

* github.com에서 회원가입

* 가입한 정보로 로그인하기

===== [원격 저장소 생성] =====


1. <https://github.com/> 접속 후 로그인 합니다.


(1) 왼쪽에 Create repository 버튼 클릭 또는 <https://github.com/new> 로 접속

(2) Create a new repository 화면에서

(2-1) Repository name : jsp

(2-2) Private 선택

☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☒  **Private**
You choose who can see and commit to this repository.

(2-3) Add a README file

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs](#)

(2-4) Add .gitignore

.gitignore template 클릭

=> Java 입력 => Java 선택

=> .gitignore template: Java ▼

Add .gitignore

.gitignore template: Java ▼

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

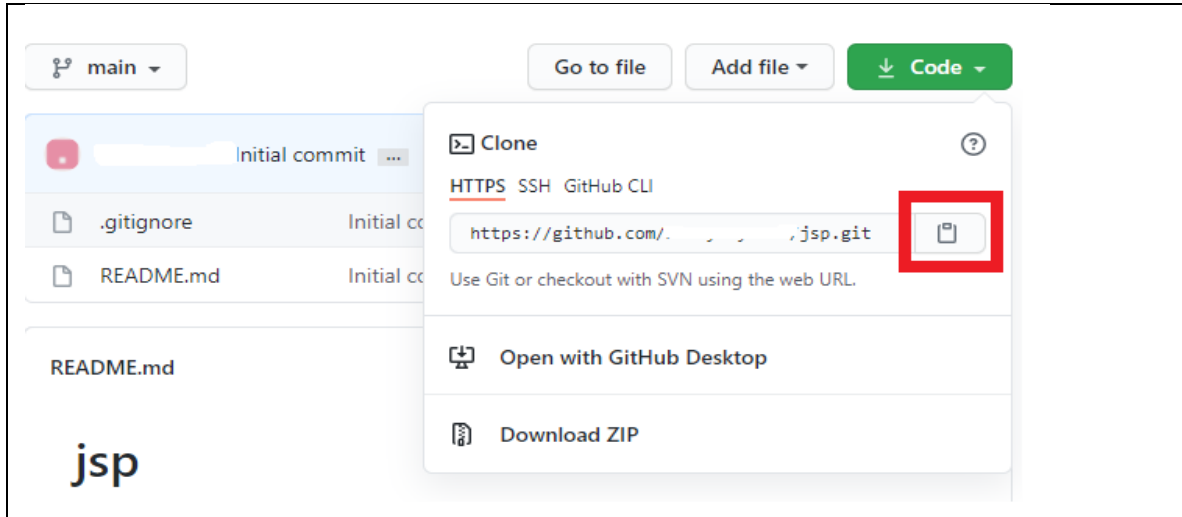
(2-5) Create Repository 버튼 클릭

Create repository

(3) 위 단계를 모두 실행 하면 <https://github.com/username/jsp>로 이동합니다.

(3-1) Code 버튼 클릭

=> <https://github.com/username/jsp.git> 옆 아이콘 클릭합니다. (주소가 복사됩니다.)



참고사항

<https://www.toptal.com/developers/gitignore> 로 접속

- ⇒ 검색창에서 **Java** 입력 후 엔터
- ⇒ Eclipse 입력 후 엔터
- ⇒ 생성 클릭



Java 입력 후 생성 클릭 후 결과 화면

```
# Created by https://www.toptal.com/developers/gitignore/api/java
# Edit at https://www.toptal.com/developers/gitignore?templates=java

### Java ###
# Compiled class file
*.class
```

```
# Log file
*.log

# BlueJ files
*.ctxt

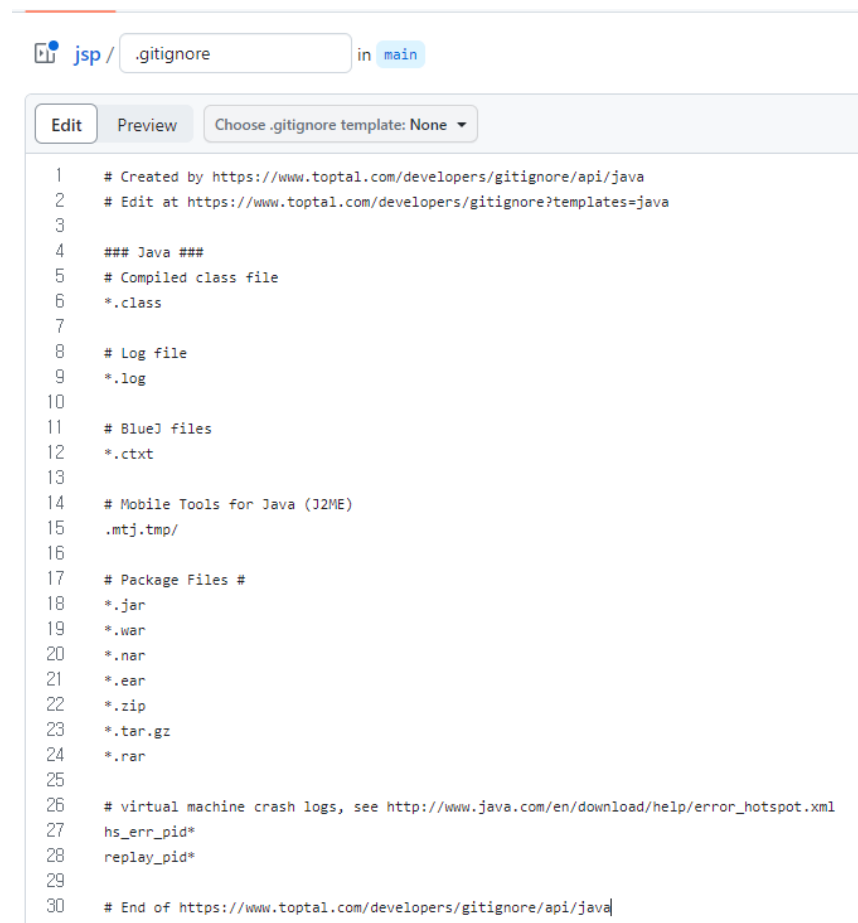
# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
replay_pid*

# End of https://www.toptal.com/developers/gitignore/api/java
```

Add file -> Create new file 선택



The screenshot shows a web-based editor for a .gitignore file. At the top, there's a breadcrumb navigation: a folder icon, 'jsp /', a text input containing '.gitignore', and 'in main'. Below this is a toolbar with three buttons: 'Edit' (active), 'Preview', and 'Choose .gitignore template: None'. The main area is a code editor with line numbers 1 through 30 on the left. The code content is identical to the one in the first block, but with line numbers added for each line.

```
1  # Created by https://www.toptal.com/developers/gitignore/api/java
2  # Edit at https://www.toptal.com/developers/gitignore?templates=java
3
4  ### Java ###
5  # Compiled class file
6  *.class
7
8  # Log file
9  *.log
10
11 # BlueJ files
12 *.ctxt
13
14 # Mobile Tools for Java (J2ME)
15 .mtj.tmp/
16
17 # Package Files #
18 *.jar
19 *.war
20 *.nar
21 *.ear
22 *.zip
23 *.tar.gz
24 *.rar
25
26 # virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
27 hs_err_pid*
28 replay_pid*
29
30 # End of https://www.toptal.com/developers/gitignore/api/java
```

Commit changes...

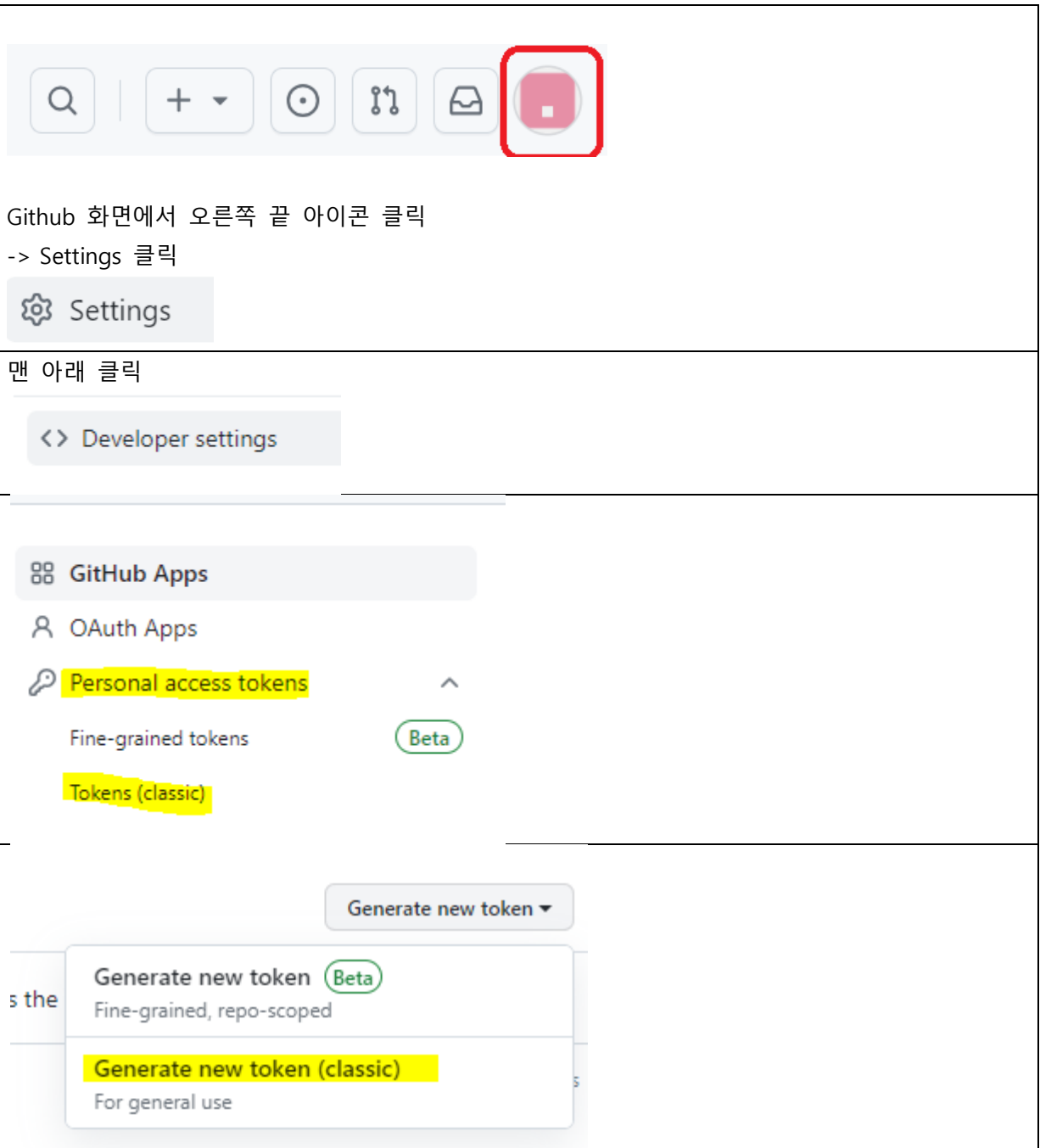
클릭

===== [토큰 생성] =====

처음 저장소를 만든 후 Eclipse에서 Github에 접속하기 위한 인증 단계를 거쳐야 합니다.

이 때 Github의 Username과 Password를 입력해야 하는데 Github 계정의 비밀번호 대신 Github에서 만든 토큰을 넣어야 인증이 됩니다.

2. 토큰 생성하기



The screenshot shows the GitHub web interface. At the top, a navigation bar contains several icons; the user profile icon (a circle with a square) is highlighted with a red square. Below this, the 'Settings' button is visible. Under 'Settings', the 'Developer settings' button is selected. In the 'Developer settings' section, 'GitHub Apps' is expanded, and 'Personal access tokens' is highlighted in yellow. Under 'Personal access tokens', 'Tokens (classic)' is highlighted in yellow. At the bottom, the 'Generate new token' button is shown, and its dropdown menu is open. The dropdown menu has two options: 'Generate new token (Beta)' (labeled 'Fine-grained, repo-scoped') and 'Generate new token (classic)' (labeled 'For general use'). The 'Generate new token (classic)' option is highlighted in yellow.

GitHub 화면에서 오른쪽 끝 아이콘 클릭
-> Settings 클릭

맨 아래 클릭

Generate new token ▼

Generate new token (Beta)
Fine-grained, repo-scoped

Generate new token (classic)
For general use

Note

What's this token for?

임의의 글자 입력

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☒ repo
 Full control of private repositories

☒ repo:status
 Access commit status

☒ repo_deployment
 Access deployment status

☒ public_repo
 Access public repositories

☒ repo:invite
 Access repository invitations

☒ security_events
 Read and write security events

Generate token

발행한 토큰을 잘 저장해 두시고 username과 비번 입력 시 비번에 넣으시면 됩니다.

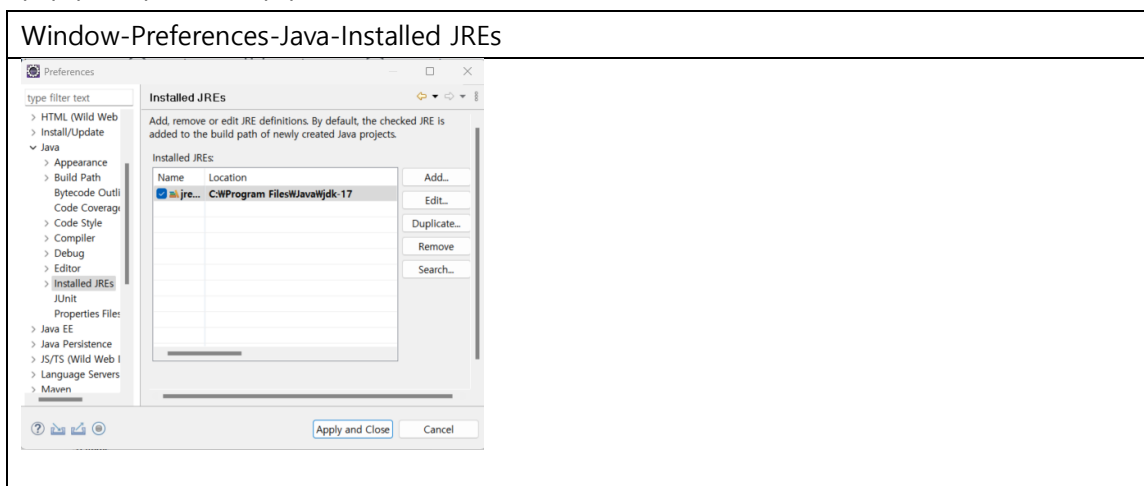
===== [지역 저장소 생성] =====

3. Eclipse에서 작업합니다.

(0) 새로운 워크 스페이스에서 시작합니다.

Eclipse [File]-[Switch Workspace]-[Other]

(1) 아래와 같이 설정합니다.



(2) Window > Perspective > Open Perspective > Other -> Git -> Open

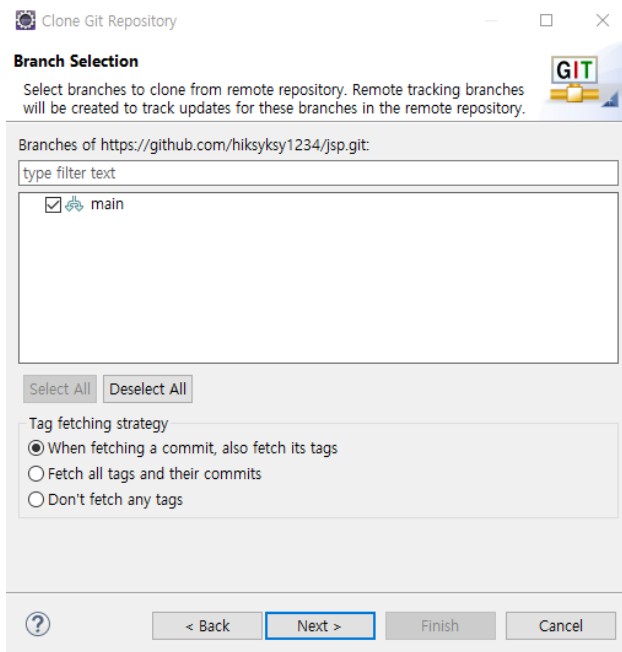
(3) Clone a Git repository 클릭



(4) URI : 6에서 복사한 주소를 붙여 넣기 합니다.

URI : https://github.com/username/jsp.git
Host : github.com
Repository path : /username/jsp.git
User : username
Password : ●●●●●●●●●●●●●●●●
Store in Secure Store 선택(암호 저장합니다.)
=> Next

(5) 원격 저장소에서 클론할 branch를 선택합니다.



(6) 로컬 저장소 관련 설정

Directory : C:\Users\사용자계정\git\jsp

Remote name: origin (원격 저장소 이름을 origin으로 사용)

=> Finish

===== [Dynamic Web Project 생성] =====

3. Eclipse에서 작업합니다.

(1) Window > Perspective > Open Perspective > Other -> JAVA EE -> Open

(2) 서버가 설치 되어 있지 않으면 설치합니다.

(3) Dynamic Web Project 생성

프로젝트 이름 : JspGit

Target runtime: Apache Tomcat v10.1

Dynamic web module version : 5.0

Finish

(4) webapp에서 NewFile.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    1. 처음으로 만든 파일
</body>
</html>
```

(5) src에서 _1 패키지 생성 -> 서블릿 Index를 생성

```
package _1;

import java.io.IOException;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/index")
public class Index extends HttpServlet {
    private static final long serialVersionUID = 1L;

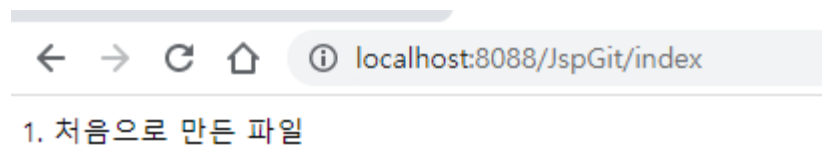
    public Index() {
        super();
    }
}
```

```

        protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
            throws ServletException, IOException {
            RequestDispatcher dispatcher =
            request.getRequestDispatcher("NewFile.jsp");
            dispatcher.forward(request, response);
        }
    }
}

```

(6) 브라우저에서 <http://localhost:8088/JspGit/index> 접속



===== [저장소와 공유하기] =====

4. 프로젝트 저장소와 공유하기

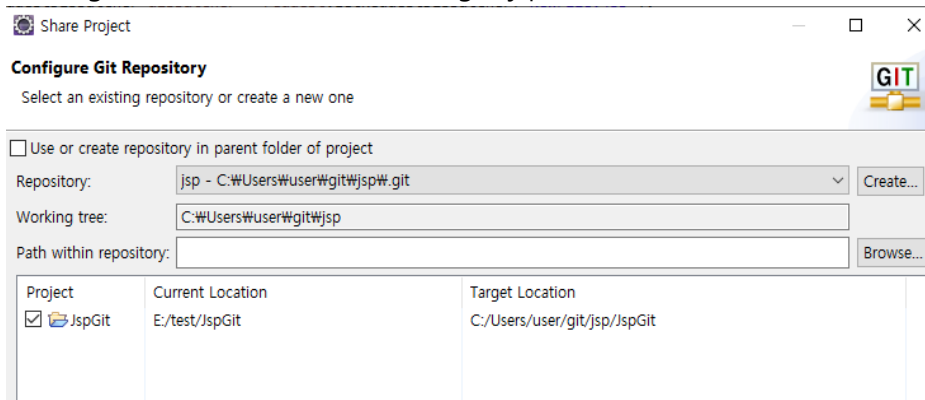
(1) 프로젝트에서 마우스 우 클릭으로 Team > Share Project

(2) Configure Git Repository에서

Repository 항목의 콤보 박스를 클릭 후 선택합니다.

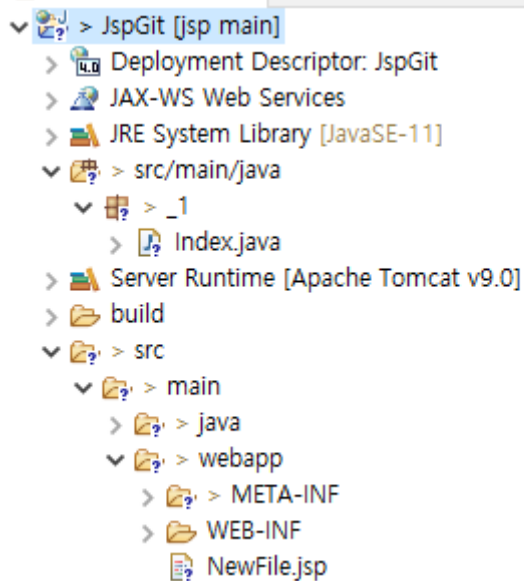
Repository : jsp-C:\Users\user\git\jsp\jsp.git

Working tree: C:\Users\user\git\jsp\jsp



(3) Finish

?는 새로 생성된 파일입니다.



(4) Git 프로젝트로 변경되었는지 확인합니다.

JspGit [jsp main]

===== [프로젝트 원격 저장소로 업로드] =====

5. 프로젝트 원격 저장소로 업로드

프로젝트 파일 중에서 내가 올리고자 하는 파일을 선택한 후 commit 합니다.

이때 선택한 파일들은 index 또는 staging area에 올려놓습니다.

참고) "index" 또는 "staging area"는 변경된 파일들이 일시적으로 저장되는 곳

(1) 프로젝트에서 마우스 우 클릭으로 Team > Commit

Unstaged Changes : 수정된 모든 파일들의 목록들이 표시되는 공간

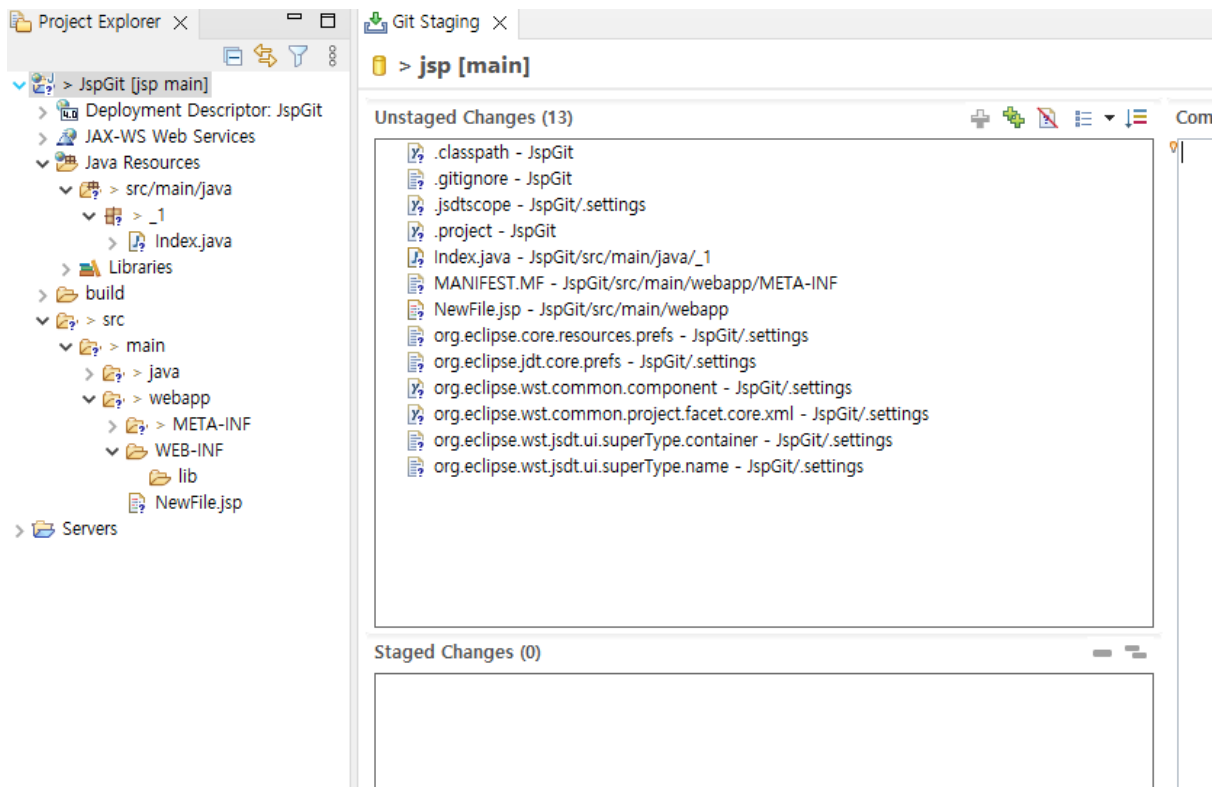
Staged Changes : commit시 만들어질 버전에 합류할 파일들만 모아놓은 공간

(2) Unstaged Changes

+ : Add Selected files

++ : Add all files

++ 클릭 -> 모든 파일 Staged Changes로 이동



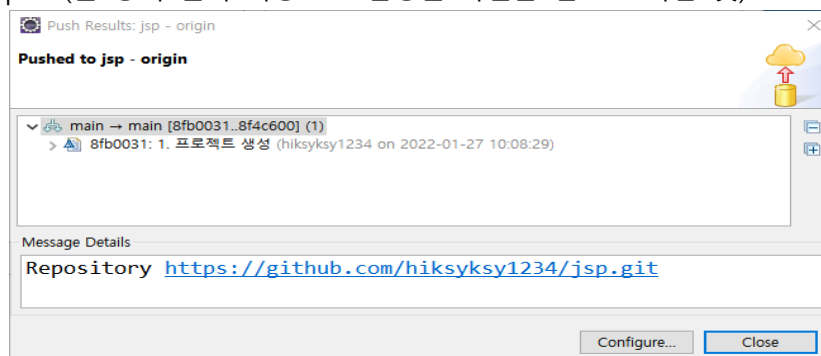
(3) Commit Message

1. 프로젝트 생성 (입력)

(4) Commit and Push

설명) Commit(파일 및 폴더의 추가/변경 사항을 저장소에 기록)

push(웹 상의 원격 저장소로 변경된 파일을 업로드 하는 것)



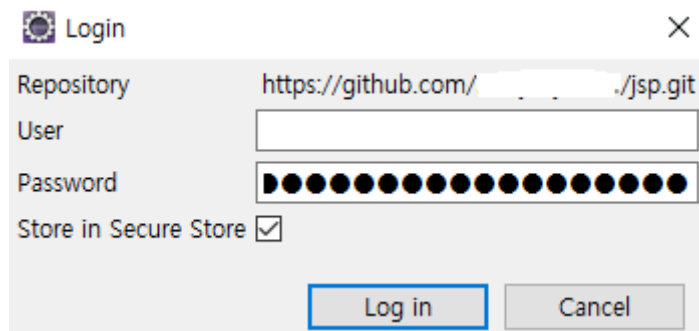
(5) Close

*** 인증 오류 나는 경우 ***

<http://github.com/username/jsp.git> not authorized 인 경우

토큰 확인합니다.

(6) Eclipse에서 Push HEAD 클릭



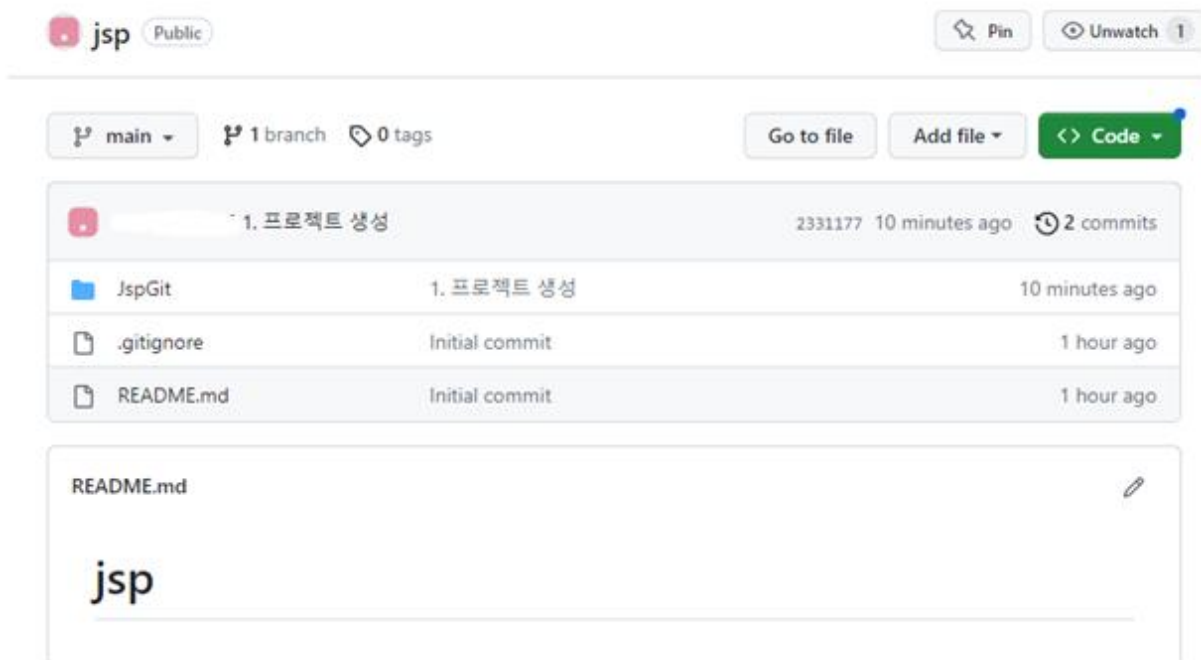
User에는 github username을 입력합니다.

Password에는 github 토큰을 넣어주세요

체크 박스 체크해 주세요



(7) <https://github.com/username/jsp> 접속 후 변경사항 확인



===== [원격 저장소에 있는 프로젝트를 Eclipse 프로젝트로 만들기] =====

팀원들은 팀장이 만들어 놓은 프로젝트를 아래와 같이 공유합니다.

6. 지역 저장소 생성

- 다른 워크스페이스에서 작업합니다.
- 서버 설치합니다.

(1) Eclipse에서 Window -> Perspective -> Open Perspective -> Other -> Git

(2) Clone a Git repository 클릭

-> Clone URI

-> URI에서 <https://github.com/username/jsp.git> 붙여 넣기

-> Next

-> Next

Directory : C:\Users\사용자계정\git\jsp2

Projects

import all existing Eclipse projects after clone finishes 선택

Clone Git Repository

Local Destination

Configure the local storage location for jsp.

Destination

Directory:

Initial branch:

☐ Clone submodules

Configuration

Remote name:

Projects

☒ Import all existing Eclipse projects after clone finishes

Working sets

☐ Add project to working sets

Working sets:

-> Finish

- JspGit [jsp2 main]
 - > Deployment Descriptor: JspGit
 - > JAX-WS Web Services
 - > Java Resources
 - > src/main/java
 - > _1
 - > Index.java
 - > Libraries
 - > build
 - > src
 - > main
 - > java
 - > _1
 - > webapp
 - > META-INF
 - > NewFile.jsp
 - > Servers

===== [지역(local) 저장소에서 새로운 branch(branch) 만들기] =====

7. branch 는 '나뉘어가지'란 뜻으로, 만들어 놓은 버전(main)의 복사본(branch)을 만들어 다른 방향으로 작업을 이어나가는 것입니다. 모든 팀원들은 각자 자신의 branch를 만들어 올리고 팀장은 이 branch들을 merge 합니다.

(1) 새로운 branch 생성

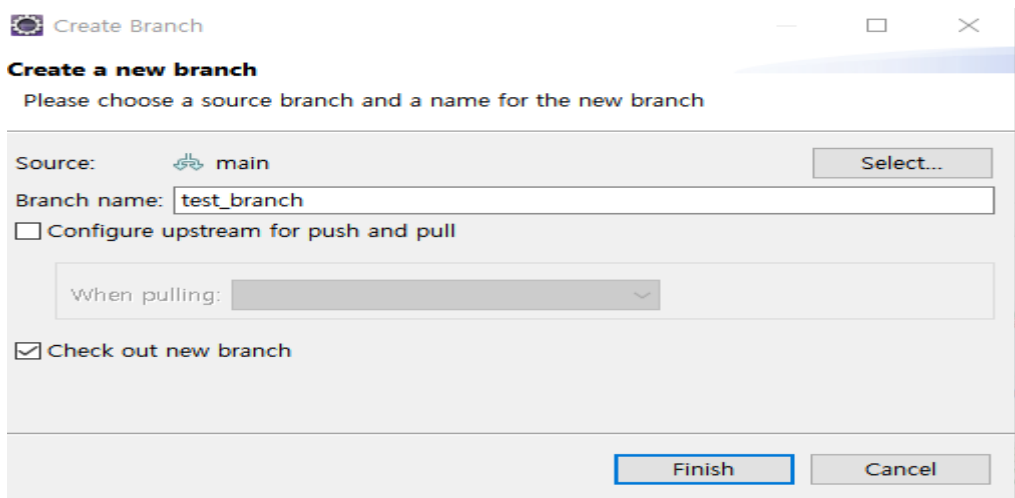
프로젝트 우 클릭 Team -> Switch To -> New Branch

=> Create a new branch에서

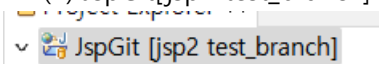
Branch name : test_branch

체크 - Check out new branch : 새로운 branch 로 이동해라

=> Finish



(2) JspGit[jsp2 test_branch]로 변경



===== [지역(local)의 새로운 branch에서 원격 저장소(test_branch)로 upload] =====

8 새로운 branch에서 파일 생성 후 지역 저장소 및 원격 저장소 upload 합니다.

(1) src 폴더에서 test 패키지 생성 후 Test.java라는 이름의 서블릿 생성

```
package test;

import java.io.IOException;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/index2")
public class Test extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Test() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.getWriter()
            .print("Served at: " + request.getContextPath() + "Test");
    }
}
```

Test.java 파일에 ?가 생깁니다.

(2) 프로젝트에서 마우스 우 클릭으로 Team -> Commit

Unstaged Changes에서 Test.java를 선택 후 Staged Changes에 add 합니다.

=> + 클릭

(3) Commit Message

2. test_branch에서 Test.java를 작성합니다.

(4) Commit and Push

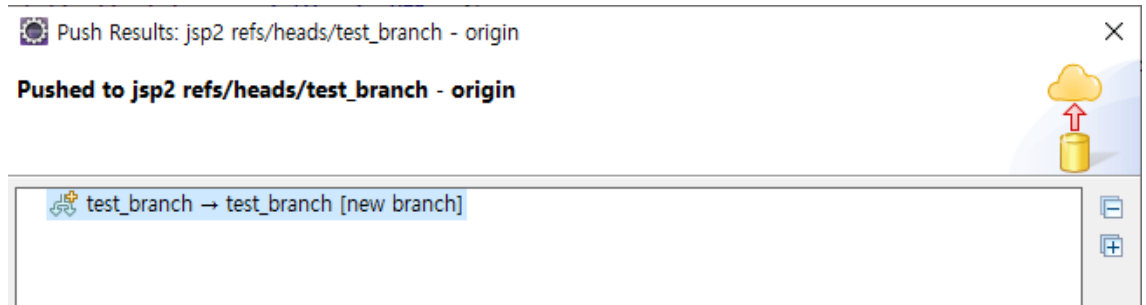
Remote : origin

Branch : test_branch

=> 원격의 test_branch로 Push합니다.

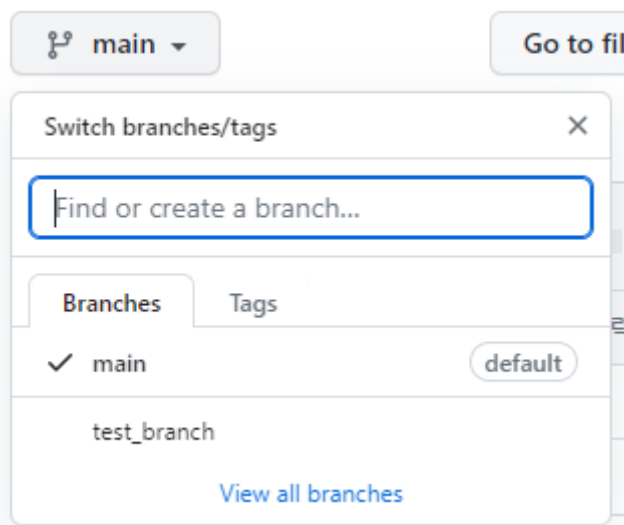
->Next

(5) Finish



(6) Close

(7) <https://github.com/username/jsp> 접속 후 변경사항 확인
=> branch는 test_branch를 선택합니다.



https://github.com/username/jsp/tree/test_branch



⇒ JspGit 선택 => src/main => java => test

https://github.com/username/jsp/tree/test_branch/JspGit/src/main/java/test

===== [origin/test_branch => origin/main 머지하기] =====

9. 원격 test_branch의 변경된 내용을 원격 main branch로 저장하기

(origin/test_branch => origin/main)

(1) 원격의 main branch 선택

<https://github.com/username/jsp/tree/main/JspGit/src/main/java/test>로 접속

404 오류 발생

main branch 에는 test 패키지가 없습니다.

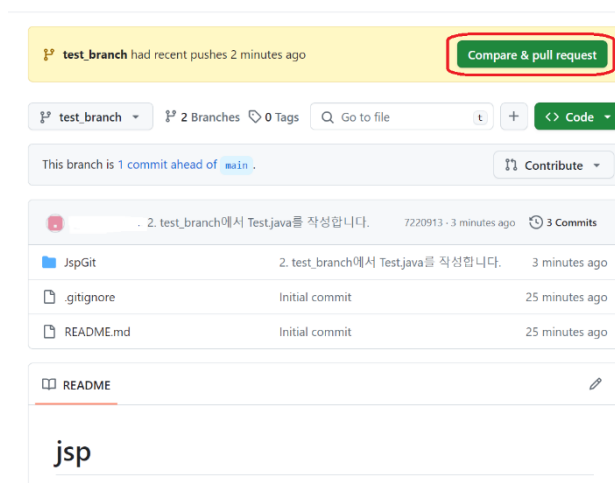
(2) <https://github.com/username/jsp>에서 Branch를 test_branch로 변경 또는

https://github.com/username/jsp/tree/test_branch로 이동

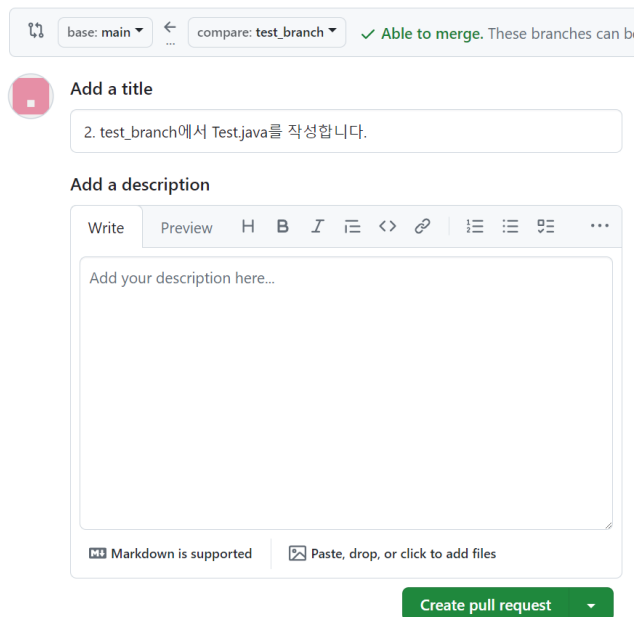
(3) Compare & pull request 버튼 클릭

용어) pull request(원격의 main branch로 머지 할 때 사용하는 단어) 버튼 클릭합니다
main branch에 저장하기 위해 main branch와 test_branch 를 비교하고 저장할 수 있도록 요청합니다.

충돌이 발생하지 않으면 Able to merge 메시지가 나타납니다.



(4) Open a pull request



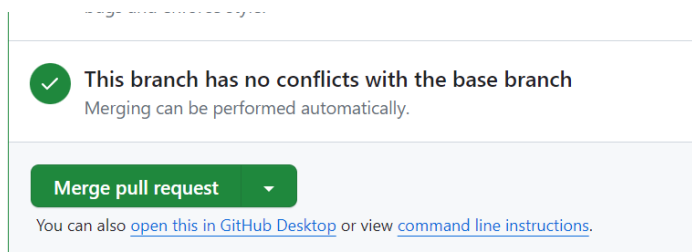
(4-1) Write 탭에서 아래와 같이 작성합니다.

2.test_branch에서 main branch로 request pull 합니다.

(4-2) Create pull request 클릭

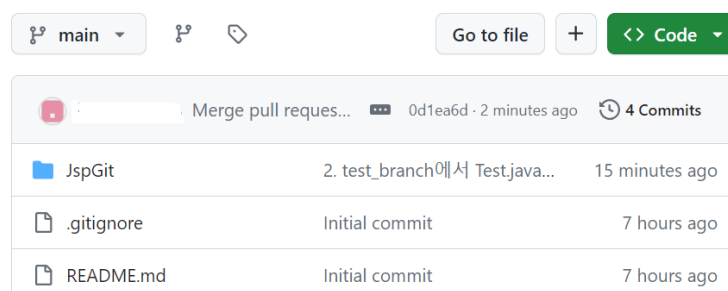
(5) <https://github.com/username/jsp/pull/1>

1) Merge pull request 클릭 (main branch와 merge 할 수 있는 명령)



2) Confirm merge 클릭(merge 하는 것을 확인합니다.)

(6) <https://github.com/username/jsp/tree/main/JspGit/src/main/java/test> 접속해 봅니다.



File	Commit Message	Time
JspGit	2. test_branch에서 Test.java...	15 minutes ago
.gitignore	Initial commit	7 hours ago
README.md	Initial commit	7 hours ago

===== [origin/main의 내용이 바뀐 경우 로컬의 test_branch에서 적용하기] =====

10. origin/main 내용 바꾸기 (다른 branch들의 내용을 병합한 경우라고 생각해 봅니다)

(1) github에서 main branch 선택합니다. (<https://github.com/username/jsp>)

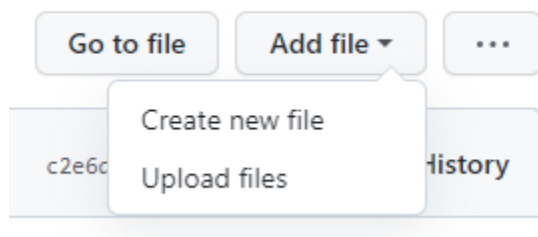
(2) JspGit 클릭 (<https://github.com/username/jsp/tree/main/JspGit>)

(3) src/main 클릭

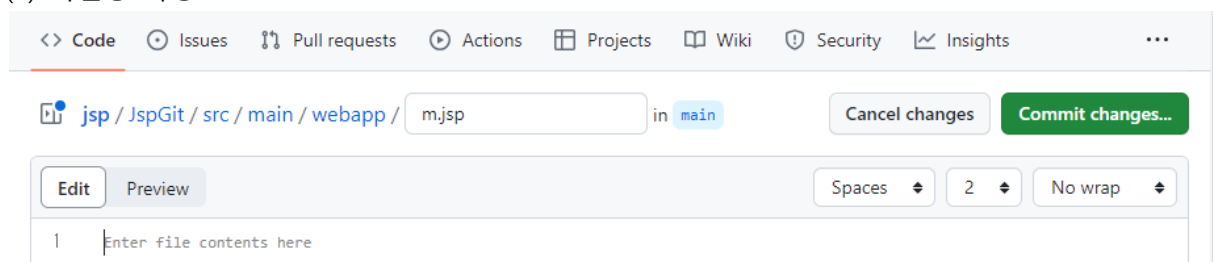
(4) webapp 클릭

(<https://github.com/username/jsp/tree/main/JspGit/src/main/webapp>)

(4) Add file => Create new file 클릭



(5) 파일명 작성



```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
```

```
<h1>m.jsp입니다.</h1>
</body>
</html>
```

(6) Commit changes... 클릭

Cancel changes

Commit changes...

(7) 아래와 같이 입력 후 Commit changes 클릭

4. main branch에서 m.jsp 생성

Commit changes ×

Commit message

Extended description

4. main branch에서 m.jsp 생성

☒ Commit directly to the main branch
☐ Create a new branch for this commit and start a pull request
[Learn more about pull requests](#)

Cancel Commit changes

m.jsp 생성되었는지 확인

Create m.jsp 33c3f58 · now History		
Name	Last commit message	Last commit date
..		
META-INF	1. 프로젝트 생성 (입력)	14 hours ago
NewFile.jsp	1. 프로젝트 생성 (입력)	14 hours ago
m.jsp	Create m.jsp	now

===== [Eclipse에서 test_branch로 10번에서 작성한 m.jsp 다운받기] =====
원격의 main branch가 변경되면 반드시 local의 test_branch는 반드시 pull을 합니다.

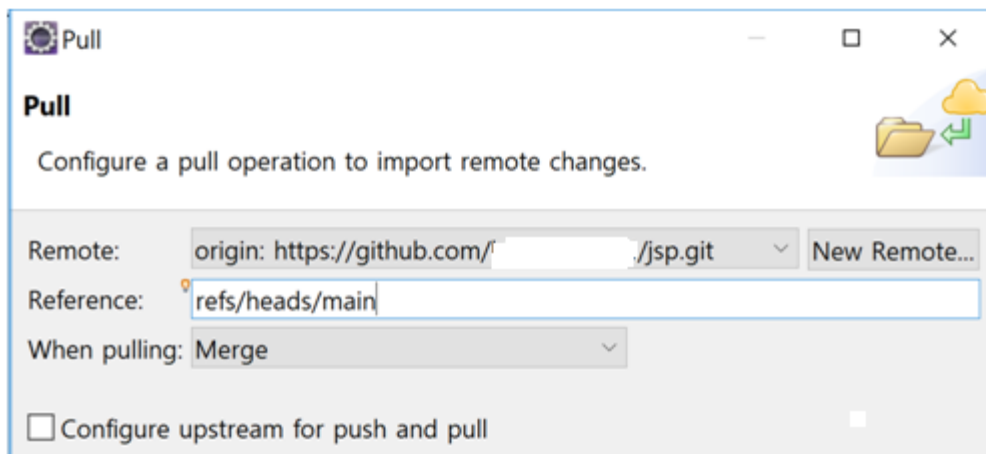
11. Eclipse의 JspGit[jsp2 test_branch] 상태에서 작업합니다.

프로젝트에서 마우스 우 클릭

=> Team -> pull,, (pull점점점 - 원격의 특정 branch를 변경할 수 있어요)


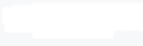



=> Reference에서 영역 설정 후 Ctrl + Space => main 선택

Reference: refs/heads/main 선택



➔ Finish -> Close

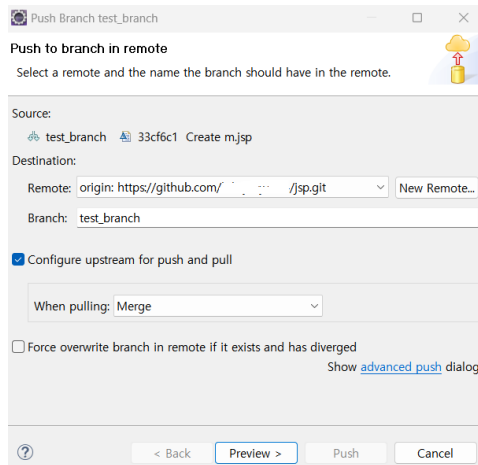
<https://github.com/username/jsp/commits/main> => 5 commits

	 Create m.jsp ...	6f3c121 3 minutes ago	🕒 5 commits
	JspGit Create m.jsp		3 minutes ago
	.gitignore Initial commit		26 minutes ago
	README.md Initial commit		26 minutes ago

===== [로컬의 test_branch => 원격 test_branch] =====

12. Eclipse 프로젝트 우 클릭

(1) Team => Push Branch "test_branch" => Preview => Push



(2) 원격의 test_branch에 접속해서 m.jsp가 생성되었는지 확인

https://github.com/username/jsp/tree/test_branch/JspGit/webapp

===== [로컬 test_branch에서 m.jsp 변경] =====

13. Eclipse에서 작업합니다.

(1) m.jsp <h2>local-1</h2> 추가

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <h2>local-1</h2>
</body>
</html>
```

(2) Commit and Push

1) 프로젝트에서 마우스 우 클릭으로 Team -> Commit

Unstaged Changes에서 m.jsp를 Staged Changes에 add 합니다.

=> + 클릭

2) Commit Message에서

5. 로컬 test_branch에서 m.jsp 내용 수정합니다.

3) Commit and Push

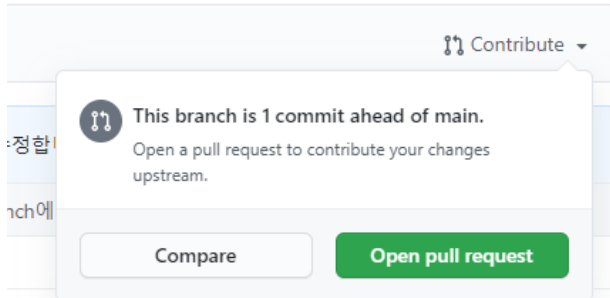
(3) https://github.com/username/jsp/blob/test_branch/JspGit/src/main/webapp/m.jsp
바뀐 내용 확인

=== [원격 test_branch => 원격 main branch로 Merge] ===

14. 원격 test_branch => 원격 main으로 Merge

(1) https://github.com/username/jsp/tree/test_branch

(2) Contribute 클릭 -> Open pull request 클릭



(3)

Write 탭

Add a description

6. 로컬 test_branch에서 m.jsp 내용 수정했습니다. (입력)

-> Create pull request 클릭

(4) Merge pull request

(5) Confirm merge

(6) <https://github.com/username/jsp/tree/main/JspGit/webapp/m.jsp> 확인

===== [로컬 충돌] =====
15. 로컬에서의 충돌 발생 예(반드시 commit and push 하기 전에 pull을 해야 합니다.)

충돌 발생 상황)

- ① 원격 test_branch와 원격 main branch, 로컬 test_branch 내용 일치
- ② 원격 main branch 수정
- ③ 로컬 test_branch 수정 -> Commit and Push
- ④ 로컬 test_branch에서 pull.. => 충돌발생

(예) 팀원1의 내용을 원격 main branch로 merge 후 다른 팀원2의 내용을 merge 했습니다.

팀원1은 변경된 main branch를 pull하지 않고

로컬에서 작업 후 test_branch로 commit and push 진행했습니다.

팀원1은 원격의 main branch가 수정된 것을 알고 pull을 했습니다.

충돌 발생합니다.

(1) 원격 test_branch와 원격 main, 로컬 test_branch 내용 일치 확인

(2) 원격 main branch 수정 ()

(<https://github.com/username/jsp/edit/main/JspGit/webapp/m.jsp>)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <h2>local-2</h2>
</body>
</html>
```

(3) Commit changes.. 클릭 -> Commit changes 클릭

(4) Eclipse의 로컬 저장소 test_branch의 m.jsp 변경하기

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
```

```

    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <h2>local-3</h2>
</body>
</html>

```

(5) Eclipse의 로컬 저장소 test_branch 수정 후

=> m.jsp를 Staged Changes에 올립니다.

=> Commit Message : 1-> 3으로 변경

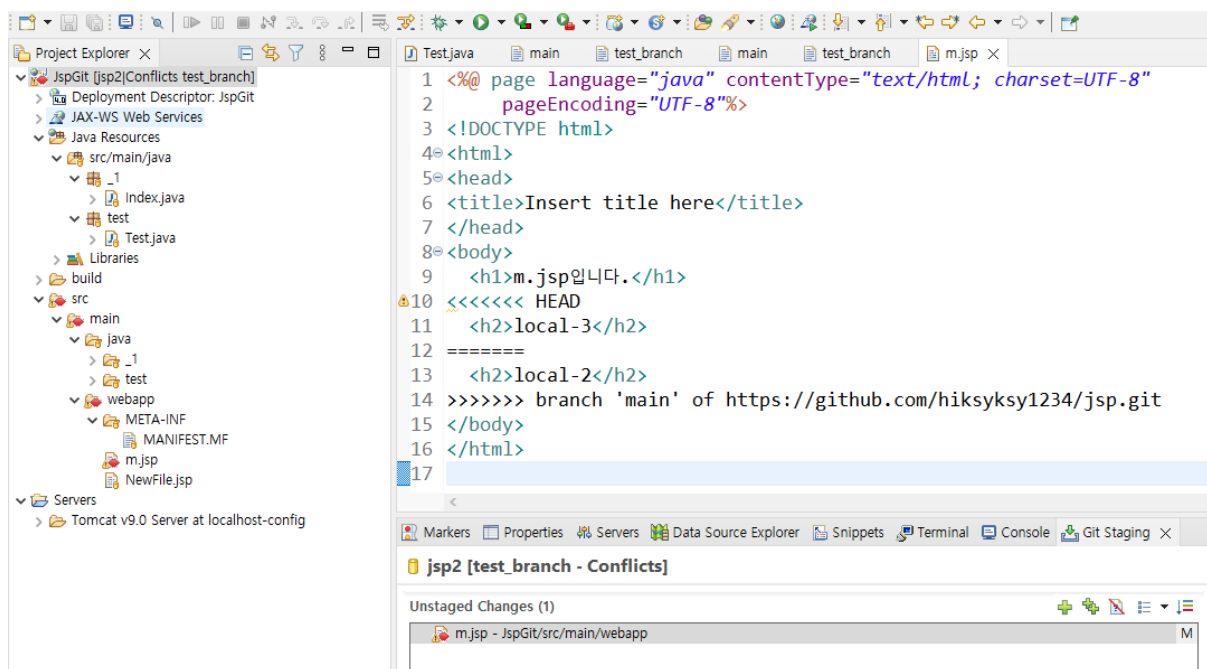
=> Commit and push

=> close

(6) 로컬 test_branch

프로젝트 우 클릭 -> Team -> pull.. => Reference : refs/heads/main => Finish

=> 충돌발생



```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
<=====
    <h2>local-2</h2>
>>>>>> branch 'main' of https://github.com/username/jsp.git
</body>
</html>

```

***** 현재 작업 중인 branch를 가리키는 HEAD *****

(7) 충돌된 파일 수정

```

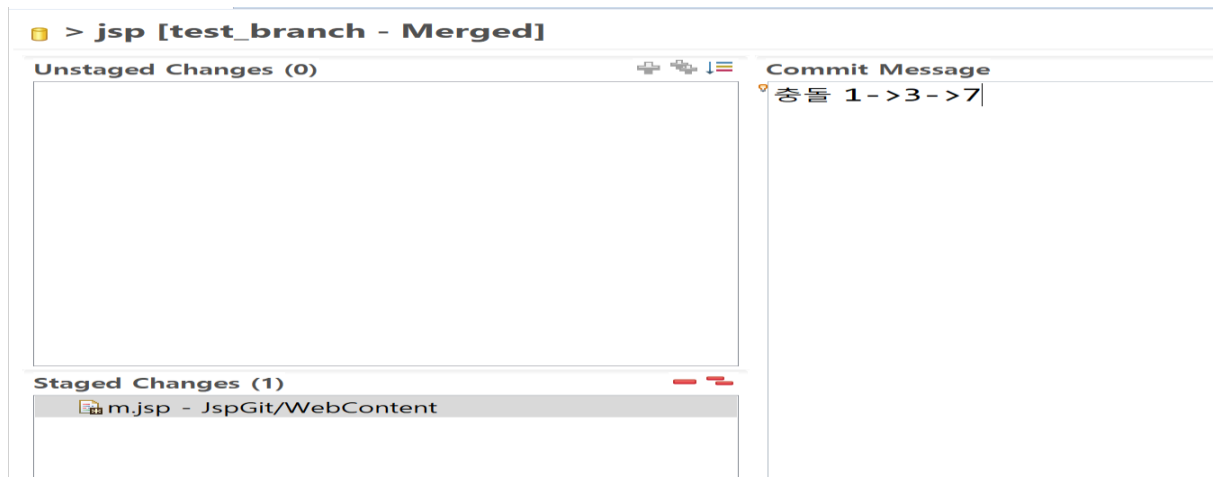
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <h2>local-7</h2>
</body>
</html>

```

(8) 다이아몬드 아이콘(충돌표시) m.jsp를

커밋 메시지 : 충돌 1->3->7

Commit and Push



➔ Close

===== [원격 충돌 경우] =====

16. 원격 충돌 경우

(1) 원격 test_branch는 로컬에서 변경 사항 적용됨

```
<body>
  <h1>m.jsp입니다.</h1>
  <h2>local-7</h2>
</body>
```

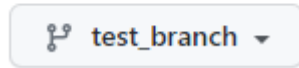
(2) 원격 test_branch의 변경을 원격에 적용하지 않은 상태에서 원격 main branch 수정

```
<h2>local-2</h2>
-> <h2>local-10</h2>
```

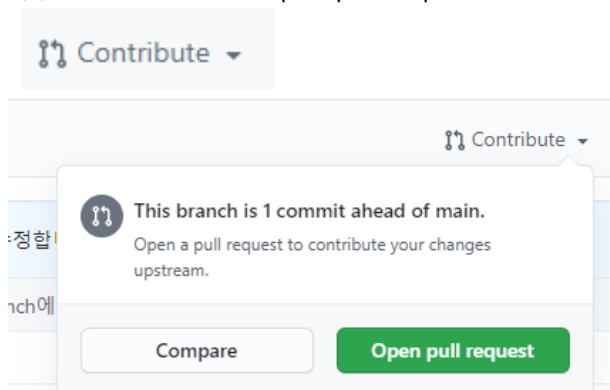
(3) Commit changes.. -> Commit changes

(4) test_branch로 이동

https://github.com/username/jsp/tree/test_branch



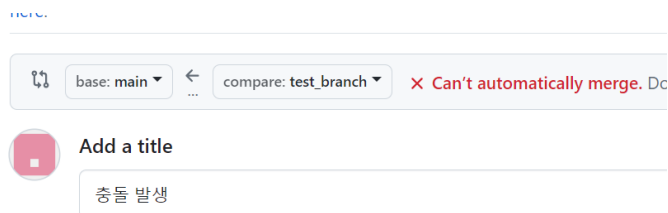
(5) Contribute => Open pull request 클릭



(6) 원격 test_branch -> 원격 main

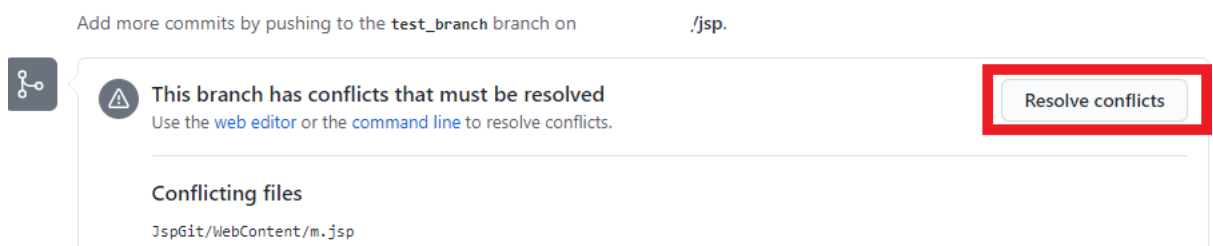
=> Open pull request 클릭하는 경우 충돌 발생

Add a title : 충돌 발생



(7) Create pull request 클릭

(8) Resolve conflicts 클릭



(9) 충돌 내용

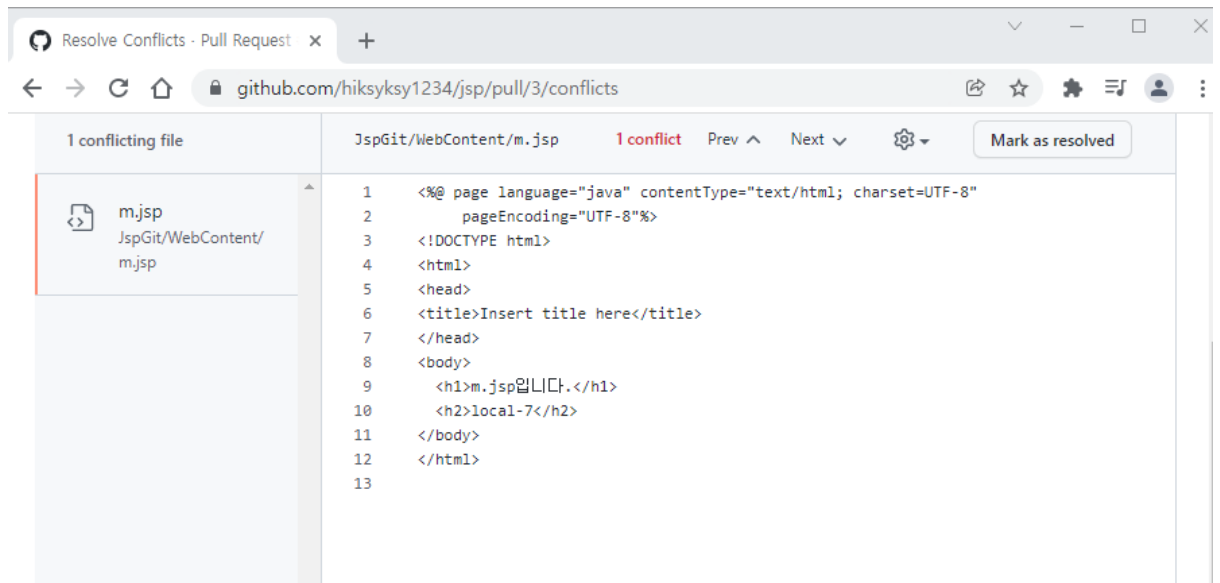
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
```

```
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <<<<<< test_branch
    <h2>local-7</h2>
    ===== main
    <h2>local-10</h2>
    >>>>>> main
</body>
</html>
```

서로 의견 조율 후 수정합니다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <h1>m.jsp입니다.</h1>
    <h2>local-7</h2>
</body>
</html>
```

(10) Mark as resolved 클릭 (충돌 부분 해결했다고 알려줍니다.)



(11) Commit merge 클릭

(12) Merge pull request 클릭

(13) Confirm merge 클릭

(14) 수정 내용 처럼 변경되었는지 아래에서 확인

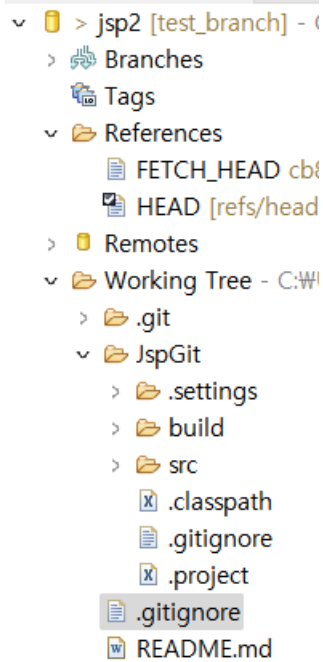
<https://github.com/username/jsp/blob/main/JspGit/src/main/webapp/m.jsp>

https://github.com/username/jsp/blob/test_branch/JspGit/src/main/webapp/m.jsp

===== [.gitignore] =====

.gitignore 파일은 Git 저장소에서 추적하지 않을 파일이나 디렉토리를 지정하는 데 사용되는 설정 파일입니다.

예) 확장자 txt인 파일은 commit 대상에서 제외하고자 하는 경우



```
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see
http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
replay_pid*

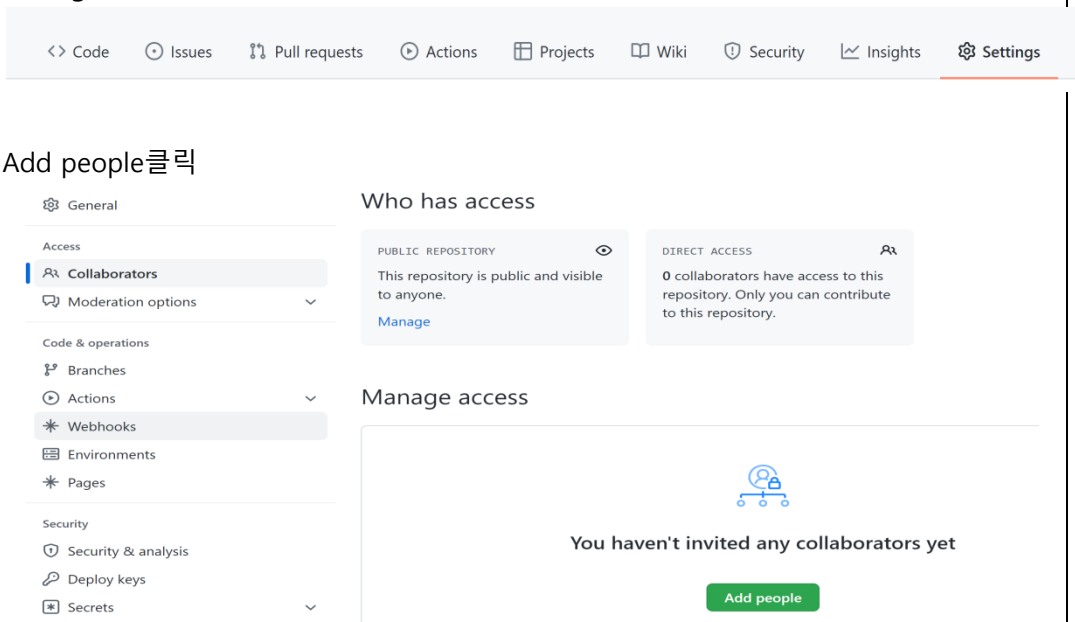
*.txt
```


===== [협 업] =====

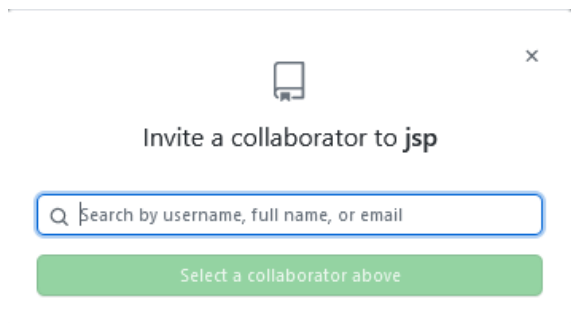
1. Master가 팀원들을 초대합니다.
2. 팀원들은 초대를 수락합니다.
3. Master가 만들어 놓은 저장소를 clone 합니다.
4. 자신의 아이디와 비밀번호를 통해 push와 pull을 할 수 있습니다.

1. 팀원 추가 (Master)

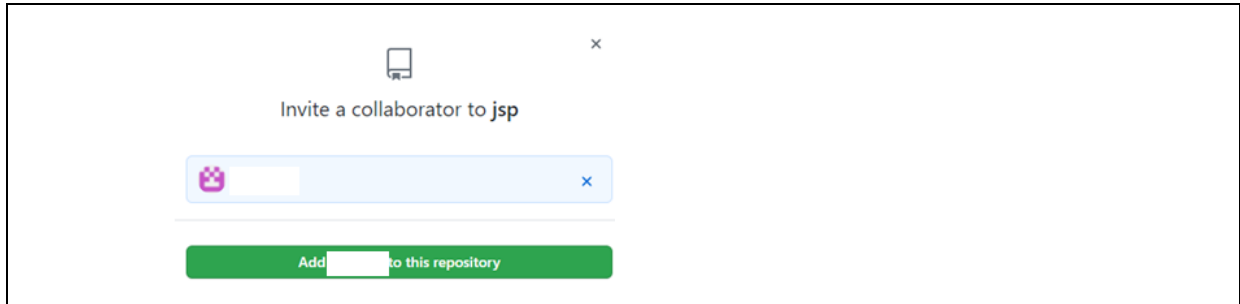
(1) Settings > Collaborators



(2) user name 또는 이메일을 통해 팀원 검색 후

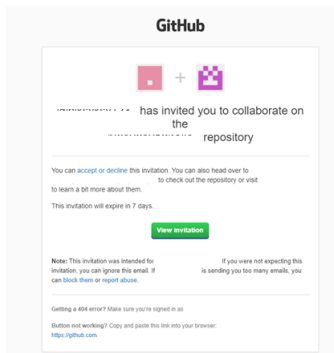


(3) Add 선택한 username to this repository 버튼 선택



2. 초대 수락 (팀원)

- (1) Github에 팀원 로그인(로그인 되어 있어야 4번 Accept Invitation 확인 할 수 있어요)
- (2) 가입 시 작성한 이메일을 확인
- (3) View invitation 클릭



- (4) Accept invitation 클릭 (반드시 팀원 로그인 되어 있어야 합니다.)

- Master는 팀원들이 추가되었는지 확인합니다.

Settings -> Manage access

저장소 clone 하기

1. 팀장은 저장소의 주소를 팀원들에서 알려줍니다.
(예) <https://github.com/팀장username/jsp.git>)
2. 팀원들은 6번 내용을 참조합니다.
 - (1) Eclipse에서 Window > Perspective > Open Perspective > Other -> Git
 - (2) Clone a Git repository 클릭
URI : 팀장의 원격 주소 붙여 넣기

Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☒ Store in Secure Store

-> Next

-> Next

Directory : C:\Users\사용자계정\git\jsp

Projects import all existing Eclipse projects after clone finishes 선택

-> Finish

1. 팀장이 프로젝트 틀을 만듭니다.
2. 팀원들은 clone합니다.
3. 실행 잘 되지는 확인합니다.
4. 각자의 .gitignore에 다음 사이트를 참고 후 넣어주세요

*<https://defacto-standard.tistory.com/252>

/classes/
.classpath
.project
.settings/

1. 각 팀원들은 오후 특정 시간(5시)에 github에 commit and push합니다. 팀원들은 대기 합니다.
2. 팀장은 모든 팀원들의 push를 확인 후 merge 합니다.
3. 팀장은 merge 후 팀원들에게 merge가 되었다는 것을 알리고 모든 팀원들은 변경된 내용이 적용될 수 있도록 pull 하도록 합니다.