

Git 기초 형상관리



개요

형상관리

❖ 정의

- 소프트웨어 개발을 통합 관리하기 위한 방법

❖ 종류

- CVS, SVN, Git 등



목차

❖ Git 기초

- 환경설정, 파일올리기, reset&revert, 브랜치

❖ Github 활용

- 회원가입, git 연동, commit, 내려받기, 동기화, ssh 연동하기

❖ 개별서버 만들기

- 환경설정, 동기화, 자동 로그인

❖ Github와 이클립스 연동

- 환경설정, commit, 동기화

❖ 개별서버 이클립스 연동



- 환경설정, commit, 동기화


환경설정

다운로드

❖ <https://git-scm.com/downloads>

Downloads

 **Mac OS X**  **Windows**

 **Linux/Unix**

Older releases are available and the [Git source repository](#) is on [GitHub](#).

GUI Clients


Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

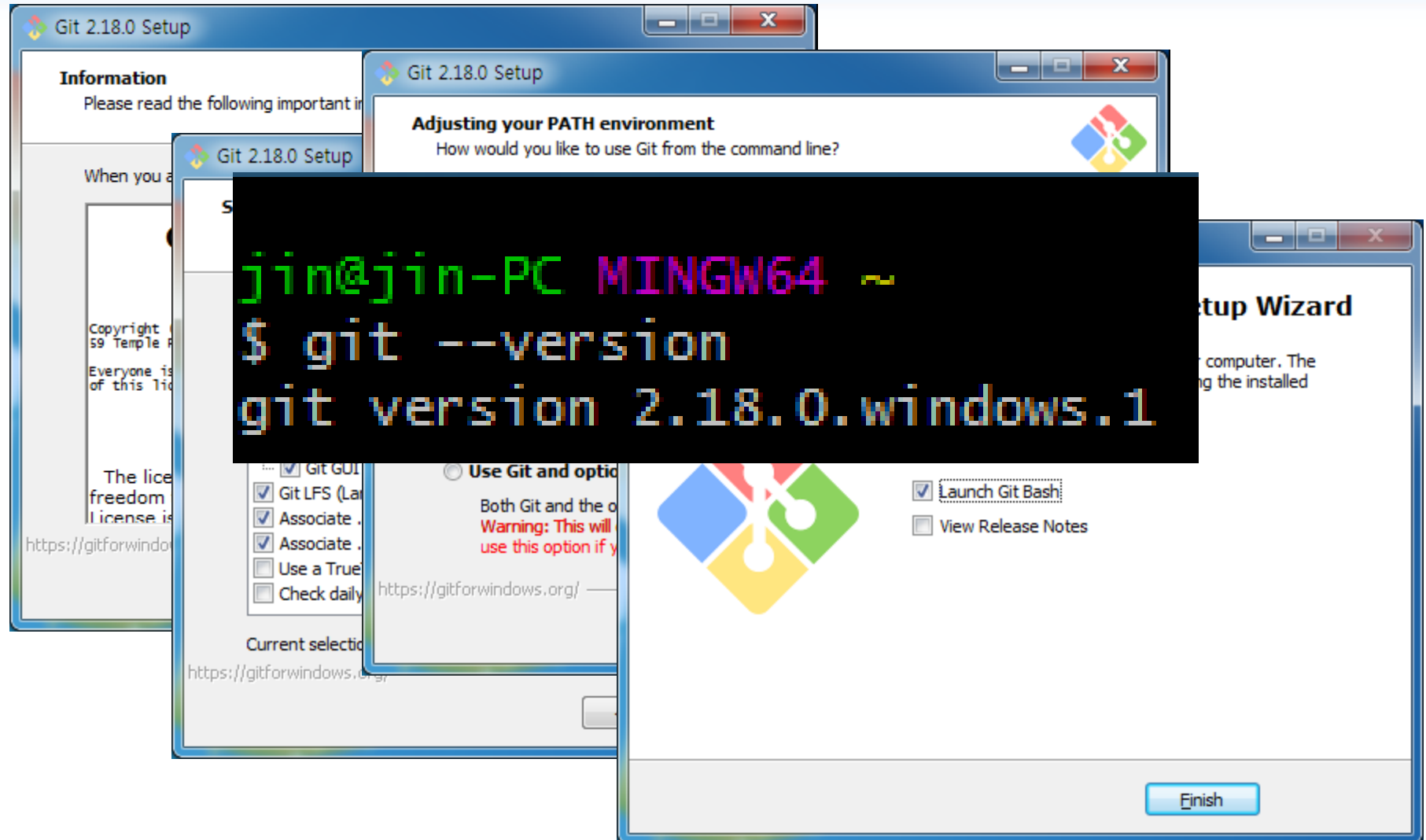
Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)



설치



사용자 등록

❖ 이해하기

- 여러 명이 사용해야 함으로 개개인을 식별하기 위해 개인 정보 등록

❖ 기본문법

git config [위치] [적용객체] [적용 데이터]
[위치]



local



global

[적용객체]

user, core

❖ 사용 예

```
git config --global user.name "yoonki.Cho"
```

```
git config --global user.email "gloss62@naver.com"
```

```
git config --global core.editor notepad
```


저장소 생성

❖ 이해하기

- 개발 소스를 관리하기 위한 저장소 생성

❖ 저장 순서

- 저장할 디렉토리 생성
- 디렉토리 초기화



실습하기

❖ 사용자 등록

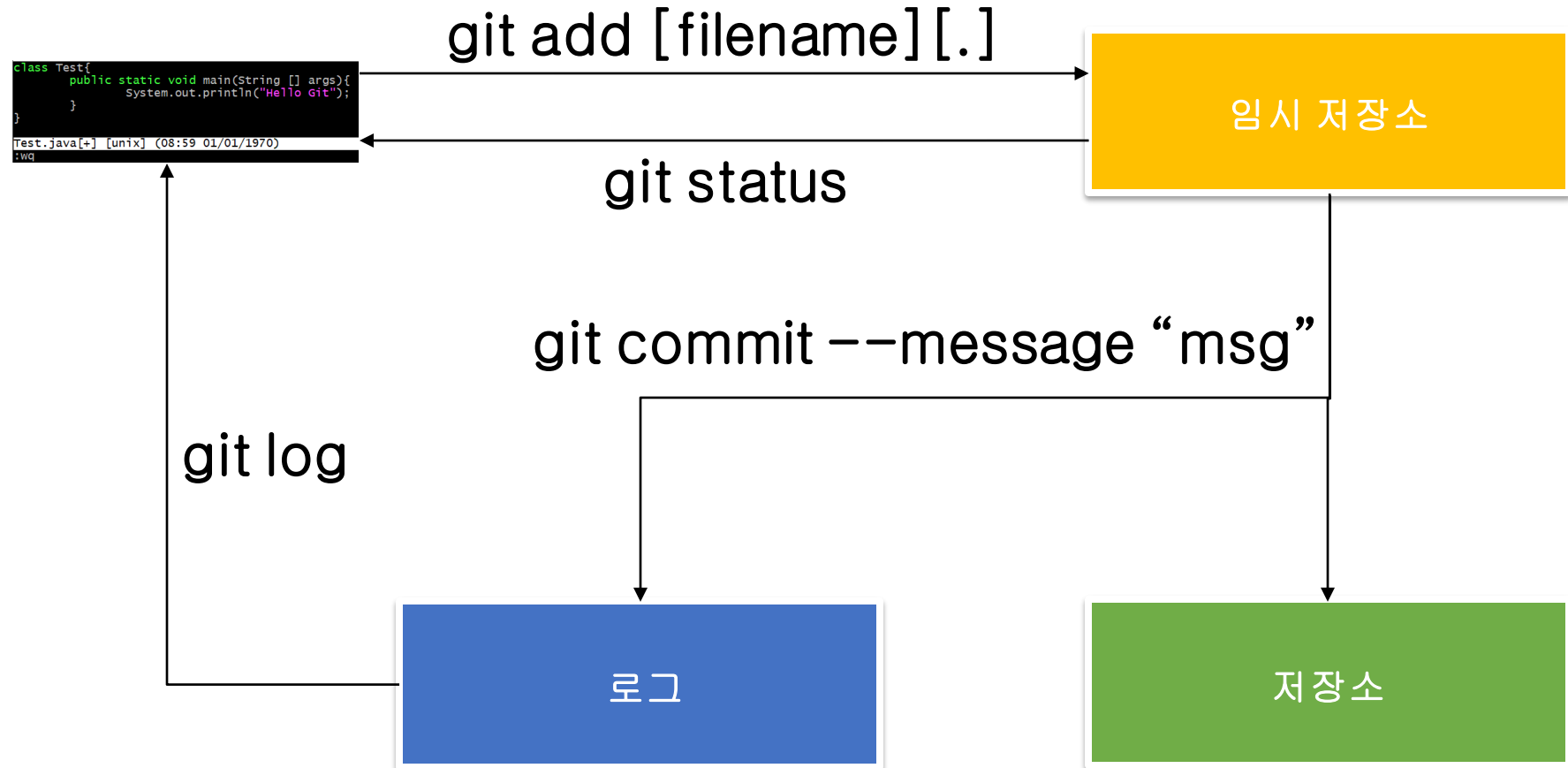
```
git config --global user.name "yoonki.Cho"  
git config --global user.email "gloss62@naver.com"  
git config --global core.editor notepad
```

❖ 저장소 생성

```
pwd  
cd d:  
mkdir git_repository  
cd git-repository  
git init  
cd .git  
ls  
ll
```

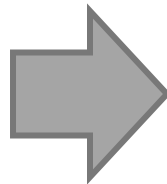
파일올리기

소스 생성

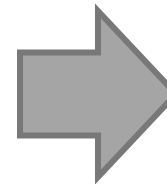


Commit message

git commit



Core.editor
[notepad]



```
# Please enter the commit message for your
# changes. Lines starting
# with '#' will be ignored, and an empty
# message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   Test.java
```

실습

- ❖ 소스 생성
- ❖ 소스 변경
- ❖ Commit
- ❖ Git log

Git staging 이해

영역 이해

❖ 작업공간(Working directory)

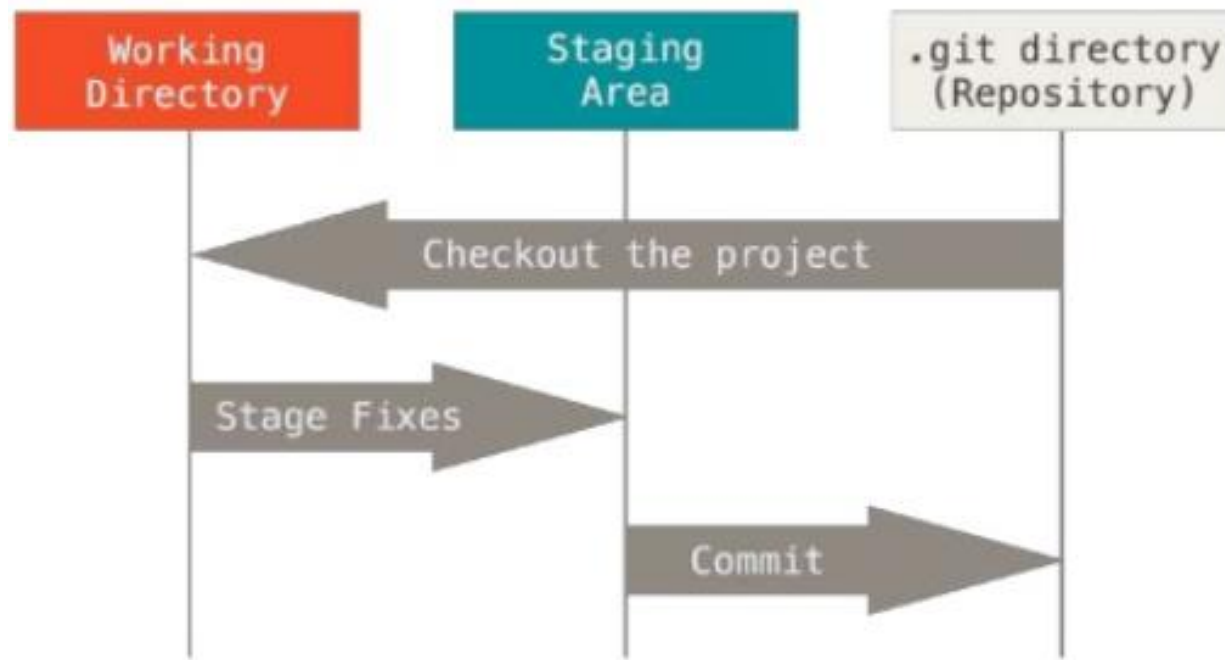
- 소스코드를 작업하는 영역으로 코드를 추가, 수정, 삭제하는 작업이 이루어지는 영역

❖ 논리 저장소(Staging Area)

- 작업공간에서 `git add` 명령을 통해 git의 논리 저장소로 파일 올리기

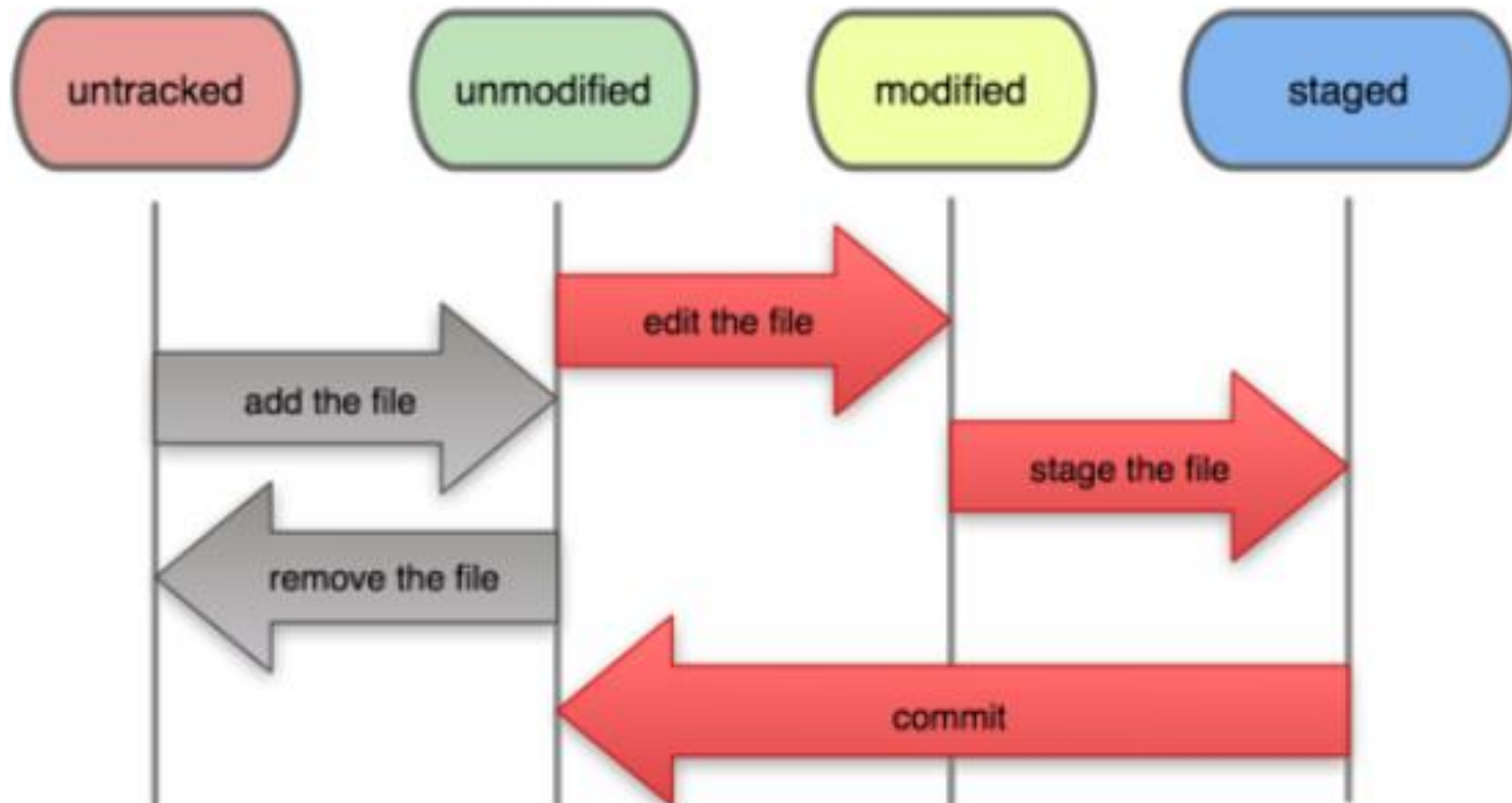
❖ 저장소(.git directory)

- `git commit`을 통해 논리 저장소에 있는 파일을 저장한다.



File Status Lifecycle

File Status Lifecycle



실습

- ❖ 두 개의 파일 생성
- ❖ 상태 확인
- ❖ add 후 상태 확인
- ❖ 기존 파일 변경 후 상태 확인
- ❖ 한 개의 파일 삭제 후 상태 확인
- ❖ 두 번째 파일 삭제 후 상태 확인
- ❖ 모든 파일 추가 후 상태 확인
- ❖ 모든 파일 commit 후 상태 확인
- ❖ 로그 확인

변경 사항 확인

변경 사항 확인

❖ 이해하기

- 파일의 추가 및 삭제된 상태를 확인

❖ 사용 예

- `git log -p`
- `git diff`
- `git diff commitID1 commitID2`

```
class Test{  
    public static void main(String [] args){  
-        System.out.println("Heo Git");  
+        System.out.println("Heo");  
        System.out.println("Hi Git");  
        System.out.println("branch1");  
    }  
}
```

실습

- ❖ `git log -p`
- ❖ `git diff`
- ❖ `git diff commitID1 commitID2`

Reset vs Revert

Reset vs Revert

❖ 이해하기

- Reset : 없던 상태로 돌림(로그 삭제), 개인저장소에 사용
- Revert : 없던 상태로 돌림(로그 유지), 서버저장소에 사용

❖ 사용 예

- `git reset --hard commitID`
- `git revert commitID`

실습

- ❖ Reset
- ❖ Revert

Branch

branch

❖ 이해하기

- 개발 시 원본을 유지하고 별도의 branch 생성
- 개발 완료 시 병합
- 병합 시 공통 파일의 경우 충돌 발생

❖ 실습하기

- 단일 브랜치 적용
- 이중 브랜치 적용
- 이중 브랜치 충돌 및 해결
- Stash
- 삭제