Lab03:
Django Tutorial I

ESM2014-41

객체지향프로그래밍 및 실습

SKKU 시스템경영공학과

조영일

Setup Django

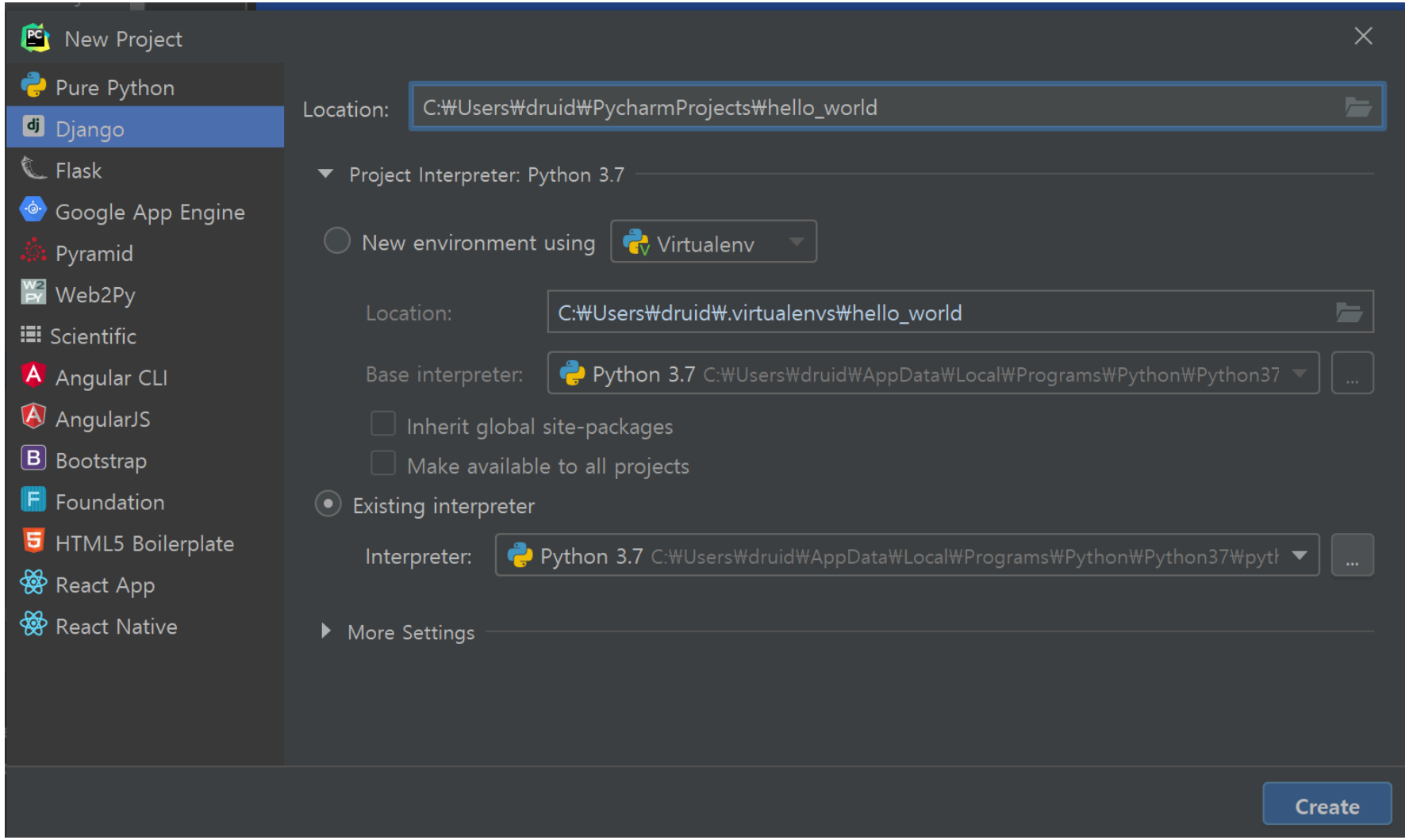
- Open command line : 시작 > 검색 > “cmd“
- “pip install django”
- “Successfully installed django-2.2.x sqlparse-0.3.x” 메시지가 출력되지 않는다면, Python 개발환경 설정이 잘 못된 것이니 강의 슬라이드 참조하여 수정 할 것

Start your first Django project in PyCharm

- PyCharm 실행
- File > New Project
- “Django” 선택
  - Location : 프로젝트가 저장될 장소(본인이 희망하는 디렉토리로 지정할 것 – “XXX\hello_world”)
  - Project Interpreter : “Existing Interpreter”선택 > 본인 컴퓨터 Python 3.7 설치 위치 지정
  - More Settings : “Application Name” > “myapp”(본인이 생성하길 희망하는 어플리케이션 이름 아무거나 입력)
- “Create” 버튼 클릭

- PyCharm이 설치된 Django Package를 토대로 Project Skeleton을 생성함
- PyCharm이 없어도 Django command를 통해 프로젝트 시작 가능

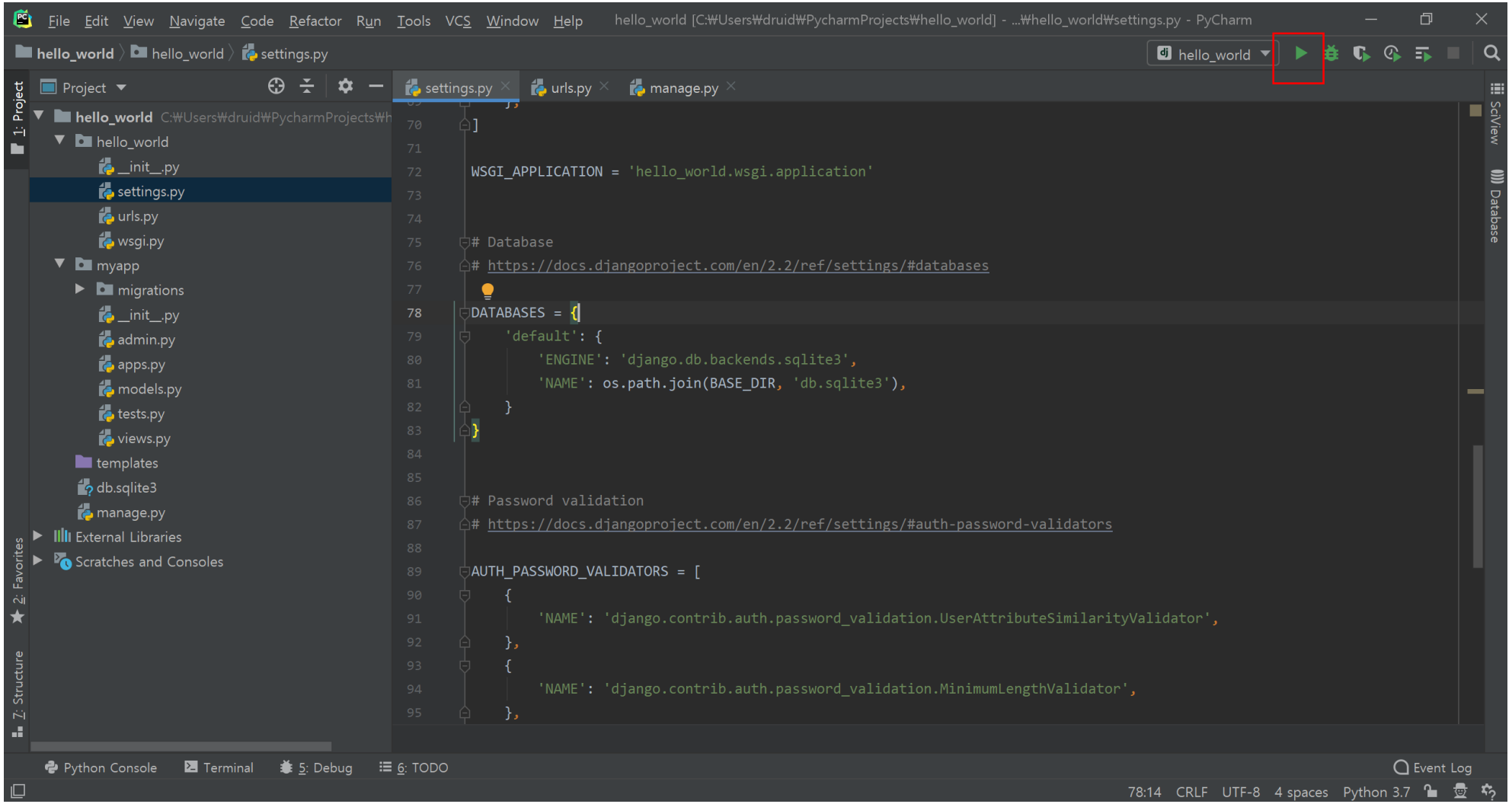
Start your first Django project in PyCharm



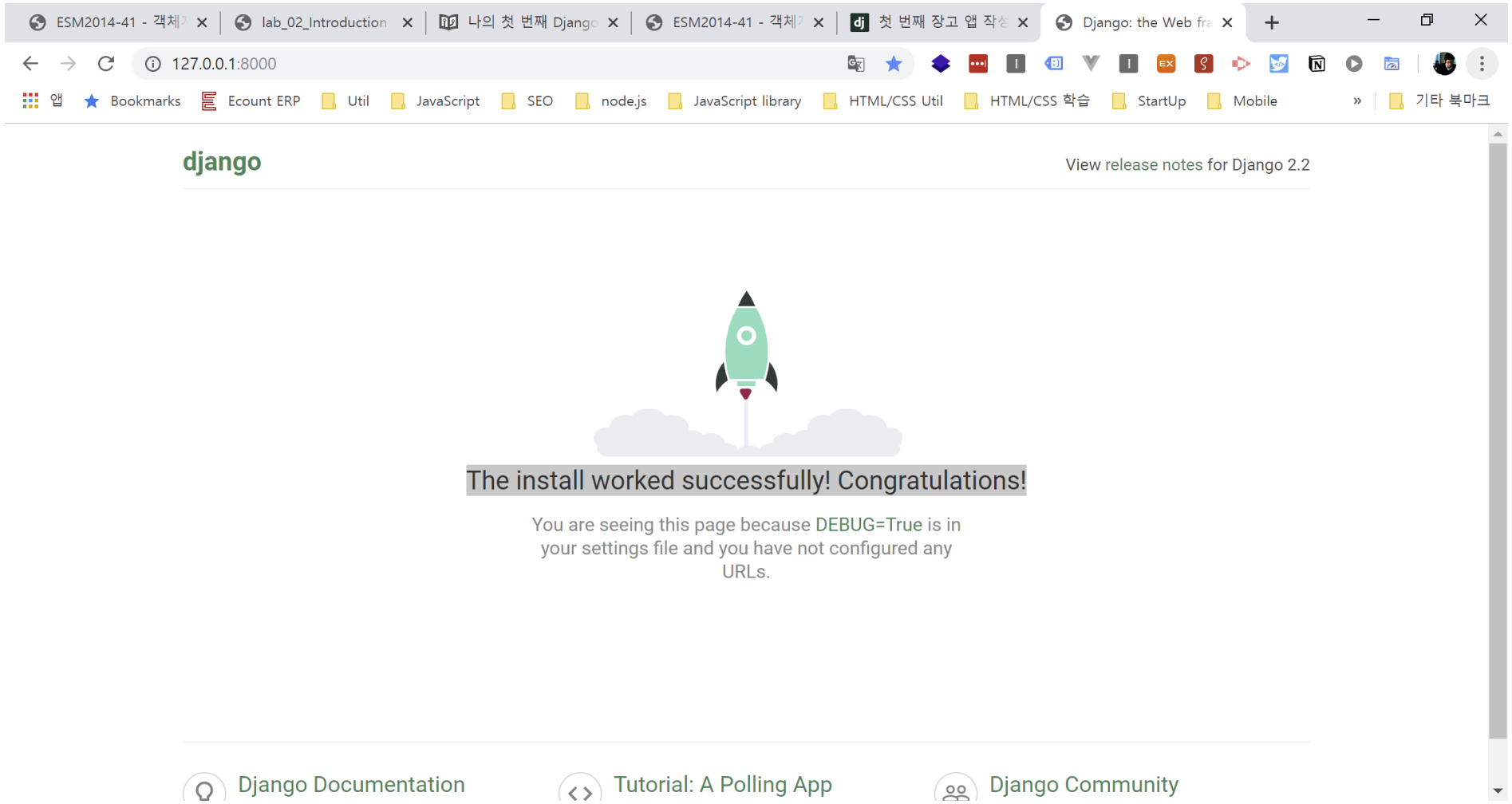
Start your first Django project in PyCharm

- “Run [project_name]” PyCharm (Shortcut : Shift + F10)
- Open your browser and Connect to http://127.0.0.1:8000

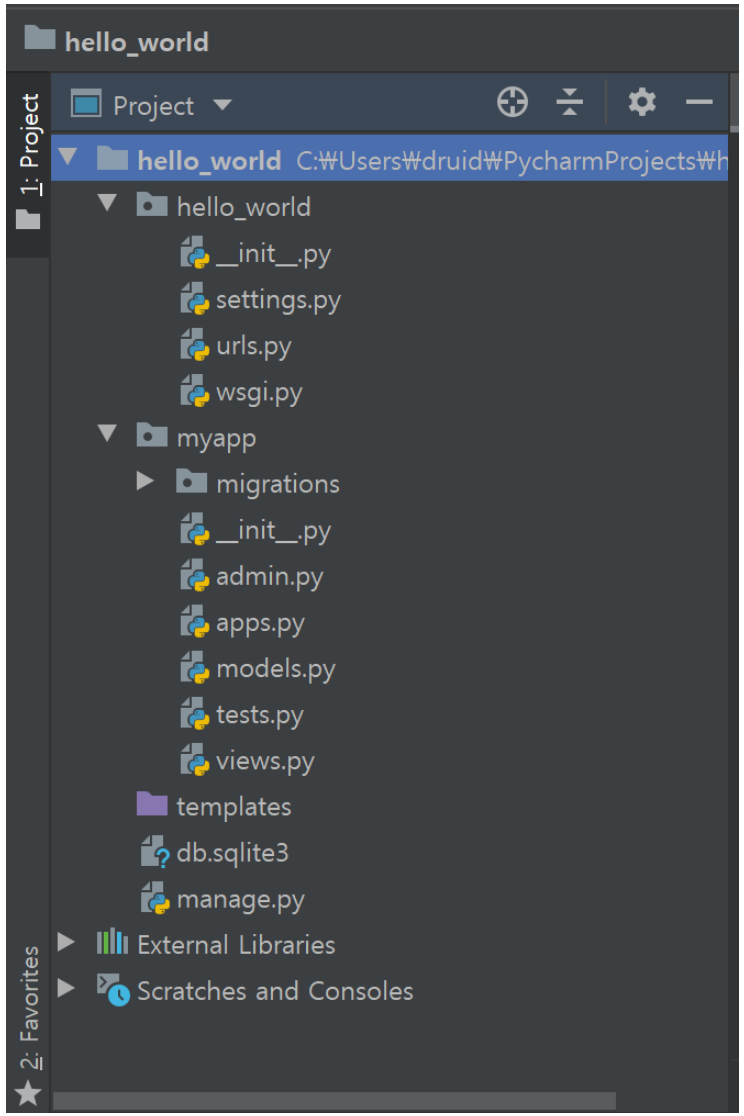
Start your first Django project in PyCharm



Start your first Django project in PyCharm

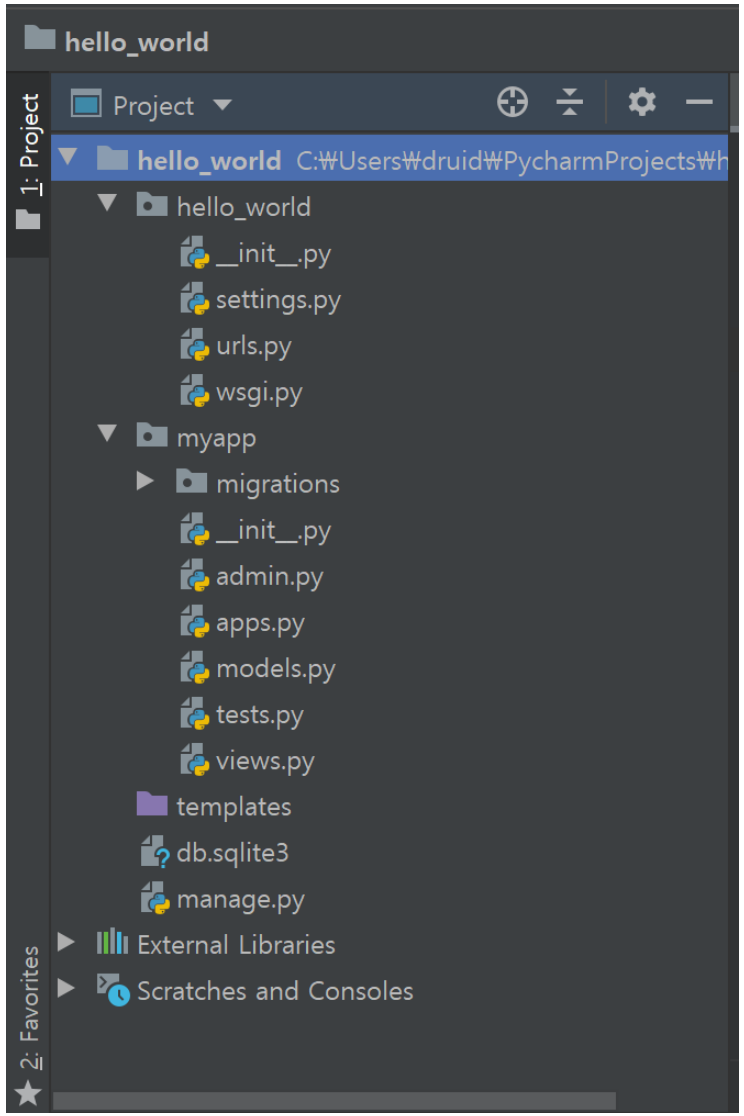


Django File Structure



- Project Name(Project Directory Name) : hello_world / App Name : myapp
- hello_world : Project Root(프로젝트의 최상위 디렉토리)
- manage.py : Django 프로젝트를 관리하기 위한 Django의 커맨드라인 유틸리티. manage.py 에 대한 자세한 정보는 django-admin and manage.py 에서 확인할 수 있음.
- hello_world/hello_world : 프로젝트를 위한 실제 Python Module이 저장되는 디렉토리

Django File Structure



- __init__.py : Python 에게 이 디렉토리가 Package임을 알려주기 위해 생성된 빈 파일

- settings.py : 해당 프로젝트와 관련된 Django 설정 값을 모아두기 위한 파일. Django에서 미리 정한 설정 들에 대해 이 파일에 정의하면 프로젝트에 반영된다. settings와 관련된 상세한 내용은 여기에서 확인 가능하다.

- urls.py : URL Dispatcher. 현재 프로젝트의 URL 목록을 정의한다. 해당 파일에 URL을 정의하고 그 URL을 처리하기 위한 View와 연결한다. 상세정보

- wsgi.py : 현재 프로젝트를 서비스하기 위한 WSGI 호환 웹 서버의 진입점. How to deploy with WSGI를 읽어보세요.

Django Project vs Application

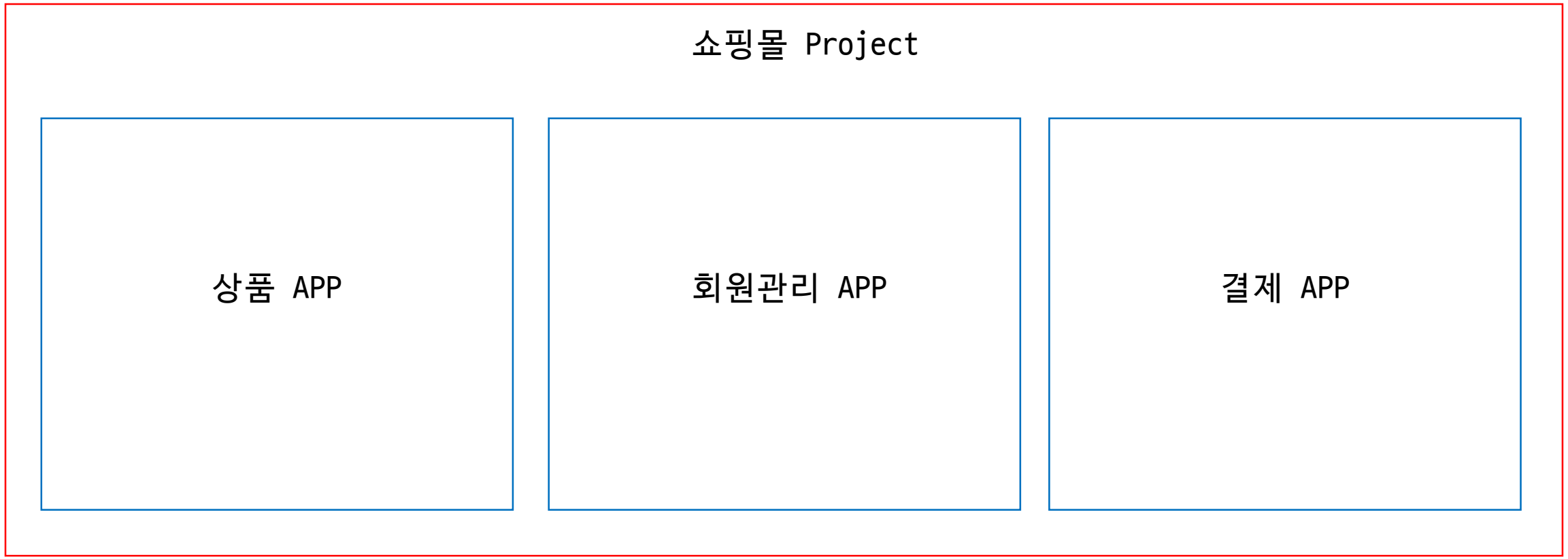
- Django는 코드의 재사용성을 높이기 위해 Project를 Application 단위로 나누어 구현하도록 한다.

- Project = Applications + Settings

- Project: 하나의 웹 서비스를 위해 여러 Application과 Settings을 결합하여 만든 결과물

- Application : 웹 사이트에서 특정한 기능을 수행하는 어플리케이션(관련 코드들의 집합), Application의 경우 구현에 따라 다수의 프로젝트에서 재사용 될 수 있다.

Django Project vs Application



Writing your first app

- 설문조사와 관련된 기능을 제공하는 “polls” 어플리케이션을 개발한다고 가정.
- 커맨드 라인 실행(“cmd“ or PyCharm > “Terminal)
- Project Root에서 아래 명령어 입력(Django 프로젝트를 관리하는 manage.py 에서 프로젝트에서 새로운 application을 생성하는 명령어)
- “python manage.py startapp polls”

Writing your first app(View)

- PyCharm에서 polls/views.py 더블클릭 하여 열기

- 다음과 같은 코드 작성

```
# Django에서 일반적인 HTTP 응답을 생성하기 위한 함수 - HttpResponse
from django.http import HttpResponse

# View - 함수 형태로 구현 할 수 있으며 항상 첫번째 argument로 request를 받는다.
def index(request):
    # Return 하는 값이 클라이언트에 HTTP Response로 전달된다.
    return HttpResponse("Hello, world. You're at the polls index.")
```

Writing your first app(View)

- PyCharm 내에서 polls 디렉토리 “마우스 오른쪽 클릭 > New > Python File” 선택 후 “urls.py” 파일 생성

- 생성된 urls.py 파일 열어서 아래와 같은 코드 작성

```
# urls.py 에서 url 경로를 지정하기 위해 필요한 함수
from django.urls import path

from polls import views


# urlpatterns 에서 실제 url 정의 목록과 그 정의를 처리하기 위한 view를 연결함.
urlpatterns = [
    path('', views.index, name='index'),
]
```

Writing your first app(View)

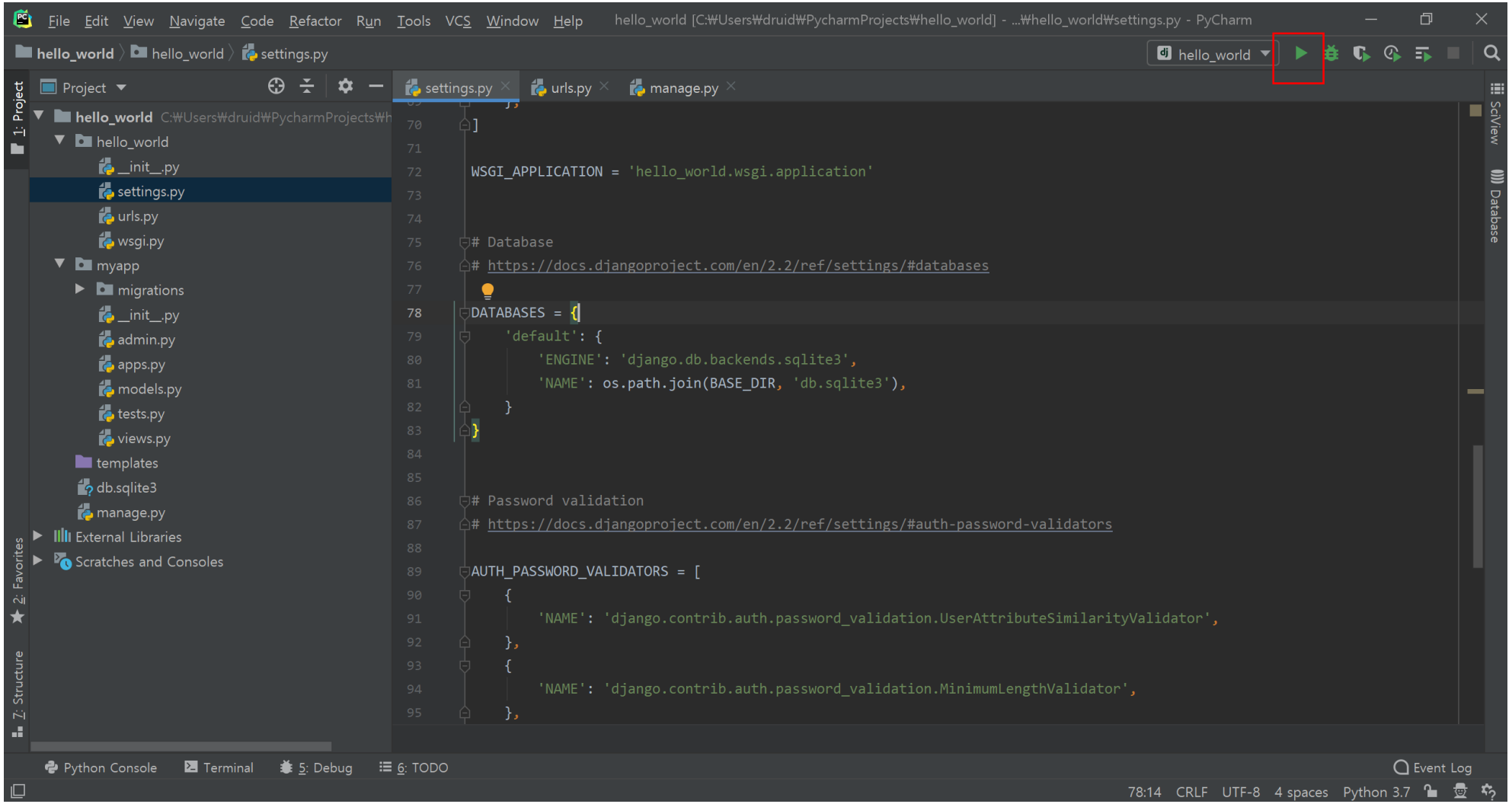
- 프로젝트 자체의 urls.py 를 수정해야함(settings.py 와 동일한 위치에 있는 urls.py)

- 해당 파일을 열고 아래와 같이 수정

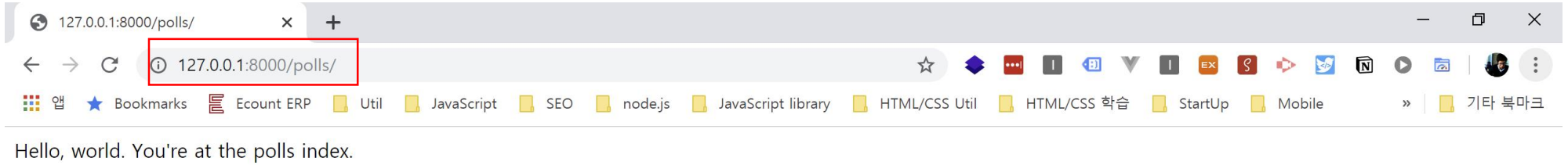
```
from django.contrib import admin
# include : 프로젝트 내에 속해 있는 application의 urls.py 파일을 그대로 가져와서 하위 경로로
사용하기 위한 모듈
from django.urls import include, path

urlpatterns = [
    # polls/ 하위 경로의 경우 polls application의 urls의 정의를 가져와서 사용함
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

Writing your first app(View) – 결과 확인



Writing your first app(View) – 결과 확인



Writing your first app(Model)

- settings.py 수정

- Model – Django ORM의 핵심. Django는 웹 서비스 구현을 위해 필요한 데이터 구조를 Model(Class) 형태로 정의하며 Model 사용을 위해서는 settings.py 의 DATABASES 값을 수정하여야 함.

- DATABASES 세팅
  - DATABASES[‘default’][‘ENGINE’] : 프로젝트에서 사용할 데이터베이스 엔진(DBMS)의 종류를 정의. 'django.db.backends.sqlite3', 'django.db.backends.postgresql', 'django.db.backends.mysql', 또는 'django.db.backends.oracle’
  - DATABASES[‘default’][‘NAME’] : 사용할 데이터베이스의 이름

Writing your first app(Model)

- Database Engine
  - Django는 기본적으로는 SQLite을 사용하도록 구성되어 있음.
  - 만약 데이터베이스를 처음 경험해보거나, Django에서 데이터베이스를 한번 경험해 보고 싶다면, SQLite가 가장 간단한 방법.
  - SQLite는 Python에서 기본으로 제공되기 때문에 별도로 설치할 필요가 없습니다.
  - SQLite는 데이터베이스를 단일 파일 형태로 저장하기 때문에, 별도의 DBMS 서버 설치가 필요 없으며 파일 하나에 모든 데이터가 저장됨.
  - 연습 프로젝트에서는 SQLite 사용, 추후 실제 프로젝트 진행 시에는 MySQL을 사용하여야함
- Database NAME
  - 사용할 데이터베이스의 이름
  - SQLite의 경우 데이터베이스가 저장될 파일 경로 지정
- Database USER, PASSWORD, HOST
  - SQLite 가 아닌 다른 Database Engine을 사용할 경우 설정이 필요한 값
  - HOST : DBMS 서버의 원격 HOST 주소
  - USER : DBMS 서버 접속을 위한 인증 정보(Username)
  - PASSWORD : DBMS 서버 접속을 위한 인증 정보(Password)

Writing your first app(Model)

- Database Engine
  - Django는 기본적으로는 SQLite을 사용하도록 구성되어 있음.
  - 만약 데이터베이스를 처음 경험해보거나, Django에서 데이터베이스를 한번 경험해 보고 싶다면, SQLite가 가장 간단한 방법.
  - SQLite는 Python에서 기본으로 제공되기 때문에 별도로 설치할 필요가 없습니다.
  - SQLite는 데이터베이스를 단일 파일 형태로 저장하기 때문에, 별도의 DBMS 서버 설치가 필요 없으며 파일 하나에 모든 데이터가 저장됨.
  - 연습 프로젝트에서는 SQLite 사용, 추후 실제 프로젝트 진행 시에는 MySQL을 사용하여야함
- Database NAME
  - 사용할 데이터베이스의 이름
  - SQLite의 경우 데이터베이스가 저장될 파일 경로 지정
- Database USER, PASSWORD, HOST
  - SQLite 가 아닌 다른 Database Engine을 사용할 경우 설정이 필요한 값
  - HOST : DBMS 서버의 원격 HOST 주소
  - USER : DBMS 서버 접속을 위한 인증 정보(Username)
  - PASSWORD : DBMS 서버 접속을 위한 인증 정보(Password)

Writing your first app(Model)

- settings.py > DATABASES : 새로운 프로젝트를 manage.py 명령어 혹은 PyCharm을 통해 프로젝트를 생성하였을 경우 SQLite를 사용하고 Project Root에 Database File을 생성하도록 기본 설정되어 있음

```
DATABASES = {
    'default': {
        # DB Engine으로 SQLite 사용
        'ENGINE': 'django.db.backends.sqlite3',
        # Project Root 에 'db.sqlite3'이라는 이름으로 데이터베이스 파일을 생성하도록 함
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Writing your first app(Model)

- settings.py > TIME_ZONE : Django 가 설치된 컴퓨터의 Time zone을 설정해야 함 (Database와 Python 시간 동기화를 위하여)

```
TIME_ZONE = 'Asia/Seoul'
```

Writing your first app(Model)

- settings.py > INSTALLED_APPS : 현재 Django 프로젝트에서 사용할 모든 Django Application의 목록. 사용자가 작성한 App 도 여기에 추가하여야 하며, 기본적으로는 settings.py 최초 생성 시 아래와 같은 Django Built-in app 이 포함되도록 되어 있다.
  - django.contrib.admin – Django가 제공하는 기본 관리자 페이지.
  - django.contrib.auth – Django가 제공하는 기본 인증 시스템.
  - django.contrib.contenttypes - 컨텐츠 타입을 위한 프레임워크.
  - django.contrib.sessions -- 세션 프레임워크.
  - django.contrib.messages -- 메세징 프레임워크.
  - django.contrib.staticfiles - 정적 파일(Static Resource)를 관리하는 프레임워크.

Writing your first app(Model)

- “python manage.py migrate” : 프로젝트에서 사용 중인 Application 중에서 Model을 사용하는 Application이 존재할 경우 사용 이전에 데이터베이스에 테이블 생성을 하여야 함. 기본 Django 앱들도 Model을 사용하므로 프로젝트를 생성하고 반드시 한번은 “python manage.py migrate”를 실행하여야 한다.

Writing your first app(Model)

- 설문조사 앱(Poll)을 위한 두개의 Model을 작성(설문조사를 추상화 한다면?)
  - Question : 설문조사 질문을 위한 Model – 질문(question)과 발행일(pub_date) 두개의 attribute를 가짐.
  - Choice : 설문조사의 응답 선택지를 위한 Model – 선택지(choice_text)와 응답갯수(votes) 두개의 attribute를 가지며, 각 Choice Model은 어떤 설문조사 질문에 대한 선택지인지 표시하기 위해 Question Model과 연관(Related)되어 있어야 한다.

```python
# 모든 Django Model은 django.db.models를 상속 받아야 한다.
from django.db import models

# 설문조사 질문을 기록 하기 위한 Model
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

# 설문조사 응답을 기록하기 위한 Model
class Choice(models.Model):
    # Django Model에서 두 Model 사이의 연관 관계를 표시하기 위한 방법
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Writing your first app(Model)

- Model class 를 정의하는 방법
  - Django model이 되기 위해서는 django.db.models 를 반드시 상속 받아야 함(from django.db import models)
  - Model attribute 선언 : [attribute 이름] = models.[attribute 유형](arguments)

- 주요 Attribute 유형
  - BooleanField : True/False 값 저장
  - CharField : 길이가 정해져 있는 문자열을 저장(max_length argument를 필수로 지정하여야 함)
  - TextField : CharField에 비해서 길이가 긴 문자열을 저장
  - IntegerField : -2147483648와 2147483647 사이의 정수 값을 저장
  - DateField : 날짜 값을 저장
  - DateTimeField : 날짜와 시간 값을 저장
  - Field Type References
  - Field Option References

Writing your first app(Model)

- poll앱의 Model 변경사항을 Database에 반영하기 위한 방법

- settings.py 의 INSTALLED_APPS에 poll 앱을 추가한다.(Django는 settings.py의 INSTALLED_APPS에 등록된 어플리케이션 만을 프로젝트에서 사용하는 어플리케이션으로 간주하고 Database에 반영한다.)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls.apps.PollsConfig',
]
```

Writing your first app(Model)

- “python manage.py makemigrations polls” : makemigrations 을 실행시킴으로서, 우리가 모델을 변경시킨 사실과(이 경우에는 새로운 모델을 만듦) 이 변경사항을 Database에 migration으로 저장시키고 싶다는 것을 Django에게 알려줌.

- “python manage.py migrate” : makemigrations 명령어를 통해 생성된 Database 변경점들을 실제로 DBMS에 반영함

- 즉, 사용자가 정의한 어플리케이션에서 Model을 추가하거나 변경한 경우 항상 다음의 순서를 통해 Database에 반영하여야 함.
  1. Application Model 추가/변경
  2. python manage.py makemigrations [app name]
  3. python manage.py migrate

Writing your first app(Model)

- 생성된 Model을 통해 실제로 데이터 저장하기
- “python manage.py shell” : Project의 Application들을 사용 할 수 있는 console을 시작하는 명령어

Writing your first app(Model)

```
>>> from polls.models import Choice, Question  # Import the model classes we just wrote.

# No questions are in the system yet.
>>> Question.objects.all()
<QuerySet []>

# Create a new Question.
# Support for time zones is enabled in the default settings file, so
# Django expects a datetime with tzinfo for pub_date. Use timezone.now()
# instead of datetime.datetime.now() and it will do the right thing.
>>> from django.utils import timezone
>>> q = Question(question_text="What's new?", pub_date=timezone.now())

# Save the object into the database. You have to call save() explicitly.
>>> q.save()

# Now it has an ID.
>>> q.id
1

# Access model field values via Python attributes.
>>> q.question_text
"What's new?"
>>> q.pub_date
datetime.datetime(2012, 2, 26, 13, 0, 0, 775217, tzinfo=<UTC>)

# Change values by changing the attributes, then calling save().
>>> q.question_text = "What's up?"
>>> q.save()

# objects.all() displays all the questions in the database.
>>> Question.objects.all()
<QuerySet [<Question: Question object (1)>]>
```

Summary

- Setup Django
- Start Django Project in PyCharm
- Django File Structure
- Django View
- Django Model

Study Topic

- Django Tutorial Topic 1(https://docs.djangoproject.com/ko/2.2/intro/tutorial01/)
- Django Tutorial Topic 2(https://docs.djangoproject.com/ko/2.2/intro/tutorial02/)
- Django Girls Tutorial(https://tutorial.djangogirls.org/ko/django/)