

Sinamon

천천히 그리고 조금씩..

ProjectManager
박정욱

BackendDeveloper
이재연

FrontendDeveloper
서장원
천재용

시나몬이란?

시나몬은 순우리말인 시나브로와
모임과 사람의 의미를 포함하고 있습니다.

1
시나브로
모르는 사이에
조금씩 조금씩



2
모임
여러 가지
오프라인 모임



3
사람
각 분야의
다양한 사람들

운영진 소개

이번 모임의 운영을 맡으신 분들입니다!!
모임에 대한 건의사항이나 문의사항이 있을 경우 이분들에게 이야기해주세요^^



Lee JaeYeon
이재연

현) 시나몬 백엔드개발자
전) (주)TNDN 백엔드개발자



Park JungWook
박정욱

현) 시나몬 대표
전) (주)다노 백엔드개발자



Park UnChol
박운철

현) LG 개발자



자기소개 및 아이디어 소개

The background of the slide features a wide-angle photograph of a coastal road. The road curves along the edge of a steep, densely forested hillside. Below the road, the ocean is visible with white-capped waves crashing against the rocks. The sky is clear and blue.

커리큘럼 소개

이번 스터디의 커리큘럼입니다.

- 1회차
Orientation
- 2회차
Django Girls Live Coding
- 3회차
Django Girls Tutorial 실습 & Django Architecture

- 
- 4회차
HTTP 통신에 대해서 알아보기
 - 5회차
Model과 Form
 - 6회차
Template & Bootstrap
 - 7회차
View
 - 8회차
OAuth2에 대해서 알아보고 Social Login을 실습



팀빌딩

목차

장고란?

왜 장고를 쓸까?

MTV 아키텍쳐

개발 환경 설정

개발툴

Git

장고란?

보안이 우수하고 유지보수가 편리한 웹사이트를
신속하게 많도록 도움을주는 파이썬 웹서버 프레임워크로 다양한 기능을 제공한다.
(관리자페이지, 로그인, 회원가입, 인증 등)

Complete(완결성 있는)

Django는 "Batteries included" 의 철학을 기반으로 합니다.
Batteries included 철학이란 전자 제품을 구입하여 포장을 풀면 배터리가 패키지에 포함되어 있으며,
즉시 사용할 수 있는 것처럼 재사용 할 수 있는 소프트웨어 패키지 / 모듈 제공을 목표로 합니다.

Scalable(확장성 있는)

Django는 컴퓨터 기반의 "shared-nothing" 아키텍처를 사용합니다.
shared-nothing 이란 각각의 아키텍처가 독립적이어서 필요하다면 교체 및 변경할 수 있는 것입니다.

Maintainable(유지보수가 쉬운)

Django 코드는 유지보수가 쉽고 재사용하기 좋게끔 하는 디자인 원칙 및 패턴들을 이용하여 작성됩니다.
특히 Don't Repeat Yourself (DRY) 원칙을 적용해서 불필요한 중복이 없고 많은 양의 코드를 줄였습니다.

왜 장고를 쓸까?

보안이 우수하고 유지보수가 편리한 웹사이트를
신속하게 많도록 도움을주는 파이썬 웹서버 프레임워크로 다양한 기능을 제공한다.
(관리자페이지, 로그인, 회원가입, 인증 등)

장고를 이용하여 만든 사이트 소개

Scalable(확장성 있는)Plus Python은 다양한 분야에서 광범위하게 활용되는 언어이다.

Data Science(Numpy, 사이킷런, Tensor Flow, Keras, pyTorch…)

웹스크롤링 (BeautifulSoup)

빌드 시스템(SCons), 빌드 보조 툴로 펄에서 파이썬으로 넘어가고 있다.

Python + Django 공부?

MTV 아키텍쳐 및 장고의 코드 흐름

MVC 패턴

- Model : DB와 같은 Data를 관리합니다.
- View : 화면을 그립니다.
- Controller : 이러한 동작을 제어합니다.
- 코드의 복잡도를 줄임과 동시에 재사용 성도 높이는 패턴입니다

유저

- 브라우저에서 login과 같은 요청을 합니다. (Http request)

URLS

- url patterns에 따라 View에 request를 전달합니다.

View

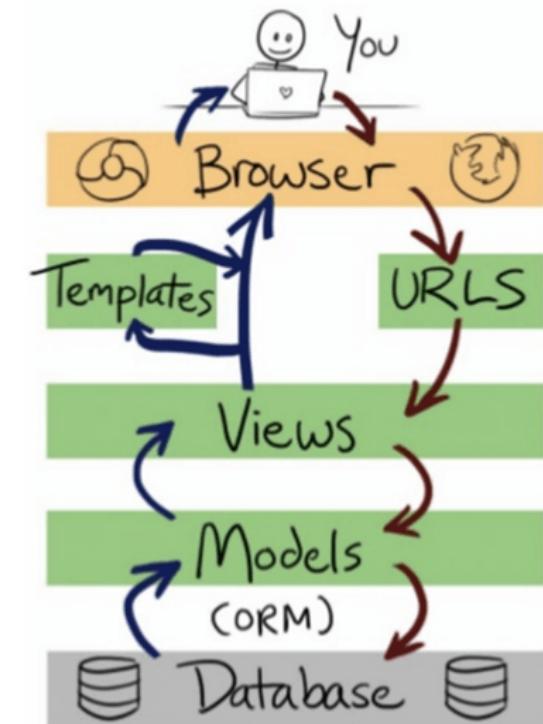
- 요청 내용에 따라 Models를 이용하여 데이터베이스에 접근하거나 다른 로직을 수행합니다.

Models

- Database를 추상화하여 (python manage.py migrate/makemigration) command를 통해 db를 생성할 수 있고, 데이터를 저장 수정 제거 등을 수행합니다.

View

- request처리를 마치면, view의 render()함수나 HttpResponse등으로 결과 값을 리턴합니다.



개발 환경 (python 설치 및 가상 환경 설정)

Python 3.7.x or 3.6.x 설치

- Python3 -V 명령어로 설치여부 확인 후, 설치 안 되어 있으면,
<https://www.python.org/downloads/>에서 설치

Python 가상 환경 설치

- virtualenvwrapper
- pipenv

가상환경 package 설치

- pip3 install pipenv
- pip3 install virtualenvwrapper

Hello Django

- mkdir djangoprj
 - cd djangoprj
 - pipenv install django==2.1
 - pipenv shell
 - (djangoprj) django-admin startproject djangoprj .
 - (djangoprj) python manage.py runserver
-
- `__init__.py` 는 빈 파일입니다. 이 파일은 Python에게 이 디렉토리를 하나의 Python 패키지로 다루도록 지시합니다.
 - `settings.py` 는 웹사이트의 모든 설정을 포함하고 있습니다.
 - `urls.py` 는 사이트의 url과 뷰의 연결을 지정해줍니다. 여기에는 모든 url 매핑 코드가 포함될 수 있지만, 특정한 어플리케이션에 매핑의 일부를 할당해주는 것입니다.
 - `wsgi.py` 는 당신의 장고 어플리케이션이 웹서버와 연결 및 소통하는 것을 돕습니다.
 - `manage.py` 스크립트는 어플리케이션을 생성하고, 데이터베이스와 작업하고, 그리고 개발 웹 서버를 시작하기 위해 사용합니다.

Start App

(djangoprj)python startapp catalog

- app은 python package로 INSTALLED_APPS 배열에 등록하여야 사용할 수 있습니다.
- DRY, django framework은 admin, auth, contenttypes, session, messages, staticfiles 관련된 것을 앱 형태로 제공해 줍니다.
- app들은 plug in 형태로 load_module함수로 실제 실행 시점에 로드됩니다.
- admin.py는 url(localhost:8000/admin)에 model을 등록할 수 있게 하는 모듈로 데이터 베이스를 쉽게 접근할 수 있게 합니다.
- apps.py는 추가된 app의 entry point입니다.
- model.py는 데이터 베이스를 추상화하는 모듈로 view에서 해당 모듈을 이용하여 디비에 데이터를 추가/제거/변경할 수 있습니다.
- tests.py는 unit test를 하는 모듈로써 습관적으로 작성해 주면 좋습니다.
- view.py는 request에 대한 적절한 로직이 추가 되는 모듈입니다.
- urls.py를 생성하여 연결할 수 있습니다.

개발툴

Visual Code

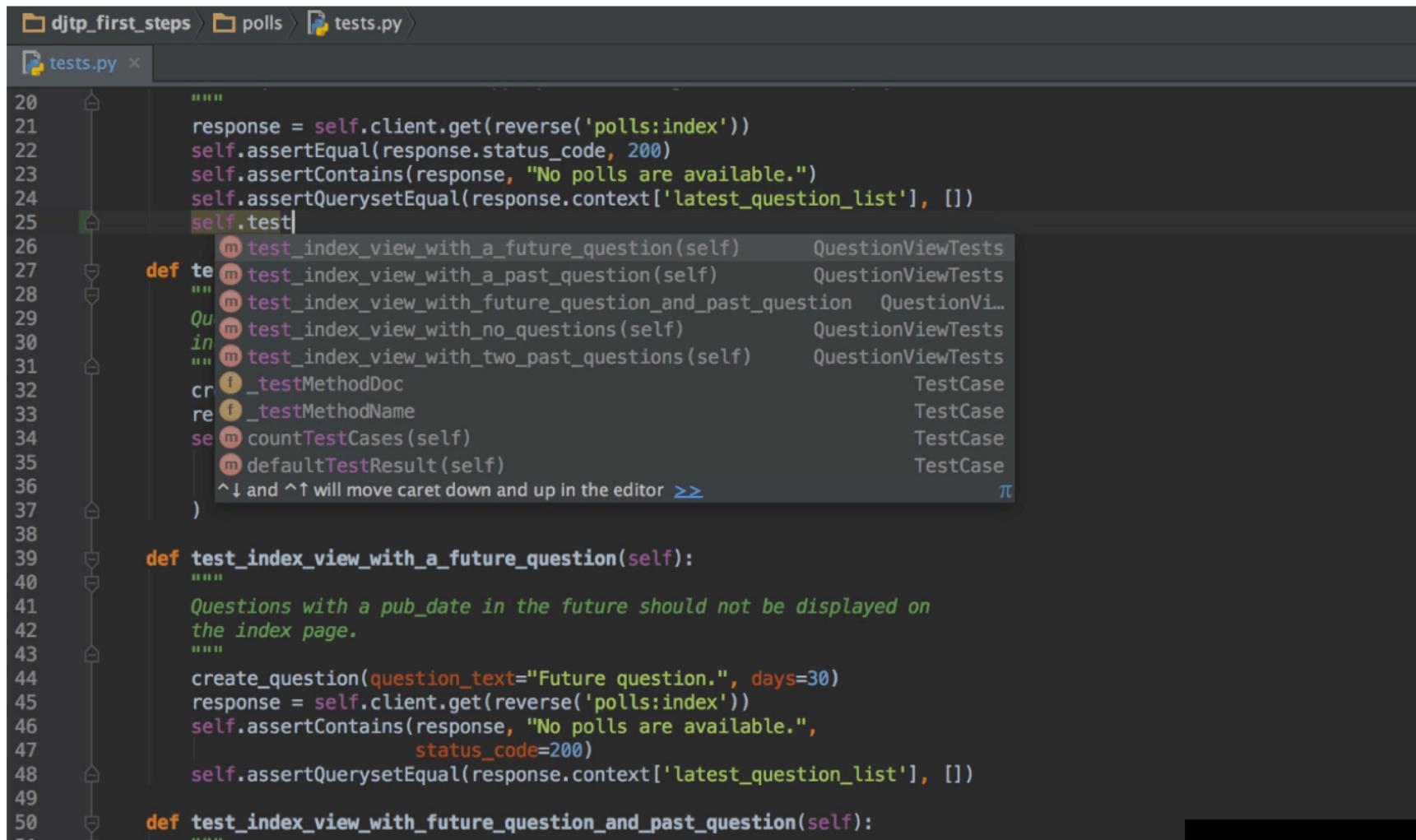
A screenshot of the Visual Studio Code interface. The title bar says "settings.py — Untitled (Workspace)". The left sidebar shows a file tree with a dark theme, displaying files like account_.helper.go, settings.py, urls.py, wsgi.py, db.sqlite3, manage.py, Pipfile, and Pipfile.lock. The main editor area shows the contents of settings.py:

```
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'de7qixy_w0+o8-s6ve=_6((tb7ewkqc!smx%4+fh3kmr8%uo(e'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
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 ]
41     'catalog',
42 ]
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
```

The status bar at the bottom shows "Python 3.7.0 64-bit ('djangoproj'): pipenv" and "Ln 40, Col 5 Spaces: 4 UTF-8 LF Python". There are also icons for analysis tools and notifications.

개발툴

PyCharm



The screenshot shows the PyCharm IDE interface with the file `tests.py` open. The code is part of a Django test suite for a poll application. The cursor is at line 25, where a test method is being defined. A code completion dropdown menu is open, listing several methods from the `TestCase` class, such as `_testMethodDoc`, `_testMethodName`, `countTestCases`, and `defaultTestResult`. The background of the code editor has a dark theme.

```
20     """
21     response = self.client.get(reverse('polls:index'))
22     self.assertEqual(response.status_code, 200)
23     self.assertContains(response, "No polls are available.")
24     self.assertQuerysetEqual(response.context['latest_question_list'], [])
25     self.test[Completion suggestions]
26         m test_index_view_with_a_future_question(self) QuestionViewTests
27     def te m test_index_view_with_a_past_question(self) QuestionViewTests
28         "" m test_index_view_with_future_question_and_past_question QuestionVi...
29     Qu m test_index_view_with_no_questions(self) QuestionViewTests
30     in m test_index_view_with_two_past_questions(self) QuestionViewTests
31     cr f _testMethodDoc
32     re f _testMethodName
33     se m countTestCases(self)
34     m defaultTestResult(self)
35     ^↓ and ^↑ will move caret down and up in the editor >>
36
37 )
38
39 def test_index_view_with_a_future_question(self):
40     """
41     Questions with a pub_date in the future should not be displayed on
42     the index page.
43     """
44     create_question(question_text="Future question.", days=30)
45     response = self.client.get(reverse('polls:index'))
46     self.assertContains(response, "No polls are available.",
47                         status_code=200)
48     self.assertQuerysetEqual(response.context['latest_question_list'], [])
49
50 def test_index_view_with_future_question_and_past_question(self):
51     """
```

개발툴

SublimeText

The screenshot shows the SublimeText interface with the following details:

- Left Sidebar (Folders):** Shows the file structure of the `django-master` repository. The `django` folder is expanded, showing its subfolders (`apps`, `bin`, `conf`, `contrib`, `core`, `db`, `dispatch`, `forms`, `jinja2`, `templates`) and files (`__init__.py`, `boundfield.py`, `fields.py`, `forms.py`, `formsets.py`, `models.py`, `renderers.py`, `utils.py`, `wIDGETS.py`, `http`, `middleware`, `template`, `templatetags`, `test`, `urls`, `utils`, `views`, `__init__.py`, `main.py`, `shortcuts.py`, `docs`, `extras`, `js_tests`, `scripts`, `tests`, `.editorconfig`, `.eslintignore`, `.eslintrc`, `.gitattributes`, `.gitignore`, `.hgignore`, `AUTHORS`, `CONTRIBUTING.rst`).
- Right Editor Window:** Displays the contents of the `forms.py` file. The code is as follows:

```
1 """
2 Form classes
3 """
4
5 import copy
6 from collections import OrderedDict
7
8 from django.core.exceptions import NON_FIELD_ERRORS, ValidationError
9 # BoundField is imported for backwards compatibility in Django 1.9
10 from django.forms.boundfield import BoundField # NOQA
11 from django.forms.fields import Field, FileField
12 # pretty_name is imported for backwards compatibility in Django 1.9
13 from django.forms.utils import ErrorDict, ErrorList, pretty_name # NOQA
14 from django.forms.widgets import Media, MediaDefiningClass
15 from django.utils.functional import cached_property
16 from django.utils.html import conditional_escape, html_safe
17 from django.utils.safestring import mark_safe
18 from django.utils.translation import gettext as _
19
20 from .renderers import get_default_renderer
21
22 __all__ = ('BaseForm', 'Form')
23
24
25 class DeclarativeFieldsMetaclass(MediaDefiningClass):
26     """Collect Fields declared on the base classes."""
27     def __new__(mcs, name, bases, attrs):
28         # Collect fields from current class.
29         current_fields = []
30         for key, value in list(attrs.items()):
31             if isinstance(value, Field):
32                 current_fields.append((key, value))
33                 attrs.pop(key)
34         attrs['declared_fields'] = OrderedDict(current_fields)
35
36         new_class = super(DeclarativeFieldsMetaclass, mcs).__new__(mcs, name, bases, attrs)
37
38         # Walk through the MRO.
39         declared_fields = OrderedDict()
40         for base in reversed(new_class.__mro__):
41             # Collect fields from base class.
42             if hasattr(base, 'declared_fields'):
43                 declared_fields.update(base.declared_fields)
44
45             # Field shadowing.
46             for attr, value in base.__dict__.items():
47                 if value is None and attr in declared_fields:
48                     declared_fields.pop(attr)
49
50         new_class.base_fields = declared_fields
51         new_class.declared_fields = declared_fields
52
53     return new_class
```

Git

- 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템입니다.
소프트웨어 개발에서 소스 코드 관리에 주로 사용 되지만, 어떠한 집합의 파일의 변경사항을 지속적으로 추적하기 위해 사용합니다.
(찾기 어려운 버그를 변경 사항을 기준으로 쉽게 찾을 수 있습니다.)
- git은 여러 사람이 수정하는 코드를 관리하기 위한 하나의 저장소를 가지는데,
이를 remote 저장소라고 하며, github, gitlab등 사이트에서
remote 저장소를 제공해 주고 있으며, “git init --bare <project name>.git”명령어를
이용하여 remote 저장소를 만들 수 있습니다.
- git clone “*” 명령어를 이용하여 remote server에 있는 코드를
로컬 PC에 다운로드할 수 있습니다.

Git

git add

- 서버에 반영하고 싶은 파일에 대해서 add하거나. git add .을 이용하여 전반 수정 파일을 add할 수 있습니다.

git commit -m "initial commit"

- commit을 하게 되면 commit id가 하나 생성되는데, 이 commit id가 git에서 관리하는 버전 단위입니다.

git status

- add를 한 파일과 add를 하지 않은 파일을 구분 가능합니다.

git log

- 커맨드로 반영된 소스의 history를 확인 가능합니다.

git diff commitid1 commitid2

- 두 commit사이의 수정 사항을 확인합니다.

git reset —hard commit-id

- 해당 commit을 제거 할 수 있는데, 이 때 수정한 소스가 모두 없어지는 것으로 주의해야 합니다.

Git

git push origin master

- remote 저장소에 수정한 commit들을 반영합니다.

origin

- remote 저장소

master

- 브랜치 이름

git pull/fetch

- 다른 개발자가 remote 저장소를 변경했을 경우 최신 사항을 업데이트하기 위해 사용하는 command입니다.

fetch

- .git 폴더 부분만 반영

git pull

- 코드 변경 사항까지 반영