Git Branch - Team

1. 팀장

1. 프로젝트 생성

- 레포지터리 생성 - README 포함
- 레포 이름 : git_flow_practice
- 프라이빗으로 생성 후 클론

2. 멤버 초대

- 팀원 , 강사님 초대
- 메인테이너 권한

3. 깃이그노어 생성

- gitignore.io에서 이그노어 생성 후 복사 붙여넣기
- python, vue, vuejs, django, vscode, windows, mac 등

4. 파이썬 가상환경 생성 후 장고 설치

-
```
$ python -m venv venv
$ source venv/Scripts/activate
$ pip install django
```

5. requirements.txt 생성

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

6. 브랜치 생성 후 생성한 브랜치로 변경

```
$ git branch dev
$ git switch dev
```

7. git add & commit

```
$ git add .
$ git commit -m "upload"
```

8. 장고 프로젝트/앱 생성

```
$ django-admin startproject pjt .
$ py manage.py startapp articles
```

9. 프로젝트에 앱 등록

```
# settings.py

INSTALLED_APPS = [
    'articles',
    'django.contrib.admin',
    ...,
]
```

10. 모델 생성 후 마이그레이션

```
# articles/models.py
from django.db import models
from django.conf import settings

class Article(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    title = models.TextField()
```

```
$ py manage.py makemigrations
$ py manage.py migrate
```

11. 더미데이터 생성

1. 익스텐션 설치 후 requirements.txt 갱신

```
$ pip install django_extensions ipython
$ pip freeze > requirements.txt
```

2. 설치한 익스텐션 등록

```
# settings.py

INSTALLED_APPS = [
    'articles',
    'django_extensions',
    'django.contrib.admin',
    ...,
]
```

3. 아티클 생성

```
$ article = Article()
$ article.title="test"
$ article.save()
```

4. 생성한 더미데이터 내보내기

```
$ python manage.py dumpdata --indent=4 articles > articles.json
```

5. 생성한 더미데이터 JSON파일 fixtures 폴더로 이동

```
$ mkdir articles/fixtures
$ mv articles.json articles/fixtures/
```

12. 지금까지 한 작업 올리기

1. git add & commit

```
$ git add .
$ git commit -m "dev"
```

2. git push

```
git push -u origin dev
```

2. 팀원

1. 클론 받아오기

- master Branch를 받아오기 때문에 비어있음

2. 브랜치 생성 후 dev 브랜치 pull 받아오기

```
$ git branch dev
$ git switch dev
$ git pull origin dev
```

- 원격 저장소에서 dev 내용 local로 pull

3. accounts 브랜치로 이동

```
$ git branch accounts
$ git switch accounts
```

4. 가상환경 생성 후 requirements 설치

```
$ py -m venv venv
$ source venv/Scripts/activate
$ pip install -r requirements.txt
```

5. accounts 앱 생성 후 등록

```
$ py manage.py startapp accounts
```

```
# settings.py

INSTALLED_APPS = [
    'articles',
    'accounts',
    'django_extensions',
    'django.contrib.admin',
    ...,
]
```

6. 커스텀 User 모델 생성 후 마이그레이션

```
# accounts/models.py

from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    pass
```

```
$ py manage.py makemigrations
$ py manage.py migrate
```

7. 슈퍼유저 생성으로 User 모델 테스트

```
$ py manage.py createsuperuser
```

8. User 덤프 데이터 생성 후 폴더 이동

```
$ python manage.py dumpdata --indent=4 accounts > users.json
```

9. 지금까지 한 작업 올리기

```
$ git add .
$ git commit -m "accounts"
$ git push origin accounts
```

10. accounts 브랜치 내용 dev 브랜치에 Merge 하기

- Merge 시 dev로 병합하는 것으로 설정되어 있는지 확인

3. 팀장

1. settings 브랜치 생성 후 이동

```
$ git branch settings
$ git switch settings
```

2. BASE 템플릿 생성 후 사용 설정

```
TEMPLATES = [
    {
        ...,
        'DIRS': [BASE_DIR/'templates'],
        ...,
    }
]
```

3. base 템플릿 생성 커밋

```
$ git add .
$ git commit -m "base template"
```

4. 완료된 accounts 작업 가져오기

```
$ git pull origin dev
```

5. settings branch git push

```
$ git push origin settings
```

6. settings 브랜치 내용 dev 브랜치에 Merge 하기

7. dev 브랜치로 변경

```
$ git switch dev
```

8. Merge 변경 내용 pull 받기

```
$ git pull origin dev
```

9. articles 브랜치 생성/이동 후 articles 모델 작성

```
$ git branch articles
$ git switch articles
```

10. 작성한 모델 마이그레이션

- 에러 발생 시 db 제거 후 다시 생성 (더미 데이터이기 때문)
- 옵션 선택 에서 1(자동생성), 1(기본값) 선택

11. 더미데이터 로드

- articles.json만 로드 시 에러 발생
  - 참조 해야할 user가 DB에 없기 때문
- users.json 먼저 로드 한 후 articles.json 생성 (관계 순서 생각)
- 또는 한번에 모두 다 생성

- 또는 articles의 dumpdata 다시 생성
  - shell_plus로 새로운 객체 생성 후
  - dumpdata 명령어로 articles.json 생성
  - 원래 위치에 덮어쓰기

12. 완료한 작업 깃에 올리고 Dev branch에 Merge

```
$ git add .
$ git commit -m "user article 1:N"
$ git push origin articles
```

4. 팀원

1. 수정된 dev Branch 받아오기

```
$ git pull origin dev
```

2. db(더미데이터) 삭제 후 다시 마이그래이션으로 생성

```
$ py manage.py migrate
$ py manage.py loaddata users.json articles.json
```

conflict 난 상황

- git pull 해서 수정 후 다시 git push
- Merge 페이지에서 Open된 Merge의 커밋 확인 후 다시 Merge 시도