Django: Web(Backend) Framework

~~장고동아리 스팍스의 전공필수~~

by ashe(김재성)

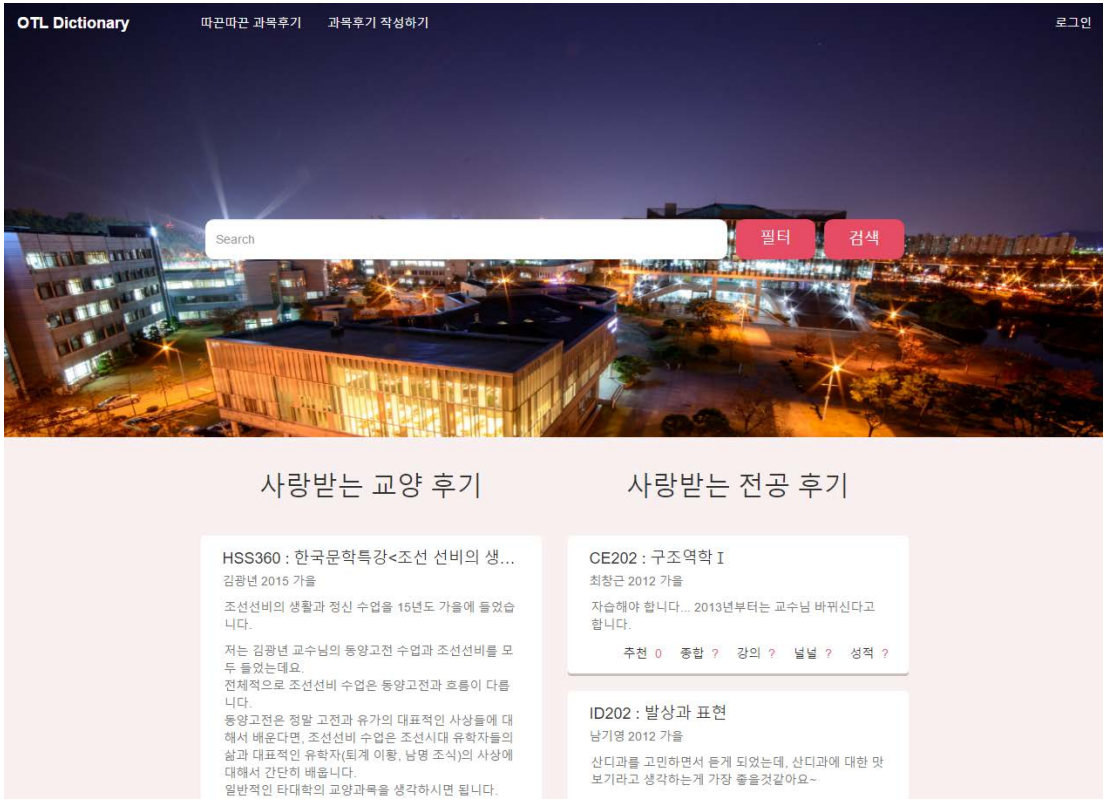
들어가기에 앞서

- 이번 신입생 교육은 (앞은 어떤지 모르겠지만) 기본적인 개념과 검색을 위한 안내 수준으로 진행됩니다. ~~꿀 빠는게 절대 아닙니다~~
- 제가 왜 세미나하는지 모를 정도로 지식이 부족하며, 이 세미나는 전문적인 검수를 1도 거치지 않았기 때문에 오류가 많을 수 있습니다. 질문과 지적은 언제든 환영입니다.
- 이번 장고 교육은 이번 세미나 1회로 끝입니다. 따라서 시간관계상 실습을 제외하였습니다.
- 당연하지만 세미나 후 매점 간식이 제공됩니다. ~~**저는 여러분의 양심과 인간됨을 신뢰함을 미리 밝힙니다**~~.

Backend? Web Framework?

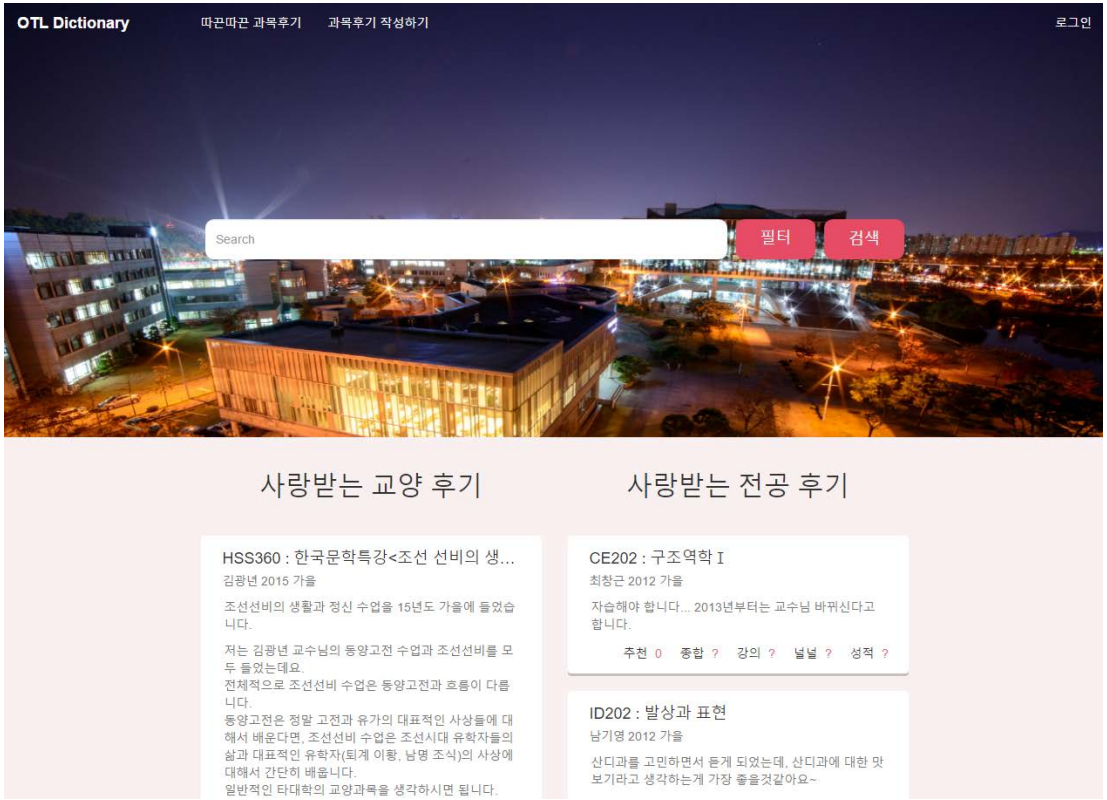
- Backend : **데이터**를 받아들여서 **조작**하고 아웃풋을 내놓는 부분을 코딩.
- Frontend : Backend에서 얻은 아웃풋을 **실제로 보여주는** 부분을 코딩
  → HTML + CSS + JavaScript
- (Server-side) Web Framework : 서버측 Backend 개발을 용이하게 해주기 위해 만들어진 틀.
  ex) Django, Flask, Node.js, JSP, Ruby On Rails
- (Client-side) Web Framework : 클라이언트측 Backend 개발을 용이하게 해주기 위해 만들어진 틀.
  ex) AngularJS, ReactJS, …
- 신입생 프로젝트에서는 어떤 프레임워크를 사용해도 무방하나 가르쳐주는(?)건

User Perspective - OTLPLUS 과목 검색



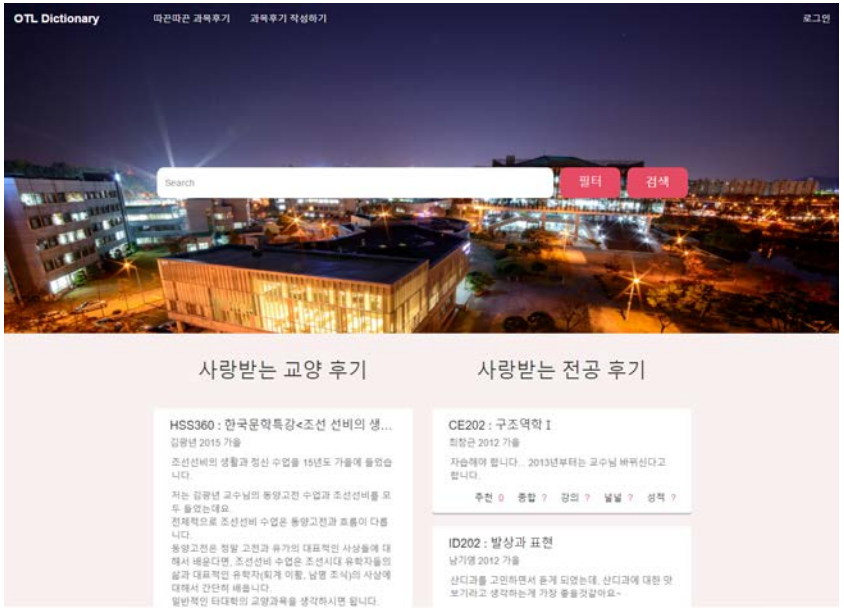
- 검색창에 과목 입력
- 검색 버튼 클릭
- **???**
- ~~PROFIT!~~ 해당하는 과목을 보여주는 페이지로 넘어감

Backend Perspective



- 검색창에 과목 입력
- 검색 버튼 클릭
- **정보를 서버로 전송, 데이터베이스를 뒤져서 해당하는 과목을 찾아냄**
- 해당하는 과목을 보여주는 페이지로 넘어감

Django Perspective



- url 입력, 그에 해당하는 페이지로 넘어감 : url → urls.py
- 검색창에 과목 입력을 입력 후 검색 클릭
- **정보를 서버로 전송, 데이터베이스를 뒤져서 해당하는 과목을 찾아냄 : input data → views.py → output data, url**
- 해당하는 과목을 보여주는 페이지로 접하고 보여줌 → output(html, data)

Pattern 1(httpredirect) : input(url, data) → urls.py → views.py → output(url, data)
Pattern 2(render) : input(url, data) → urls.py → views.py → output(html, data)

Virtual Environment

- 프로젝트 개발에 필요한 프로그램들을 서버에 모두 깐다?
  → 프로젝트1, 2, 3, …,n에 필요한 프로그램들이 서버에 모두 스까져 있다.
  → 서버가 날아가면 전부 날아가고 관리하기도 불편. 결정적으로 **버전 충돌!**

- 하나의 폴더에 개발에 필요한 프로그램(**requirements.txt**에 작성)을 모두 깔고 그놈을 독립시켜서 관리한다.

- $ virtualenv [가상환경이름] → 가상환경(폴더) 생성

- source [가상환경이름]/bin/activate → 가상환경 적용

- $ pip install -r [requirements.txt] → requirements.txt에 해당하는 프로그램들을 가상환경폴더 안에 설치 또는 $ pip install [파이썬 라이브러리 이름]

requirements.txt example

```
Django==1.9.7
Pillow==2.8.2
wheel==0.24.0
pyodbc==3.0.10
requests==2.9.1
MySQL-python==1.2.5
```

손으로 직접 작성하지말고 pip freeze라는 툴을 사용하여 자동으로 작성하도록 하는게 좋다.
https://pip.pypa.io/en/stable/reference/pip_freeze/

프로젝트 생성, 실행, 앱 생성, 앱을 프로젝트에 추가

- $ django-admin startproject [프로젝트명] → 프로젝트(폴더) 생성
- (이제부터 계속 프로젝트 폴더 내에서) $ python manage.py runserver 0.0.0.0:[포트번호]
  → localhost:포트번호 로 장고 프로젝트 실행. 브라우저로 접속 가능.
- $ python manage.py startapp [앱 이름] → 프로젝트폴더/apps 폴더 아래에 app에 해당하는 폴더 생성.
- $ vi [프로젝트이름]/settings.py에 다음 부분 작성.
  INSTALLED_APPS = (

  ...,
  '[앱이름]',
)

urls.py

```
from django.conf.urls import url
from django.http import HttpResponseRedirect
urlpatterns = [
 url(r'^delete/$','apps.review.views.ReviewDelete'),
 url(r'^like/$','apps.review.views.ReviewLike'),
 url(r'^refresh/$','apps.review.views.ReviewRefresh'),
 url(r'^portal/$','apps.review.views.ReviewPortal'),
 url(r'^dictionary/([^/]+)/$', 'apps.review.views.dictionary'),
 ]
```

search : django HttpResponseRedirect (example)

views.py

```
from django.shortcuts import render, redirect, render_to_response
from django.template import RequestContext
from apps.session.models import UserProfile
from apps.subject.models import Course, Lecture, Department, CourseFiltered, Professor
from apps.review.models import Comment,CommentVote, MajorBestComment, LiberalBestComment

def search_view_first_again(request):
    drugs = [
        "하하! 다시보니 반갑군요!",
        "튜토리얼의 협곡에 오신 것을 환영합니다.",
        "안뇽! 튜토리얼에 온 걸...화녕행!!!!",
        "안녕! 친구들! 튜토리얼이 왔어!",
        "튜토리얼이 진다..",
        "어서 와요! 꽤 보고싶었다구요?",
        "아이고, 이게 누구신가!",
            ]
    return render(request, 'review/tutorial-main-2.html', {'hello_message': drugs[random.randint(0, len(drugs)-1)],})
```

Template

- Frontend 소스파일 : html 파일들이 들어있는 곳.
- [프로젝트명]/apps/[앱이름]/template/에 위치 → 지정이 가능하다.
-
```
TEMPLATES = [
      {
            …
            ‘DIRS’: [
                  os.path.join(BASE_DIR, ‘templates’),
            ],
            …
      },
]
```

Introducing Jinja2: Django Template Language

- 기본목적 : views.py로부터 넘겨받은 데이터를 이용해 페이지를 보여준다.
- 페이지를 보여주는 것은 html로 작성. 넘겨받은 데이터를 활용하는 것은 Jinja2로 가능하다. {% %}, {{}} ← looks like Japanese Jinja!
- views의 아웃풋
  return render(request, 'html파일명', {'변수명' : '값', ...})
- 해당 html에서 불러오려면 {{변수명}}이라고 쓰면 된다.
- **Django Template Language와 Jinja2가 미묘하게 다를 수 있으므로 검색 시에는 django template ...으로 검색하기를 추천.**

Tags

for-loop:

```
{% for <var> in <list> %}
...
{% endfor %}
```

if-statement:

```
{% if <condition> %}
...
{% else %}
...
{% endif %}
```

search : django template tag,
https://docs.djangoproject.com/en/1.10/topics/templates/ → 도큐멘트는 항상 몸에 이롭습니다.

Models

- 웹 페이지의 수많은 데이터들 = 파이썬 오브젝트.
- 오브젝트의 청사진 = 클래스(스키마)를 정의해야 한다.
- 각 앱별 디렉터리 아래의 models.py에 작성.

models.py 작성

```python
# -*- coding: utf-8 -*-
from django.db import models
from apps.subject.models import Course, Lecture, Professor
from apps.session.models import UserProfile

class Comment(models.Model):
    course = models.ForeignKey(Course, db_index=True)
    lecture = models.ForeignKey(Lecture, db_index=True)

    comment = models.CharField(max_length=65536)
```

```python
@classmethod
def u_create(cls, course, lecture, comment, grade, load, speech, writer):
    professors = lecture.professor.all()
    related_list = [course]+[lecture]+list(professors)
    for related in related_list:
        related.grade_sum += grade*3
        related.load_sum += load*3
        related.speech_sum += speech*3
        related.comment_num += 1
        related.avg_update()
        related.save()
    new = cls(course=course, lecture=lecture, comment=comment, grade=grade, lo
    new.save()
    return new
```

- https://docs.djangoproject.com/en/1.10/topics/db/models/
- 필요한 것을 import하고, 클래스의 attribute와 메소드를 작성한다.
- 멤버변수는 models.CharField처럼 타입을 정의하거나, 이미 정의된 변수에 models.ForeignKey를 써서 연결시킬 수 있고, db_index, max_length

Model 적용(migration)

- DB 환경 세팅: [프로젝트명]/[프로젝트명]/settings.py
- sqlite3

```
# Database
# https://docs.djangoproject.com/en/1.8/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Model 적용(migration) ~~Pain in the ass~~

- 기본과정
  python manage.py check
  python manage.py makemigrations [앱 이름]
  python manage.py migrate

- migration이 꼬였을 경우에는 sql 파일을 까봐서 migration 로그를 삭제하는 등의 행위가 필요할 수도 있다. → 다른 문제도 많음. 삽질의 시작.

오브젝트 생성, 필터

- python manage.py shell → 파이썬 쉘에서 다양한 오브젝트 명령어를 시험해 볼 수 있다.

- 오브젝트 생성
  from apps.subject.models import Lecture
  l1 = Lecture(생성자에 적합한 인수들...)
  l1.save()

- 오브젝트 반환
  lecture_list = Lecture.objects.all() # 모든 오브젝트들의 리스트 리턴
  lecture_list_filtered = lecture_list.filter([attribute name] = [attribute value])
  # 필터에 해당하는 오브젝트들의 리스트 반환

오브젝트 변경, 삭제

- 오브젝트 변경
  lecture1 = lecture_list[0]
  lecture1.credit += 1
  lecture1.save()
- 오브젝트 삭제
  lecture1.delete()

Django Admin

- settings.py INSTALLED_APPS에 'django.contrib.admin' 추가
- urls.py urlpatterns에 url(r'^admin/', include(admin.site.urls')) 추가
- [프로젝트명]/[앱이름]/admin.py에 (다음은 otlplus 예시)
  from.subject.models import [클래스이름]

  class LectureAdmin(admin.ModelAdmin):
  list_display = ([attributes])
  list_filter = ([attributes for filtering])
  admin.site.register(Lecture, LectureAdmin)
- python manage.py createsuperuser로 슈퍼유저 만들고 런서버 후 admin url

Django Users

- 사용자를 관리하기 위해 User 모델을 사용한다.
- https://docs.djangoproject.com/en/1.10/topics/auth/
- 위 documentation 참조.