오픈소스SW – 12주차 실습

202204495 홍창희

실습

강의자료를 따라 터미널 또는 커맨드에서 임시처리와 워킹 디렉터리 청소에 관련된 실습 절차를 그대로 수행하여 보고서를 제출.

```
hch@HCHui-MacBookPro ~ % cd desktop
hch@HCHui-MacBookPro desktop % mkdir gitstudy07
hch@HCHui-MacBookPro desktop % cd gitstudy07
hch@HCHui-MacBookPro gitstudy07 % git init
 Initialized empty Git repository in /Users/hch/Desktop/gitstudy07/.git/
```

→ 스태시 실습을 하기 위한 로컬 저장소를 생성하고, git init 명령어를 통해 저장소를 초기화 했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % code stash.htm
```


```
hch@HCHui-MacBookPro gitstudy07 % git add stash.htm
hch@HCHui-MacBookPro gitstudy07 % git commit -m "first"
[main (root-commit) c806bbe] first
 1 file changed, 1 insertion(+)
 create mode 100644 stash.htm
```

→ 스태시 실습을 위해 stash.htm 파일을 작성 후 저장하고, 생성한 파일을 git add, git commit 명령어를 통해 등록 및 커밋했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git status
On branch main
nothing to commit, working tree clean
```

→ git status 명령어를 통해 상태를 확인해봤더니, 커밋을 했기때문에 워킹 디렉터리가 깨끗한 상태인 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git checkout -b feature
Switched to a new branch 'feature'
```

→ 내가 개발자로서 일하고 있고, 수정 작업 도중에 또 다른 수정 요청을 받았을 경우라고 가정하자. 이때 현재 작업은 잠시 멈추고, 또 다른 요청을 반영하려면 새로운 브랜치가 필요하므로 git checkout -b 명령어를 통해 feature 브랜치를 만들고 이동했다.

```
hch@HCHui-MacBookPro gitstudy07 % code stash.htm
```


```
hch@HCHui-MacBookPro gitstudy07 % git commit -am "new feature start"
[feature 02c2ac6] new feature start
 1 file changed, 2 insertions(+), 1 deletion(-)
```

→ 새로운 feature 브랜치에서 stash.htm 파일을 수정하고 저장한 뒤, “new feature start” 라는 메세지와 함께 커밋했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % code stash.htm
```


```
hch@HCHui-MacBookPro gitstudy07 % git status
On branch feature
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:   stash.htm

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

→ feature 브랜치에서 파일을 한 번 더 수정 후 저장하고 커밋은 하지 않았습니다. 그렇기에 git status 명령어를 통해 상태를 확인했을 때 modified 상태인 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git checkout master
error: pathspec 'master' did not match any file(s) known to git
hch@HCHui-MacBookPro gitstudy07 % git checkout main
error: Your local changes to the following files would be overwritten by checkout:
        stash.htm
Please commit your changes or stash them before you switch branches.
Aborting
```

→ 위와 같은 상황에서 다른 버그를 수정하기 위해 다른 브랜치에서 작업을 하려고 main 브랜치로 체크아웃 하려고하니, 오류 메시지가 출력되었습니다. 이는 워킹 디렉토리에 커밋하지 않은 작업이 남아있어 현재 브랜치를 변경할 수 없다는 내용이었습니다.

→ 위와 같은 상황에서 스태시를 활용하면 유용합니다.

(워킹 디렉터리와 스테이지를 깔끔하게 정리할 수 있기 때문에.)

스태시를 저장하는 방법:

→ git stash 명령어

```
hch@HCHui-MacBookPro gitstudy07 % git status
On branch feature
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:   stash.htm

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

→ git status 명령어를 통해 알 수 있듯이, 현재 작업중인 브랜치의 워킹 디렉토리를 정리하지 못한 상태입니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash
Saved working directory and index state WIP on feature: 02c2ac6 new feature start
```

```
hch@HCHui-MacBookPro gitstudy07 % git status
On branch feature
nothing to commit, working tree clean
```

→ git stash 명령어를 통해 작업 중인 내역들을 스택에 저장했습니다. 그 뒤에 git status 명령어를 통해 상태를 확인해보니 깨끗하게 정리된 상태인 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git checkout main
Switched to branch 'main'
```

→ 워킹 디렉토리가 깨끗한 상태이므로 이제 현재 브랜치를 변경할 수 있었습니다. git checkout 명령어를 통해 main 브랜치로 이동했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % cat .git/refs/stash
c947fa9b2b6b923130df9ed18f534153369971be
```

→ (참고) 깃은 스태시된 객체들을 .git/refs/stash 에 저장한다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
stash@{0}: WIP on feature: 02c2ac6 new feature start
```

→ 현재 main 브랜치에서 git stash list 명령어를 통해 스태시 목록을 확인해보니, feature 브랜치에서 저장한 스태시를 볼 수 있었다. (강의자료에선 main 에서 저장된 스태시가 출력된다고 나오지만 이는 틀린 것이다.)

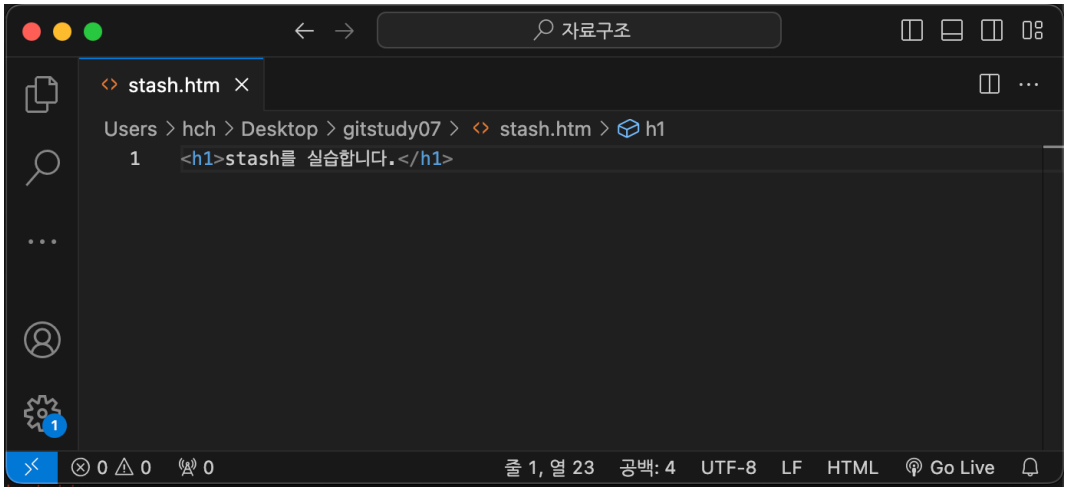
```
hch@HCHui-MacBookPro gitstudy07 % git stash show
 stash.htm | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

→ git stash show 명령어를 사용하면 현재 워킹 디렉토리 내용과 스태시된 내용 간의 파일 변화를 간략하게 출력해주는 것을 확인했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash show -p stash@{0}
diff --git a/stash.htm b/stash.htm
index 8088c30..5804e8c 100644
--- a/stash.htm
+++ b/stash.htm
@@ -1,2 +1,3 @@
 <h1>스태시를 실습합니다.</h1>
-<h2>새로운 기능을 시작합니다.</h2>
\ No newline at end of file
+<h2>새로운 기능을 시작합니다.</h2>
+새로운 기능의 너무 복잡하네요. 시간이 많이 걸릴 것 같습니다.
\ No newline at end of file
```

→ git stash show 명령어에 -p 옵션을 추가하면 상세 차이점도 알려주는 것을 확인했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % code stash.htm
```




```
hch@HCHui-MacBookPro gitstudy07 % git commit -am "bug fix main"
[main be2af41] bug fix main
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
hch@HCHui-MacBookPro gitstudy07 % git checkout feature
Switched to branch 'feature'
```

→ 현재 브랜치인 main 브랜치에서 stash.htm 파일을 수정(스태시 -> stash)하고 저장한 뒤, "bug fix main" 메시지와 함께 커밋했습니다. 그 후에 원래 브랜치 feature 브랜치로 이동했습니다.

저장된 스태시를 불러오는 방법:

→ pop 옵션(스택에서 삭제 O), apply 옵션(스택에서 삭제 X)

```
hch@HCHui-MacBookPro gitstudy07 % git stash pop
On branch feature
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:   stash.htm

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (c947fa9b2b6b923130df9ed18f534153369971be)
```

→ 현재 브랜치인 feature 브랜치에서, 임시 저장된 스태시를 불러오기 위해 git stash pop 명령어를 사용했습니다. modified 상태인 것을 볼 수 있었고, 다시 작업중인 상태가 된 것을 알 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
```

→ pop 옵션을 사용한 뒤에 git stash list 명령어를 통해 스태시 목록을 확인해보니 내용이 없는 것을 볼 수 있었습니다. pop 옵션을 사용하면 스태시 저장 스택에서 항목을 가져오는 동시에 저장된 항목을 삭제하기 때문이었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash
Saved working directory and index state WIP on feature: 02c2ac6 new feature start
```

→ 현재 스태시로 이전 내용을 복원한 상태이기에 워킹 디렉토리의 작업내용이 수정된 상태로 남아있는데, 실습을 위해 현재의 임시 작업 내용을 다시 스태시 했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
stash@{0}: WIP on feature: 02c2ac6 new feature start
```

→ git stash list 명령어를 통해 스태시 목록을 확인해보니 스태시 내용이 하나 있는 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash branch test
Switched to a new branch 'test'
On branch test
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:   stash.htm

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (01629ab5bf46c96fa4f4d411384fe9241e8c3716)
```

→ git stash branch test 명령어를 통해 새로운 test 브랜치를 생성하고 동시에 스태시를 적용했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git branch
  feature
  main
* test
```

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
```

→ 이제 현재 브랜치는 test 브랜치가 되었는데, git stash list 명령어를 통해 스태시 목록을 확인해보니 스태시가 적용되었기에 저장된 스택이 삭제되어 스태시 목록이 비어있는 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash
Saved working directory and index state WIP on test: 02c2ac6 new feature start
```

→ 스태시 복원 명령어 중 하나인 apply 옵션을 실습하기 위해 test 브랜치에서 임시 작업 내용을 다시 스태시로 저장했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git checkout feature
Switched to branch 'feature'
```

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
stash@{0}: WIP on test: 02c2ac6 new feature start
```

→ feature 브랜치로 이동한 후에도 스태시에 저장된 내용을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash apply stash@{0}
On branch feature
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:   stash.htm

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

→ git stash apply stash@{0} 명령어를 통해 스태시 번호를 이용해서 스태시를 복원했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git stash list
stash@{0}: WIP on test: 02c2ac6 new feature start
```

→ apply 옵션을 사용해서 스태시를 복원하게되면 스택에서 삭제시키지 않고 복사하는 것과 비슷하게 동작하기때문에 스태시 목록에 0번 스태시가 그대로 남아있는 것을 확인할 수 있었습니다.

스태시를 삭제하는 방법:

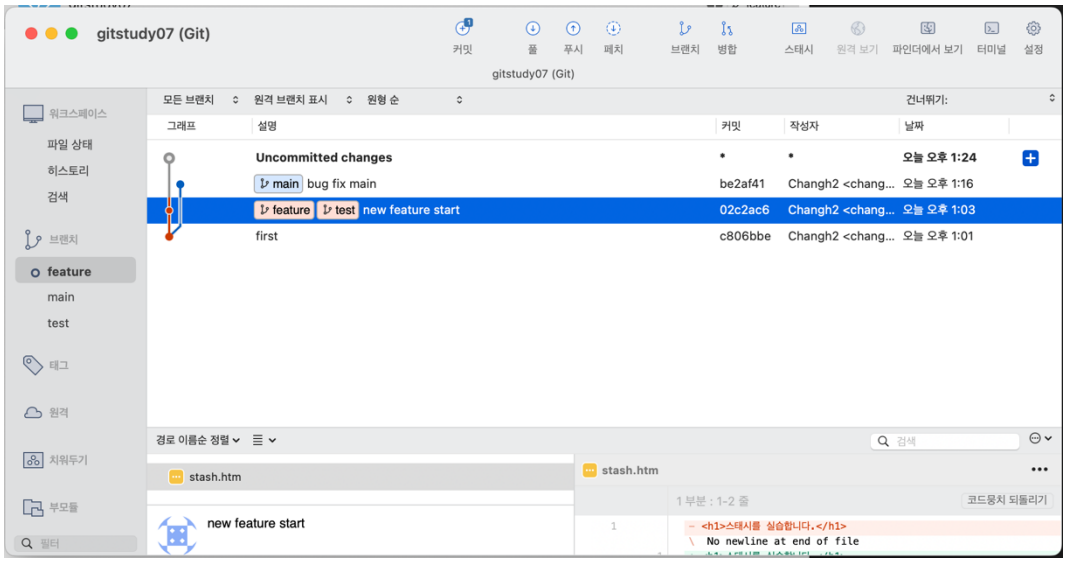
→ git stash drop 명령어

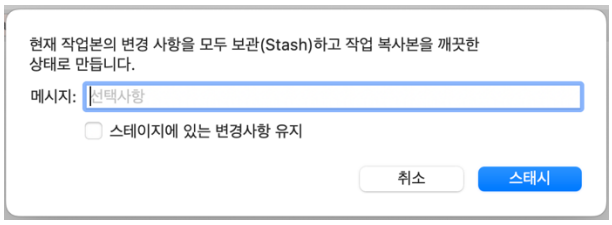
```
hch@HCHui-MacBookPro gitstudy07 % git stash list
stash@{0}: WIP on test: 02c2ac6 new feature start
```

```
hch@HCHui-MacBookPro gitstudy07 % git stash drop
Dropped refs/stash@{0} (8140f3a7d442fb546a641d2f1946d975abf50f71)
```

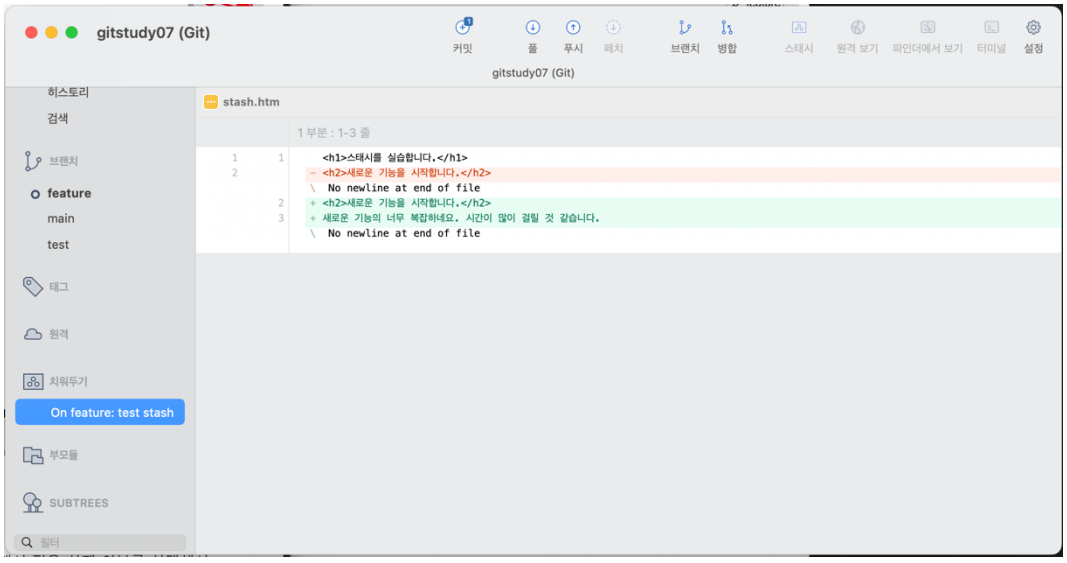
→ git stash drop 명령어를 통해 스태시 목록에 있던 0번 스태시를 삭제했습니다.

위의 스태시와 관련된 모든 과정은 소스트리에서도 가능하다.





: 스태시 저장



: 스태시 적용/삭제

워킹 디렉토리 청소

→ git clean 명령어에 몇가지 옵션을 꼭 같이 사용

```
hch@HCHui-MacBookPro gitstudy07 % git branch
* feature
  main
  test
```

```
hch@HCHui-MacBookPro gitstudy07 % echo "this is temp" > temp.htm
```

```
hch@HCHui-MacBookPro gitstudy07 % git status
On branch feature
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        temp.htm

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

→ 실습을 위해 feature 브랜치에 새로운 temp.htm 파일을 하나 생성했습니다. git status 명령어를 통해 상태를 확인하니 강의자료와는 다르게 stash.htm 파일이 modified 상태가 아니였는데, 위의 소스트리를 통한 실습을 하는 과정에서 스태시를 저장하고 삭제했기 때문인 것으로 추측됩니다.

```
hch@HCHui-MacBookPro gitstudy07 % git clean
fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to clean
```

→ 워킹 디렉토리 청소를 위해서 git clean 명령어 만을 사용하게 되면 오류 메시지가 출력되는데, clean 명령어는 민감해서 몇 가지 옵션을 같이 사용해야만 하기 때문입니다.

```
hch@HCHui-MacBookPro gitstudy07 % git clean -help
usage: git clean [-d] [-f] [-i] [-n] [-q] [-e <pattern>] [-x | -X] [--] [<pathspec>...]

    -q, --quiet           do not print names of files removed
    -n, --dry-run         dry run
    -f, --force           force
    -i, --interactive     interactive cleaning
    -d                    remove whole directories
    -e, --exclude <pattern>
                          add <pattern> to ignore rules
    -x                    remove ignored files, too
    -X                    remove only ignored files
```

→ 옵션에 대한 도움말을 보기 위해서 -help 옵션을 사용했습니다.

```
hch@HCHui-MacBookPro gitstudy07 % git clean -f
Removing temp.htm
```

→ -f 옵션을 사용하여 강제로 삭제했습니다.