

230116_Django_Day1

▼ 웹 사이트 개발에 필요한 3가지

1. 비즈니스 로직
2. 화면
3. 데이터 저장소

- github.io같은 경우는 데이터베이스를 쓰지 않고 파일으로만 작성.
- 하지만 장고는 데이터베이스를 쓸 때 쿼리 구조로 사용
 - 기본 ORM이 제공되어 있다.
- 웹 사이트의 크기가 커지게 되는 경우 구조화된 코드가 필요하다.
 - 즉 프레임워크 사용 Flask, Django 등.
 - @(데코레이터) Flask에 있던 기능들이 다 쪼개져 있다.
- Library vs Framework
 - Library는 기존에 있던 기능을 빌려 쓰는 것.
 - Framework 는 형태가 짜여져 있어서 그 안에 데이터만 넣으면 됨.
 - 재사용성이 어렵다.
- MVC 패턴 - MTV(Model / Template / View)

```
pip install django # django 설치
django-admin startproject shinhanapp # shinhanapp 폴더 만들기
cd shinhanapp # shinhanapp 디렉터리로 이동
pip freeze > requirements.txt # requirement.txt에 pip 버전 설정
django-admin
python manage.py startapp member # member 디렉터리, 앱 생성(추가) (startapp 관련??)
python manage.py runserver # 초기값 127.0.0.1:8000 주소에 서버를 연다. 실행시
# django에서 success 관련된 내용이 출력
python manage.py makemigrations # migrate전에 먼저 입력 (commit-push 느낌)
python manage.py migrate # 어떤 데이터든 변경사항이 있을 시 명령어 실행한다.
```

- settings.py
 - DATABASES
 - sqlite3 장점

You have 1 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): member.

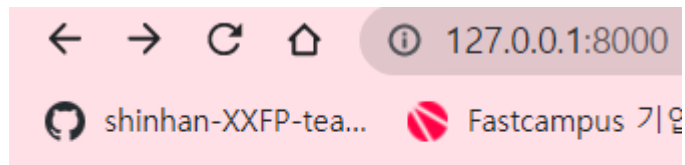
Run 'python manage.py migrate' to apply them.

- 위와 같은 에러코드 발생시
 - python manage.py migrate(반영되지 않은 값들이 있어서 반영 시켜주는 코드)

▼ 화면 출력 바꾸기(실습)

```
shinhanapp > member > views.py > main
1  from django.shortcuts import render
2  from django.http.response import HttpResponseRedirect
3
4  # Create your views here.
5
6
7  def main(request):
8      return HttpResponseRedirect('Hello!')
```

```
shinhanapp > shinhanapp > urls.py > ...
14  """
15  2. Add a URL to urlpatterns: pat
16  """
17  from django.contrib import admin
18  from django.urls import path
19  from member.views import main
20
21  urlpatterns = [
22      path('admin/', admin.site.urls),
23      path('', main),
24  ]
```



Hello!

- 성공적인 출력
 - 데이터 베이스 변경

```
shinhanapp > member > models.py > Member
1  from django.db import models
2
3  # Create your models here.
4
5  class Member(models.Model):
6      name = models.CharField(max_length=128, verbose_name='이름')
7      age = models.IntegerField(verbose_name='나이')
```

- python manage.py makemigrations
 - 데이터베이스 없는 값을 추가할 시 makemigrations을 해주면 다음과 같이 뜬다.

```
It is impossible to add a non-nullable field 'age' to member without specifying a default. This is because the database needs something to populate existing rows.
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit and manually define a default value in models.py.
Select an option: 1
```

- 기본값도 없고 비워있어서 무슨 값인지 모르겠다.
- 여기서 1번 입력하고 내가 직접 기본값(30)을 쓴다.

```
1) Quit and manually define a default value in models.py.
Select an option: 1
Please enter the default value as valid Python.
The datetime and django.utils.timezone modules are available, so it is possible to provide e.g. timezone.now as a value.
Type 'exit' to exit this prompt
>>> 30
Migrations for 'member':
  member/migrations/0002_member_age.py
  - Add field age to member
```

```

shinhanapp > member > migrations > 0002_member_age.py > ...
1  # Generated by Django 4.1.5 on 2023-01-16 00:53
2
3  from django.db import migrations, models
4
5
6  class Migration(migrations.Migration):
7
8      dependencies = [
9          ('member', '0001_initial'),
10     ]
11
12     operations = [
13         migrations.AddField(
14             model_name='member',
15             name='age',
16             field=models.IntegerField(default=30, verbose_name='나이'),
17             preserve_default=False,
18         ),
19     ]

```

- 새 파일이 생겼고 의존성이 생겼다 0001이 먼저 실행한 후 실행하겠다.
- 디폴트 값이 30이 들어갔다.

```

(venv) C:\Users\user\Desktop\2023\shinhanapp>python
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
KeyboardInterrupt
>>> quit()

```

- python을 잘 못 들어갔을 시 다른 명령어가 통하지 않으므로 quit()을 통해 나온다.
- `__pycache__`

```

└─ __pycache__
   ├── __init__.cpython-310.pyc
   ├── admin.cpython-310.pyc
   ├── apps.cpython-310.pyc
   ├── models.cpython-310.pyc
   └── views.cpython-310.pyc

```

- manage.py를 빼고 모든 파일은 어디서 가져다 쓰는(import) 파일인데 그 때 만들어지는 파일
- 컴파일이 안되는 인터프리터인 파이썬에 대해 컴파일을 미리 선언해주는 파일

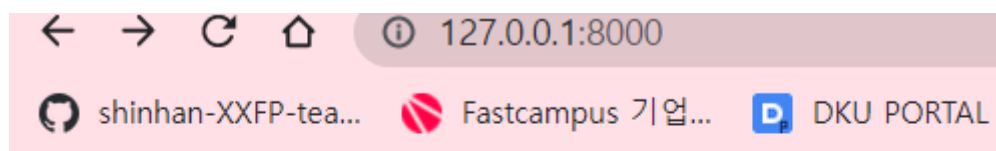
- 내가 어떤 파일이든 import하는 시점에 생긴다.
- 출력 바꾸기(기존에 세이브하는 값 말고 값 가져오는 출력)
- views.py

```
shinhanapp > member > views.py > main
1  from django.shortcuts import render
2  from .models import Member
3
4  # Create your views here.
5
6
7  def main(request):
8      member = Member.objects.get(pk=1)
9
10     return render(request, 'index.html', { 'member': member })
```

- pk(primary key) 값을 가져오는 것

```
shinhanapp > member > templates > index.html > html
1  <html>
2      <body>
3          <h1>Hello python! {{ member.name }} {{ member.age }}</h1>
4      </body>
5  </html>
```

- 출력 결과



Hello python! 테스트 30

shinhanapp > db.sqlite3

Search tables... Reset Filters Records: 7

Tables (12)	id	name	age
> django_migrations	1	테스트	30
> sqlite_sequence	2	테스트	30
> auth_group_permissions	3	테스트	30
> auth_user_groups	4	테스트	30
> auth_user_user_permissions	5	테스트	30
> django_admin_log	6	테스트	30
> django_content_type	7	테스트	40
> auth_permission			
> auth_group			
> auth_user			
> django_session			
> member_member			

- 데이터베이스 확인 가능

```
member = Member.objects.get(pk=100)
```

- pk를 100하면 오류가 난다. `Member matching query does not exist.`
- 모든 키 다 출력하기
 - `views.py`.
 - `all()`함수 사용

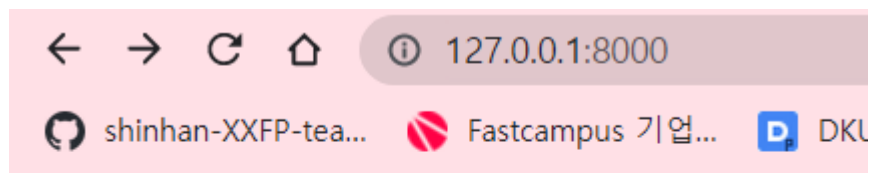
```
def main(request):
    member = Member.objects.all()

    return render(request, 'index.html', { 'member': member })
```

- index.html 수정

```
shinhanapp > member > templates > index.html > html
1 <html>
2   <body>
3     <h1>Hello python!</h1>
4     <ul>
5       {% for member in member %}
6       <li>{{member.name}} {{member.age}}</li>
7       {% endfor %}
8     </ul>
9   </body>
10 </html>
```

- {% %} 문법은 띄어쓰기가 아닌 그대로 쓰는 게 중요
- 출력 결과



Hello python!

- 테스트 30
- 테스트 30
- 테스트 30
- 테스트 30
- 테스트 30
- 테스트 30
- 테스트 40

▼ filter()함수 사용

- views.py

```
def main(request):
    member = Member.objects.filter(age__gte=35)

    return render(request, 'index.html', { 'member': member })
```

- age가 35를 기준으로 gte(크거나같다)만 가져오는 것.

Hello python!

• 테스트 40

- member = Member.objects.filter(name="신한")
 - 이런식으로 바꾸면 값이 안나오면 ⇒ 왜냐 없기 때문에
- member = Member.objects.filter(name__contains="테스")
 - 값이 나온다. name은 "테스트" __contains "테스"를 포함하기 때문이다.

▼ get()함수 사용

- member = Member.objects.get(name="테스트")
- 위 코드 실행시 에러 발생
- `get() returned more than one Member -- it returned 7!`
 - get는 조건의 검사가 하나만 나올 때 가져오는 함수이기 때문에 에러 발생.(그렇기 때문에 pk를 써서 가져온다.)
 - 그 뒤에 .first()(값을 하나만 꺼내는 것) 같은 것을 써도 동일.

▼ order_by함수

- member = Member.objects.filter(name__contains="테스").order_by('-age')
 - 역순시 - 사용
- 함수 chaining
 - .으로 이어서 계속 함수를 쓰는 것(성능이 안 좋을 수는 있지만 가능은 하다.)

▼ product startapp 만들기

- python manage.py start

▼ Filed Type

Field Type	설명
CharField	제한된 문자열 필드 타입. 최대 길이를 max_length 옵션에 지정해야 한다. 문자열의 특별한 용도에 따라 CharField의 파생클래스로서, 이메일 주소를 체크를 하는 EmailField, IP 주소를 체크를 하는 GenericIPAddressField, 콤마로 정수를 분리한 CommaSeparatedIntegerField, 특정 폴더의 파일 패스를 표현하는 FilePathField, URL을 표현하는 URLField 등이 있다.
TextField	대용량 문자열을 갖는 필드
IntegerField	32 비트 정수형 필드. 정수 사이즈에 따라 BigIntegerField, SmallIntegerField 을 사용할 수도 있다.
BooleanField	true/false 필드. Null 을 허용하기 위해서는 NullBooleanField를 사용한다.
DateTimeField	날짜와 시간을 갖는 필드. 날짜만 가질 경우는 DateField, 시간만 가질 경우는 TimeField를 사용한다.
DecimalField	소숫점을 갖는 decimal 필드
BinaryField	바이너리 데이터를 저장하는 필드
FileField	파일 업로드 필드
ImageField	FileField의 파생클래스로서 이미지 파일인지 체크한다.
UUIDField	GUID (UUID)를 저장하는 필드

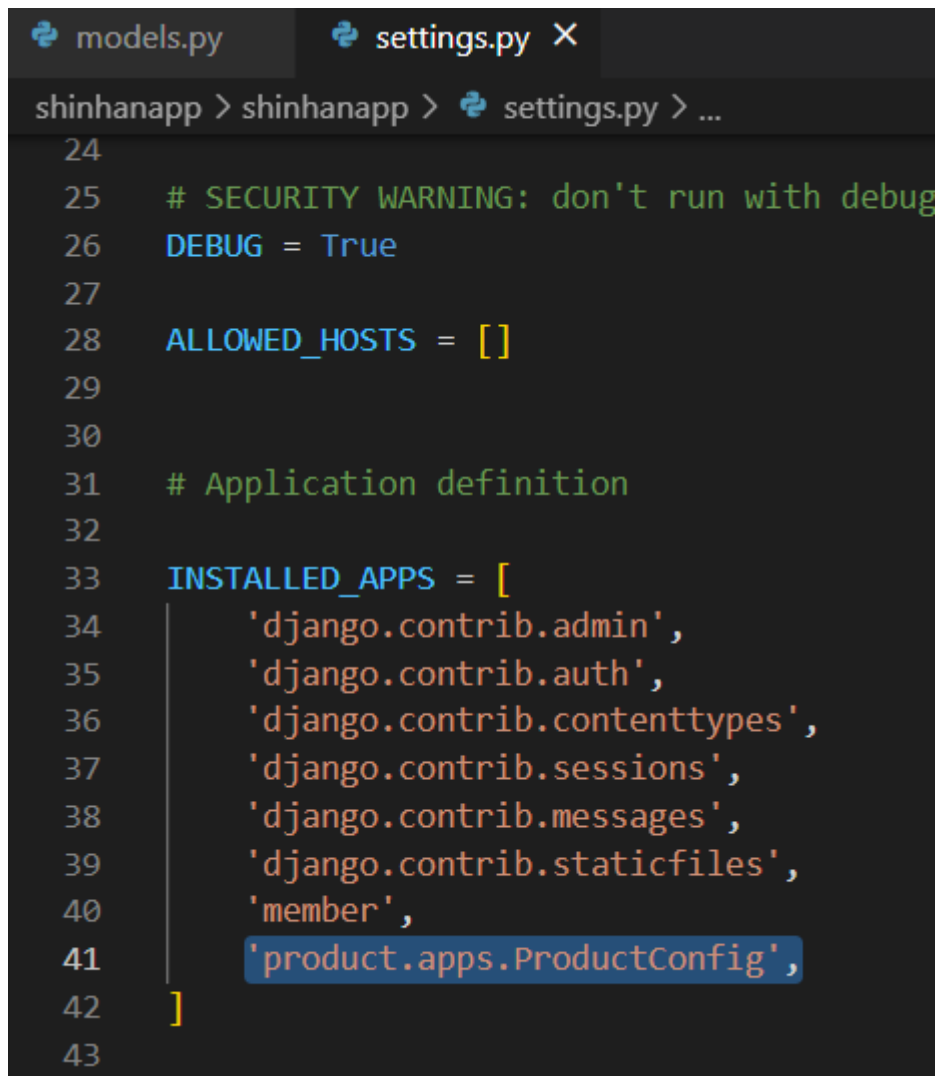
- model 초기 설정

```

models.py × settings.py
shinhanapp > product > models.py > Product > Meta
1  from django.db import models
2
3  # Create your models here.
4
5  class Product(models.Model):
6      # title, content, price, location
7      title = models.CharField(max_length=128, verbose_name='상품명')
8      content = models.TextField(verbose_name="상품내용")
9      price = models.IntegerField(verbose_name='가격')
10     location = models.CharField(max_length=128, verbose_name='위치')
11
12     class Meta:
13         db_table = 'shinhan_product'

```

- 경로 추가하기(설정파일으로 추가해달라.)

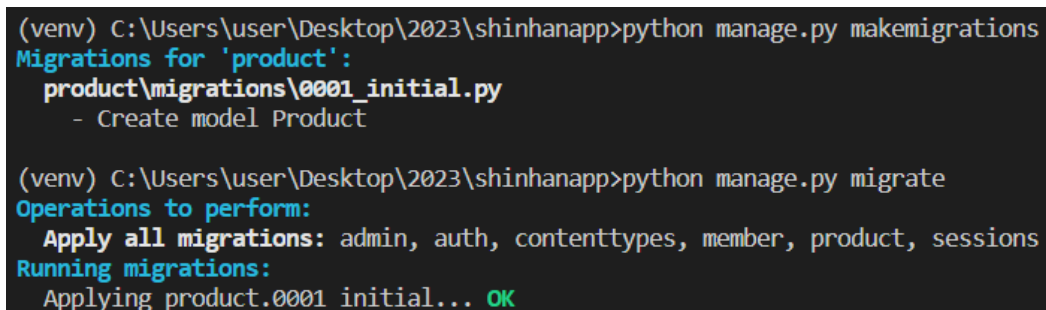


```

shinhanapp > shinhanapp > settings.py > ...
24
25 # SECURITY WARNING: don't run with debug
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'member',
41     'product.apps.ProductConfig',
42 ]
43

```

- 변경사항 저장



```

(venv) C:\Users\user\Desktop\2023\shinhanapp>python manage.py makemigrations
Migrations for 'product':
  product\migrations\0001_initial.py
    - Create model Product

(venv) C:\Users\user\Desktop\2023\shinhanapp>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, member, product, sessions
Running migrations:
  Applying product.0001_initial... OK

```

- class meta로 설정했던 것처럼 설정한 이름으로 데이터베이스가 만들어짐

```
member_member  
> shinhan_product
```

- python manage.py createsuperuser
 - <http://127.0.0.1:8000/admin/>
 - admin 12345678로 생성
- 한글 설정

```
models.py db.sqlite3 settings.py X  
shinhanapp > shinhanapp > settings.py > ...  
107  
108 LANGUAGE_CODE = 'ko-kr'  
109  
110 TIME_ZONE = 'Asia/Seoul'  
111  
112 USE_I18N = True  
113  
114 USE_TZ = False  
115
```

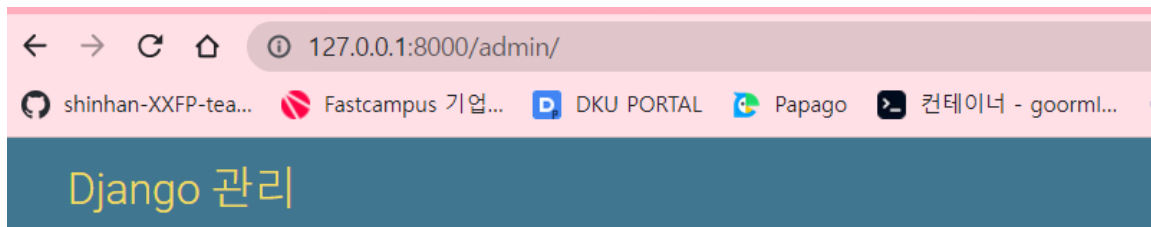
- 실행화면



- member 생성

```
models.py × settings.py admin.py ...\product admin.py ...\member ×
shinhanapp > member > admin.py > MemberAdmin
1 from django.contrib import admin
2 from .models import Member
3
4 # Register your models here.
5
6 @admin.register(Member)
7 class MemberAdmin(admin.ModelAdmin):
8     pass
```

- 실행화면



사이트 관리

MEMBER	
Members	+ 추가 ✎ 변경
인증 및 권한	
그룹	+ 추가 ✎ 변경
사용자(들)	+ 추가 ✎ 변경

```
models.py ...\product  settings.py  admin.py  models.py ...\member
shinhanapp > member > models.py > Member > Meta
1  from django.db import models
2
3  # Create your models here.
4
5  class Member(models.Model):
6      name = models.CharField(max_length=128, verbose_name='이름')
7      age = models.IntegerField(verbose_name='나이')
8
9      class Meta:
10         verbose_name='회원'
11         verbose_name_plural = '회원'
```

회원
회원
+ 추가 ✎ 변경

- 상품 등록하기

상품
상품
+ 추가 ✎ 변경

- admin.py

```
models.py ...\product  settings.py  admin.py ...\member  admin.py ...\product X
shinhanapp > product > admin.py > ProductAdmin
1  from django.contrib import admin
2  from .models import Product
3
4  # Register your models here.
5
6  @admin.register(Product)
7  class ProductAdmin(admin.ModelAdmin):
8      pass
```

- apps.py

```
shinhanapp > product > apps.py > ...
1  from django.apps import AppConfig
2
3
4  class ProductConfig(AppConfig):
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'product'
7      verbose_name = '상품'
8
```

- models.py

```
class Meta:
    db_table = 'shinhan_product'
    verbose_name='상품'
    verbose_name_plural = '상품'
```

- member>models.py

```
def __str__(self):
    return f"{self.name}: {self.age}세"
```

- 이름 그대로 바뀜.

변경할 회원 선택

액션:

- ☐ 회원
- ☐ 테스트: 40세
- ☐ 테스트: 30세
- ☐ 테스트: 30세
- ☐ 테스트: 30세
- ☐ 테스트: 30세
- ☐ 테스트: 30세
- ☐ 테스트: 30세

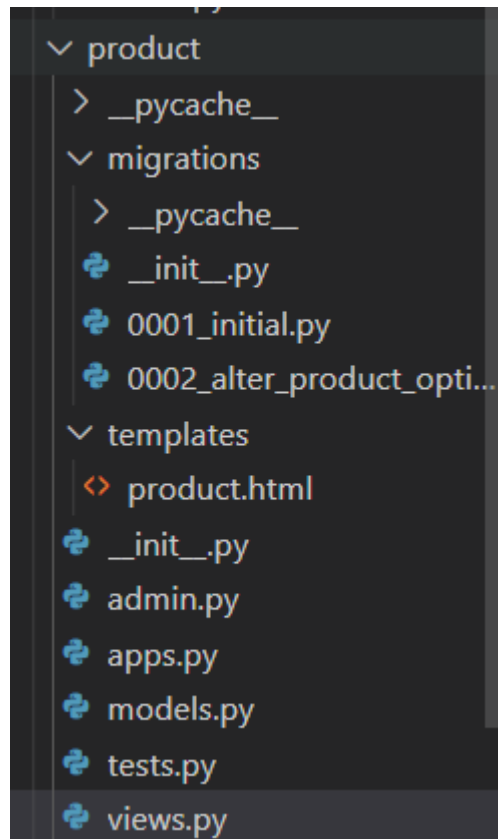
•

• 매직 메소드 (Magic method)

- 클래스 안에 정의된 함수를 우리는 특별히 '메소드(method)'라고 부릅니다. 메소드 중에서 `__`로 시작해서 `__`로 끝나는 메소드들이 있는데 이를 매직 메소드 또는 특별 메소드(special method)라고 부릅니다. 가장 유명한 매직 메소드에는 `__init__`이라는 생성자가 있습니다. 생성자는 어떤 클래스의 인스턴스(객체)가 생성될 때 파이썬 인터프리터에 의해 자동으로 호출되는 메소드였습니다. 이처럼 파이썬의 매직 메소드는 특별한 기능을 제공합니다.

▼ 상품 목록 출력하기

- product 폴더 안에 templates폴더를 만들고 product.html 파일 만들기.



- views.py 수정

```
shinhanapp > product > views.py > main
1  from django.shortcuts import render
2  from .models import Product
3
4  # Create your views here.
5
6
7  # templates 폴더 만들고 index.html --
8  # main 함수 만들어서 상품 리스트 나오게 하기
9  # 상품 리스트에는 한줄로 상품명, 가격, 장소 나오게 하기
10
11 def main(request):
12     products = Product.objects.all()
13     return render(request, 'product.html', { 'products': products })
```

- models.py


```

shinhanapp > product > models.py > Product
1  from django.db import models
2
3  # Create your models here.
4
5  class Product(models.Model):
6      # title, content, price, location
7      title = models.CharField(max_length=128, verbose_name='상품명')
8      content = models.TextField(verbose_name="상품내용")
9      price = models.IntegerField(verbose_name='가격')
10     location = models.CharField(max_length=128, verbose_name='위치')
11
12     class Meta:
13         db_table = 'shinhan_product'
14         verbose_name='상품'
15         verbose_name_plural = '상품'

```

- shinhanapp>urls.py

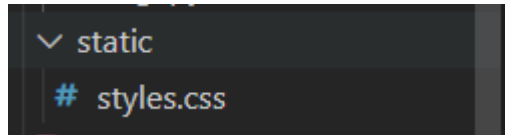
```

shinhanapp > shinhanapp > urls.py > ...
14     2. Add a URL to urlpatterns: path(
15     """
16     from django.contrib import admin
17     from django.urls import path
18
19     from product.views import main
20
21     urlpatterns = [
22         path('admin/', admin.site.urls),
23         path('', main),
24     ]

```

▼ css 설정

- static디렉터리 및 파일 생성



- styles.css

```
shinhanapp > static > # styles.css > body
1  @font-face {
2      font-family: 'KOTRAHOPE';
3      src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2110@1.0/KOTRAHOPE.woff2') format('woff2');
4      font-weight: normal;
5      font-style: normal;
6  }
7
8  @font-face {
9      font-family: 'GangwonEdu_OTFBoldA';
10     src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2201-2@1.0/GangwonEdu_OTFBoldA.woff') format('woff');
11     font-weight: normal;
12     font-style: normal;
13 }
14
15 body {
16     font-family: 'GangwonEdu_OTFBoldA';
17 }
```

- settings.py

```
shinhanapp > shinhanapp > settings.py > ...
116
117 # Static files (CSS, JavaScript, Images)
118 # https://docs.djangoproject.com/en/4.1/howto/static-files/
119
120 STATIC_URL = 'static/'
121 STATIC_DIR = BASE_DIR / 'static'
122 STATICFILES_DIRS = [STATIC_DIR]
123
124 MEDIA_URL = 'media/'
125 MEDIA_ROOT = BASE_DIR / 'media'
126
127 # Default primary key field type
128 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
129
130 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
131
```

-

▼ 당당마켓 만들기

- <https://getbootstrap.kr/docs/5.2/utilities/flex/>
- 부트스트랩 사용
- Django 공식 문서 참고

- class에 값을 집어넣는 방식으로 사용

jQuery 3.x

- jQuery Core 3.6.3 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)

- minified 사용
- urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('product/<int:pk>/', detail),
    path('', main),
]
```

- 각 url마다 쓰이는 함수들이 다름.
- <http://127.0.0.1:8000/> >> main함수 출력
- <http://127.0.0.1:8000/product/1/> >> detail함수 출력
- 과제:/product/write/
 - 페이지 작성(기능까진 x) flask 연동
 - 주소 찾기 까지 해오면 더 좋다.
- 현재 작업
- 과제 완성

Django

This document describes Django's built-in template tags and filters. It is recommended that you use the automatic documentation , if available, as this will also include

 <https://docs.djangoproject.com/en/4.1/ref/templates/builtins/>

