Git 을 통한 버전 관리 입문

본 슬라이드 및 그 부속 메모는 소프트웨어 개발의 협업에서 가장 인기 있고 널리 쓰이는 도구인 Git 를 입문하는 것을 목적으로 한다.

(LibreOffice 를 쓴다면, 메뉴의 “보기 → 메모”를 선택하면 각 슬라이드의 메모를 열람 가능하다.)

저작권

넓은 곳에서 좁은 곳으로

- 프로젝트의 복잡도 관리와 협업
- 충돌-동기화-통합의 문제와 VCS
- VCS 가 부적절한 경우와 적절한 경우
- codebase 란 무엇인가?
- Git 의 적요
- Git 를 쓰는 방법

여러 사람이 하나의 프로젝트를 만든다는 것

- ‘직교성’과 작업 배분의 문제
- 충돌의 문제
- 동기화의 문제
- 통합의 문제

VCS(버전 관리 시스템)은 이 문제들을 프로그램이 인식하여 서술하도록 만들고, 프로그래머를 도와 프로그래머가 해결하도록 하는 툴이다.

VCS 를 안 쓸 경우의 “충돌, 동기화, 통합”

- 압축 파일의 형태
- 카*오톡 같은 메신저의 첨부 파일
- SMB 같은 공유 폴더 형태

- Dr*p**x, G**gl* Dr*ve, Nextcloud 같은 폴더 동기화 툴을 이용
- USB 메모리로 수동으로 이동
- diff/patch 를 수동으로 작성하여 배포

… 결과는?

“model_최종.zip, model_최종2.zip, model_진짜최종.zip, model_마지막.zip ...”
“core.java.김철수의_20231012_충돌_버전”
“분명히 수정 했는데 왜 수정 파일이 사라졌지?”

VCS 사용이 부적절한 경우

- 무거운 binary 파일을 자주 갱신해야 하는 경우 (e.g. 포*샵 작업 원본)
- 주 수정 대상이 plain text 가 아닌 경우 (e.g. 워드프로세서로 작성하는 문서)
- ...

→ 일반적으로 diff/patch 의 형태로 관리가 힘든 객체 전반

VCS 를 써서 **이득이 있는 부분과 그렇지 않은 부분을 나눠서** 관리! (단, diff/patch 관리가 힘든 객체의 history 관리가 충분히 유익한 경우는 Git LFS 같은 형태를 고려할 수도 있음.)

VCS 사용이 적절한 경우

주로 plain text 로 구성되어 있는 codebase

→ 프로그램 코드, 문서화!

- history 를 효율적으로 관리할 수 있는 객체 → diff/patch
- diff/patch 를 유의미하게 적용할 수 있는 객체 → plain text
- 생성물이 아닌 프로그래머가 작성한 객체 → codebase

codebase 란 무엇인가?

- 일반적으로는 생성물이 아닌 원시코드
- VCS 의 history 로 기록될 필요성이 있는 객체
- 프로젝트에 필수적인 부품

codebase 의 정의는 프로그래머나 프로젝트에 따라서 달라질 수 있으며 궁극적으로는 “무엇을 프로젝트의 일부로 다룰 것인가”의 결정에 따른다.

VCS 바깥에서 관리되는 codebase

- diff/patch 적용이 곤란하지만 codebase 인 것들 (주로 binary 파일들)
  - 프로젝트에서 사용되는 그림, 소리, 영상 등
  - 기계 학습의 모델 결과물 파일
- 비록 VCS 로 관리되지는 않지만 “충돌-동기화-통합”을 처리해야 하는 대상들

Rsync, Unison, Git LFS 같은 동기화 툴을 사용하여 통합!

무슨 VCS 를 선택해야 할까?

- CVS
- SVN
- Mercurial
- Bazaar
- …
- Git

기존의 프로젝트가 이미
사용하고 있는 VCS 가 있다.
→ 그대로 사용

새로 시작하는 프로젝트다.
→ 고민하지 말고 Git

‘Git’를 읽고 적어보자

- Git 은 ‘깃’ 혹은 ‘기트’라 읽는다.
- 일반 문서 내에서는 capitalize 하여 Git 로 적는 것이 원칙이지만 git 이라 적는 경우도 많다. GIT 이라 적는 경우는 드물다.
- CLI 환경에서 Git 의 호출은 항상 git 로 한다. Git 는 비문이다.

Git 의 문서화

- 공식 문서: https://git-scm.com/
- Unix 환경에서는 매뉴얼 페이지를 제공한다: man git, man gittutorial, man giteveryday 등
- 모를 때는 일단 git commit --help 처럼 help 를 해보면 간략한 사용법이 나온다.
- Unix 환경에서는 각 subcommand 에 대해서 매뉴얼이 있다: man git-commit

프로그래머는 컴퓨터 안에 문서화를 넣어 두고 모를 때마다 찾아볼 수 있어야 한다.

Git CLI 와 다른 Git frontends

- Git 의 CLI 명령은 Git 의 기본이면서 최소한의 조작 방법이다.
- IDE 를 쓸 경우 IDE 가 제공하는 Git frontend 를 통해 사용한다.

Git 의 CLI 명령은 어떤 경우라도 적용 가능하며, 다른 Git frontend 들은 내부적으로는 사실상 Git CLI 명령을 실행하고 있다.