오픈소스SW – 9주차 실습

202204495 홍창희

실습

강의자료를 따라 터미널 또는 커맨드에서 브랜치와 관련된 실습 절차를 그대로 수행하여 보고서를 제출.

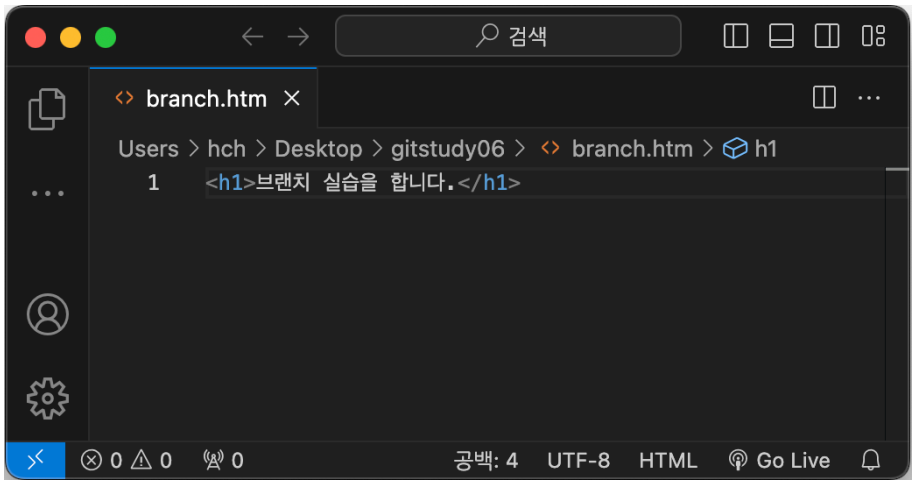
```
hch@HCHui-MacBookPro ~ % cd desktop
hch@HCHui-MacBookPro desktop % mkdir gitstudy06
hch@HCHui-MacBookPro desktop % cd gitstudy06
hch@HCHui-MacBookPro gitstudy06 % git init
Initialized empty Git repository in /Users/hch/Desktop/gitstudy06/.git/
hch@HCHui-MacBookPro gitstudy06 % git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

→ 실습을 위한 폴더를 만들고 저장소를 초기화하여 실습준비를 마쳤습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git branch
```

→ 강의자료에선 git branch 를 입력하면 *master 브랜치가 나온다고 설명하지만, 실제론 처음에 브랜치가 master 한개 뿐일땐 나오지 않는다는 것도 확인했습니다.

```
hch@HCHui-MacBookPro gitstudy06 % code branch.htm
```



→ code 명령어를 통해 저장소에 branch.htm 파일을 만들었습니다

```
[hch@HCHui-MacBookPro gitstudy06 % git add branch.htm
[hch@HCHui-MacBookPro gitstudy06 % git commit -m "first"
[main (root-commit) 3b25186] first
 1 file changed, 1 insertion(+)
 create mode 100644 branch.htm
```

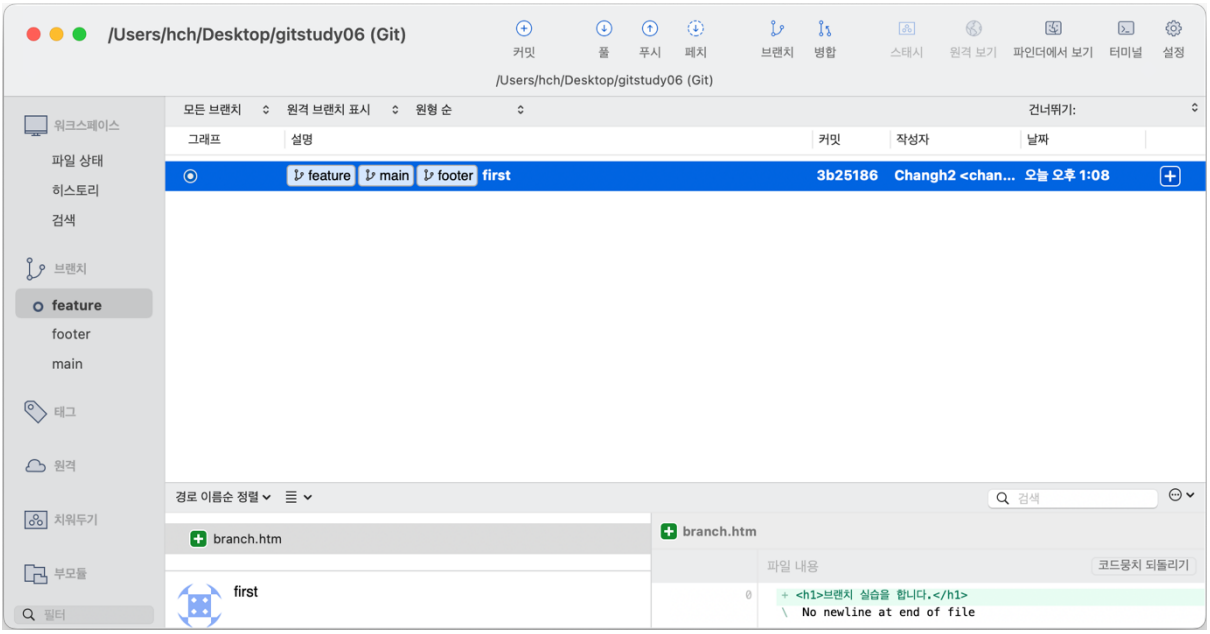
→ add 명령어를 통해 tracked상태가 되게 하고, "first" 메세지의 커밋을 추가했습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git branch footer
[hch@HCHui-MacBookPro gitstudy06 % git branch
  footer
* main
```

→ 커밋이 있는 상태에서 git branch 명령어를 통해 새로운 브랜치 'footer'를 생성했습니다. 그 뒤 git branch 명령어를 통해 브랜치 목록과 현재 브랜치를 확인해보니 footer와 main 브랜치가 있고, 현재 브랜치는 main인 것을 볼 수 있었습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git branch footer
fatal: a branch named 'footer' already exists
```

→ git branch 명령어를 통해 아까 만든 footer 브랜치와 같은 이름의 브랜치를 만들려고 하니 오류가 발생하는 것을 볼 수 있었고, 이를 통해 브랜치 이름은 중복해서 사용하지 않아야 한다는 것을 알 수 있었습니다.



→ 소스트리를 통해 feature 브랜치를 생성했고, 브랜치 목록으로 feature, footer, main 이 있고, 현재 브랜치는 feature가 된 것을 확인했습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git branch
* feature
  footer
  main
```

→ 소스트리에서 확인한 것과 동일하게 터미널에서도 git branch 명령어를 통해 브랜치 목록과 현재 브랜치를 확인할 수 있었습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git log
commit 3b2518669c35189af7f37b5b4e027b0ec24829fa (HEAD -> feature, main, footer)
Author: Changh2 <changhi9701@gmail.com>
Date:   Fri May 3 13:08:58 2024 +0900

    first
```

→ git log 명령어를 통해 커밋 로그를 확인했습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git rev-parse footer
3b2518669c35189af7f37b5b4e027b0ec24829fa
```

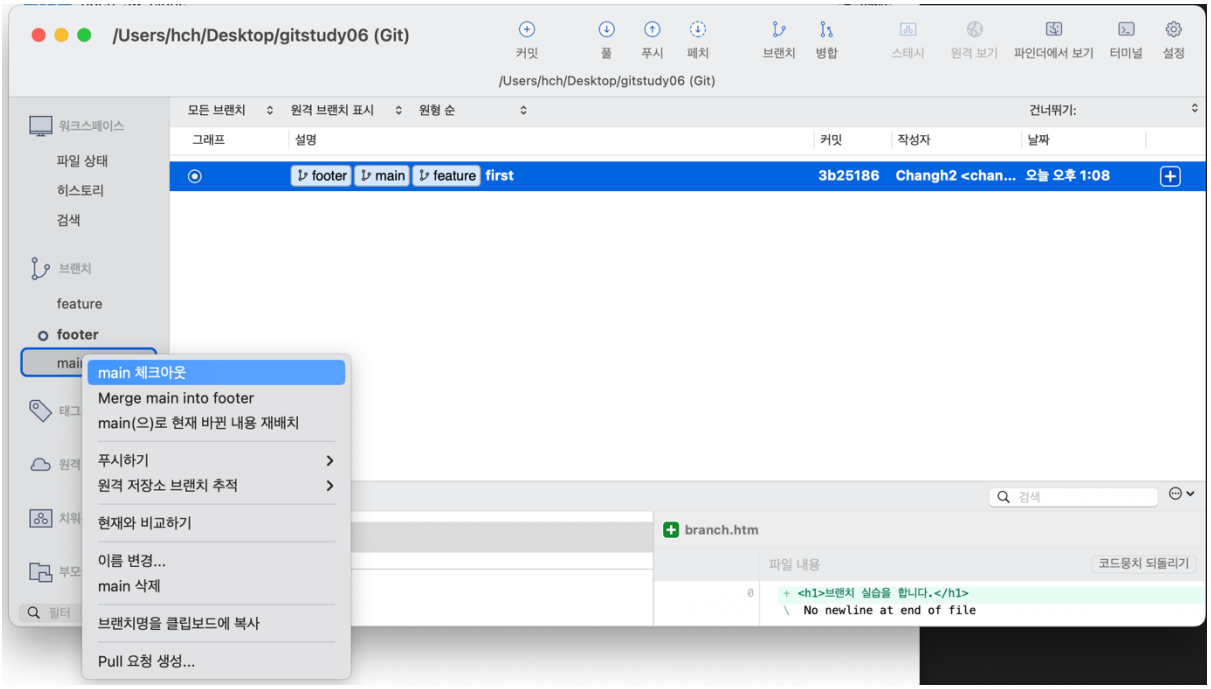
→ git rev-parse 명령어를 통해 좀 전에 만들었던 footer 브랜치의 커밋 해시를 확인해보니, 브랜치의 해시 값과 브랜치를 생성한 기준 커밋의 해시 값이 '3b25186...' 으로 동일한 것을 확인할 수 있었습니다.

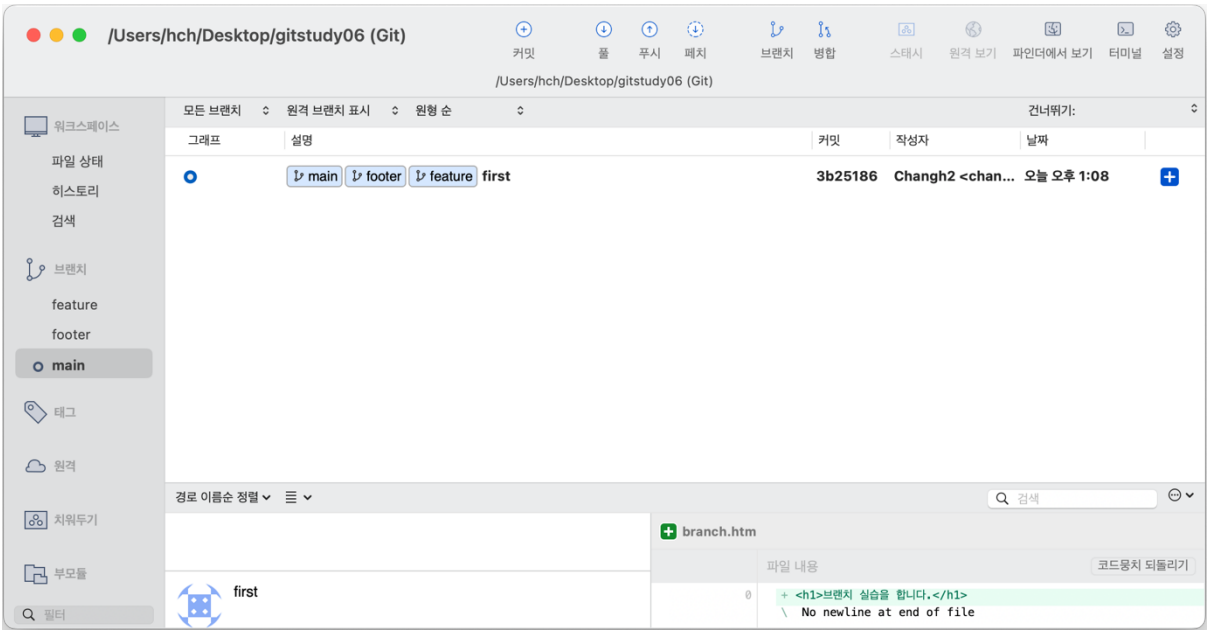
```
[hch@HCHui-MacBookPro gitstudy06 % git branch -v
* feature 3b25186 first
  footer  3b25186 first
  main    3b25186 first
```

→ 간단한 브랜치 이름만 출력하는 git branch 명령어와 달리, 뒤에 -v 를 붙여 git branch -v 명령어를 사용하면 브랜치 이름, 커밋 ID, 커밋 메세지를 같이 볼 수 있다는 것을 확인했습니다.

```
[hch@HCHui-MacBookPro gitstudy06 % git checkout footer
Switched to branch 'footer'
[hch@HCHui-MacBookPro gitstudy06 % git branch -v
  feature 3b25186 first
* footer  3b25186 first
  main    3b25186 first
```

→ git checkout 명령어를 통해 현재 브랜치를 footer 브랜치로 이동했고, git branch -v 명령어를 통해 브랜치 목록을 확인해보니, 현재 브랜치가 footer로 잘 변경된 것을 볼 수 있었습니다.





→ 소스트리에서도 브랜치를 이동(변경)해봤습니다. footer 브랜치에서 main 브랜치로 잘 변경된 것을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git branch
  feature
  footer
* main
```

→ 변경된 현재 브랜치는 터미널에서도 동일하게 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout -
Switched to branch 'footer'
hch@HCHui-MacBookPro gitstudy06 % git branch
  feature
* footer
  main
```

→ git checkout – 명령어를 통해 이전 브랜치로 돌아갔습니다. git branch 명령어를 통해 main 브랜치에서 footer 브랜치로 되돌아 간 것을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout main
Switched to branch 'main'
hch@HCHui-MacBookPro gitstudy06 % code branch.htm
```

→ main 브랜치로 이동한 뒤, code 명령어를 통해 코드를 수정하고 저장했습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git status
On branch 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:   branch.htm

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

→ 저장은 했지만 커밋은 하지 않은 상태인 것을 git status 명령어를 통해 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout footer
M       branch.htm
Switched to branch 'footer'
```

→ 위의 상태에서 브랜치를 변경하니 파일명 앞에 M 표시를 띄워 modified 상태인 것을 경고해주는걸 볼 수 있었습니다,

```
hch@HCHui-MacBookPro gitstudy06 % git checkout -
M       branch.htm
Switched to branch 'main'
hch@HCHui-MacBookPro gitstudy06 % git commit -am "main working..."
[main e951d2b] main working...
 1 file changed, 2 insertions(+), 1 deletion(-)
hch@HCHui-MacBookPro gitstudy06 % git checkout footer
Switched to branch 'footer'
```

→ 브랜치 간에 정상적으로 이동하려면 남아있는 작업들을 정리해 주어야 하므로, 이전 브랜치인 main 브랜치로 되돌아가 수정된 내용을 커밋했고, footer 브랜치로 다시 이동했습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git log --graph --all
* commit e951d2b5a5fdbf594aea31574a434859001006a8 (main)
| Author: Changh2 <changhi9701@gmail.com>
| Date:   Fri May 3 13:21:19 2024 +0900
|
|     main working...
|
* commit 3b2518669c35189af7f37b5b4e027b0ec24829fa (HEAD -> footer, feature)
  Author: Changh2 <changhi9701@gmail.com>
  Date:   Fri May 3 13:08:58 2024 +0900

      first
```

→ 브랜치 흐름을 보기 위해 git log –graph –all 명령어를 입력했고, 잘 출력되는 것을 확인했습니다.

```
hch@HCHui-MacBookPro gitstudy06 % cat branch.htm
<h1>브랜치 실습을 합니다.</h1>%
```

→ cat 명령어를 통해 현재 브랜치인 footer 브랜치의 소스 코드 내용을 확인해보니, 첫 번째 커밋까지만 적용되어있는 것을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout main
Switched to branch 'main'
hch@HCHui-MacBookPro gitstudy06 % cat branch.htm
<h1>브랜치 실습을 합니다.</h1>
<h2>메인 워킹 디렉터리 작업 중</h2>%
```

→ checkout 명령어를 통해 main 브랜치로 이동한 후, cat 명령어를 통해 소스 코드 내용을 확인해보니 두번째 커밋까지 적용되어있는 것을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout footer
Switched to branch 'footer'
hch@HCHui-MacBookPro gitstudy06 % git log --graph --all
* commit e951d2b5a5fdbf594aea31574a4348590001006a8 (main)
| Author: Changh2 <changhi9701@gmail.com>
| Date:   Fri May 3 13:21:19 2024 +0900
|
|     main working...
|
* commit 3b2518669c35189af7f37b5b4e027b0ec24829fa (HEAD -> footer, feature)
  Author: Changh2 <changhi9701@gmail.com>
  Date:   Fri May 3 13:08:58 2024 +0900

      first
```

→ checkout 명령어를 통해 footer 브랜치로 이동한 후, git log –graph –all 명령어를 통해 커밋 로그를 확인해보니, 현재 HEAD가 footer 브랜치의 마지막 커밋, 즉 '3b25186'을 가리킨다는 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout main
Switched to branch 'main'
hch@HCHui-MacBookPro gitstudy06 % git log --graph --all
* commit e951d2b5a5fdbf594aea31574a4348590001006a8 (HEAD -> main)
| Author: Changh2 <changhi9701@gmail.com>
| Date:   Fri May 3 13:21:19 2024 +0900
|
|     main working...
|
* commit 3b2518669c35189af7f37b5b4e027b0ec24829fa (footer, feature)
  Author: Changh2 <changhi9701@gmail.com>
  Date:   Fri May 3 13:08:58 2024 +0900

      first
```

→ checkout 명령어를 통해 main 브랜치로 이동한 후, git log –graph –all 명령어를 통해 커밋 로그를 확인해보니, 현재 HEAD가 main 브랜치의 마지막 커밋, 즉 'e951d2b'를 가리킨다는 것을 볼 수 있었습니다,

```
hch@HCHui-MacBookPro gitstudy06 % git checkout -b hotfix
Switched to a new branch 'hotfix'
hch@HCHui-MacBookPro gitstudy06 % git branch -v
  feature 3b25186 first
  footer  3b25186 first
* hotfix  e951d2b main working...
  main    e951d2b main working...
```

→ checkout -b 명령어를 통해 hotfix 브랜치를 생성하는 동시에 체크아웃(이동)했고, git branch -v 명령어를 통해 브랜치 목록을 확인해보니 hotfix 브랜치가 추가되고, 현재 브랜치가 hotfix인 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout e951d2b
Note: switching to 'e951d2b'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at e951d2b main working...
```

```
hch@HCHui-MacBookPro gitstudy06 % git branch -v
* (HEAD detached at e951d2b) e951d2b main working...
  feature                    3b25186 first
  footer                     3b25186 first
  hotfix                     e951d2b main working...
  main                       e951d2b main working...
```

→ 커밋 해시키로 HEAD를 직접 체크아웃(이동)시켰습니다.
그 뒤 git branch -v 명령어를 통해 브랜치 목록을 확인해보니, 위와 같이 '(HEAD detached at e951d2b)' 이름의 'e951d2b' 해시키를 갖는 새로운 브랜치가 생기고, 이동된 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout HEAD~1
Previous HEAD position was e951d2b main working...
HEAD is now at 3b25186 first
```

```
hch@HCHui-MacBookPro gitstudy06 % git branch -v
* (HEAD detached at 3b25186) 3b25186 first
  feature                    3b25186 first
  footer                     3b25186 first
  hotfix                     e951d2b main working...
  main                       e951d2b main working...
```

→ HEAD~1 명령어를 통해 현재의 한 단계 전 커밋으로 이동했습니다.
그 뒤 git branch -v 명령어를 통해 브랜치 목록을 확인해보니, 위와 같이 '(HEAD detached at 3b25186)' 이름의 '3b25186' 해시키를 갖는 새로운 브랜치가 생기고, 이동된 것을 볼 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout HEAD~5
error: pathspec 'HEAD~5' did not match any file(s) known to git
```

→ HEAD~5 명령어를 입력하면, 5번째 전 단계의 커밋은 존재하지 않기에 에러를 띄우는 것을 확인할 수 있었습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout hotfix
Previous HEAD position was 3b25186 first
Switched to branch 'hotfix'
hch@HCHui-MacBookPro gitstudy06 % git checkout e951d2b
Note: switching to 'e951d2b'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at e951d2b main working...
```

→ 실습자료에선 위의 HEAD~1 명령어는 직접 입력하지 않은 상태이기에, 다시 hotfix 브랜치로 이동하고, 'e951d2b' 해시키로 직접 체크아웃(이동)했습니다.

```
hch@HCHui-MacBookPro gitstudy06 % git checkout -
Switched to branch 'hotfix'
hch@HCHui-MacBookPro gitstudy06 % git checkout main
Switched to branch 'main'
```

→ checkout – 명령어를 입력하니 이전 브랜치인 hotfix로 잘 돌아오는 것을 확인했고, main 브랜치로 이동시켰습니다.