

Python Django

5강

Python Django

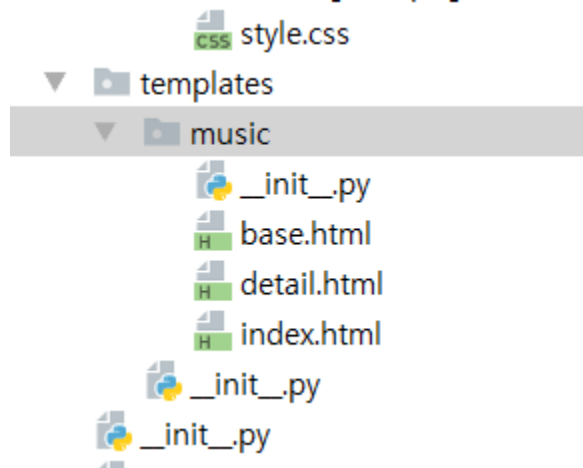
1. Base Template 만들기

우리가 만든 네비게이터 -> 모든 view 에 표현되어야 할 요소

⇒ 이렇게 모든 view에 표시 해야할 요소들을 하나의 base template에 넣어
그것을 다른 html 파일에 연결 시킬 것임 (이것이 base template의 역할)

1) Template/music 안에 base.html 생성 후

스크립트 부분은 <head> 안의 <title>아래, 우리가 만든 <nav>는 <body> 안에 넣어주기!



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
  {% load staticfiles %}
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
  <link href="https://fonts.googleapis.com/css?family=Satisfy|Slabo+27px" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="{% static 'music/style.css' %}" />
  <script src = "http://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-inverse" ...>

</body>
</html>
```

Python Django

1. Base Template 만들기

2) Base.html에 block 위치를 설정해주기

block : 우리가 실제로 다른 기능을 할 부분에 블록을 이용해 다른 html의 코드를 적용하도록 해줌
→ Base.html안의 block 부분에 index.html, detail.html에서 설정해준 해당 블록이 들어가게 됨!

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>{% block title %}MLWJIC{% endblock %}</title>
  {% load staticfiles %}
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
  <link href="https://fonts.googleapis.com/css?family=Satisfy|Slabo+27px" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="{% static 'music/style.css' %}" />
  <script src = "http://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-inverse"...>
  {% block body %}
  {% endblock %}
</body>
</html>
```

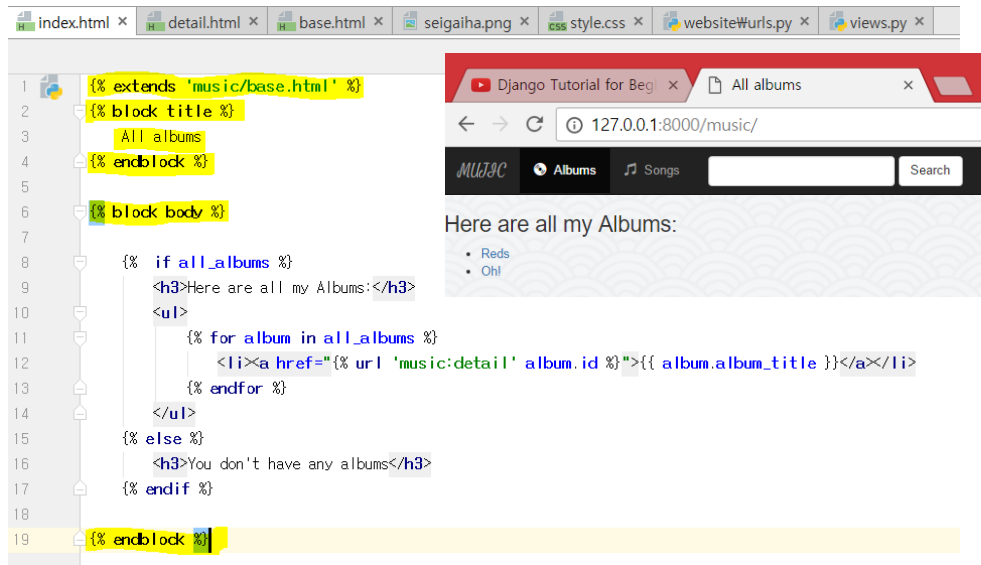
보면 이해 가니까
일단 base.html에
이렇게 block을 추가해 봅시다 ^.^

Python Django

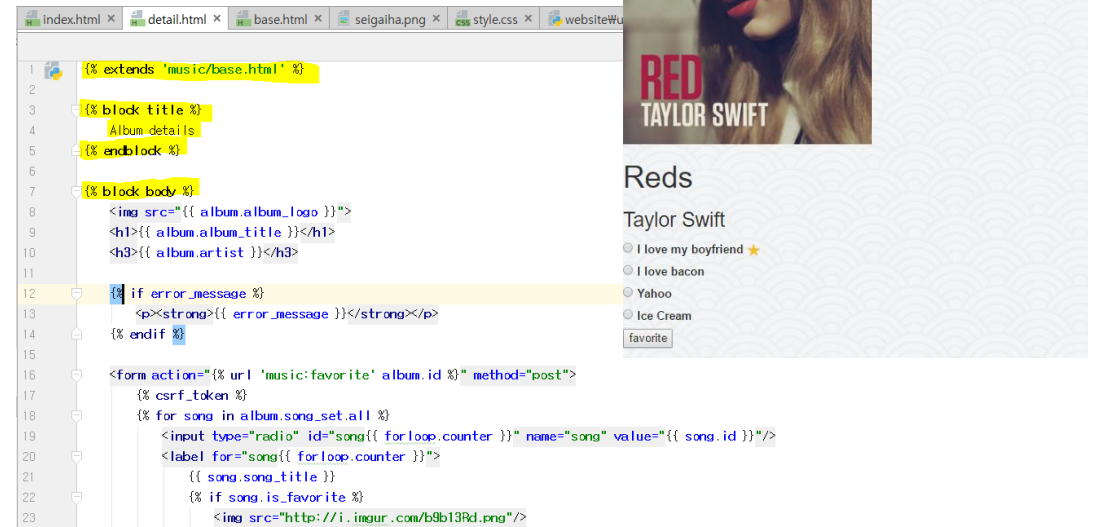
1. Base Template 만들기

3) 하위 템플릿과 베이스 템플릿 연결하기

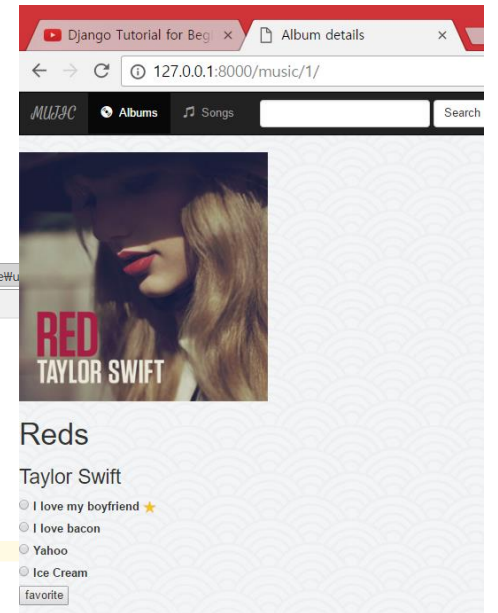
다른 하위 템플릿에 base template 연결해주기 / block부분이 될 곳 설정해주기!



```
1 {% extends 'music/base.html' %}
2 {% block title %}
3     All albums
4 {% endblock %}
5
6 {% block body %}
7
8     {% if all_albums %}
9         <h3>Here are all my Albums:</h3>
10        <ul>
11            {% for album in all_albums %}
12                <li><a href="{% url 'music:detail' album.id %}">{{ album.album_title }}</a></li>
13            {% endfor %}
14        </ul>
15    {% else %}
16        <h3>You don't have any albums</h3>
17    {% endif %}
18
19 {% endblock %}
```



```
1 {% extends 'music/base.html' %}
2
3 {% block title %}
4     Album details
5 {% endblock %}
6
7 {% block body %}
8     
9     <h1>{{ album.album_title }}</h1>
10    <h3>{{ album.artist }}</h3>
11
12    {% if error_message %}
13        <p><strong>{{ error_message }}</strong></p>
14    {% endif %}
15
16    <form action="{% url 'music:favorite' album.id %}" method="post">
17        {% csrf_token %}
18        {% for song in album.song_set.all %}
19            <input type="radio" id="song{{ forloop.counter }}" name="song" value="{{ song.id }}" />
20            <label for="song{{ forloop.counter }}">
21                {{ song.song_title }}
22                {% if song.is_favorite %}
23                    
24                {% endif %}
25            </label>
26        {% endfor %}
27    </form>
```



이렇게 해주면 base 템플릿에 지정해 준 해당 이름의 block 위치에 하위 템플릿의 코드가 들어가 각 url에 맞게 view가 작동이 됩니다!!

Python Django

View 클래스에 대한 설명은 여기 더 자세히 나와있어요!
<http://ruaa.me/django-view/>

2. Generic View

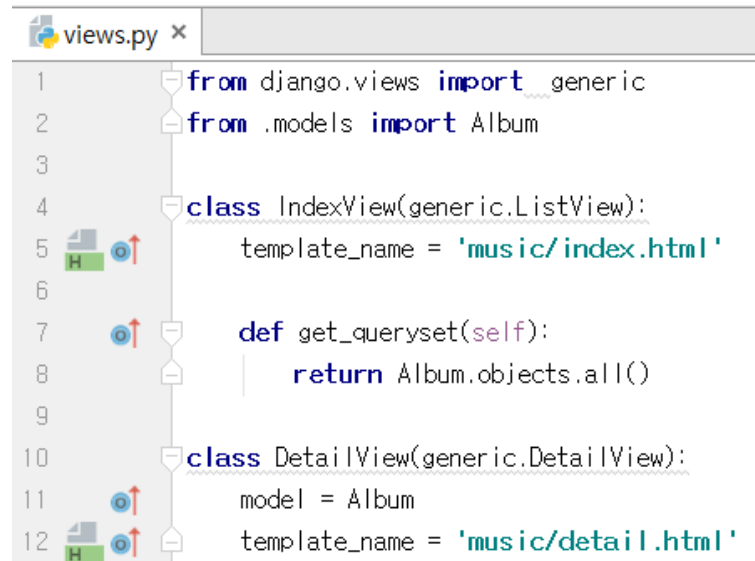
보통 우리가 사용하는 모든 사이트의 형식이 대체로 비슷하다! (List -> 각 항목에 대한 Detail)

-> 장고에서는 이런 view 형식을 라이브러리로 제공한다

더 간단한 새로운 view를 써 주기 위해 여태 했던 view.py의 모든 것을 삭제 해보겠습니다 8ㄱ8

삭제 후에

- 1) Generic view 라이브러리 불러오기
+ model도 필요하니까 같이 불러와 줘시다
- 2) 우리가 필요한 형태의 generic view를 불러오기
-> 그것을 사용할 때 class 상속으로서 불러온다
- 3) 해당 view를 실행시킬 때에 사용할 html을 연결 시켜주기
-> template_name 에 연결 url 설정
- 4) ListView -> queryset function 만들기
/ DetailView -> 사용할 model을 불러와 주기



```
views.py x
1  from django.views import generic
2  from .models import Album
3
4  class IndexView(generic.ListView):
5      template_name = 'music/index.html'
6
7      def get_queryset(self):
8          return Album.objects.all()
9
10 class DetailView(generic.DetailView):
11     model = Album
12     template_name = 'music/detail.html'
```

Python Django

2. Generic View

Before

```
index.html x detail.html x views.py x urls.py x style.css x
1 from django.shortcuts import render, get_object_or_404
2 from .models import Album, Song
3
4 def index(request):
5     all_albums = Album.objects.all()
6     return render(request, 'music/index.html', {'all_albums': all_albums})
7
8 def detail(request, album_id):
9     album = get_object_or_404(Album, pk=album_id)
10    return render(request, 'music/detail.html', {'album': album})
11
```

After

```
views.py x
1 from django.views import generic
2 from .models import Album
3
4 class IndexView(generic.ListView):
5     template_name = 'music/index.html'
6
7     def get_queryset(self):
8         return Album.objects.all()
9
10 class DetailView(generic.DetailView):
11     model = Album
12     template_name = 'music/detail.html'
13
```

Python Django

2. Generic View

5) urls.py 수정해주기

- 기존에 각 url 실행시 function을 넘겨주는 것 대신 이제는 view class의 method 를 넘겨주게 될 것임!
- > as_view : 해당 class에 진입하기 위한 진입 메서드
- Detail 함수에 같이 넘겨주던 앨범 id를 'pk'로 바꿔 주기

```
views.py x  urls.py x
1  from django.conf.urls import url
2  from . import views
3
4  app_name='music'
5
6  urlpatterns = [
7      # /music/
8      url(r'^$', views.IndexView.as_view(), name='index'),
9      # /music/IDNum/
10     url(r'^(?P<pk>[0-9]+)/$', views.DetailView.as_view(), name='detail'),
```

* pk 로 바꾸지 않으면 이런 에러 발생!

```
<  →  ↻  127.0.0.1:8000/music/1/
AttributeError at /music/1/
Generic detail view DetailView must be called with either an object pk or a slug.
```

Python Django

2. Generic View

6) 연결된 html 파일 수정

Index.html -> album의 정보를 html에 연결시켜 보내줬었는데,
과거에는 function에서 정해진 이름대로 연결이 되었음 하지만 이제는 기본 default 이름이 'object_list' 가 됨

```
index.html x detail.html x views.py x urls.py x
1 {% extends 'music/base.html' %}
2 {% block title %}
3     All albums
4 {% endblock %}
5
6 {% block body %}
7
8     {% if object_list %}
9         <h3>Here are all my Albums:</h3>
10        <ul>
11            {% for album in object_list %}
12                <li><a href="{% url 'music:detail' %}" %}
13            {% endfor %}
14        </ul>
15    {% else %}
16        <h3>You don't have any albums</h3>
17    {% endif %}
18
19 {% endblock %}
```

- 하지만 object_list 라는 이름을 재정의 할 수 있음
-> context_object_name 을 다른 이름으로 재정의 해주면 끝!

```
index.html x detail.html x views.py x urls.py x
op#대학동아리#포리프#파이썬스터디#Django#website3
2 from .models import Album
3
4 class IndexView(generic.ListView):
5     template_name = 'music/index.html'
6     context_object_name = 'all_albums'
7
8     def get_queryset(self):
9         return Album.objects.all()
10
11 class DetailView(generic.DetailView):
12     model = Album
13     template_name = 'music/detail.html'
```

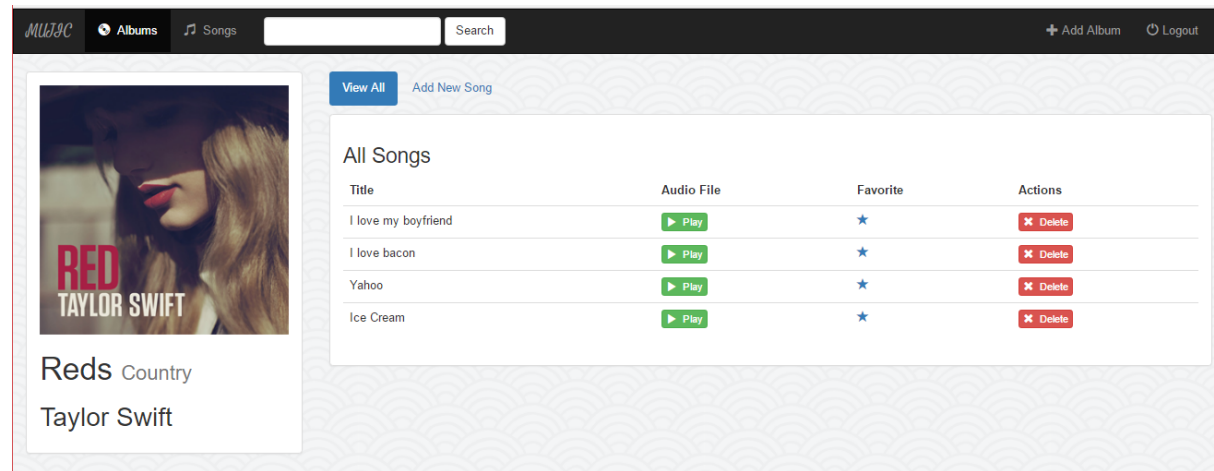
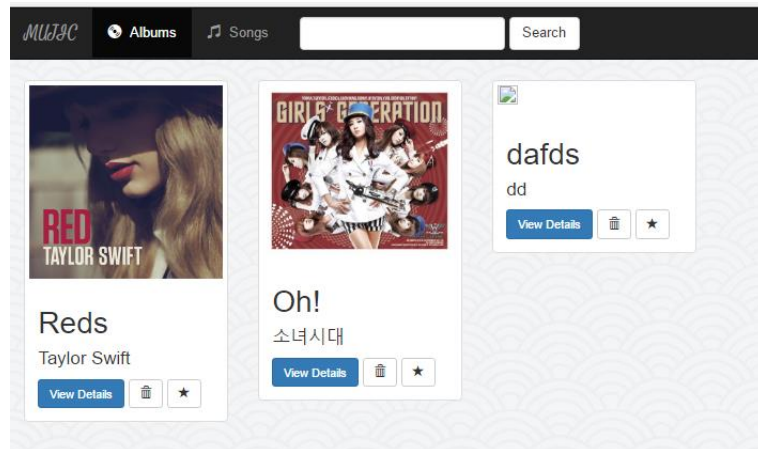
아직 detail 부분은 에러떠요..! 맞게 잘한거랍니다

Python Django

잠깐만!

다음 내용으로 들어가기 앞서서, 새로운 html 코드가 필요합니다

여기서는 기본적으로 부트스트랩을 다루는 것을 기반으로 하기 때문에,
이후에 갑자기 보지 못한 인터페이스가 등장하게 됩니다!!!! (저도 그래서 멘붕...)
그래서 여러분께 그 인터페이스가 적용 되어있는 html 파일을 나눠 드릴 테니
원래 내용에 맞춰 붙여넣기 해주시면 됩니다
각각 index, detail에 적용하면 아래와 같은 화면이 나오게 됩니다!



Python Django

3. Model Forms - Create View

Model Forms : 사용자가 정보를 입력, 수정 할 수 있도록 마련해 놓는 하나의 형식

-> 사용자들이 정보를 입력, 수정 할 수 있을 뿐만 아니라 그 정보를 손쉽게 admin을 통해 관리도 가능!

1) model.py 에 새로운 라이브러리 추가 + get_absolute_url 메서드 생성

```
models.py x
1 from django.db import models
2 from django.core.urlresolvers import reverse
3
4 class Album(models.Model):
5     album_title = models.CharField(max_length=500)
6     artist = models.CharField(max_length=250)
7     genre = models.CharField(max_length=100)
8     album_logo = models.CharField(max_length=1000)
9
10     def get_absolute_url(self):
11         return reverse('music:detail', kwargs={'pk':self.pk})
12
13     def __str__(self):
14         return self.album_title + ' - ' + self.artist
15
```

get_absolute_url

-> 그 해당 앨범의 detail 페이지의 url 을 넘겨줄 수 있도록 함

reverse(view, 그에 필요한 keyword -> kwargs)

* Key 가 그 class 안에 있으므로 self.pk 로 불러와 준다!

Python Django

3. Model Forms - Create View

2) views.py 에 새로운 라이브러리 추가 + CreateView 이용한 class 생성

```
models.py x views.py x
1 from django.views import generic
2 from django.views.generic.edit import CreateView, UpdateView, DeleteView
3 from .models import Album
4
5 class IndexView(generic.ListView):
6     template_name = 'music/index.html'
7     context_object_name = 'all_albums'
8     def get_queryset(self):
9         return Album.objects.all()
10
11 class DetailView(generic.DetailView):
12     model = Album
13     template_name = 'music/detail.html'
14
15 class AlbumCreate(CreateView):
16     model = Album
17     fields = ['artist', 'album_title', 'genre', 'album_logo']
```

Model -> 내가 실제로 생성한 모델의 form
Field -> 사용자에게 추가하게 할 영역들
(model 이름 그대로 가져오면 좋음)

3) urls.py 에서 음악추가 url 생성 해주기

```
#/music/album/add/
url(r'^album/add/$', views.AlbumCreate.as_view(), name='album-add')
```

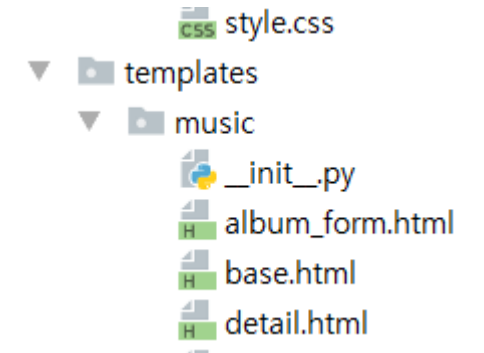
Python Django

3. Model Forms - Create View

4) Album-add를 위한 새로운 html 파일 생성

-> 형식은 '**Model Name**_form -> 따라서 여기서는 album_form

5) album_form.html 작성해주기



```
models.py x views.py x urls.py x album_form.html x
1 {% extends 'music/base.html' %}
2 {% block title %}Add a New Album{% endblock %}
3
4 {% block body %}
5 <form class="form-horizontal" action="" method="post" enctype="multipart/form-data">
6     {% csrf_token %}
7     {% include 'music/form-template.html' %}
8     <div class="form-group">
9         <div class="col-sm-offset-2 col-sm-10">
10             <button type="submit" class="btn btn-success">Submit</button>
11         </div>
12     </div>
13 </form>
14 {% endblock %}
```

Base.html을 extend 해준 후

각 블록 부분에 해당하는 내용을 추가해주면 ok

Block title -> 이름

Block body -> <form> 생성

‘music/form-template’ :

실제로는 <label><input>의 집합체이지만

이것은 다른 html에도 계속 쓰일 형식이므로

Generic view로써 만들어주고

include 하는 형식으로 써먹어주면 편하다!

따라서 일단은 써주고 나중에 추가해서 써줄 것임

Python Django

3. Model Forms - Create View

6) form-template.html 생성 후 작성

```
models.py x views.py x urls.py x album_form.html x form-template.html x
1 {% for field in form %}
2
3     <div class="form-group">
4         <div class="col-sm-offset-2 col-sm-10">
5             <span class="text-danger small">{{ field.errors }}</span>
6         </div>
7         <label class="control-label col-sm-2">{{ field.label_tag }}</label>
8         <div class="col-sm-2">{{ field }}
9         </div>
10    </div>
11
12 {% endfor %}
```

〈div〉를 field 개수 만큼 for 문으로 돌려주고 있음

Field.errors -> 입력한 필드명이 에러일 때

〈span〉이 나타나도록 함

Field.labet_tag -> field 명을 label 로서 나타내줌

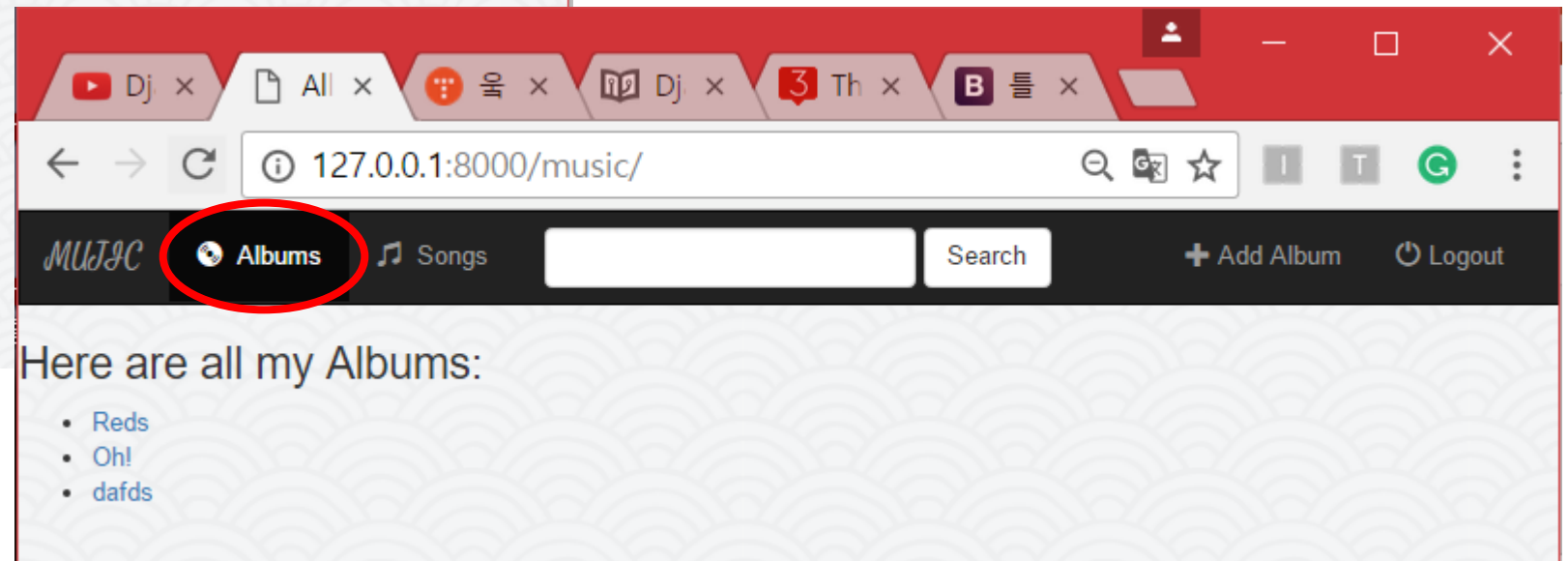
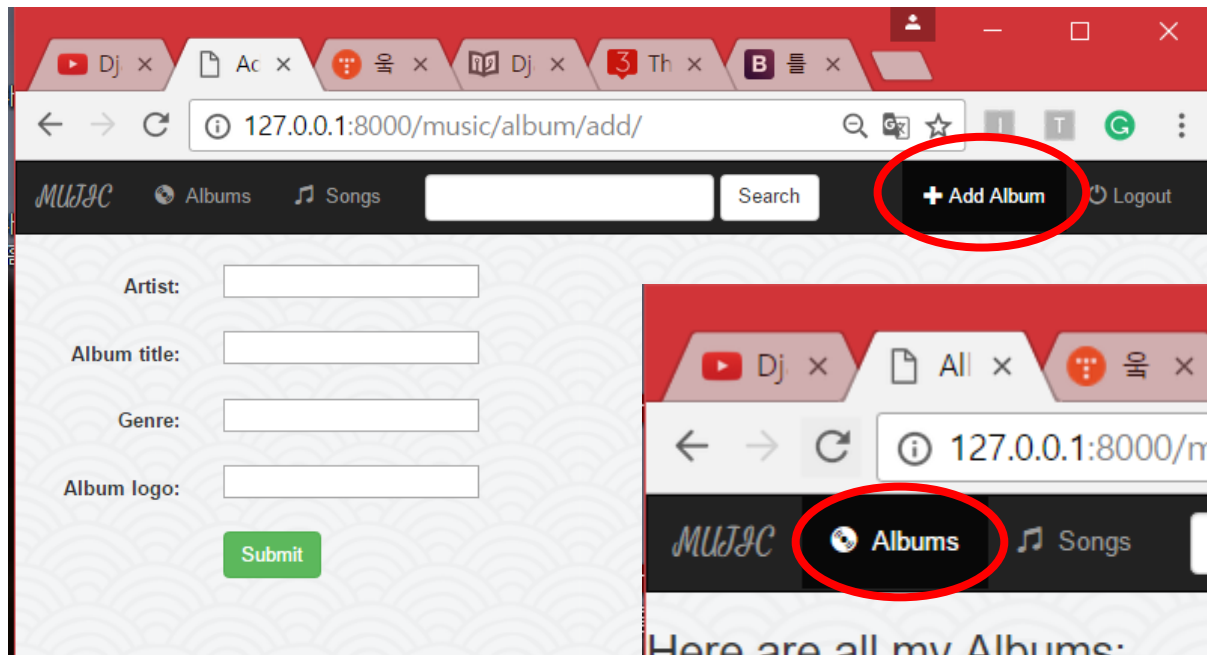
Field -> 입력 input box

7) base.html에서 add album 버튼에 url 연결 시켜주기

```
<ul class="nav navbar-nav navbar-right">
    <li class="">
        <a href="{% url 'music:album-add' %}">
            <span class="glyphicon glyphicon-plus" aria-hidden="true"></span> Add Album
        </a>
    </li>
```

Python Django

★중간 과제★ 이렇게 해당 탭의 사이트에 들어갔을 때 그 버튼만 활성화 되도록 만들어 보세요!



Python Django

4. Model Forms - Update/Delete View

Create View 처럼 Update, Delete View 도 설정할 수 있다!

1) views.py 에 reserve_lazy 라이브러리 추가 + update, delete view class 생성하기

```
models.py x views.py x urls.py x
1 from django.views import generic
2 from django.views.generic.edit import CreateView, UpdateView, DeleteView
3 from django.core.urlresolvers import reverse_lazy
4 from .models import Album
```

```
class AlbumCreate(CreateView):
    model = Album
    fields = ['artist', 'album_title', 'genre', 'album_logo']

class AlbumUpdate(UpdateView):
    model = Album
    fields = ['artist', 'album_title', 'genre', 'album_logo']

class AlbumDelete(DeleteView):
    model = Album
    success_url = reverse_lazy('music:index') Successful_url -> 삭제가 된 후에 돌아갈 url을 지정
```

Python Django

4. Model Forms - Update/Delete View

2) urls.py생성한 views 에 대한 url 생성하기

```
#!/music/album/add/  
url(r'^album/add/$', views.AlbumCreate.as_view(), name='album-add'),  
#!/music/album/pk/  
url(r'^album/(?P<pk>[0-9]+)/$', views.AlbumUpdate.as_view(), name='album-update'),  
#!/music/album/pk/delete  
url(r'^album/(?P<pk>[0-9]+)/delete/$', views.AlbumDelete.as_view(), name='album-delete')
```

위의 #의 형태로 url이 만들어지도록 내용을 붙여 놓고, 연결할 메서드와 이름명을 잘 맞춰서 바꿔주면 된다!

3) 후에 index.html에 delete url 연결 시켜주기

```
<!-- Delete Album -->  
<form action="{% url 'music:album-delete' album.id %}" method="post" style="...">  
    {% csrf_token %}  
    <input type="hidden" name="album_id" value="{{ album.id }}" />  
    <button type="submit" class="btn btn-default btn-sm">  
        <span class="glyphicon glyphicon-trash"></span>  
    </button>  
</form>
```

Url을 연결 시켜주면, 쓰레기통 버튼이 delete로서 활성화된다!