Lab5. Django Project Automatic Deployment with Jenkins I

1. Jenkins Repository의 [Lab1.Django 설치하기.pdf]
2. Jenkins Repository의 [Lab2.Django Project 시작하기.pdf]의 4번까지 실행

3. requirements 파일 생성
    1)프로젝트와 동일한 pip와 pip를 이용한 설치 및 패키지의 버전을 맞추기 위해 requirements.txt 파일을 생성한다.
    2)최상위 디렉토리(manage.py 파일이 있는 디렉토리)로 이동하여 현재 가상환경에 설치된 pip 목록을 requrements.txt에 저장

```
$ pip freeze > requirements.txt
```

    3)requirements.txt 목록 확인

```
$ cat requirements.txt
asgiref==3.6.0
Django==4.2.1
sqlparse==0.4.4
tzdata==2023.3
```

4. Local Repository 생성
    -상위 디렉토리로 이동

```
$ cd ..
$ pwd
    ~/DjangoHome

$ ls
    --Windows의 경우
        Include          Lib          myproject          Scripts          pyvenv.cfg
    --macOS의 경우
        bin              Lib          myproject          pyvenv.cfg
```

    -git 초기화

```
$ git init
    -Windows의 경우
        Initialized empty Git repository in D:/DjangoHome/.git/
    -macOS의 경우
        Initialized empty Git repository in /Users/{계정}/DjangoHome/.git/
```

    -git 필요 환경 설정

```
$ git config user.name 'henry'
$ git config user.email 'javaexpert@nate.com'
```

5. .gitignore 파일 생성
    1)Github에 push해야 하기 때문에 gitignore 파일이 필요하다.
    2)아래의 사이트에서 생성한다.
        https://www.gitignore.io/

    3)Django, venv, Python 입력 후 [생성] 버튼 클릭

```
# Created by https://www.toptal.com/developers/gitignore/api/django,venv,python
# Edit at https://www.toptal.com/developers/gitignore?templates=django,venv,python

### Django ###
*.log
*.pot
*.pyc
__pycache__/
local_settings.py
db.sqlite3
db.sqlite3-journal
media

# If your build process includes running collectstatic, then you probably don't need or want to include staticfiles/
# in your Git repository. Update and uncomment the following line accordingly.
# <django-project-name>/staticfiles/

### Django.Python Stack ###
# Byte-compiled / optimized / DLL files
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
```

```
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo

# Django stuff:

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.
#Pipfile.lock

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include poetry.lock in version control.
#   This is especially recommended for binary packages to ensure reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control
#poetry.lock

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
#pdm.lock
#   pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide
.pdm.toml

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that can
#  be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more nuclear
#  option (not recommended) you can uncomment the following to ignore the entire idea folder.
#.idea/

### Python ###
# Byte-compiled / optimized / DLL files

# C extensions

# Distribution / packaging

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.

# Installer logs

# Unit test / coverage reports

# Translations

# Django stuff:

# Flask stuff:

# Scrapy stuff:

# Sphinx documentation

# PyBuilder

# Jupyter Notebook

# IPython

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include poetry.lock in version control.
#   This is especially recommended for binary packages to ensure reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
#   pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm

# Celery stuff

# SageMath parsed files

# Environments

# Spyder project settings

# Rope project settings

# mkdocs documentation

# mypy

# Pyre type checker

# pytype static type analyzer

# Cython debug symbols

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that can
#  be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more nuclear
#  option (not recommended) you can uncomment the following to ignore the entire idea folder.

### Python Patch ###
# Poetry local configuration file - https://python-poetry.org/docs/configuration/#local-configuration
poetry.toml

# ruff
.ruff_cache/

# LSP config files
pyrightconfig.json

### venv ###
# Virtualenv
# http://iamzed.com/2009/05/07/a-primer-on-virtualenv/
[Bb]in
[Ii]nclude
[Ll]ib
[Ll]ib64
[Ll]ocal
[Ss]cripts
pyvenv.cfg
pip-selfcheck.json

# End of https://www.toptal.com/developers/gitignore/api/django,venv,python
```

4)생성한 gitignore를 .gitignore 파일에 저장

```
$ nano .gitignore
```

5)Commit 및 Push 하기

-현재 디렉토리 확인

```
    $ pwd
    ~/DjangoHome

    $ ls
```

```
                ※.git 디렉토리는 보이지 않는 것이 정상
                --Window의 경우
                    Include     Lib        myproject         Scripts          .gitignore          pyvenv.cfg
                --macOS의 경우
                    bin               Lib           myproject            .gitignore         pyvenv.cfg

        -git add
            $ git add .

        -git commit
            $ git commit -m 'Django Project Create'

        -Github에서 DjangoHome Repository 생성
            --README.md 생성 하지 않음.
            --Public

        -Github에 remote 설정하고 push 하기
            $ git remote add origin https://github.com/gitinstructor/DjangoHome.git
            $ git branch -M main
            $ git push -u origin main
            Enumerating objects: 17, done.
            Counting objects: 100% (17/17), done.
            Delta compression using up to 8 threads
            Compressing objects: 100% (16/16), done.
            Writing objects: 100% (17/17), 5.25 KiB | 2.62 MiB/s, done.
            Total 17 (delta 3), reused 0 (delta 0), pack-reused 0
            remote: Resolving deltas: 100% (3/3), done.
            To https://github.com/gitinstructor/DjangoHome.git
             * [new branch]      main -> main
            branch 'main' set up to track 'origin/main'.

    6)Github에서 확인
        https://github.com/{Github계정}/DjangoHome
            -myproject
            -.gitignore


6. EC2 Instance에 Tabby로 접속하기
7. Github Repository Clone 및 몇 가지 설정하기
    1)Github의 DjangoHome Repository Clone 하기
        $ git clone https://github.com/gitinstructor/DjangoHome.git
        Cloning into 'DjangoHome'...
        Username for 'https://github.com': gitinstructor
        Password for 'https://gitinstructor@github.com':     <----Personal Access Token
        remote: Enumerating objects: 12, done.
        remote: Counting objects: 100% (12/12), done.
        remote: Compressing objects: 100% (10/10), done.
        remote: Total 12 (delta 1), reused 12 (delta 1), pack-reused 0
        Receiving objects: 100% (12/12), 4.83 KiB | 706.00 KiB/s, done.
        Resolving deltas: 100% (1/1), done.

    2)몇 가지 설정하기
        -날짜 시간 설정
            $ sudo timedatectl set-timezone 'Asia/Seoul'

        -Ubuntu Update
            $ sudo apt-get update
            $ sudo apt-get upgrade -y
            $ sudo apt-get dist-upgrade

        -Nginx 설치
            $ sudo apt install -y nginx

        -pip upgrade
            $ python --version
            $ pip list
            $ python -m pip install --upgrade pip

         -Green unicorn package 설치
            $ pip instasll gunicorn
            Defaulting to user installation because normal site-packages is not writeable
            Collecting gunicorn
              Downloading gunicorn-20.1.0-py3-none-any.whl (79 kB)
                 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.5/79.5 kB 2.5 MB/s eta 0:00:00
            Requirement already satisfied: setuptools>=3.0 in /usr/lib/python3/dist-packages (from gunicorn) (59.6.0)
            Installing collected packages: gunicorn
            Successfully installed gunicorn-20.1.0

    3)가상환경 생성 후 해당 프로젝트로 이동
        -가상환경 생성
            $ virtualenv DjangoHome
            created virtual environment CPython3.10.6.final.0-64 in 501ms
              creator CPython3Posix(dest=/home/ubuntu/DjangoHome, clear=False, no_vcs_ignore=False, global=False)
              seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy,
              app_data_dir=/home/ubuntu/.local/share/virtualenv)
                added seed packages: pip==23.1.2, setuptools==67.6.1, wheel==0.40.0
              activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
```

-가상환경 활성화하기

```
    $ source ./DjangoHome/bin/activate
```

-해당 디렉토리로 이동하기

```
    $ cd DjangoHome
```

4)git 설정

```
$ ls -al
```

※.git은 보이지 않을 수 있음.

```
bin        lib        myproject          pyvenv.cfg          .git           .gitignore

$ git config user.name 'henry'
$ git config user.email 'javaexpert@nate.com'
```

5)필요 패키지 설치

```
$ cd myproject

$ ls
manage.py           myproject            requirements.txt

$ pip install -r requirements.txt
Collecting asgiref==3.6.0 (from -r requirements.txt (line 1))
  Using cached asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting Django==4.2.1 (from -r requirements.txt (line 2))
  Using cached Django-4.2.1-py3-none-any.whl (8.0 MB)
Collecting sqlparse==0.4.4 (from -r requirements.txt (line 3))
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting tzdata==2023.3 (from -r requirements.txt (line 4))
  Using cached tzdata-2023.3-py2.py3-none-any.whl (341 kB)
Installing collected packages: tzdata, sqlparse, asgiref, Django
Successfully installed Django-4.2.1 asgiref-3.6.0 sqlparse-0.4.4 tzdata-2023.3
```

6)Django Project 시작하기

```
$ python manage.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 17, 2023 - 05:07:24
Django version 4.2.1, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

7)Web Browser로 확인하기

http://ec2-43-201-250-26.ap-northeast-2.compute.amazonaws.com:8000

8)기본테이블 생성하기

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

9)기본테이블 확인하기

```
$ ls
db.sqlite3          manage.py          myproject          requirements.txt
```

10)관리자 계정 생성

```
$ python manage.py createsuperuser
Username (leave blank to user 'ubuntu') : admin
Email address: javaexpert@nate.com
Password : ********
Password (again) : ********
This password is too common.
Bypass password validation and create user anyway? [y/N] : y
Superuser created successfully.
```

8. 관리자모드 접속하기

-웹브라우저에서 확인

```
$ python manage.py runserver 0:8000

    http://{EC2 퍼블릭 IPv4 DNS}}:8000/admin
    Username: admin
    Password :
```

9. Django Project에 main App 생성하기

1)manage.py가 있는 최상위 디렉토리에서 다음과 같이 main App을 생성한다.

```
$ python manage.py startapp main
```

2)myproject > settings.py 수정하기

```
...
INSTALLED_APPS = [
    'main',       <----여기
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

...
LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'     <---TimeZone 설정
...
```

3)index.html 파일 생성하기

-새로 생성된 main 디렉토리로 이동

```
$ cd ..
$ cd main
```

-templates 디렉토리 생성 후 이동

```
$ mkdir templates
$ cd templates
```

-main 디렉토리 생성 후 이동

```
$ mkdir main
$ cd main
```

-index.html 파일 생성

```
$ nano index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to My Django Project</title>
</head>
<body>
    <h1>Hello Django Project</h1>
</body>
</html>
```

4)main > views.py 수정하기

```
$ nano views.py
    from django.shortcuts import render

    # Create your views here.
    def index(request):
        return render(request, "main/index.html")
```

5)myproject > urls.py 수정하기

```
$ nano urls.py
"""
URL configuration for myproject project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from main import views as main_views           <---여기 수정

urlpatterns = [
    path('', main_views.index, name='index'),   <---여기 수정
    path('admin/', admin.site.urls),
]
```

6)기존 Model들을 migrate하고, Django Project 시작하기

```
    $ python manage.py makemigrations && python manage.py migrate
    No changes detected
    Operations to perform:
      Apply all migrations: admin, auth, contenttypes, sessions
    Running migrations:
      No migrations to apply.

    $ python manage.py runserver 0:8000
```

7)Web Browser를 통해 확인하기

```
    http://{EC2 퍼블릭 IPv4 DNS}}:8000
```

10. 변경사항 Github에 push

```
$ git status
    On branch main
    Your branch is up to date with 'origin/main'.

    Changes not staged for commit:
      (use "git add <file>..." to update what will be committed)
      (use "git restore <file>..." to discard changes in working directory)
            modified:   myproject/myproject/settings.py
            modified:   myproject/myproject/urls.py

    Untracked files:
      (use "git add <file>..." to include in what will be committed)
            myproject/main/

    no changes added to commit (use "git add" and/or "git commit -a")

$ git add .
$ git status
    On branch main
    Your branch is up to date with 'origin/main'.

    Changes to be committed:
      (use "git restore --staged <file>..." to unstage)
            new file:   myproject/main/__init__.py
            new file:   myproject/main/admin.py
            new file:   myproject/main/apps.py
            new file:   myproject/main/migrations/__init__.py
            new file:   myproject/main/models.py
            new file:   myproject/main/templates/main/index.html
            new file:   myproject/main/tests.py
            new file:   myproject/main/views.py
            modified:   myproject/myproject/settings.py
            modified:   myproject/myproject/urls.py

$ git commit -m 'Add main App to Django Project'
    [main f11b7f7] Add main App to Django Project
     10 files changed, 37 insertions(+), 2 deletions(-)
     create mode 100644 myproject/main/__init__.py
     create mode 100644 myproject/main/admin.py
     create mode 100644 myproject/main/apps.py
     create mode 100644 myproject/main/migrations/__init__.py
     create mode 100644 myproject/main/models.py
     create mode 100644 myproject/main/templates/main/index.html
     create mode 100644 myproject/main/tests.py
     create mode 100644 myproject/main/views.py

$ git push origin main
    Username for 'https://github.com': gitinstructor
    Password for 'https://gitinstructor@github.com':
```

```
    Enumerating objects: 21, done.
    Counting objects: 100% (21/21), done.
    Compressing objects: 100% (13/13), done.
    Writing objects: 100% (16/16), 1.82 KiB | 930.00 KiB/s, done.
    Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
    remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
    To https://github.com/gitinstructor/DjangoHome.git
       cf69166..f11b7f7  main -> main
```

11. uWSGI 서버 연결하기

  1)Django repository의 [Lab.uWSGI 서버 연결하기.pdf]를 참조

  2)성공하면 Github에 Push한다.

```
    $ git status
        On branch main
        Your branch is up to date with 'origin/main'.

        Untracked files:
          (use "git add <file>..." to include in what will be committed)
                myproject/.config/

        nothing added to commit but untracked files present (use "git add" to track)

    $ git add .

    $ git status
        On branch main
        Your branch is up to date with 'origin/main'.

        Changes to be committed:
          (use "git restore --staged <file>..." to unstage)
                new file:   myproject/.config/uwsgi/myproject.ini

    $ git commit -m "Add myproject.ini File"
        [main 6d34994] Add myproject.ini File
         1 file changed, 16 insertions(+)
         create mode 100644 myproject/.config/uwsgi/myproject.ini

    $ git push origin main
        Username for 'https://github.com': gitinstructor
        Password for 'https://gitinstructor@github.com':
        Enumerating objects: 8, done.
        Counting objects: 100% (8/8), done.
        Compressing objects: 100% (4/4), done.
        Writing objects: 100% (6/6), 646 bytes | 646.00 KiB/s, done.
        Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
        remote: Resolving deltas: 100% (1/1), completed with 1 local object.
        To https://github.com/gitinstructor/DjangoHome.git
           f11b7f7..6d34994  main -> main
```

12. nginx 연결하기

  1)Django repository의 [Lab.nginx 연결하기.pdf]를 참조