.category  
'Movie'  
>>> a.pk  
1  
>>> a.id  
1  
>>> b = Video()  
>>> b.category = 'Drama'  
>>> b.save()  
>>> b.pk  
2  
>>> Video.objects.all()  
<QuerySet [<Video: Video object>, <Video: Video object>]>
```

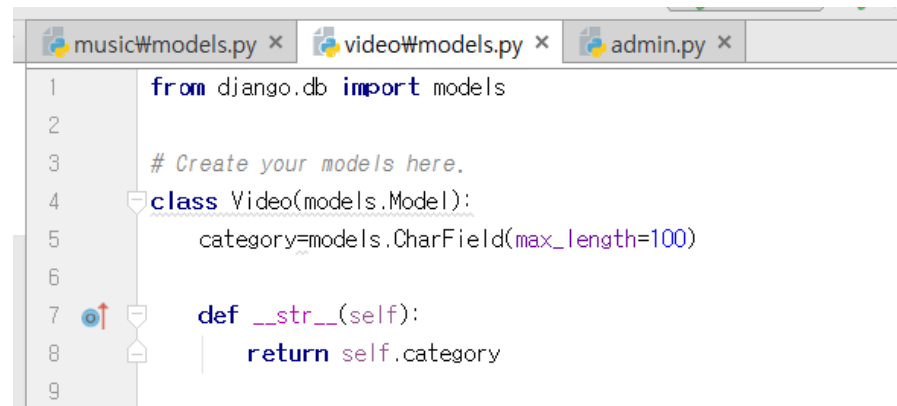

Python Django

1. 데이터베이스 API

DB의 객체를 나타냈을 경우 ‘[<Video: Video object>, <Video: Video object>]’

-> 그 객체 요소가 명확히 무엇인지 알 수 없음
따라서 이걸 나타내 줄 수 있는 메서드 구현 필요!

Models.py의 class 안에 함수 선언
def __str__(self):
return return 할 형식



```
1 from django.db import models
2
3 # Create your models here.
4 class Video(models.Model):
5     category=models.CharField(max_length=100)
6
7     def __str__(self):
8         return self.category
9
```

```
C:\Users\강민 주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>python manage.py shell
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from video.models import Video
>>> Video.objects.all()
<QuerySet [<Video: Movie>, <Video: Drama>]>
```

Python Django

1. 데이터베이스 API

DB의 객체 **filter** 사용하기

-> DB 객체들 중에 내가 원하는 것만 표시하도록 할 수 있음!

>>> Video.objects.filter(id=1) -> id가 1인 object 출력

>>> Video.objects.filter(category__startswith = 'D') -> D로 시작하는 object 출력

```
C:\Users\강민 주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>python manage.py shell
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from video.models import Video, Movie, Drama
>>> Video.objects.filter(id=1)
<QuerySet [<Video: Movie>]>
>>> Video.objects.filter(id=2)
<QuerySet [<Video: Drama>]>
>>> Video.objects.filter(id=3)
<QuerySet []>
>>> Video.objects.filter(category__startswith='D')
<QuerySet [<Video: Drama>]>
>>>
```

Python Django

2. Admin 사용하기

Administrator를 이용해서 DB를 보다 간편하게 관리 할 수 있음!

-> 따라서 Admin 사용자를 만들어 보자

CMD 창에서 프로젝트 파일 위치 >

`python manage.py createsuperuser`

```
C:\Users\강민주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>python manage.py createsuperuser
Username: Minju
Email address: kmju1997@gmail.com
Password:
Password (again):
The password is too similar to the email address.
Password:
Password (again):
The password is too similar to the email address.
Password:
Password (again):
Superuser created successfully.

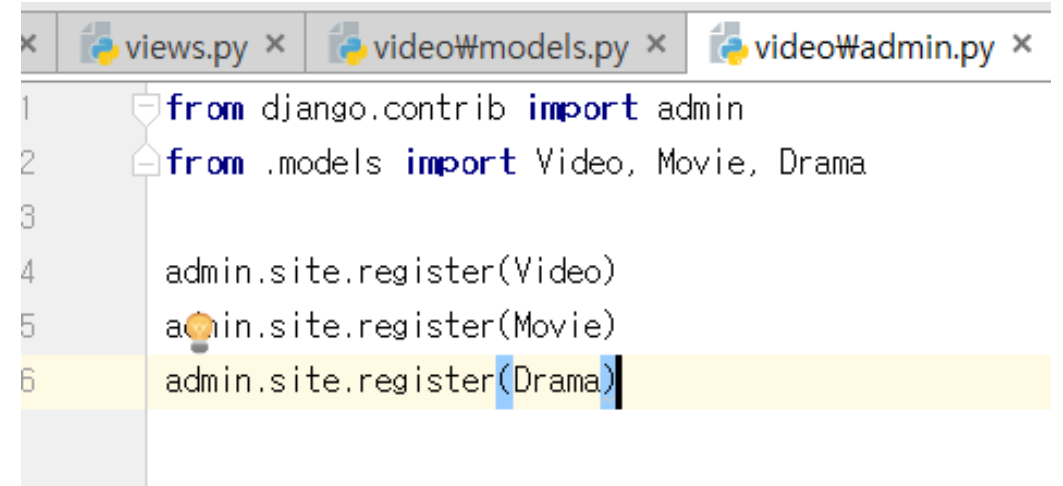
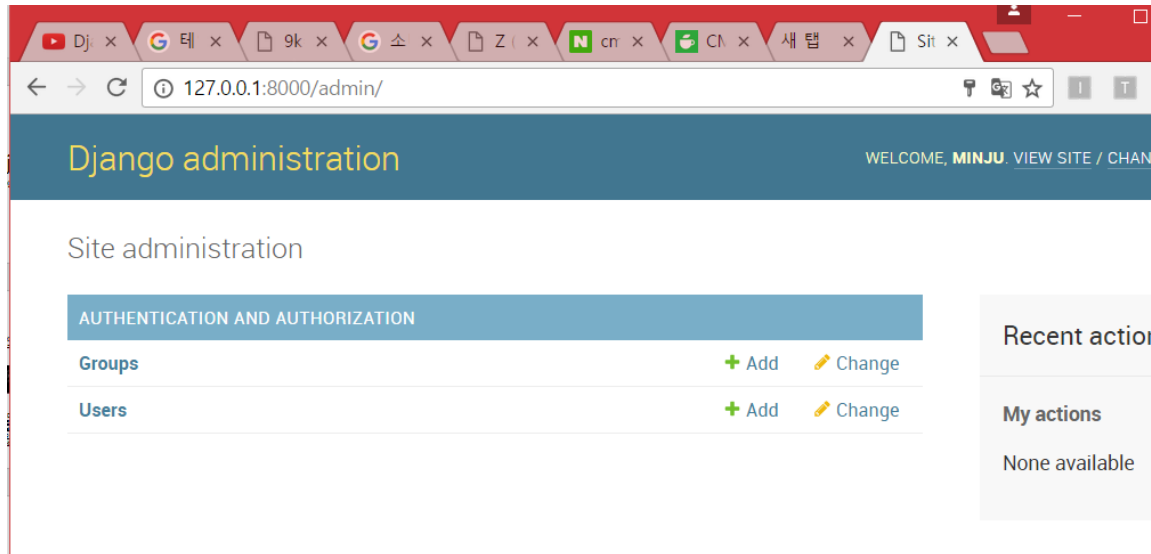
C:\Users\강민주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>
```

Python Django

2. Admin 사용하기

서버를 실행시키고 /admin url로 들어가면 로그인 화면이 뜨게 됨

-> 로그인을 하고 들어가면 아래처럼 admin 사용 User를 관리 할 수 있음!



Admin 페이지에서 DB를 관리하도록 하기 위해서는 admin.py 에 위와 같이 추가해줘야 함 ↑

적용 후, 이곳에서 삭제, 수정, 보기 전부 가능!

Python Django

3. 새로운 View 생성

이제부터 만들 새로운 view :

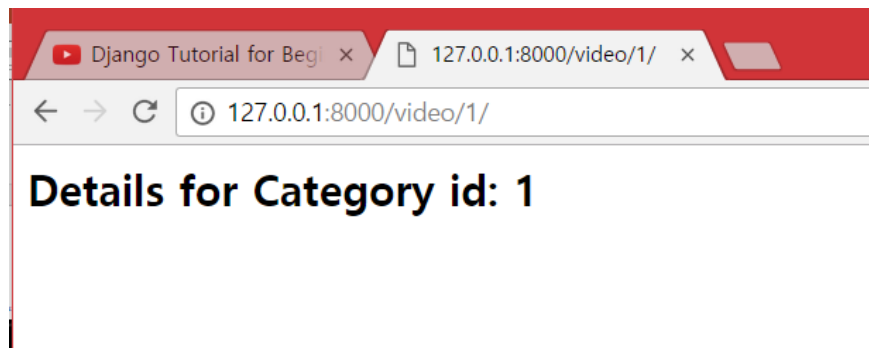
해당 카테고리에 대한 url 생성 + view를 통해 카테고리의 id가 표현

해당 app의 url.py에

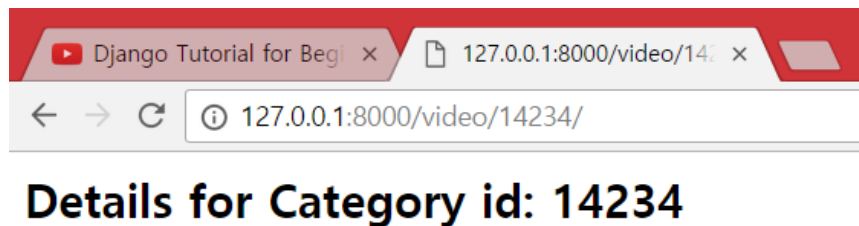
```
url(r'^(?P<category_id>[0-9]+)/$', views.detail, name='detail')
```

해당 app의 views.py에

```
def detail(request, category_id):  
    return HttpResponse("<h2>Details for Category id: " + str(category_id) + "</h2>")
```



이건 그냥 아무 번호나 치면 그 번호에 맞추어 문구가 바뀌게 됨



Python Django

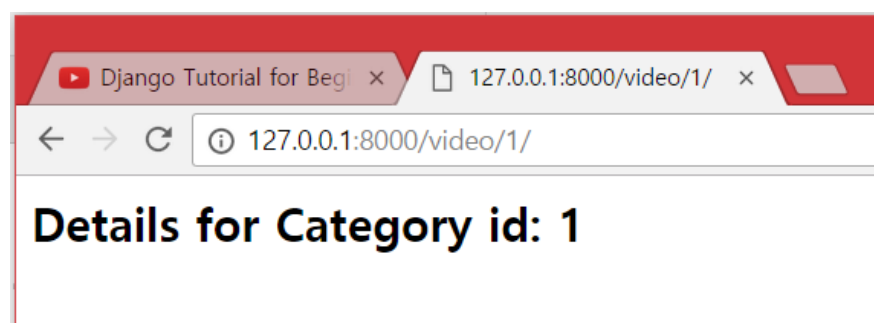
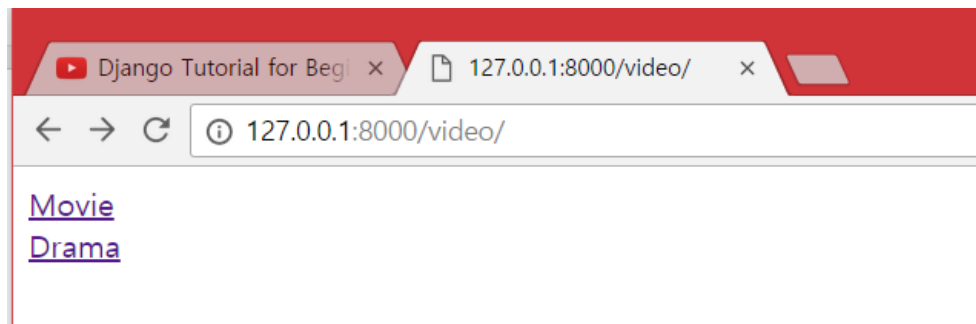
3. 새로운 View 생성 - DB연결

Video category 목록을 view에서 볼 수 있도록 만들기

해당 app의 views.py에

```
from .models import Video

def index(request):
    all_category = Video.objects.all()
    html = ''
    for category in all_category :
        url = '/video/' + str(category.id) + '/'
        html += '<a href = "' + url + '">' + category.category + '</a><br>'
    return HttpResponse(html)
```



Python Django

4. 템플릿

Template : generic html documents -> view를 예쁘게 꾸며주는 기본적인 웹 틀!

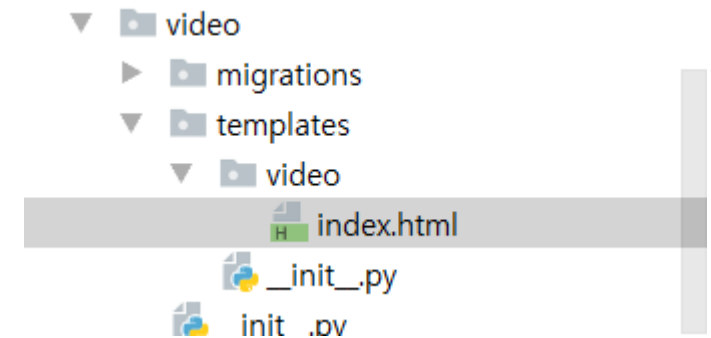
웹디자이너가 템플릿을 만들어주면

우리가 python 코드로 되어있는 DB와 그 템플릿을 엮어서 완전한 웹을 제작할 수 있도록 함!

1) 해당 app의 views.py에 template library loader추가 + index 함수 수정

```
from django.http import HttpResponse
from django.template import loader
from .models import Video

def index(request):
    all_category = Video.objects.all()
    template = loader.get_template('')
    return HttpResponse('')
```



2) App 파일 안에 templates 폴더 생성 후 안에 app과 같은 이름의 디렉토리 폴더 생성

3) 해당 디렉토리 폴더 안에 html 파일생성 (후에 여기다가 템플릿을 받아와서 넣어주는 식으로 하면 OK)

Python Django

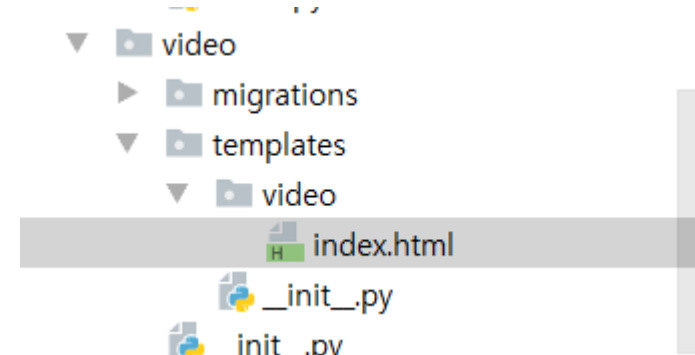
4. 템플릿

Template : generic html documents -> view를 예쁘게 꾸며져서 만들어져 있는 기본적인 웹 틀!
웹디자이너가 템플릿을 만들어주면
우리가 python 코드로 되어있는 DB와 그 템플릿을 엮어서 완전한 웹을 제작할 수 있도록 함!

1) 해당 app의 views.py에 template library loader추가 + index 함수 수정

```
from django.http import HttpResponse
from django.template import loader
from .models import Video

def index(request):
    all_category = Video.objects.all()
    template = loader.get_template('')
    return HttpResponse('')
```

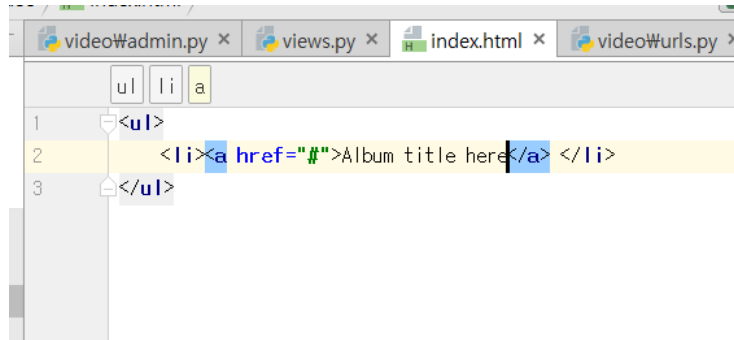


- 2) App 파일 안에 templates 폴더 생성 후 안에 app과 같은 이름의 디렉토리 폴더 생성
- 3) 해당 디렉토리 폴더 안에 html 파일생성 (후에 여기다가 템플릿을 받아와서 넣어주는 식으로 하면 OK)

Python Django

4. 템플릿

예를 들어 이런 식의 템플릿을 받았다고 한다면 이에 맞춰서 코드를 연결 시켜주면 됨!



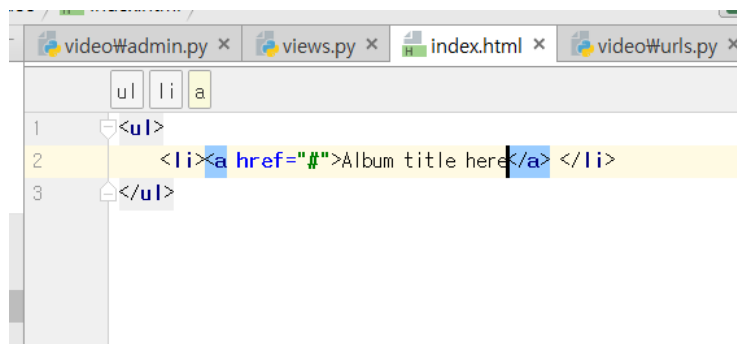
- 1) Views.py 의 index 함수에 template 수정 -> html template를 연결 시켜줌
- 2) 넘겨줄 정보에 대한 dictionary 만들기 -> 템플릿에 표시해 줄 정보를 list화
- 3) 마지막에 그 정보를 render해서 넘겨주기

```
def index(request):
    all_category = Video.objects.all()
    template = loader.get_template('video/index.html')
    context = { 'all_category' : all_category,
    }
    return HttpResponse(template.render(context,request))
```

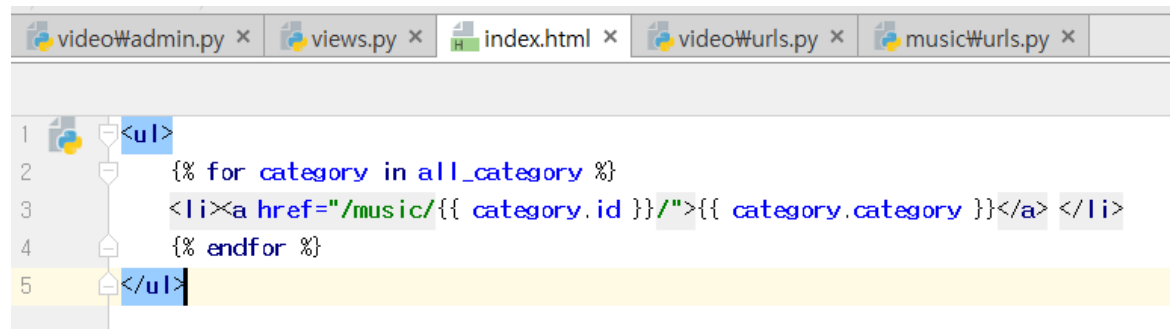
Python Django

4. 템플릿

4) 후에 html 파일에서 연결한 정보들을 사용해서 코드를 작성해줌



```
1 <ul>
2 <li><a href="#">Album title here</a> </li>
3 </ul>
```

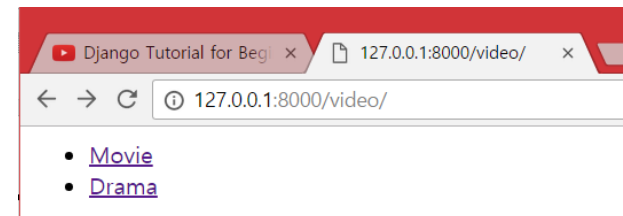


```
1 <ul>
2 {% for category in all_category %}
3 <li><a href="/music/{{ category.id }}">{{ category.category }}</a> </li>
4 {% endfor %}
5 </ul>
```

{% %} 나 {{ }}는 python 코드를 html 안에다 작성하고 적용 시킬 때 사용!

{% %} -> if 나 for 같은 python 문법을 쓰려고 할 때 사용

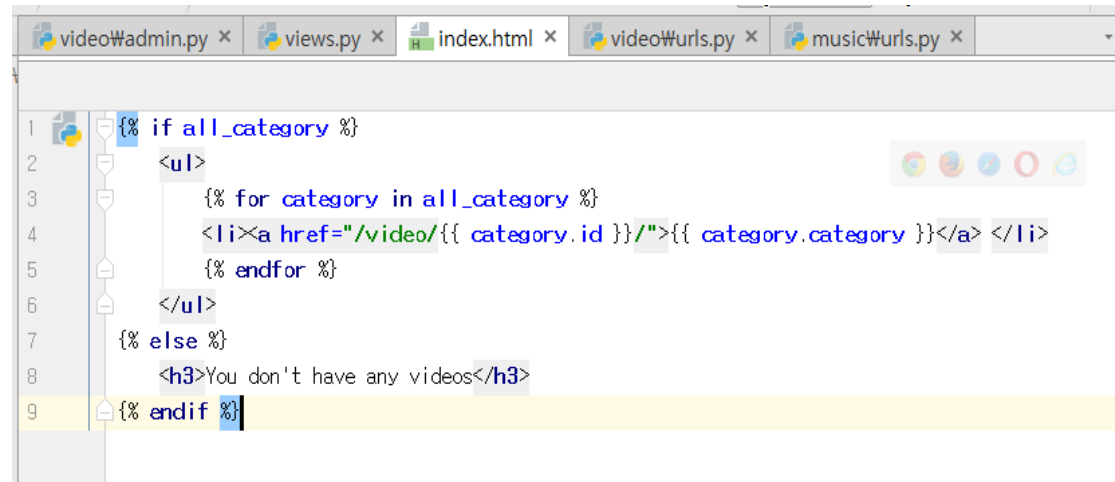
{{ }} -> html에 적용시킬 variable을 넣으려고 할 때 사용



Python Django

4. 템플릿

4) * 만약에 카테고리가 없다면? -> 그 때의 경우 화면 설정 필요



```
1 {% if all_category %}
2     <ul>
3         {% for category in all_category %}
4         <li><a href="/video/{{ category.id }}">{{ category.category }}</a> </li>
5         {% endfor %}
6     </ul>
7 {% else %}
8     <h3>You don't have any videos</h3>
9 {% endif %}
```

-> all_category가 비어 있는 경우에는 else의 화면대로 view가 나타나도록 함!

Python Django

5. 템플릿 shortcut

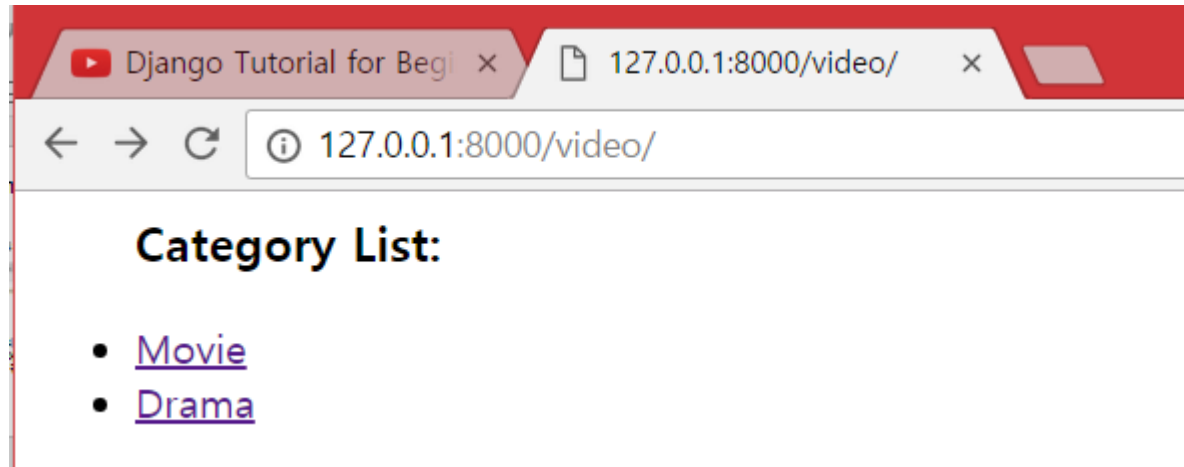
아까 했던 template 설정을 훨씬 더 간단하게 만들 수 있음!

App의 views.py 에서

- 1) Shortcut Django 라이브러리를 불러온다
- 2) HttpResponse 대신에 render 함수를 return해주도록 index함수를 수정한다
-> render 함수는 템플릿에 설정을 적용시키고 HttpResponse를 반환하는 역할을 해 줌

```
from django.http import HttpResponse
from django.shortcuts import render
from .models import Video

def index(request):
    all_category = Video.objects.all()
    context = { 'all_category' : all_category,
    }
    return render(request, 'video/index.html', context)
```



Python Django

6. 404 HTTP 에러 만들기

현재 설정까지는 존재하지 않는 id를 url에 입력했을 때에도 페이지 view가 나타난다

-> 하지만 원래는 이렇게 되어서는 안되므로 id가 존재하지 않을 때에는 404 에러가 뜨도록 만들어 줄 것!

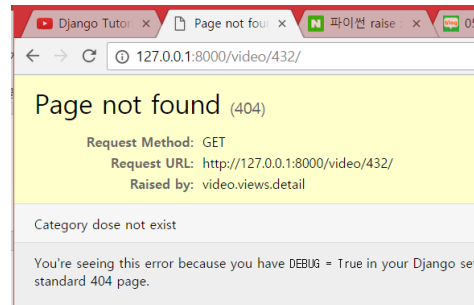
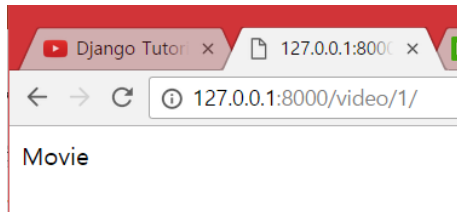
App의 views.py 에서

- 1) Http404 import 해준다
- 2) Detail view를 위한 html파일을 생성
- 3) Try except 문을 이용해서 예외 처리 -> 예외일 경우 http404오류가 나오도록 한다
- 4) 마지막에 render를 이용해 detail.html에 연결
- 5) Detail.html 에 {{ category }} 작성
-> 해당 카테고리 이름이 뜬다

```
from django.http import Http404
from django.shortcuts import render
from .models import Video
```

```
def index(request):
    all_category = Video.objects.all()
    context = { 'all_category' : all_category,
    }
    return render(request, 'video/index.html', context)
```

```
def detail(request, category_id):
    try :
        category = Video.objects.get(pk=category_id)
    except Video.DoesNotExist:
        raise Http404("Category dose not exist")
    return render(request, 'video/detail.html', { 'category' : category, })
```



Python Django

7. 각 카테고리의 video 데이터 생성

이제는 각 카테고리에 video 데이터를 넣어줘서 해당 카테고리로 들어가면 카테고리의 영상 목록이 나타나도록 만들어 줄 것임!

- 1) 각 카테고리에 따른 model 생성해주기
- 2) admin에서 DB를 사용 할 수 있도록 해당 model을 적용시키기
- 3) 해당 table의 표현형 만들어 주기

```
videoWadmin.py x models.py x views.py x
1 from django.contrib import admin
2 from .models import Video, Movie, Drama
3
4 admin.site.register(Video)
5 admin.site.register(Movie)
6 admin.site.register(Drama)
```

```
videoWadmin.py x models.py x views.py x detail.html x index.html x
8 return self.category
9
10
11 class Movie(models.Model):
12     category = models.ForeignKey(Video, on_delete=models.CASCADE)
13     movie_title = models.CharField(max_length=200)
14     director = models.CharField(max_length=250)
15     genre = models.CharField(max_length=100)
16     file_type = models.CharField(max_length=10)
17
18     def __str__(self):
19         return self.movie_title
20
21 class Drama(models.Model):
22     category = models.ForeignKey(Video, on_delete=models.CASCADE)
23     drama_title = models.CharField(max_length=200)
24     writer = models.CharField(max_length=250)
25     genre = models.CharField(max_length=100)
26     file_type = models.CharField(max_length=10)
27
28     def __str__(self):
29         return self.drama_title
```

Python Django

7. 각 카테고리의 video 데이터 생성

이제는 각 카테고리의 video 데이터를 넣어줘서 카테고리와 동영상을 연결해 줄 것!

4) CMD창에서 shell을 켜서 해당 카테고리에 대한 video 작품들을 생성해주기

```
C:\Users\강민주\Desktop\대학\동아리\포리프\파이썬스터디\Django\website>python manage.py shell
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from video.models import Video, Drama, Movie
>>> Video1 = Video.objects.get(pk=1)
>>> Video1.category
'Movie'
>>> movie = Movie()
>>> movie.category = Video1
>>> movie.file_type = 'avi'
>>> movie.movie_title = 'Monster'
>>> movie.save()
>>>
```

Select movie to change

Action: 0 of 1 selected

- ☐ MOVIE
- ☒ **Monster**

1 movie

Python Django

7. 각 카테고리의 video 데이터 생성

더 쉽게 생성을 위해서는 create() 함수 사용 / 아래와 같이 count나 연결된 model 출력, 해당 카테고리의 모든 영화 출력 가능
>>> Video1.movie_set.create(movie_title = '너의 이름은', file_type = 'avi')

```
>>> Video1.movie_set.create(movie_title = '너의 이름은', file_type = 'avi')
<Movie: 너의 이름은>
>>>
>>> movie = Video1.movie_set.create(movie_title = '타짜', file_type = 'avi')
>>> movie.category
<Video: Movie>
>>> Video1.movie_set.all()
<QuerySet [<Movie: Monster>, <Movie: 너의 이름은>, <Movie: 타짜>]>
>>>
>>> Video1.movie_set.count()
3
>>>
```

Select movie to change

Action: 0 of 3 selected

☐ MOVIE

☐ 타짜

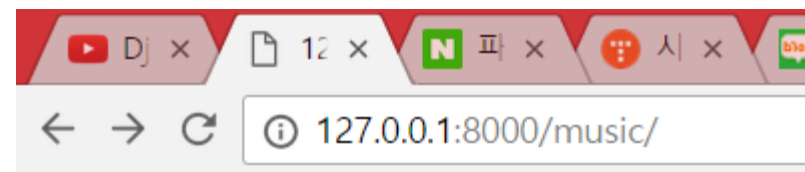
☐ 너의 이름은

☐ Monster

3 movies

알려준 곳까지 전부 해보고,
앨범 목록을 볼 수 있도록 view 구성하기! + 노래 5개 만들기
단, model 구성을 아래 제시와 같이 통일, view는 이렇게 나오도록!

```
videoWadmin.py x videoWmodels.py x musicWmodels.py x views.py
1 from django.db import models
2
3 class Album(models.Model):
4     album_title = models.CharField(max_length=500)
5     artist = models.CharField(max_length=250)
6     genre = models.CharField(max_length=100)
7     album_logo = models.CharField(max_length=1000)
8
9
10
11
12 class Song(models.Model):
13     album = models.ForeignKey(Album, on_delete=models.CASCADE)
14     file_type = models.CharField(max_length=10)
15     song_title = models.CharField(max_length=200)
16
```



Here are all my Albums:

- [Reds](#)
- [Oh!](#)