Ask Django

배포 준비 (공통)

Agenda

- 배포란?
- / 주소 처리 : redirect 또는 템플릿 처리
- 404/500 예외 템플릿
- static files 주의사항
- requirements.txt 분기

배포란?

- 우리가 만든 장고 프로젝트는 다양한 환경에서 구동됩니다.
  - 최소한 개발환경과 서비스 서버환경, 2가지.
- 개발환경 : 프로그램 성능은 낮더라도, 개발생산성에 집중
  - 소스코드 변경 시, 개발서버 자동 재시작
  - static/media 파일을 장고에서 직접 서빙 지원
  - 오류 발생 시, 오류정보 직접 노출
- 서비스 서버환경 : 프로그램 성능을 극대화
  - 소스코드가 변경되더라도, 개발서버 자동 재시작 X
  - static/media 파일은 장고에서 직접 서빙 X (외부 웹서버에 맡김)
  - 오류가 발생하더라도, 오류정보를 직접 노출 X

보통 서비스 환경에서 추가로 세팅하는 것들

- settings.DEBUG = False
  - 이 옵션을 켜두면, 메모리에 SQL쿼리 실행내역이 계속 쌓여요. 쌓이고 쌓이다가 메모리가 부족하게 되면, 서비스 불능 상태.
  - 이 옵션을 켜두면, 오류발생 시 유저에게 불필요하게 서버정보가 제공됩니다.
- 별도 static/media 세팅
  - AWS를 쓴다면, 거의 AWS S3 #ref 사용
- 별도의 데이터베이스 서버 세팅 (NOT SQLite3)
  - 보통 Django를 쓴다면 PostgreSQL, MySQL, MariaDB 추천
- 캐시 서버 세팅
  - memcached, redis

/ 주소를 아직 처리하지 않고 있다면 ?

선택1) 다른 경로로 redirect 시켜주세요.

```
# 프로젝트/views.py
from django.shortcuts import redirect

def root(request):
    return redirect('blog:index')  # 이동할 URL Pattern이나 URL을 지정

# 프로젝트/urls.py
from .views import root
# from django.views.generic import RedirectView

urlpatterns = [
    url(r'^$', root, name='root'),                                                  # 방법1) 커스텀 뷰 지정
    # url(r'^$', RedirectView.as_view(pattern_name='blog:index'), name='root'),   # 방법2) CBV 지정
]
```

/ 주소를 아직 처리하지 않고 있다면 ?

선택2) 템플릿을 만들어주세요.

```
# 프로젝트/templates/root.html 에 템플릿 생성

# 프로젝트/views.py
from django.shortcuts import render

def root(request):
    return render('root.html')

# 프로젝트/urls.py
from .views import root
# from django.views.generic import TemplateView

urlpatterns = [
    url(r'^$', root, name='root'),  # 방법1) 커스텀 뷰 지정
    # url(r'^$', TemplateView.as_view(template_name='root.html'), name='root'),  # 방법2) CBV 지정
]
```

404/400 예외 템플릿

404 예외

- django.views.defaults.page_not_found #src 뷰에서 처리
- settings.DEBUG=True 환경에서는 노란바탕의 예외정보가 노출
- 실서비스(settings.DEBUG=False)에서는 404.html 템플릿이 ...
  - 있을 경우 : 렌더링 (재미난 404 템플릿으로 채워보세요. #Funny404)
  - 없을 경우 : 아래 템플릿으로, 무미건조한 렌더링

```
<h1>Not Found</h1>
<p>The requested URL {{ request_path }} was not found on this server.</p>
```

500 예외

- django.views.defaults.server_error #src 뷰에서 처리
- settings.DEBUG=True 환경에서는 노란바탕의 예외정보가 노출
- 실서비스(settings.DEBUG=False)에서는 500.html 템플릿이 ...
  - 있을 경우 : 렌더링 (재미난 500 템플릿으로 채워보세요.)
  - 없을 경우 : 아래 템플릿으로, 무미건조한 렌더링

```
<h1>Server Error (500)</h1>
```

settings.STATICFILES_DIRS 설정 확인

- settings.STATICFILES_DIRS에 명시된 디렉토리는 필히 존재해야합니다.
  - 디렉토리가 없을 경우, python3 manage.py collectstatic 명령이 실패합니다. (OSError 예외 발생)
- WHY? : git에서는 빈 디렉토리는 버전관리 대상에 넣지 않습니다. 그러므로 로컬 개발환경에서는 존재했던 디렉토리라도, 빈 디렉토리일 경우 버전관리대상에서 제외되어, 배포서버 상에서는 해당 디렉토리가 존재하지 않게 됩니다.
  - 그러므로, STATICFILES_DIRS에 지정된 디렉토리에는 최소한 1개의 파일이라도 필히 넣어주세요.

```
# settings
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'askdjango', 'static'),
]
```

requirements.txt

의존성있는 라이브러리 관리

- 프로젝트 세팅할 때마다 일일이 설치하는 것은 너무 번거롭습니다.
- 라이브러리 수가 수십개가 될 수도 있습니다.
- 주의 : 아래 명령은 **예시일 뿐** 실제로 수행하지마세요.

```
# 개발용
쉘입력> pip3 install django django-debug-toolbar django-extensions django-bootstrap3 pilkit \
          django-imagekit django-polymorphic django-allauth django-taggit \
          django-taggit-templatetags pylibmc django-filter django-admin-honeypot \
          django-admin-tools feedparser lxml pyyaml pytz pillow wand qrcode django-bootstrap3 \
          django-crispy-forms django-widget-tweaks==1.4.1 jinja2 django-jinja requests \
          beautifulsoup4 nbconvert jupyter_client pdfrw pypdf2 reportlab django-ses

# 배포용
쉘입력> pip3 install django django-extensions django-bootstrap3 pilkit \
          django-imagekit django-polymorphic django-allauth django-taggit \
          django-taggit-templatetags pylibmc django-filter django-admin-honeypot \
          django-admin-tools feedparser lxml pyyaml pytz pillow wand qrcode django-bootstrap3 \
          django-crispy-forms django-widget-tweaks==1.4.1 jinja2 django-jinja requests \
          beautifulsoup4 nbconvert jupyter_client pdfrw pypdf2 reportlab django-ses \
          psycopg2 uwsgi
```

requirements.txt

- pip 유틸리티에서는 설치할 팩키지 목록을 파일 1개를 통해 공급받을 수 있습니다.
- requirements.txt 예시

```
django>=1.10.6
django-debug-toolbar
django-extensions
django-bootstrap3
```

쉘입력> `pip install -r requirements.txt`

- 이를 위해 일반적으로 쓰이는 파일명이 **requirements.txt** 입니다.
  - 다른 파일명/파일경로이어도 실행에는 지장이 없습니다.

실행환경에 따라 다양한 requirements.txt가 필요

- 실행환경 별로 필요한 라이브러리가 다를 수 있습니다.
- 이때, 별도 개발문서를 만들어서 실행환경 별로 필요한 라이브러리를 나열하지 않습니다.
- 프로젝트 내에 다양한 버전의 requirements.txt를 만듭니다.
  - 개발용
  - 서비스 2.0 개발용
  - 배포용 (일반)
  - 배포용 (AWS)
  - 배포용 (Heroku)

프로젝트 내 requirements.txt 생성 예시

- requirements.txt : 대개 배포용(reqs/prod.txt)을 참조
- reqs 디렉토리
  - common.txt : 공통 라이브러리
  - dev.txt : 개발용 라이브러리
  - dev_2.0.txt : 서비스 2.0용 개발용 라이브러리
  - prod.txt : 배포용 라이브러리
  - prod_aws.txt : AWS 배포용 (AWS에서만 필요한 라이브러리가 있을 경우)
  - prod_heroku.txt : Heroku 배포용 (AWS에서만 필요한 라이브러리가 있을 경우)

- 프로젝트/requirements.txt : reqs/prod.txt을 참조

```
-r reqs/prod.txt
```

- 프로젝트/reqs/common.txt : 공통 라이브러리는 common.txt에 두어, 중복지정을 피합니다.

```
django>=1.10.4
django-extensions
django-allauth
django-bootstrap3
pillow
pilkit
django-imagekit
```

- 프로젝트/reqs/dev.txt : 개발용 라이브러리

```
-r common.txt
django-debug-toolbar
```

- 프로젝트/reqs/prod.txt : 배포용 라이브러리

```
-r common.txt
psycopg2
uwsgi
raven
```

라이브러리 설치할 때

```
# 먼저, 해당 프로젝트 디렉토리로 이동한 후에 ...
쉘입력> cd 프로젝트경로

# 개발 셋업할 때
쉘입력> pip3 install -r reqs/dev.txt

# 서비스 2.0 개발 셋업할 때
쉘입력> pip3 install -r reqs/dev_2.0.txt

# 배포 셋업할 때
쉘입력> pip3 install -r reqs/prod.txt

# AWS에 배포 셋업할 때
쉘입력> pip3 install -r reqs/prod_aws.txt

# Heroku에 배포 셋업할 때
쉘입력> pip3 install -r reqs/prod_heroku.txt
```

장고 소스코드 공통 배포셋업

1차 완료.