

Django vs Flask

Django vs Flask, 까봅시다

- 까다(동사): 껍질 따위를 벗기다.
 - “(속되게) 남의 결함을 들추어 비난하다.” 아닙니다..
- Django와 Flask는 파이썬으로 웹 서비스를 개발할 때 가장 보편적으로 쓰이는 도구
- 장고의 풍부하지만 자칫 과다할 수 있는
- 플라스크의 간결하지만 다소 부족한 것들

김도현

- 선린 인터넷 고등학교

김도현

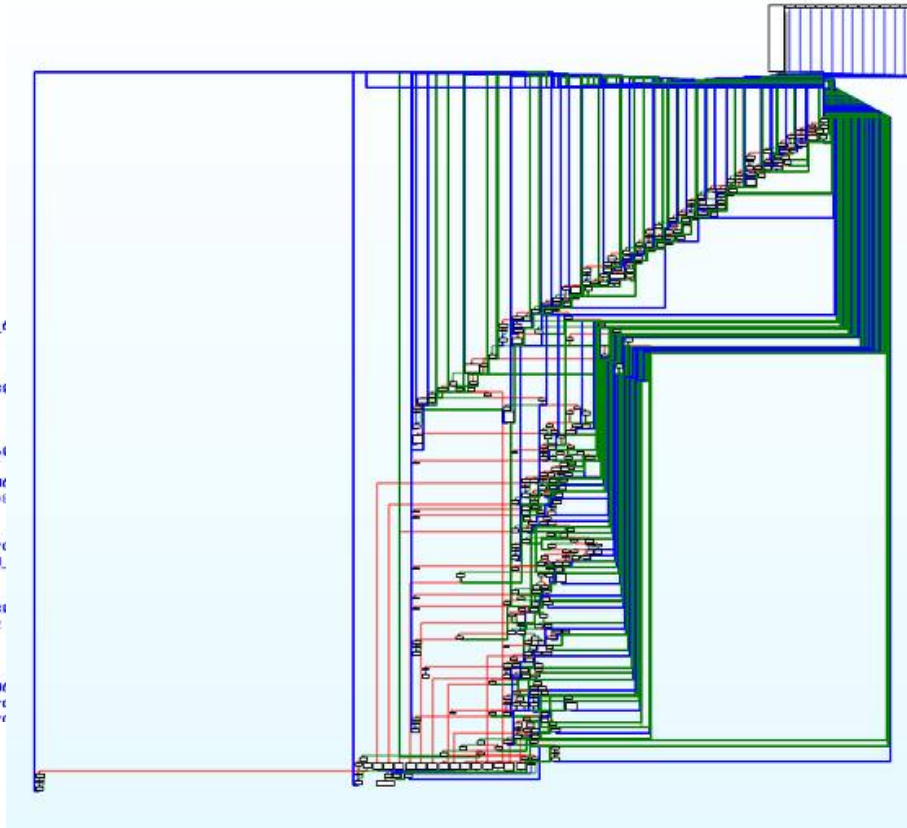
- 선린 인터넷 고등학교
- 해커에서 개발자로

```
v2 = a2;
v3 = sub_400670(a1, 3LL);
v4 = -(sub_400670(a1, 2LL)
 * ((~dword_603060[4 * v2] & 0x6EFBE940 | dword_603060[4 * v2] & 0x910416B2) ^ (~dword_603060[4 * v2 + 3] & 0x6EFBE94
 - v3));
v5 = sub_400670(a1, 2LL);
v6 = dword_603060[4 * a2 + 3];
v7 = v5 * (v6 & ~dword_603060[1 - -4 * v2] | dword_603060[1 - -4 * v2] & ~v6) + v4;
v8 = 1 - -4 * a2;
v9 = v7
 * sub_400670(a1, 2LL)
 * ((~dword_603060[4 * v2 + 2] & 0x43B37B1 | dword_603060[4 * v2 + 2] & 0xFBC4C84E) ^ (~dword_603060[4 * v2 + 3] & 0x43B3
 * a1)
 * ((~dword_603060[4 * v2] & 0x59C0DD4B | dword_603060[4 * v2] & 0xA63F22B4) ^ (~dword_603060[4 * v2 + 3] & 0x59C0DD4B |
 * ((~dword_603060[v8] & 0xBA0892CF | dword_603060[v8] & 0x45F76D30) ^ (~dword_603060[4 * v2 + 3] & 0xBA0892CF | dword_60
v10 = v9
 + a1
 * ((~dword_603060[2 - -4 * v2] & 0xEFD60ECS | dword_603060[2 - -4 * v2] & 0x1B29F137) ^ (~dword_603060[4 * v2 + 3] & 0x
 * ((~dword_603060[4 * v2 + 1] & 0x6F5BEFB1 | dword_603060[4 * v2 + 1] & 0x9BA4107E) ^ (~dword_603060[4 * v2 + 3] & 0x6F
v11 = v10
 + a1
 * ((~dword_603060[4 * v2] & 0x50C0D70D | dword_603060[4 * v2] & 0xAF3F28F2) ^ (~dword_603060[4 * v2 + 3] & 0x50C0D70D |
 * (dword_603060[4 * v2 + 3] & ~dword_603060[4 * v2 + 2] | dword_603060[4 * v2 + 2] & ~dword_603060[4 * v2 + 3]));
v12 = dword_603060[4 * a2 + 3];
v13 = ~dword_603060[4 * a2 + 3];
return v11
 + (v12 & ~dword_603060[4 * v2] | v13 & dword_603060[4 * v2])
 * ((~dword_603060[4 * v2 + 1] & 0x7E68346 | dword_603060[4 * v2 + 1] & 0x8C19FCB5) ^ (~dword_603060[4 * v2 + 3] & 0x7
 * ((~dword_603060[4 * v2 + 2] & 0xA623FFC0 | dword_603060[4 * v2 + 2] & 0x53DC1B3F) ^ (~v13 & 0xA623FFC0 | v12 & 0x53DC
```


김도현

- 선린 인터넷 고등학교
- 해커에서 개발자로

```
v2 = a2;  
v3 = sub_400670(a1, 3LL);  
v4 = -(sub_400670(a1, 2LL)  
    * ((~dword_603060[4 * v2] & 0x6EFBE940 | dword_603060[4 * v2]));  
v5 = sub_400670(a1, 2LL);  
v6 = dword_603060[4 * v2 + 3];  
v7 = v5 * (v6 & ~dword_603060[1 - -4 * v2] | dword_603060[1 - -4 * v2]);  
v8 = 1 - -4 * a2;  
v9 = v7  
    * sub_400670(a1, 2LL)  
    * ((~dword_603060[4 * v2 + 2] & 0x43B37B1 | dword_603060[4 * v2 + 2])  
    * ((~dword_603060[4 * v2] & 0x59C0DD4B | dword_603060[4 * v2])  
    * ((~dword_603060[v8] & 0xB08092CF | dword_603060[v8])  
v10 = v9  
    * a1  
    * ((~dword_603060[2 - -4 * v2] & 0xEFD60EC8 | dword_603060[2 - -4 * v2])  
    * ((~dword_603060[4 * v2 + 1] & 0x6F5BEFB1 | dword_603060[4 * v2 + 1])  
v11 = v10  
    * a1  
    * ((~dword_603060[4 * v2] & 0x50C0D7DD | dword_603060[4 * v2])  
    * (dword_603060[4 * v2 + 3] & ~dword_603060[4 * v2])  
v12 = dword_603060[4 * a2 + 3];  
v13 = ~dword_603060[4 * a2 + 3];  
return v11  
    * (v12 & ~dword_603060[4 * v2] | v13 & dword_603060[4 * v2])  
    * ((~dword_603060[4 * v2 + 1] & 0x72E6046 | dword_603060[4 * v2 + 1])  
    * ((~dword_603060[4 * v2 + 2] & 0x4C23FFC0 | dword_603060[4 * v2 + 2])
```



지난 1년 동안..

 Overview

 Repositories

 Public activity

 Edit profile

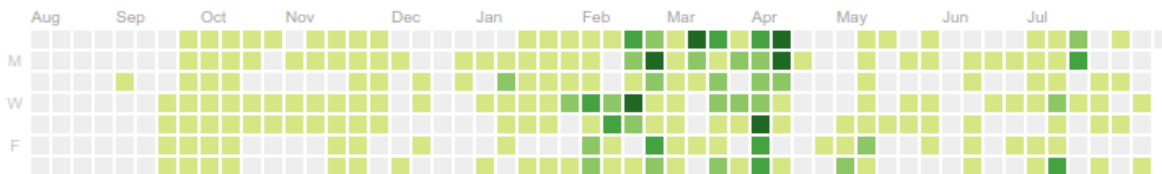
Pinned repositories

Customize your pinned repositories

≡	pennersr/django-allauth	2,512 ★
	Integrated set of Django applications addressing authentication, registration, account management as well as 3...	
≡	axelpale/minimal-django-file-upload-example	338 ★
	Source code for example at http://stackoverflow.com/questions/5871730/need-a-minimal-django-file-upload-ex...	
≡	pallets/flask	22,021 ★
	A microframework based on Werkzeug, Jinja2 and good intentions	
≡	nerogit.github.io	7 ★
	nero's blog	

1,648 contributions in the last year

Contribution settings ▾  



Summary of pull requests, issues opened, and commits. [Learn how we count contributions.](#)

Less    More

6 / 85

지난 1년 동안..

- Base64 Encode & Decode (Django)
- ELF File Format 시각화 (Flask)
- 해킹대회 운영 사이트 수정 (Django)
- MP3 관리 및 스트리밍 서비스 (Django)
- 파트라슈 (Python)
- github issue to trello
 - 터미널에서 모든거 다할수있게
 - Slack
 - Shell
- FB Messenger
- 단어 검색 기록 저장 (Django, Flask)
- 페이스북 게시물 저장 (Django, Flask)
- 위치정보를 통한 지각체크 앱 (Django, Flask)
- Django Fileupload, form example (Django)
- 상황별로 알맞은 음악 추천해주는 스트리밍 서비스 (Django)
- 광고관련 서비스 (Django)
- 후기 공유 사이트 (Django)
- 고졸 취업관련정보 크롤링 (Flask)
- konlpy를 이용한 회계 자동화 (jupyter)
- 슬라이드 쉐어 다운로더 (Flask)
- 교내 커뮤니티 사이트 (Django)
- Github to gist
- Smart Teen App Challenge 진행중 (Django)
- 카카오톡 시간별 대화량 분석 (SQLAlchemy)

웹이란 무엇일까요?

웹이란 무엇일까요?

인터넷에 연결된 컴퓨터들을 통해

정보를 공유할 수 있는

전 세계적인 정보 공간

Web Framework ?

Framework ?

Framework ?

“ 프로그래밍에서 특정 운영 체제를 위한 응용 프로그램 표준 구조를 구현하는 클래스와 라이브러리 모임 - wikipedia

Framework ?

‘ 프로그래밍에서 특정 운영 체제를 위한 응용 프로그램 표준 구조를 구현하는 클래스와 라이브러리 모임 - wikipedia

‘ 같은 작업을 반복하지 않도록 뭔가를 개발할 때 필수적으로 거쳐야 할 과정들을 우아하게 해결해 놓은 도구들의 모임, 혹은 구조

- Library ?

Library vs Framework

검색해 보시면 됩니다

Library vs Framework

검색해 보시면 됩니다

~~야크쉐이빙은 프로그래밍 할때만..~~

만약 프레임워크를 안쓰고 개발한다면?

- HTTP 프로토콜을 일일이 파싱한 다음에
- GET, POST에 따라 동작을 나누고
- HTML을 렌더링 하는 엔진을 만들어 내가 직접 html을 읽어 포맷스 트링에 맞추어 내가 원하는 변수를 html에 표시하고
- 웹서버를 거쳐 사용자에게 배포

DB는 ?

SQL로 한땀한땀 테이블 만들고

SQL로 한땀한땀 테이블 만들고
수정할 일 생기면 손수 ALTER 구문으로

SQL로 한땀한땀 테이블 만들고
수정할 일 생기면 손수 ALTER 구문으로
Join, Sub query, View, ...

SQL로 한땀한땀 테이블 만들고
수정할 일 생기면 손수 ALTER 구문으로
Join, Sub query, View, ...
혹시나 SQL Injection이라도 날아온다면..

똑같은 작업의 반복을 피하고



똑같은 작업의 반복을 피하고
해결하기 어려운 문제를 줄이며

똑같은 작업의 반복을 피하고
해결하기 어려운 문제를 줄이며
마감 시간 내에 주진 작업을 끝내기 위해

똑같은 작업의 반복을 피하고
해결하기 어려운 문제를 줄이며
마감 시간 내에 주진 작업을 끝내기 위해
내가 구현하기 벅찬 기능들을 쓰려고

똑같은 작업의 반복을 피하고
해결하기 어려운 문제를 줄이며
마감 시간 내에 주진 작업을 끝내기 위해
내가 구현하기 벅찬 기능들을 쓰려고
그래서 씁니다. **Django와 Flask**

Django vs Flask

	언제 (Initial Release)	슬로건	누가 쓰고있나
	April 1, 2010	The web framework for perfectionists with deadlines.	Disqus, Instagram, Pinterest, 요기요
	21 July 2005	Web development, One drop at a time	BeatPacking Company, StyleShare, 로켓펀치, 번개장터

Django vs Flask - 1

- Django ORM vs SQLAlchemy

Django vs Flask - 1

- Django ORM vs SQLAlchemy
- Form vs flask-wtf

Django vs Flask - 1

- Django ORM vs SQLAlchemy
- Form vs flask-wtf
- Template Engine vs Jinja2

Django vs Flask - 1

- Django ORM vs SQLAlchemy
- Form vs flask-wtf
- Template Engine vs Jinja2
- Admin Page vs flask-admin

Django vs Flask - 1

- Django ORM vs SQLAlchemy
- Form vs flask-wtf
- Template Engine vs Jinja2
- Admin Page vs flask-admin
- south(migrate, makemigrations) vs alembic

Django vs Flask - 2

- Middleware vs `before_request`, `after_request`

Django vs Flask - 2

- Middleware vs `before_request`, `after_request`
- `manage.py` vs `flask-scripts`

Django vs Flask - 2

- Middleware vs `before_request`, `after_request`
- `manage.py` vs `flask-scripts`
- `manage.py test` vs `unittest` or `flask-test`

Django vs Flask - 2

- Middleware vs `before_request`, `after_request`
- `manage.py` vs `flask-scripts`
- `manage.py test` vs `unittest` or `flask-test`
- `def view(request)` vs `flask.request`

Django vs Flask - 2

- Middleware vs `before_request`, `after_request`
- `manage.py` vs `flask-scripts`
- `manage.py test` vs `unittest` or `flask-test`
- `def view(request)` vs `flask.request`
- `urls.py` vs `flask.g.user`

Django vs Flask - 2

- `Middleware` vs `before_request`, `after_request`
- `manage.py` vs `flask-scripts`
- `manage.py test` vs `unittest` or `flask-test`
- `def view(request)` vs `flask.request`
- `urls.py` vs `flask.g.user`
- `django.contrib.messages` vs `flask.flash`

다른게 있긴 함?

1. 로그인, 회원가입
2. REST API
3. ORM - Many2Many
4. Django가 좋을때
5. Flask가 좋을때

로그인, 회원가입

- 기본 유저 모델과 pk
- 로그인 방식의 비교

Django 의 기본 유저 모델

```
class User(AbstractUser):  
    """  
    Users within the Django authentication system are represented by this  
    model.  
    Username, passwd and email are required. Other fields are optional.  
    """  
    class Meta(AbstractUser.Meta):  
        swappable = 'AUTH_USER_MODEL'
```

Django 의 기본 유저 모델

```
class AbstractUser(AbstractBaseUser, PermissionsMixin):
    username = models.CharField(...)
    first_name = models.CharField(_('first name'), max_length=30, blank=True)
    last_name = models.CharField(_('last name'), max_length=30, blank=True)
    email = models.EmailField(_('email address'), blank=True)
    is_staff = models.BooleanField(...)
    is_active = models.BooleanField(...)
    date_joined = models.DateTimeField(...)
    objects = UserManager()
    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['email']
    class Meta:
        ...
        abstract = True

    def get_full_name(self):
        full_name = '%s %s' % (self.first_name, self.last_name)
        return full_name.strip()

    def get_short_name(self):
        return self.first_name
```

Login - Django

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User

def login_view(request):
    if request.method == 'POST':
        data = request.POST
        user = authenticate(username=data['username'], passwd=data['passwd'])
        if user is not None:
            login(request, user)
            return redirect('/')
        else:
            context_data = {'message': '아이디 또는 비밀번호를 확인해주세요!'}
            return render(request, 'login.html', context_data)
    else:
        return render(request, 'login.html')
```

Login - Flask

```
@account_bp.route('/login', methods=['GET', 'POST'])
def login():
    passwd = request.form['passwd']
    userid = request.form['id']
    u = db.session.query(User).filter_by(
        userid=userid,
        passwd=passwd,
        active=True
    ).first()
    if u is not None:
        login_user(u)
        return redirect(url_for('foo.index'))
    else:
        return redirect(url_for('.login'))
```

ManyToMany - SQLAlchemy(Flask)

```
users_groups = db.Table('users_groups',
    db.Column("user_id", db.Integer, db.ForeignKey("User.id")),
    db.Column("group_id", db.Integer, db.ForeignKey("Group.id"))
)

class User(db.Model):
    __tablename__ = "User"
    id = db.Column(db.Integer, primary_key=True)
    group_id = db.Column(db.Integer, db.ForeignKey('Group.id'))
    groups = db.relationship('Group', secondary=users_groups, backref='users')

class Group
    __tablename__ = "Group"
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(30), default="plz set group name")
```

ManyToMany - Django

```
from django.db import models

class Group(models.Model):
    ...

class User(models.Model):
    group = models.ManyToManyField('Group', related_name='users')
```


get_object_or_404, get_or_create

```
def post_view(request, post_pk):  
    post = get_object_or_404(Post, pk=post_pk)  
    ...
```

```
local1, created = Local1.objects.get_or_create(name=local1)  
local2, created = Local2.objects.get_or_create(name=local2, parent_addr=local1)  
local3, created = Local3.objects.get_or_create(name=local3, parent_addr=local2)
```

jsonify, return "Hello"

```
@app.route('/')  
def index():  
    return "Hello!"  
  
@app.route('/hello')  
def hello():  
    from flask import jsonify  
    data['message'] = "Hello!"  
    return jsonify(data)
```

jsonify, return "Hello"

```
@app.route('/')  
def index():  
    return "Hello!"  
  
@app.route('/hello')  
def hello():  
    from flask import jsonify  
    data['message'] = "Hello!"  
    return jsonify(data)
```

```
def hello(requests):  
    data['message'] = "Hello!"  
    json_data = json.dumps(data)  
    content_type = "application/json"  
    return HttpResponse(json_data, content_type=content_type)
```

REST API

- 흔한 작업은 Django
- 뭔가 변동이 많고 다른 테이블을 많이 참조할 듯 싶으면 Flask

```
GET /api/tags/
```

```
HTTP 200 OK
```

```
Allow: GET, HEAD, OPTIONS
```

```
Content-Type: application/json
```

```
Vary: Accept
```

```
[
  {
    "url": "https://sunrin.io/api/tags/5/",
    "name": "android",
    "teams": [
      "https://sunrin.io/api/teams/1/",
      "https://sunrin.io/api/teams/2/",
      "https://sunrin.io/api/teams/3/"
    ],
    "users": []
  },
  {
    "url": "https://sunrin.io/api/tags/8/",
    "name": "APPJAM",
    "teams": [
```

코드에서 드러나지 않는 것들

1. 데이터를 입력, 관리하는 사람이 개발자가 아닌 경우

어디서 본건 많아서..

- 엑셀파일로 데이터 등록/수정할 수 있게 해주세요
- 관리자가 승인한 글만 올라오게 해주세요
- 간단한 CMS도 좀..

딱 여기까지면 그냥 django admin 쓰세요

딱 여기까지면 그냥 django admin 쓰세요

저거보다 더 많으면 django나 flask나..

딱 여기까지면 그냥 django admin 쓰세요

저거보다 더 많으면 django나 flask나..

admin custom은 한번 해보고

딱 여기까지면 그냥 django admin 쓰세요

저거보다 더 많으면 django나 flask나..

admin custom은 한번 해보고

아, 할게 못되는구나 한다음

딱 여기까지면 그냥 django admin 쓰세요

저거보다 더 많으면 django나 flask나..

admin custom은 한번 해보고

아, 할게 못되는구나 한다음

다음부터 상황에 맞게 결정하면 됨

2. 그래도 코드는 돈다

2. (왜 돌아가는지 몰라도) 그래도 코드는 돈다

main... main을 보자! - Django

```
import os
import sys
from django.conf.urls import url
from django.http import HttpResponse

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "__name__")

DEBUG = True

SECRET_KEY = 'foo'

ROOT_URLCONF = [
    url(r'^$', lambda x: HttpResponse('Hello World'), ),
]

if __name__ == "__main__":
    from django.core.management import execute_from_command_line

    execute_from_command_line(sys.argv)
```

- DjangoCon 2014- Anatomy of a Django Project

main... main을 보자! - Flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```


암시적이지만 많은 것들

VS

명시적이지만 부족한 것들

3. 자고 일어났더니 개발자가 바뀌어 있어요

교내 커뮤니티 개발 후기

- 마감까진 50여시간 정도 남음
- 구현할건 80% 남았는데 기간은 20% 남음
- 테스트 짤 시간이 없으니 사람을 믿어야 하는 상황
- ~~시연하다 터지면 내 상금은 너한테 받을꺼다~~

Flask프로젝트일 경우..

- 프로젝트 전반적인 구조 파악
- flask-sqlalchemy or just SQLAlchemy

4. 이게 최선입니까?

get_or_create 함수가 필요하다.

```
def get_or_create(session, model, defaults=None, **kwargs):
    instance = session.query(model).filter_by(**kwargs).first()
    if instance:
        return instance, False
    else:
        params = dict((k, v) for k, v in kwargs.iteritems() if not isinstance(v, ClauseElement))
        params.update(defaults or {})
        instance = model(**params)
        session.add(instance)
        return instance, True
```

```
def get_or_create(session, model, **kwargs):
    instance = session.query(model).filter_by(**kwargs).first()
    if instance:
        return instance
    else:
        instance = model(**kwargs)
        session.add(instance)
        session.commit()
        return instance
```

```
def get_one_or_create(session,
    model,
    create_method='',
    create_method_kwargs=None,
    **kwargs):
    try:
        return session.query(model).filter_by(**kwargs).one(), False
    except NoResultFound:
        kwargs.update(create_method_kwargs or {})
        created = getattr(model, create_method, model)(**kwargs)
        try:
            session.add(created)
            session.flush()
            return created, True
        except IntegrityError:
            session.rollback()
            return session.query(model).filter_by(**kwargs).one(), True
```

여러 생각들

- 처음 하시는 분이면 장난감부터 만들어봐야겠죠. Flask를 딱히 추천하는 건 아닌데 (애초에 Python 말고 다른 언어도 많죠) 풀스택 프레임워크는 나중에 해보시길 권합니다. 필요성을 느끼지 못하고 쓰는 도구는 제대로 활용할 수 없으니까요. - dahlia
- Flask 가 Micro Web Framework 이라고해서, 초보자들이 하기 쉽다는 뜻은 아니라고 생각합니다. 작다는 것이 쉽다는 뜻은 아니니까요. 자유도가 높은 반면, 내가 조립을 해야 된다는 것이, 처음 시작하는 분들에게는 어려움이 있을 수 있습니다. - ask Django

저는 Django로 웹개발을 처음 시작했습니다.

~~PHP도 하긴했지만 4년전에 한거라..~~

흔히 고민하는 문제

- 세션을 저장하는 장소
- REST ?
- ORM
- Migration Tools
- Project settings

일반적인 상황에서 가장 좋은 방법들!

- 세션을 저장하는 장소
- REST ?
- ORM
- Migration Tools
- Project settings

이런 것들을 처음부터 고민 하기엔
제가 많이 부족했습니다.

기존의 방법들을 학습해 나가며 **Django**

기존의 방법들을 학습해 나가며 **Django**

그 원리를 이해하고

기존의 방법들을 학습해 나가며 **Django**

그 원리를 이해하고

상황별로 더 나은 방법을 고민할 수 있을 때 **Flask**

마치며

더 많은 언어, 더 많은 도구를 다뤄
보신 분들은

더 좋은 통찰을 가지고 계실 것이라
생각합니다.

평생 한 언어, 혹은 한 프레임워크만
쓸 것이 아니라면

여러 방면에서 나에게 맞는 도구가
무엇인지

고민해보는 과정은 충분히 가치 있
다고 생각합니다.

프로그래밍에 있어 더 나은 방법을
고민하는 과정은

프로그래밍에 있어 더 나은 방법을
고민하는 과정은

끊임없는 비교

들어주셔서 감사합니다.

nero.union12@gmail.com
<http://facebook.com/dohyun26>