Django 치트시트

1. 프로젝트 및 앱 설정

- 새 프로젝트 시작: `django-admin startproject <project_name>`
- 새 앱 생성: `python manage.py startapp <app_name>`
- `settings.py`에 앱 등록:

```
INSTALLED_APPS = [
    ...
    '<app_name>',
]
```

- 개발 서버 실행: `python manage.py runserver`

2. 모델 (Models)

`models.py` 파일에 데이터베이스 스키마를 정의합니다.

- 모델 정의:

```
from django.db import models

class Post(models.Model):
    title =
models.CharField(max_length=200)
    content = models.TextField()
    created_at =
models.DateTimeField(auto_now_add=True)

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

- 주요 필드 타입:
  - `CharField`, `TextField`, `IntegerField`, `FloatField`, `BooleanField`
  - `DateTimeField`, `DateField`, `TimeField`
  - `ForeignKey`, `ManyToManyField`, `OneToOneField`
- 마이그레이션:
  - `python manage.py makemigrations`: 모델 변경사항에 대한 마이그레이션 파일 생성.
  - `python manage.py migrate`: 마이그레이션을 데이터베이스에 적용.

3. 쿼리셋 (QuerySet) API

데이터베이스와 상호작용하는 방법.

- 모든 객체 가져오기: `Post.objects.all()`
- 특정 객체 가져오기 (기본 키): `Post.objects.get(pk=1)`
- 필터링: `Post.objects.filter(title__startswith='Django')`, `Post.objects.exclude(created_at__year=2023)`
- 정렬: `Post.objects.order_by('-created_at')`
- 객체 생성: `Post.objects.create(title='New Post', content='Content')`
- 객체 저장 및 업데이트:

```
post = Post.objects.get(pk=1)
post.title = 'Updated Title'
post.save()
```

- 객체 삭제: `post.delete()`

4. 뷰 (Views)

`views.py` 파일에 비즈니스 로직을 작성합니다.

- 함수 기반 뷰 (Function-Based Views):

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

def post_list(request):
    posts = Post.objects.all()
    return render(request, 'app_name/
post_list.html', {'posts': posts})
```

- 클래스 기반 뷰 (Class-Based Views):

```
from django.views.generic import
ListView
from .models import Post

class PostListView(ListView):
    model = Post
    template_name = 'app_name/
post_list.html'
    context_object_name = 'posts'
```

5. URL

`urls.py` 파일에서 URL 패턴과 뷰를 연결합니다.

- 프로젝트 `urls.py`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('app/',
include('app_name.urls')),
]
```

- 앱 `urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.post_list,
name='post_list'),
    # 클래스 기반 뷰 사용 시
    # path('',
views.PostListView.as_view(),
name='post_list'),
]
```

6. 템플릿 (Templates)

HTML 파일에 Django 템플릿 언어(DTL)를 사용합니다.

- 변수 출력: `{{ my_variable }}`
- 태그 사용:

```
{% for post in posts %}
    <h2>{{ post.title }}</h2>
    <p>{{ post.content|
truncatewords:30 }}</p>
{% endfor %}

{% if user.is_authenticated %}
    <p>Welcome, {{ user.username }}!</
p>
{% endif %}
```

- URL 참조: `{% url 'post_list' %}`
- 템플릿 상속:
  - `{% extends 'base.html' %}`
  - `{% block content %}{% endblock %}`

7. 폼 (Forms)

`forms.py` 파일에 폼을 정의합니다.

```
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'content']
```

- 뷰에서 폼 처리:

```
def post_create(request):
    if request.method == 'POST':
        form = PostForm(request.POST)
        if form.is_valid():
            form.save()
            # ...
    else:
        form = PostForm()
    return render(request, 'app_name/
post_form.html', {'form': form})
```

- 템플릿에서 폼 렌더링:

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</
button>
</form>
```

8. 관리자 (Admin)

`admin.py` 파일에 모델을 등록하여 관리자 페이지에서 관리합니다.

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

admin.site.register(Post)
```

- 관리자 계정 생성: `python manage.py createsuperuser`