

버전 관리 툴

Contents

□ 버전 관리 툴이란?

□ CVS

- CVS란?
- CVS를 왜 사용하는가?
- CVS 사용하기

□ SVN

- SVN란?
 - SVN 설치하기
 - SVN 사용하기
 - RabbitVCS
-

프로젝트 관리 툴이란?

□ 버전 관리(revision control)

- 소스 코드 관리라고도 한다.
- 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다.
- 소프트웨어 개발에서 팀 단위로 개발 중인 소스 관리에 사용
- 문서의 변경 사항들에 숫자나 문자로 이뤄진 “버전”을 부여하여 구분한다.
- “버전” 통해서 시간적으로 변경 사항과 그 변경 사항을 작성한 작업자를 추적할 수 있다.

버전 관리를 왜 사용하는가?

□ 공동 프로젝트 관리

- 버전 관리를 사용하면 대규모의 프로젝트에서 각 모듈 개발자가 자신의 버전을 유지하면서 전체 프로젝트에 참여할 수 있도록 할 수 있음

□ 프로젝트 백업

- 버전관리는 중간에 서버가 있어서, 프로젝트 데이터의 저장소 역할을 하여 실수로 자신의 프로젝트가 날아간다고 해도 염려할 필요가 없음

□ 데이터 동기화

- 한 명이 여러 공간에서 작업을 하는 경우, 한 곳에서 작업 후 서버에 등록하고, 다른 곳에서 다시 작업을 시작할 때 서버에 등록된 최신의 데이터를 받아와서 작업을 함으로써 데이터 동기화가 가능

버전의 의미

□ 버전의 의미

1. 어떤 소프트웨어가 몇 번 개정되었는지를 나타내는 번호. 기능이 보완되거나 추가될 때 버전을 올려 나감
2. 한 소프트웨어를 서로 다른 시스템 환경에서 사용할 수 있도록 각각 제작된 프로그램을 이르는 말

ex) MS-DOS 6.2

→ 버전의 수치는 개정의 정도를 의미한다. 맨 처음 소프트웨어가 공식적으로 발표되면 보통 1.0이란 버전이 붙게 되고, 그 기능이 개선되면서 점점 버전 숫자가 커지게 되는데, 소수점 위의 숫자는 큰 개정을 의미하고, 소수점 아래의 숫자는 약간의 개정을 의미한다.

버전의 의미

버전 1.1 main.c

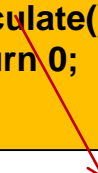
```
main.c

#include "diary.h"
Int main()
{
    memo();
    calendar();
    return 0;
}
```

버전 1.2 main.c

```
main.c

#include "diary.h"
Int main()
{
    memo();
    calculate();
    return 0;
}
```

 **calendar();** 삭제되거나
calculate();라는 새로운
소스가 추가됨.

- ◆ 위와 같이 처음 1.1버전의 **main.c**의 파일이 수정되어 1.2버전의 **main.c**파일이 되어 버전업된다.

버전 관리에서 사용하는 용어들

□ 체크 아웃(Check Out)

- 저장소(Repository)에서 파일을 가져온다.

□ 체크 인(Check In, Commit)

- 체크아웃(Check Out)한 파일의 수정이 끝난 경우 저장소에 새로운 버전으로 갱신 하는 일이다.
- 이때 이전에 갱신된 것이 있는 경우 충돌(Conflict)을 알려주며 diff 도구를 이용해 수정하고 commit하는 과정을 거치게 된다.

□ 가져오기(Import)

- 버전 관리가 되고 있지 않은 로컬 디렉토리의 파일을 처음으로 저장소에 복사한다.

□ 저장소(Repository)

- 파일의 현재 버전과 변경 이력 정보를 저장하는 저장소
-

Contents

☐ 버전 관리 툴이란?

☐ CVS

- CVS란?
- CVS 설치하기
- CVS 사용하기

☐ SVN

- SVN란?
 - SVN 설치하기
 - SVN 사용하기
-

CVS란?

- ❑ Concurrent Version System 의 줄임말
- ❑ 공동 버전 시스템
- ❑ "공동으로 진행하는 프로젝트의 버전 관리 시스템"



CVS 설치하기

□ CVS 서버 설치하기

- `sudo apt-get install cvs`
- `sudo apt-get install xinetd`
- `sudo apt-get install apache2-utils`

□ CVS 유저 추가

```
#sudo adduser cvs
```

- ## □ 계정을 생성 한 후, cvs 계정 홈폴더에 프로젝트 파일들을 보관하는 저장소(Repository)를 생성

```
#sudo mkdir /home/cvs/repository
```

저장소 디렉토리 생성 및 초기화

```
jyn@jyn-sslslab: ~  
File Edit View Search Terminal Help  
jyn@jyn-sslslab:~$ sudo adduser cvs  
[sudo] password for jyn:  
Adding user `cvs' ...  
Adding new group `cvs' (1001) ...  
Adding new user `cvs' (1001) with group `cvs' ...  
Creating home directory `/home/cvs' ...  
Copying files from `/etc/skel' ...  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Changing the user information for cvs  
Enter the new value, or press ENTER for the default  
    Full Name []: cvs  
    Room Number []:  
    Work Phone []:  
    Home Phone []:  
    Other []:  
Is the information correct? [Y/n] y  
jyn@jyn-sslslab:~$ sudo mkdir /home/cvs/repository  
jyn@jyn-sslslab:~$
```

CVS 유저 생성 및 권한 설정

- 저장소 디렉토리의 **CVS**에서 사용하는 제어파일들이 들어있는 디렉토리를 생성하기 위해 초기화

- -d 옵션 : cvs 저장소의 위치

```
#sudo cvs -d /home/cvs/repository init
```

- 초기화한 후, 시스템에 **CVS**유저를 등록하고 저장소 디렉토리에 대한 **CVS**유저의 접근 권한을 설정

```
#sudo chown -R cvs.cvs /home/cvs/repository  
#sudo chgrp -R cvs /home/cvs/repository
```

- -R 옵션 : 경로와 그 하위까지 모두 바꿈

CVS사용자 등록

- **CVS** 서버에 인증을 거쳐 프로젝트를 파일을 받아올 때 인증에 유효한 사용자를 생성
- **CVS**사용자는 시스템 사용자와 같이 실제 시스템 사용 권한을 가지고 있는 대상은 아니며 단지 **CVS**서버만이 인지하는 사용자임
- **CVS** 사용자를 등록하는 방법은 저장소 디렉토리의 **CVSROOT** 디렉토리 내에 **passwd**파일을 생성하면 됨

CVS사용자 등록

- 사용자 등록을 하기 위해 **passwd** 파일 생성

```
#cd /home/cvs/repository/CVSROOT  
#sudo htpasswd -c passwd [등록할 사용자 명]  
암호 입력
```

- **passwd** 파일 수정

```
#sudo vi passwd
```



```
cvs:JzA129Z7vWqpg
```

 →

```
cvs:JzA129Z7vWqpg:cvs
```

첫 번째는 사용자 ID, 두 번째 JzA129Z7vWqpg는 암호화된 사용자의 암호
세 번째에 추가하는 cvs 는 서버에 파일을 기록할 때 파일의 권한을 가지는 계정

□ 기본 명령어 형식

```
#cvs [cvs 옵션] 명령 [명령 옵션과 인자]
```

□ 모든 CVS 명령은 저장소의 위치를 알아야 수행됨

- 환경변수 CVSROOT 의 값으로 저장소의 위치 지정
- 또는 명령어 마다 -d 옵션 사용

CVS 명령의 종류

전체 명령	동의어	명령 실행
login		cvس 서버에 로그인
logout		.cvspass 파일에서 저장소 제거
import		프로젝트 파일 등록
checkout	co, get	프로젝트 파일 가져오기
commit	ci	프로젝트 파일 수정 후 cvs 서버에 반영
update	up	CVS 서버의 최신 버전을 작업 디렉토리에 반영
add	new	파일 또는 디렉토리 추가
remove	m, delete	파일 삭제

CVS 명령의 종류(계속)

전체 명령	동의어	명령 실행
diff		파일 버전에 따른 차이점 비교
log	rlog	파일 로그 보기
annotate	ann	행별 정보 출력(작성 날짜, 작성자 등)
status		파일 상태 보기
history		각종 히스토리 보기
tag		프로젝트 파일에 태깅하기
rtag		저장소 디렉토리에 태깅하기
release		모듈 release

☐ 버전 관리 툴이란?

☐ CVS

- CVS란?
- CVS 설치하기
- CVS 사용하기

☐ SVN

- SVN란?
 - SVN Repository 생성
 - SVN 설치하기
 - SVN 사용하기
-

SVN이란?

☐ Subversion

- CVS의 단점을 개선해서 CVS를 대체하기 위해서 개발
- 장점
 - ☐ 원자적 commit을 지원
 - ☐ 빠른 속도
 - ☐ 기본적으로 웹 인터페이스
 - ☐ 디렉토리 버전, 파일 이름 변경 지원
 - ☐ 관리자 툴 지원
 - ☐ 강력한 사용자 접근 권한 관리

SVN Repository 생성

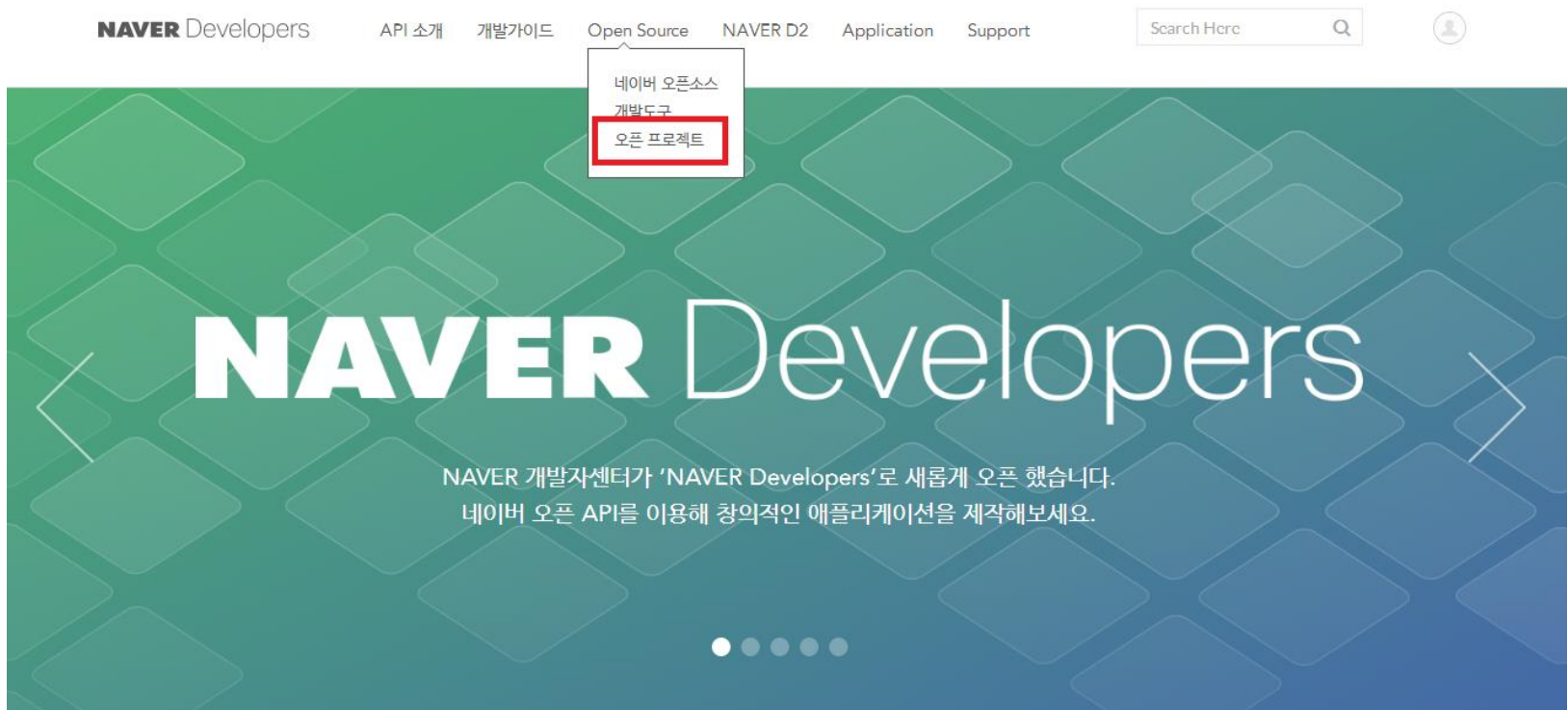
□ SVN 저장소 생성하기

- 네이버 개발자 센터 (<http://developers.naver.com/>)
- 네이버 계정으로 로그인하기



SVN Repository 생성

- SVN 저장소 생성하기
 - 나의 프로젝트 등록하기
 - 상단 메뉴의 'Open Source' > '오픈 프로젝트' 선택



SVN Repository 생성

□ SVN 저장소 생성하기

- 좌측 메뉴의 '프로젝트 등록' 선택
- 프로젝트 이름과 아이디를 생성
- 아이디는 저장소에 접근할 때 URL 주소로 사용

NAVER Developers

API 소개

개발가이드

Open Source

NAVER D2

Application

Notice

프로젝트 검색



프로젝트 정보

대시보드 >

회원정보 >

자동알림 관리 >

프로젝트 등록 >

프로젝트 등록

아래의 설명을 주의깊게 읽어보시고, 완전하고 자세한 내용을 제공해 주십시오. 프로젝트 공개설명은 필수입니다.

프로젝트 이름

Linux Application Test

프로젝트 아이디

rilyaus-gd

프로젝트 공개 설명

Linux Application SVN Test

코드관리시스템

☐ Mercurial ☐ Git ☒ SVN



고도한자원*을 사용하거나 타인의 지적재산권, 저작권등을 침해할 경우 이용이 제한될 수 있음에 동의합니다.

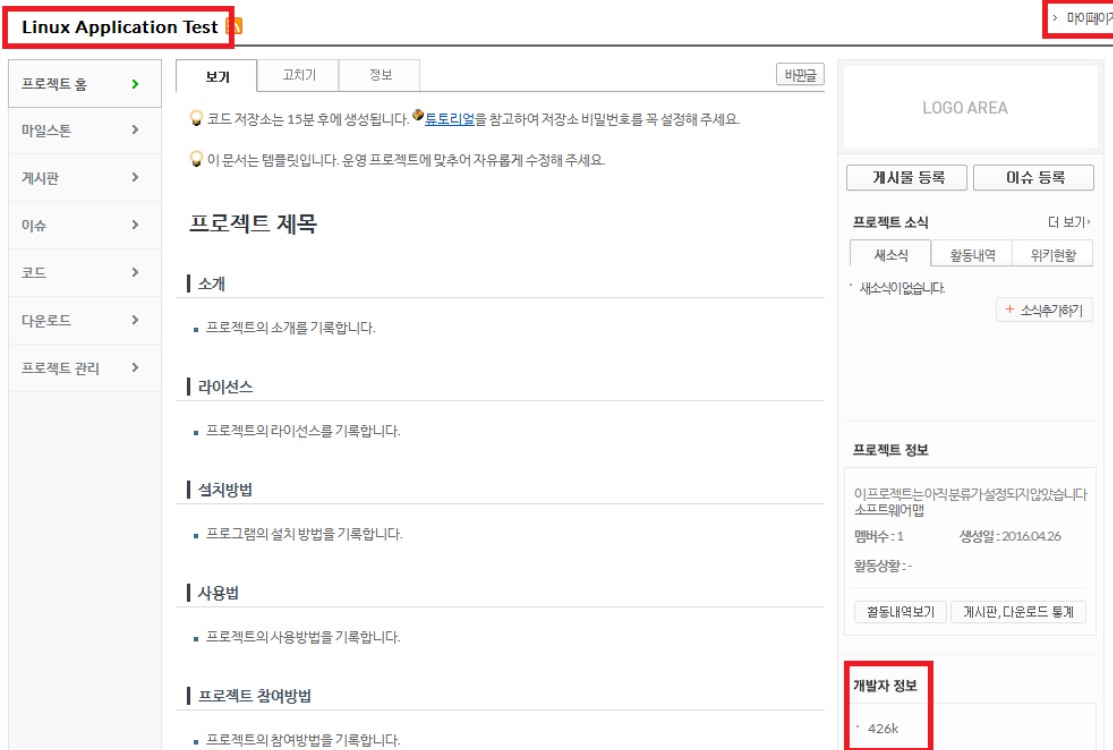
*저장소당 2GB, 단일 파일 100MB 미만

확인

SVN Repository 생성

□ SVN 저장소 생성하기

- 프로젝트 정보를 모두 입력하면 다음의 창이 나타납니다.
- 개발자 정보: 본인의 네이버 로그인 아이디
- 좌측 상단: 프로젝트 생성시 설정한 프로젝트 명



Linux Application Test

로그인 회원가입

프로젝트 소식

로그인 회원가입

프로젝트 정보

이 프로젝트는 아직 분류가 설정되지 않았습니다
소프트웨어 템플릿

멤버수: 1 생성일: 2016.04.26

활동상황: -

활동내역 보기 게시물, 다운로드 통계

개발자 정보

426k

마이페이지 선택

SVN Repository 생성

□ SVN 저장소 생성하기

- 마이페이지를 선택하면 다음의 창이 나타납니다.
- 좌측 메뉴의 '회원정보' 선택

| 마이페이지

대시보드 >

회원정보 >

자동알림 관리 >

프로젝트 등록 >

| 내가 참여한 프로젝트

프로젝트	게시물	멤버수	등급	상태변경
Linux Application Test	0	1	관리자	

| 할당받은 이슈

게시판분류	요약	번호	중요도
제출하신 문제점 추적 항목이 없습니다.			

SVN Repository 생성

□ SVN 저장소 생성하기

- 별명의 경우 기본값으로 본인의 아이디가 설정되어 있음
- ‘코드저장소 비밀번호 설정’ 선택

| 마이페이지

대시보드 >	사용자아이디	426k	
회원정보 >	별명	<input type="text" value="426k"/>	<input type="button" value="코드저장소 비밀번호 설정"/>
자동알림 관리 >	홈페이지	<input type="text" value="http://blog.naver.com/426k"/>	
프로젝트 등록 >	메일받기	<input checked="" type="checkbox"/> 네이버 개발자센터에서 보내는 알림 메일을 받습니다	

SVN Repository 생성

□ SVN 저장소 생성하기

- SVN Repository에 접근하기 위해 필요한 비밀번호
- 비밀번호 설정 후 '비밀번호 갱신' 선택

| 비밀번호 변경

대시보드 >	설정된 코드저장소 비밀번호는 15분 후에 적용됩니다	
회원정보 >	새로운 비밀번호 (6글자 이상)	<input type="password"/>
	새로운 비밀번호 (한번더)	<input type="password"/>
자동알림 관리 >	<input type="button" value="비밀번호 갱신"/>	
프로젝트 등록 >		

SVN Repository 생성

□ SVN 저장소 멤버 추가하기

- 마이페이지의 대시보드 화면에서 내가 참여한 프로젝트 선택

마이페이지

대시보드 >

회원정보 >

자동알림 관리 >

프로젝트 등록 >

내가 참여한 프로젝트

프로젝트	게시물	멤버수	등급	상태변경
Linux Application Test	0	1	관리자	


할당받은 이슈

게시판분류	요약	번호	중요도
제출하신 문제점 추적 항목이 없습니다.			

SVN Repository 생성

□ SVN 저장소 멤버 추가하기

- 좌측 메뉴의 '프로젝트 관리' > '멤버 관리' 선택
- 사용자 아이디 입력 후 우측의 '추가' 버튼 선택
- 추가 완료 후 '확인' 버튼 선택

Linux Application Test 

> 마이페이지

프로젝트 이름	닉네임	사용자아이디	등급	탈퇴/추가
프로젝트 홈	426k	426k	관리	탈퇴
마일스톤	사용자 추가	<input type="text"/>	관리	추가
게시판	확인			
이슈				
코드				
다운로드				
프로젝트 관리				
공개정보				
멤버관리				
등급관리				

SVN 설치하기

☐ Subversion 설치

- subversion, libapache2-svn 패키지 설치
 - ☐ `sudo apt-get install subversion libapache2-svn`
 - ☐ 시냅틱 패키지 관리자에서 선택

SVN 서버에 프로젝트 파일 등록

- 아래의 파일들을 ~/Diary에 생성

diary.h

```
#include<stdio.h>
int memo();
int calendar();
```

memo.c

```
#include "diary.h"
int memo()
{
    printf("function memo.\n");
    return 0;
}
```

main.c

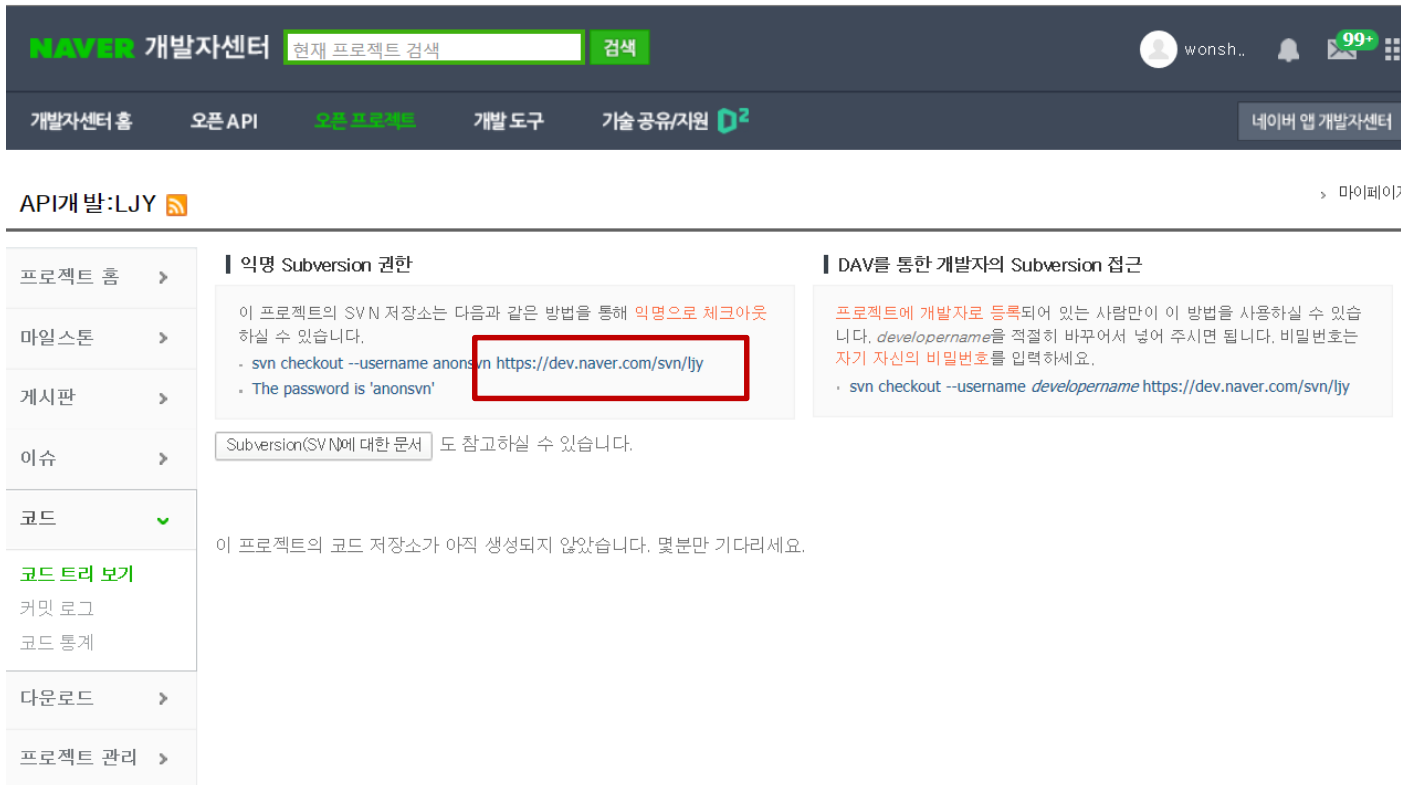
```
#include "diary.h"
Int main()
{
    memo();
    calendar();
    return 0;
}
```

calendar.c

```
#include "diary.h"
Int calendar()
{
    printf("function calendar.\n");
    return 0;
}
```

□ Subversion

- 본인이 설정한 Project_name 확인하기
 - 마이페이지 → 대시보드 → 프로젝트 선택 → 코드
 - 상자 안의 주소를 명령어 전달 인자로 입력



The screenshot shows the Naver Developer Center (NAVER 개발자센터) interface. The top navigation bar includes links for '개발자센터 홈', '오픈 API', '오픈 프로젝트', '개발 도구', and '기술 공유/자원'. The main content area is titled 'API개발:LJY' and features a sidebar with navigation links: '프로젝트 홈', '마일스톤', '게시판', '이슈', '코드' (selected), '코드 트리 보기', '커밋 로그', '코드 통계', '다운로드', and '프로젝트 관리'.

The main content area is divided into two columns. The left column, titled '익명 Subversion 권한', contains the following text:

이 프로젝트의 SVN 저장소는 다음과 같은 방법을 통해 **익명으로 체크아웃**하실 수 있습니다.

- `svn checkout --username anonsvn https://dev.naver.com/svn/ljy`
- The password is 'anonsvn'

The URL `https://dev.naver.com/svn/ljy` is highlighted with a red box. Below this, a link to 'Subversion(SVN)에 대한 문서' is provided.

The right column, titled 'DAV를 통한 개발자의 Subversion 접근', contains the following text:

프로젝트에 **개발자로** 등록되어 있는 사람만이 이 방법을 이용하실 수 있습니다. `developername`을 적절히 바꾸어서 넣어 주시면 됩니다. 비밀번호는 **자기 자신의 비밀번호**를 입력하세요.

- `svn checkout --username developername https://dev.naver.com/svn/ljy`

At the bottom of the main content area, a message states: '이 프로젝트의 코드 저장소가 아직 생성되지 않았습니다. 몇분만 기다리세요.'

□ Subversion

■ 명령어

- `svn import` [디렉토리 이름] [<https://dev.naver.com/svn/프로젝트이름>]
- 프로젝트 서버 설정 후 디렉토리 안에 있는 파일을 처음 등록

```
jnoo@jnoo-virtual-machine:~/Diary$ svn import /home/jnoo/Diary https://dev.naver.com/svn/kwucsos
추가      /home/jnoo/Diary/main.c
추가      /home/jnoo/Diary/calendar.c
추가      /home/jnoo/Diary/svn-commit.2.tmp
추가      /home/jnoo/Diary/svn-commit.tmp
추가      /home/jnoo/Diary/memo.c
추가      /home/jnoo/Diary/diary.h
```

```
GNU nano 2.2.4                                파일 : svn-commit.3.tmp
[LOG]import Diary Project
--이 줄 이하는 자동으로 제거됩니다--
A      .
```

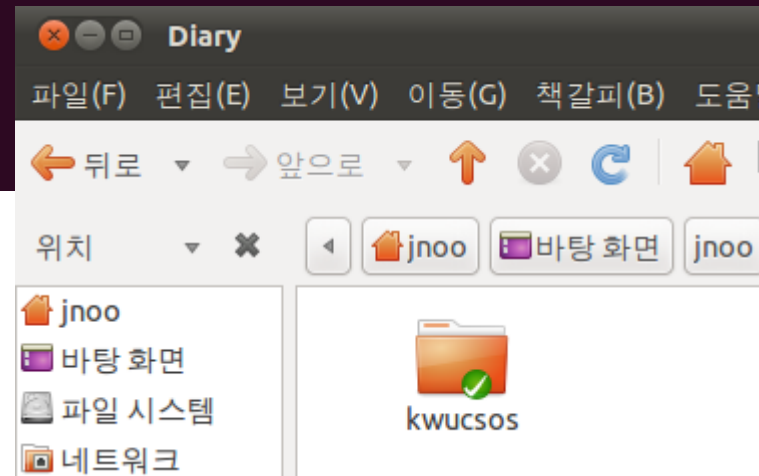

□ Subversion

■ 명령어

- `svn checkout --username [ID] [https://dev.naver.com/svn/프로젝트 이름]`

- 프로젝트 저장소에 있는 파일을 받아오는 명령

```
jnoo@jnoo-virtual-machine:~/svn/Diary$ svn checkout --username parammiru2 https://dev.naver.com/svn/kwucsos
A      kwucsos/trunk
A      kwucsos/trunk/main.c
A      kwucsos/branches
A      kwucsos/main.c
A      kwucsos/calendar.c
A      kwucsos/svn-commit.2.tmp
A      kwucsos/svn-commit.tmp
A      kwucsos/diary.h
A      kwucsos/memo.c
A      kwucsos/tags
체크아웃된 리비전 3.
```



□ Subversion

■ 명령어

□ svn add [파일 이름]

- 새로 파일을 생성했을 때, 관리하고자 하는 대상으로 추가시키는 명령어로, 파일 생성 후 이 명령으로 추가해준 후에 **commit** 을 해야 서버에 등록이 됨

```
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$ ls
addfile.c  branches  calendar.c  diary.h  main.c  memo.c  svn-commit.2.tmp  svn-commit.tmp
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$ svn add addfile.c
A      addfile.c
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$
```

□ Subversion

■ 명령어

□ svn ci

- commit 명령
- 현재 폴더에 있는 파일들을 comment를 달아 서버로 업로드
- -m 옵션으로 바로 comment 작성 가능
 - ex) `svn ci -m "COMMENT"`

```
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$ svn ci
로그 메시지가 변경되지 않았거나 지정되지 않았습니다
취소 (A), 계속 (C), 수정 (E):
c
추가          addfile.c
파일 데이터 전송 중 .
커밋된 리비전 4.
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$
```

□ Subversion

■ 명령어

□ svn up

- update
- 내 폴더에 있는 파일들을 현재 서버에 있는 최신 버전으로 업데이트 하는 명령
- 항상 저장소에 적용된 최신의 파일들을 현재 작업 디렉토리에 적용 시켜야 함.
- 저장소와 작업 디렉토리의 파일 버전 등의 차이로 문제가 발생 할 수 있기 때문

```
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$  
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$ svn up  
리비전 4.  
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$
```

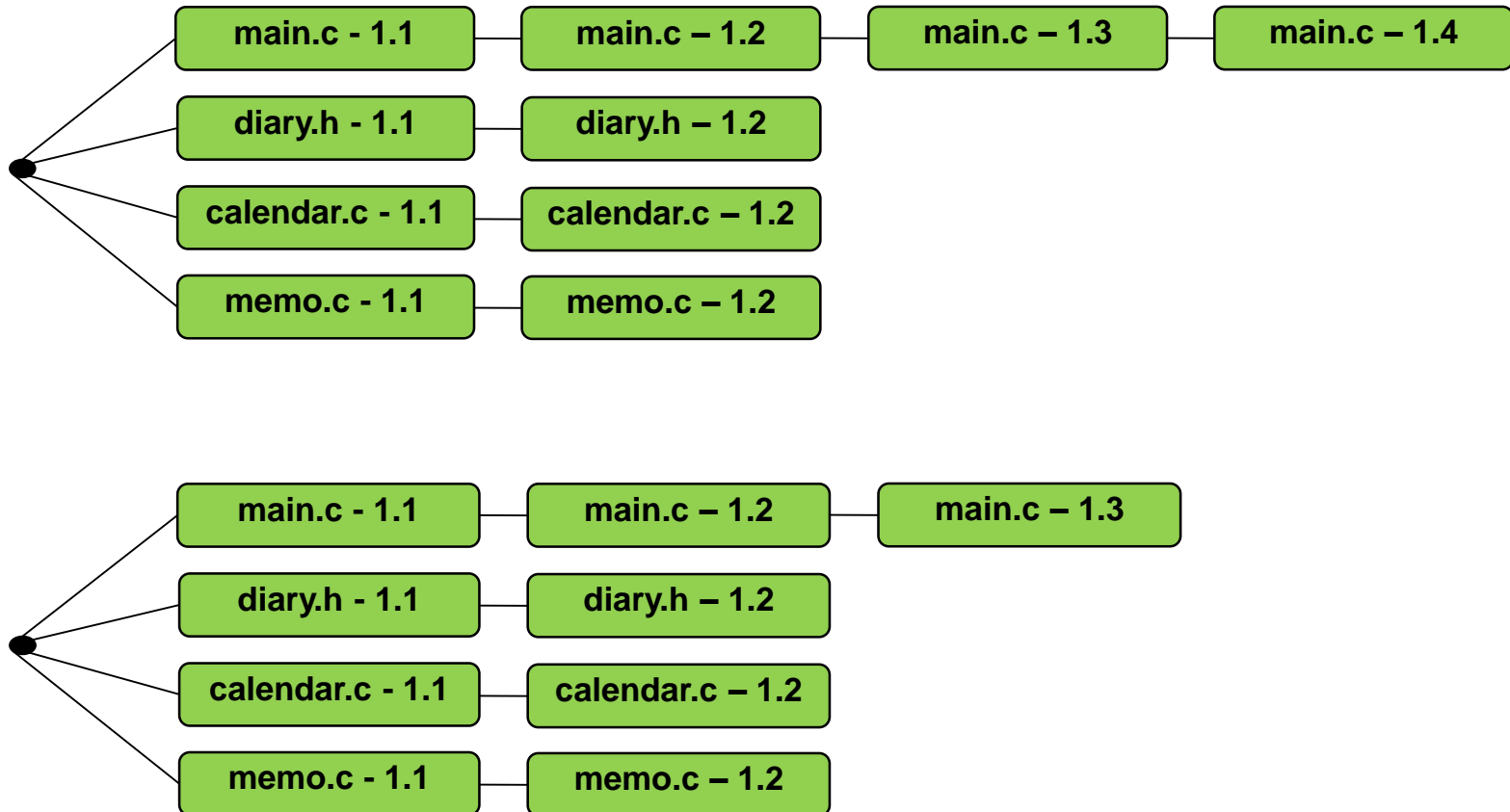
소스 충돌 해결

- 만약 두 사람이 같은 파일을 수정하고 **commit**했는데 수정한 부분이 다르다면 **svn**이 알아서 관리해줌

- 만약 같은 파일의 같은 부분을 수정할 경우 소스 충돌이 발생함. 이 경우 **svn**는 어떤 부분에서 소스충돌이 발생했는지에 대해서 개발자에게 알리고 개발자가 결정하게 함

- 충돌이 발생한 소스를 수정한 후 다시 **svn commit** 명령으로 반영할 경우 충돌 상황은 종료

소스 충돌 해결



소스 충돌 상태에서의 노드 상태

소스 충돌 해결

```
jnoo@jnoo-virtual-machine: ~/svn/Diary/kwucsos
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
.c
jnoo@jnoo-virtual-machine:~/svn/Diary/kwucsos$ cat main.c
#include "diary.h"
int main()
{
    memo(); //JNOO
    calendar(); //JNOO
    return 0;
}
```

```
root@jnoo-virtual-machine: /home/jnoo/svn/Diary/kwucsos
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
calendar.c memo.c tags
root@jnoo-virtual-machine:/home/jnoo/svn/Diary/kwucsos# cat main.c
#include "diary.h"
int main()
{
    memo(); //ROOT
    calendar(); //ROOT
    return 0;
}
```

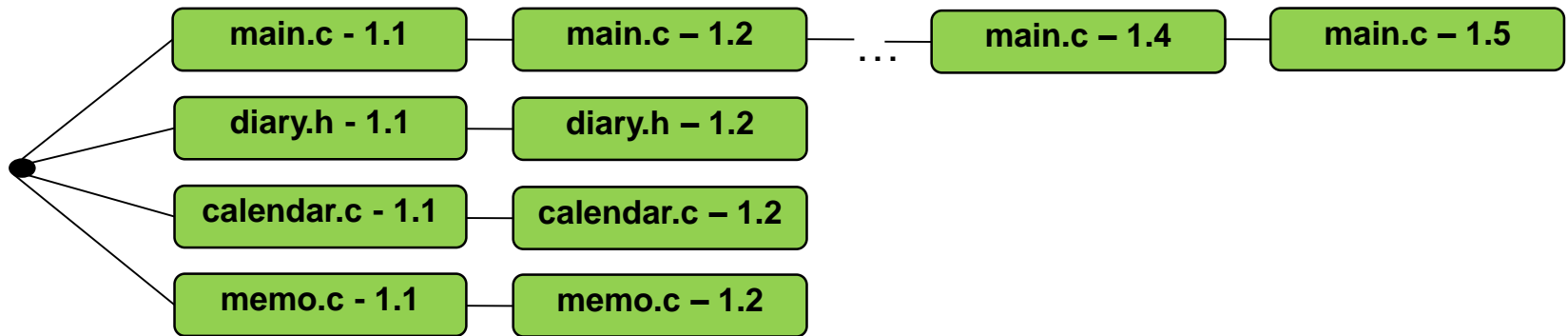
소스 충돌 해결

```
root@jnoo-virtual-machine:~/svn/kwucsos# svn up
충돌 발견됨: 'main.c'.
선택: (p) 다음에 시도, (df) diff-full, (e) edit,
      (mc) mine-conflict, (tc) theirs-conflict,
      (s) 모든 옵션: p
C      main.c
업데이트된 리비전 7.
충돌 상황 요약:
  본문 충돌 개수: 1
```

```
root@jnoo-virtual-machine: ~/svn/kwucsos
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
root@jnoo-virtual-machine:~/svn/kwucsos# cat main.c
#include "diary.h"
int main()
{
<<<<<<< .mine
      memo(); //ROOT
      calendar(); //ROOT
=====
      memo(); //JNOO
      calendar(); //JNOO
>>>>>>> .r7
      return 0;
}
```

충돌이 일어난 파일을 열어보면
다음과 같이 충돌이 일어난 부분에
대한 설명이 있다.

소스 충돌 해결



소스 충돌 해결을 위해 **main.c** 파일을 수정 후 다시 업데이트한 경우의 노드 상태

현재 시점의 소스 태깅하기

- **Trunk** 디렉토리에 프로젝트 파일 복사하기
- **Tag** 만들기
 - Trunk 와 branches의 특정시간을 가리키는 포인터
 - 주로 프로젝트의 릴리즈 주기에 맞추어서, 각 주기 코드를 보존하기 위해 사용
 - Tag는 보통 해당 릴리즈 주기에서 가장 안정화 됐다고 판단하는 시점에 적용
 - <사용방법>

```
$ svn copy calendar.c diary.h main.c memo.c trunk
```

```
$ svn copy trunk tags/0.1 (버전을 입력)
```

```
$ svn ci
```

현재 시점의 소스 태깅하기

□ Tag 만들기

```
addfile.c  calendar.c  kwucsos  nemo.c  new my- proj  svn-commit.tmp  trunk
branches  diary.h      main.c    nyproj  svn-commit.2.tmp  tags
conso@conso-virtual-machine: ~/Diary/kwucsos$ svn copy trunk tags/0.1
A      tags/0.1
conso@conso-virtual-machine: ~/Diary/kwucsos$ svn ci
```

```
conso@conso-virtual-machine: ~/Diary/kwucsos$ svn ci
```

```
Log message unchanged or not specified
```

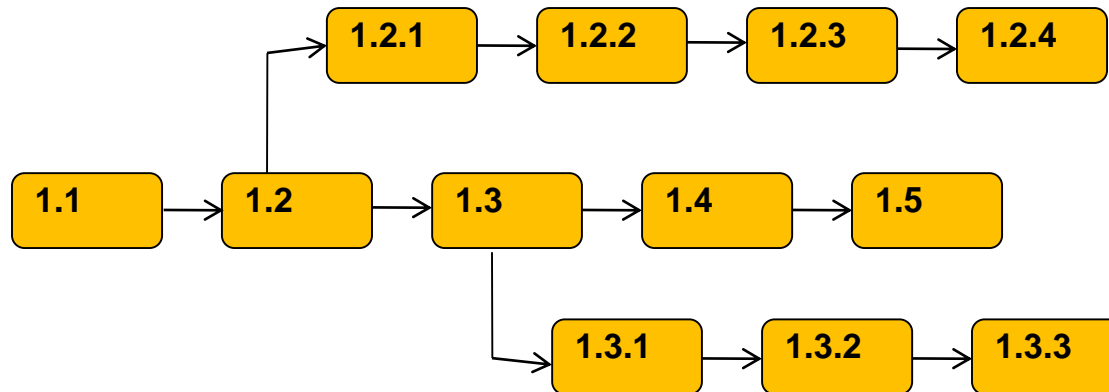
```
(a) abort, (c) continue, (e) edit:
```

```
c
```

```
Adding      tags/0.1
```

브랜치하기

- 브랜치란 메인 버전의 흐름에서 가지를 친다는 의미
- 브랜치를 왜 하는가?
 - 기존의 메인 버전에 새로운 기능을 추가하거나 혹은 색다른 버전으로 개발하고자 할 때 메인 버전의 소스를 보존하기 위해서 브랜치를 함



□ Subversion

■ Branch 만들기

- 1. svn copy [https://dev.naver.com/svn/프로젝트 이름 /trunk] [branches/새 branch 폴더 이름]
- 2. svn commit

```
conso@conso-virtual-machine: ~/Directory/kwucsos$ svn copy myproj branches/new-my-proj
A      branches/new-my-proj
conso@conso-virtual-machine: ~/Directory/kwucsos$ svn ci branches/

Log message unchanged or not specified
(a) abort, (c) continue, (e) edit:
c
Adding      branches/new-my-proj

Committed revision 9.
conso@conso-virtual-machine: ~/Directory/kwucsos$
```

■ Branch 파일 다운로드

- svn checkout [https://dev.naver.com/svn/프로젝트 이름 /branches] [저장할 디렉토리 이름]

브랜치한 소스 머징하기

- 새로운 기능을 추가하여 테스트하기 위해서 브랜치를 했다면 테스트가 올바르게 끝나게 되는 경우 메인 버전과 머징하게 됨

- **Subversion**

- 소스 Merge 하기

- `svn merge [option] [branch 디렉토리 주소] [trunk 디렉토리 주소]`

- ** 디렉토리는 둘다 URL을 의미함

- `--dry-run` 옵션 : 실제 명령은 수행하지 않고 시도만 함

```
conso@conso-virtual-machine: ~/Directory/kwucsos/new-ny-proj $ svn diff https://dev.naver.com/svn/kwucsos/branches/new-ny-proj \https://dev.naver.com/svn/kwucsos/trunk
conso@conso-virtual-machine: ~/Directory/kwucsos/new-ny-proj $
```

어떤 소스가 변경되었는지 비교하기 위해 실행

```
conso@conso-virtual-machine: ~/Directory/kwucsos/new-ny-proj $
conso@conso-virtual-machine: ~/Directory/kwucsos/new-ny-proj $ svn merge --dry-run https://dev.naver.com/svn/kwucsos/branches/new-ny-proj
https://dev.naver.com/svn/kwucsos/trunk
conso@conso-virtual-machine: ~/Directory/kwucsos/new-ny-proj $
```

Branches 의 소스를 **trunk**로 머징하는 명령어
(`--dry-run` 옵션은 어떤 소스가 바뀌었는지 표시를 해줌)

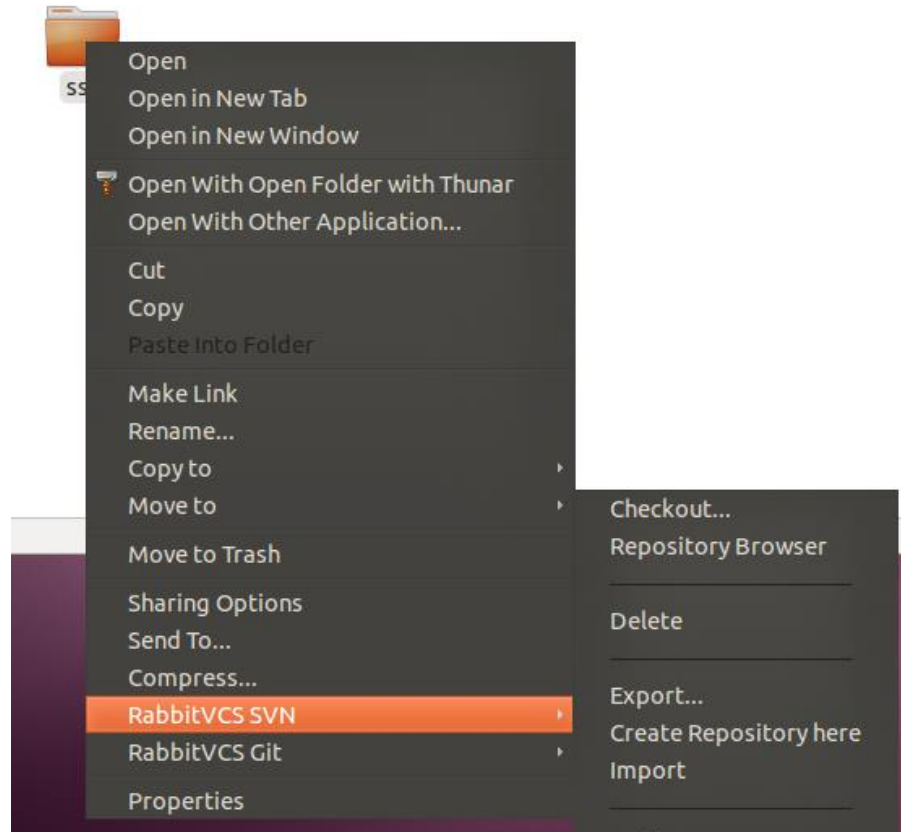
☐ Subversion

☒ RabbitVCS

- ☐ Linux 용 SVN GUI 프로그램
- ☐ <http://wiki.rabbitvcs.org/wiki/install/ubuntu>
- ☐ 터미널에서 다음 명령어로 설치
 - ☒ `sudo add-apt-repository ppa:rabbitvcs/ppa`
 - ☒ `sudo apt-get update`
 - ☒ `sudo apt-get install rabbitvcs-core rabbitvcs-nautilus3`
- ☐ 설치 후 재부팅하면 적용된다.

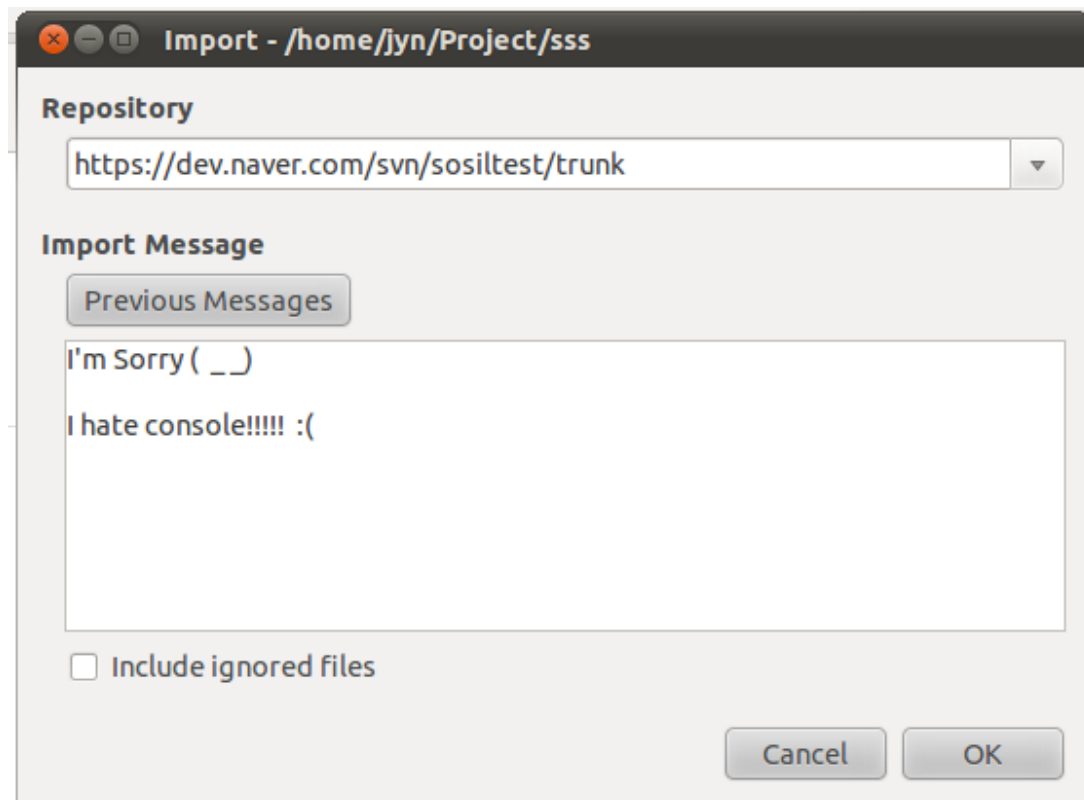
□ Import

- sss 폴더 안에 작성한 파일이 있고, 이것을 import 시킬 때,
sss폴더 오른쪽 클릭 > RabbitVCS SVN > import



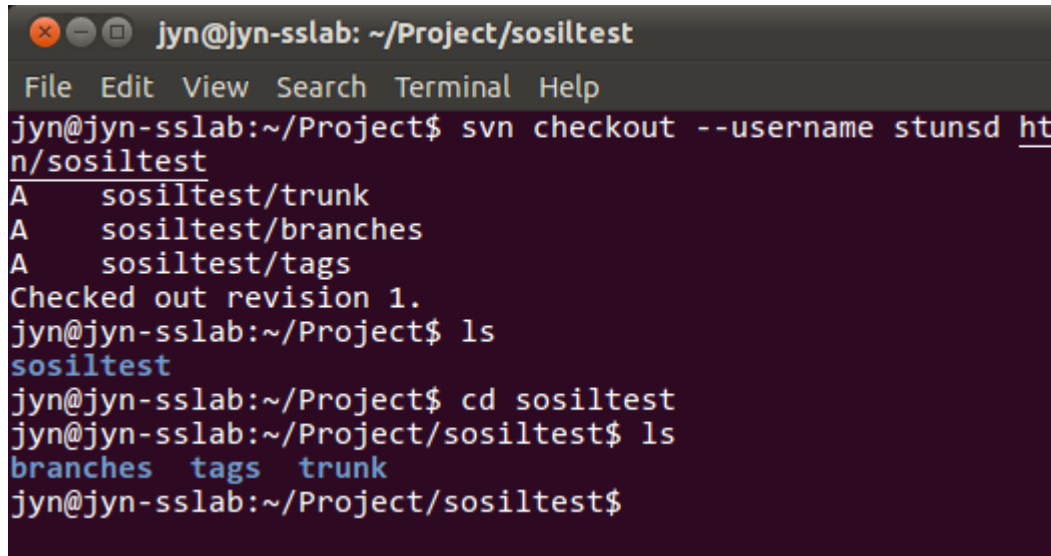
RabbitVCS

- ❑ Import
- ❑ Repository에 SVN 주소 입력
- ❑ 그리고 OK 를 눌러준다.



❑ Checkout

- ~/ 밑에 폴더를 하나 생성 후 checkout 을 하면
 - ❑ `svn checkout --username [ID] [https://dev.naver.com/svn/프로젝트 이름]`

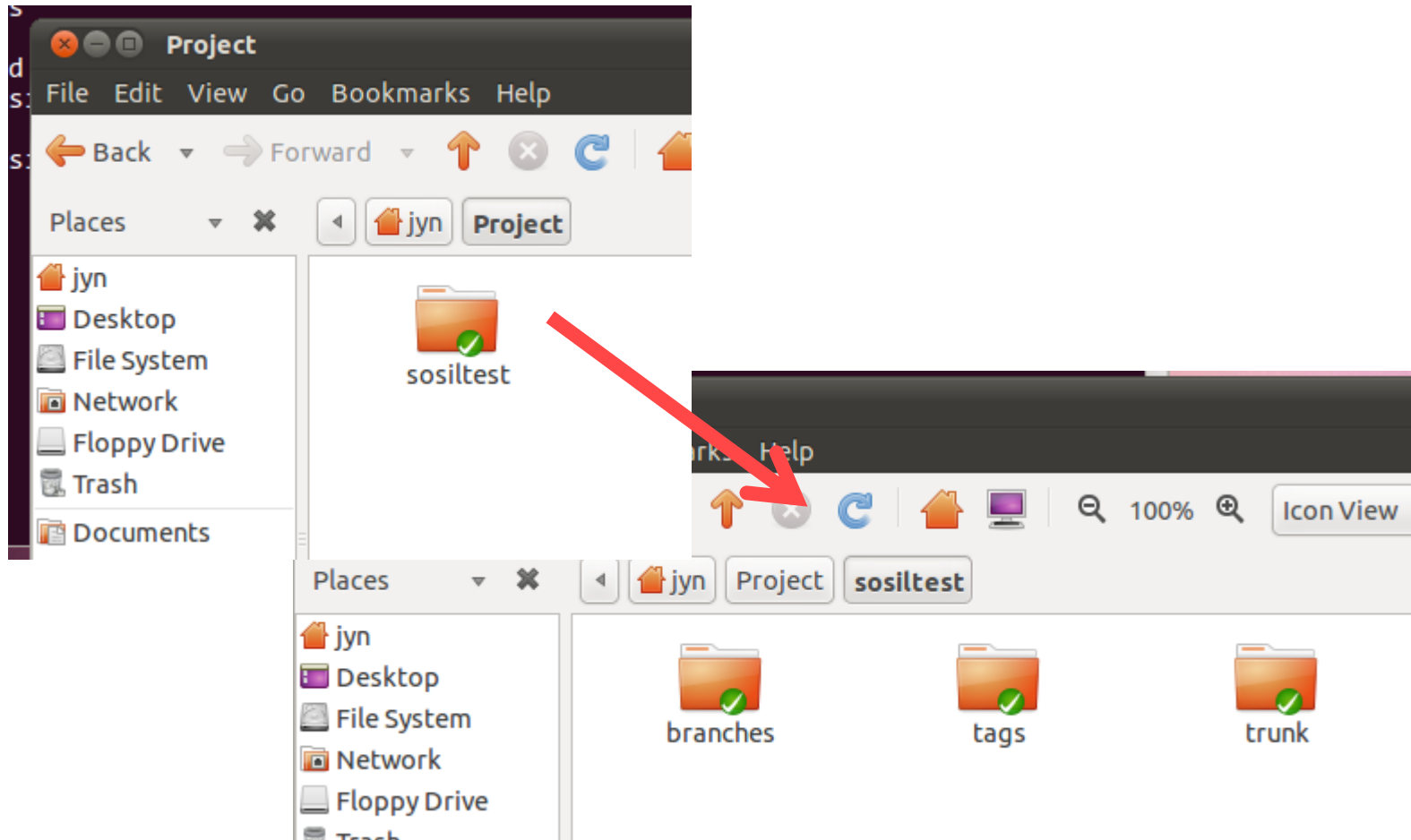


```
jyn@jyn-sslslab: ~/Project/sosiltest
File Edit View Search Terminal Help
jyn@jyn-sslslab:~/Project$ svn checkout --username stunsd ht
n/sosiltest
A      sosiltest/trunk
A      sosiltest/branches
A      sosiltest/tags
Checked out revision 1.
jyn@jyn-sslslab:~/Project$ ls
sosiltest
jyn@jyn-sslslab:~/Project$ cd sosiltest
jyn@jyn-sslslab:~/Project/sosiltest$ ls
branches  tags  trunk
jyn@jyn-sslslab:~/Project/sosiltest$
```

- 저장소의 폴더 구조를 그대로 다운 받게 된다
- 그리고 리눅스의 탐색기(`nautilus`) 를 열어보면

RabbitVCS

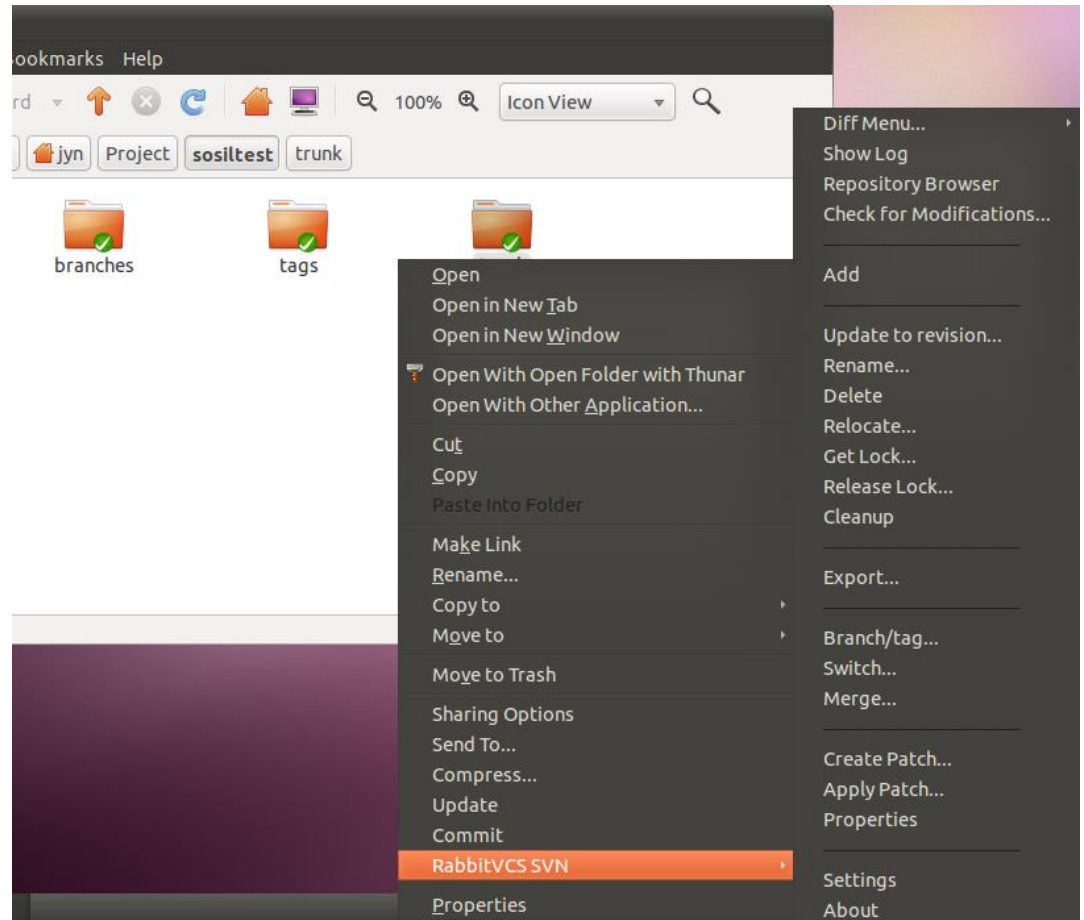
☐ Checkout



RabbitVCS

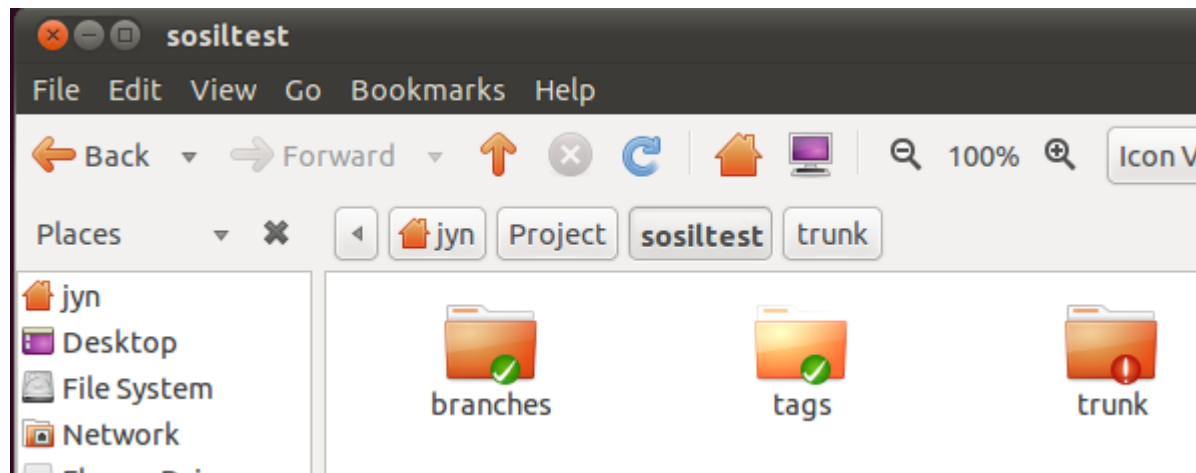
□ Subversion

- 오른쪽 클릭을
하면 각종 명령을
볼 수 있다



□ Commit

- 예) checkout으로 받은 trunk 폴더 안에 diary.h 를 수정했다고 가정한다



- Trunk 폴더에 느낌표가 뜬 것을 볼 수 있다
- Trunk 폴더를 우 클릭 한 후 Commit 메뉴 클릭
- 메시지 작성 후 확인

□ Subversion

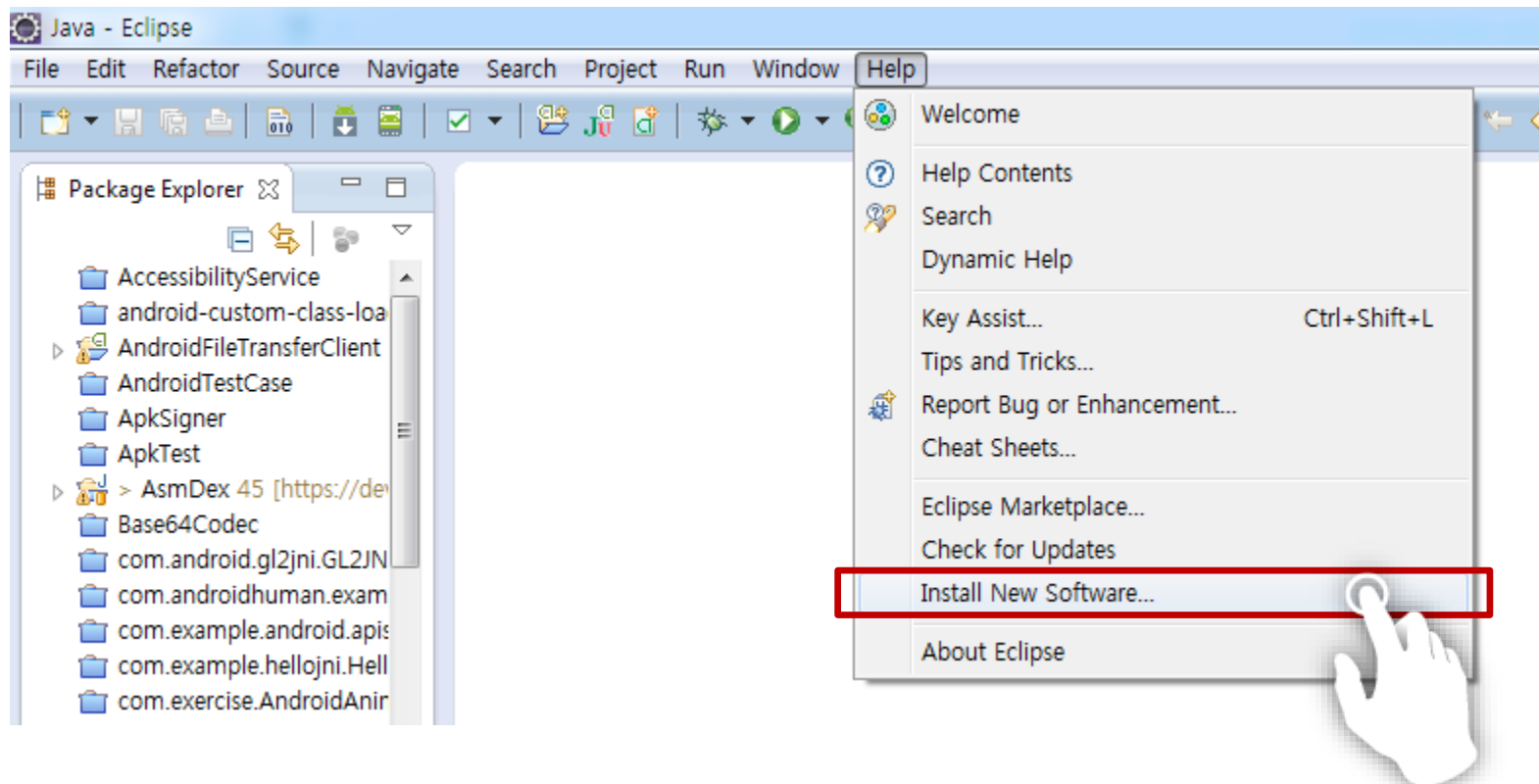
- 나머지 명령(update, add 등 ...)을 다 편하게 사용할 수 있음
- 웹 브라우저로 접속하면 버전 정보 등을 살펴볼 수 있음

Eclipse addon - Subversive

□ Subversive 설치방법

■ Step 1. Eclipse Marketplace 설치 (있으면 생략)

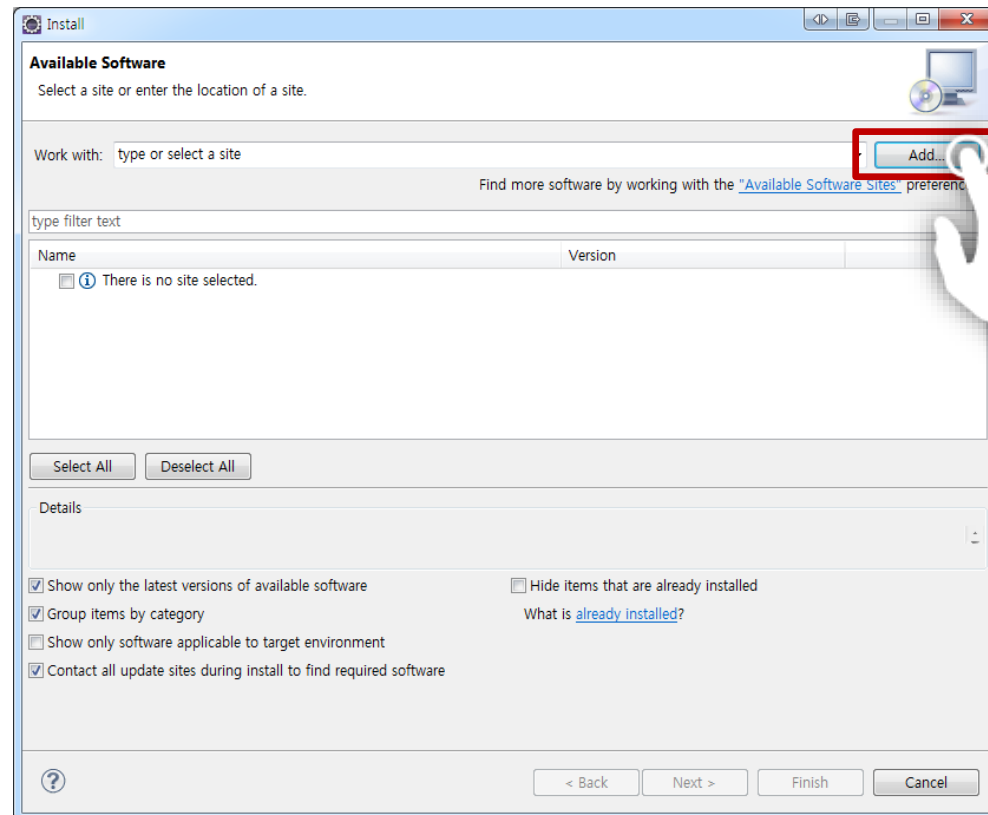
□ Help > Install New Software



Eclipse addon - Subversive

□ Subversive 설치방법

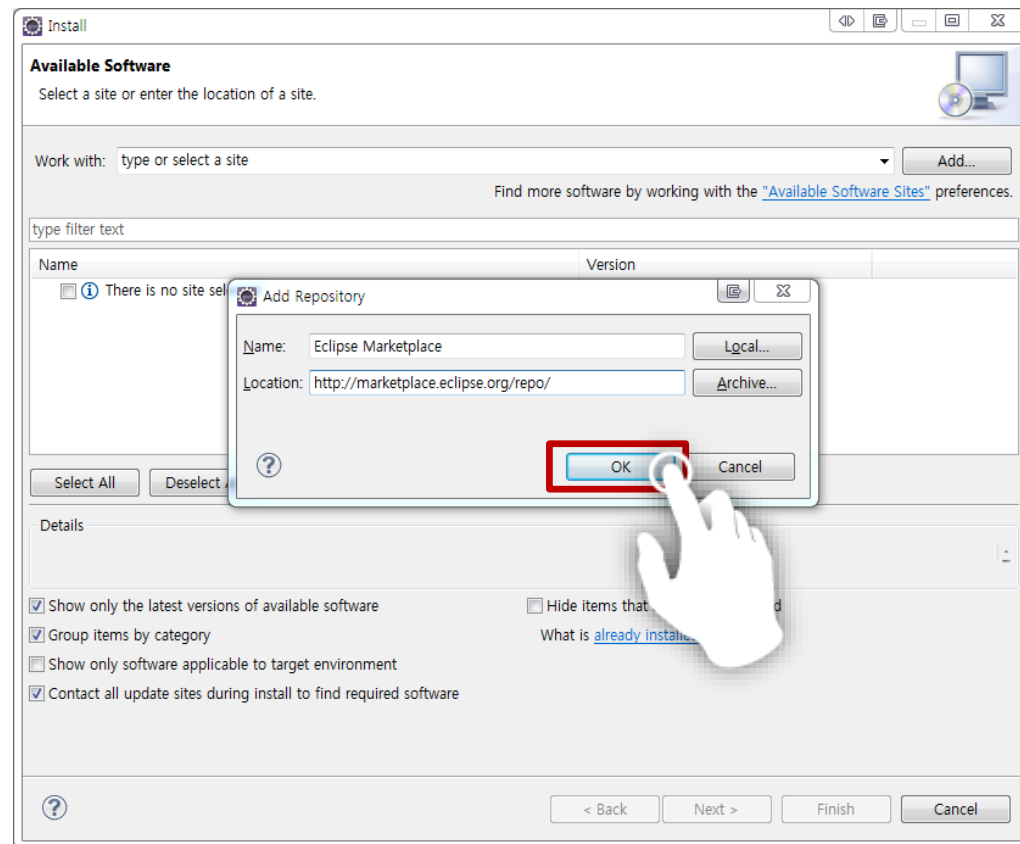
- Install > Add



Eclipse addon - Subversive

❑ Subversive 설치방법

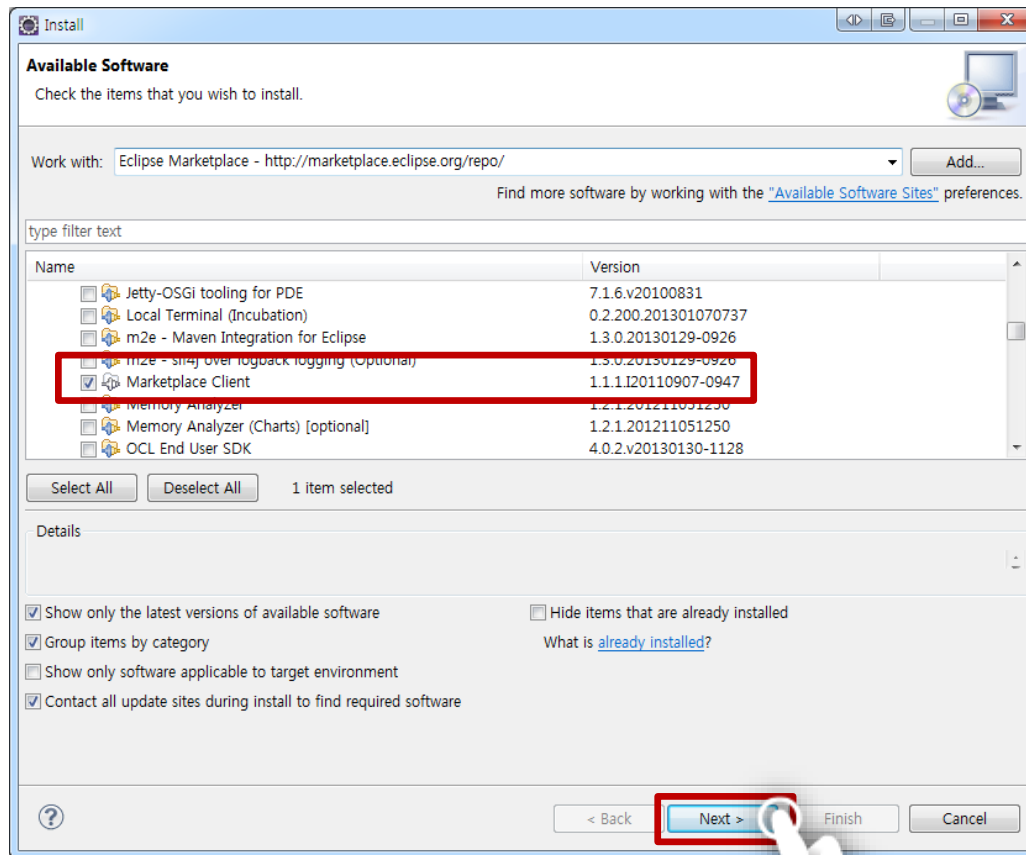
■ 아래와 같이 입력



Eclipse addon - Subversive

□ Subversive 설치방법

- General Purpose Tools > Marketplace Client 체크
- 동의 후 설치

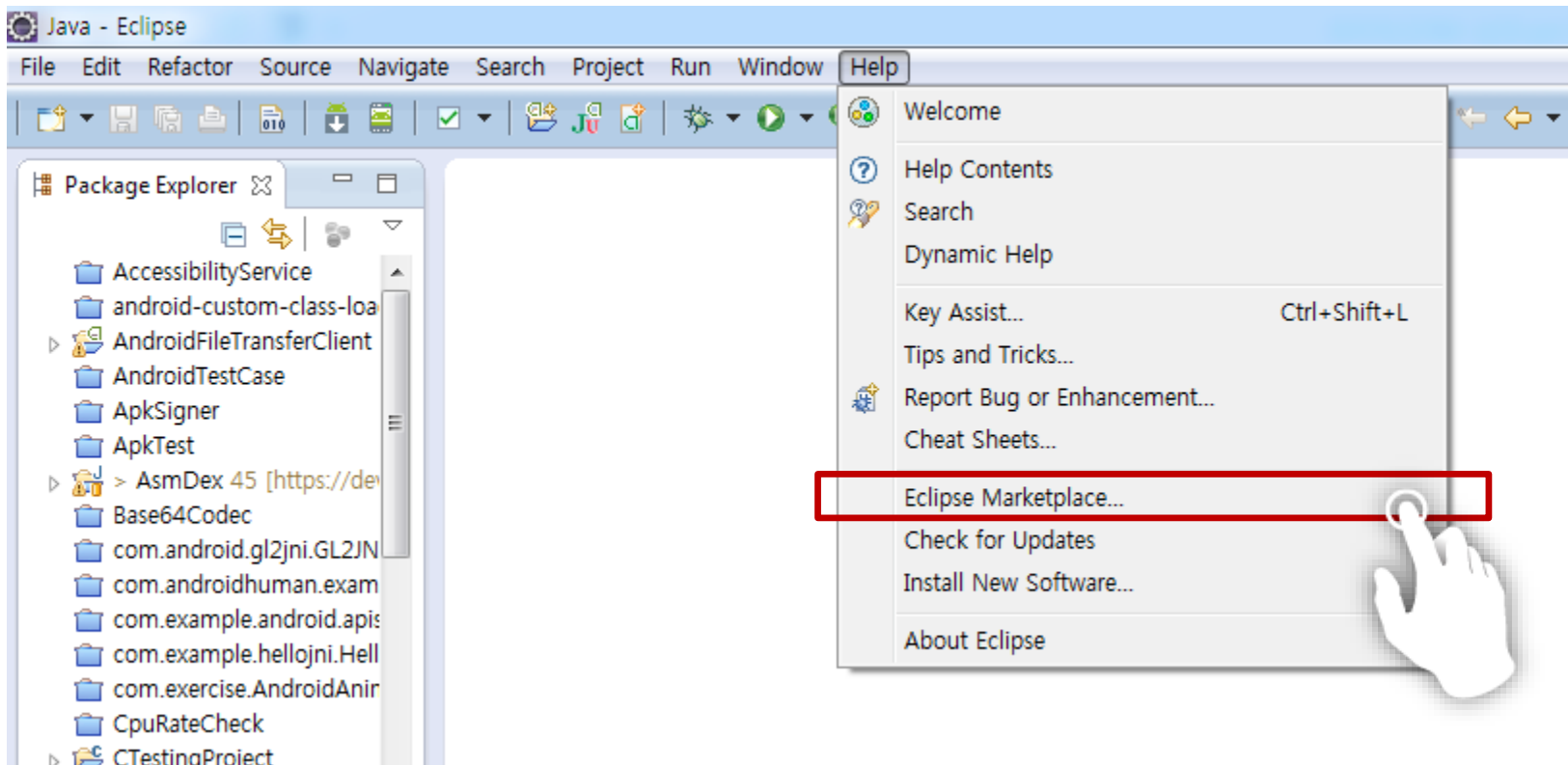


Eclipse addon - Subversive

□ Subversive 설치방법

■ Step 2. Marketplace에서 설치

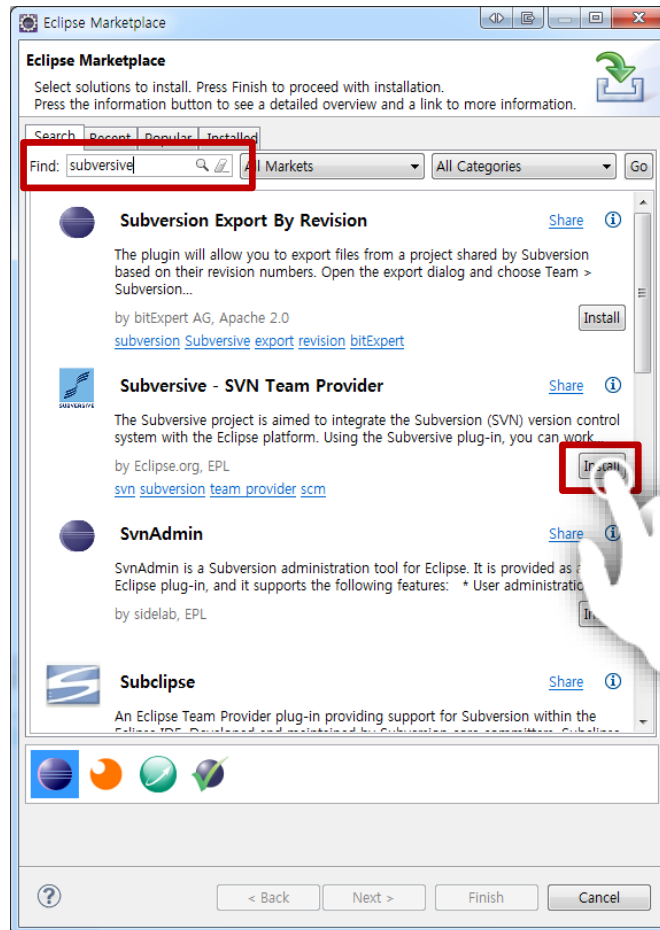
□ Help > Eclipse Marketplace



Eclipse addon - Subversive

❑ Subversive 설치방법

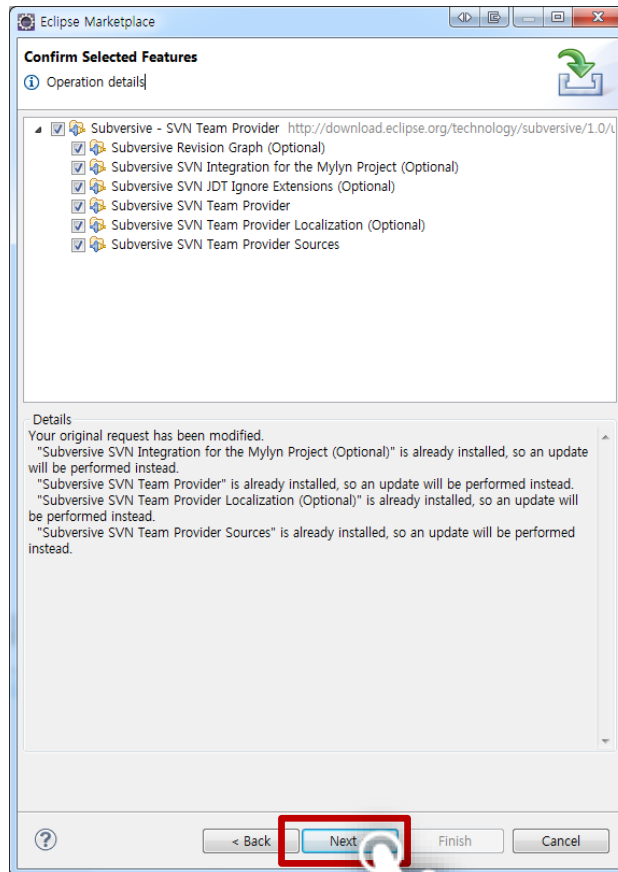
■ Find subversive, Install



Eclipse addon - Subversive

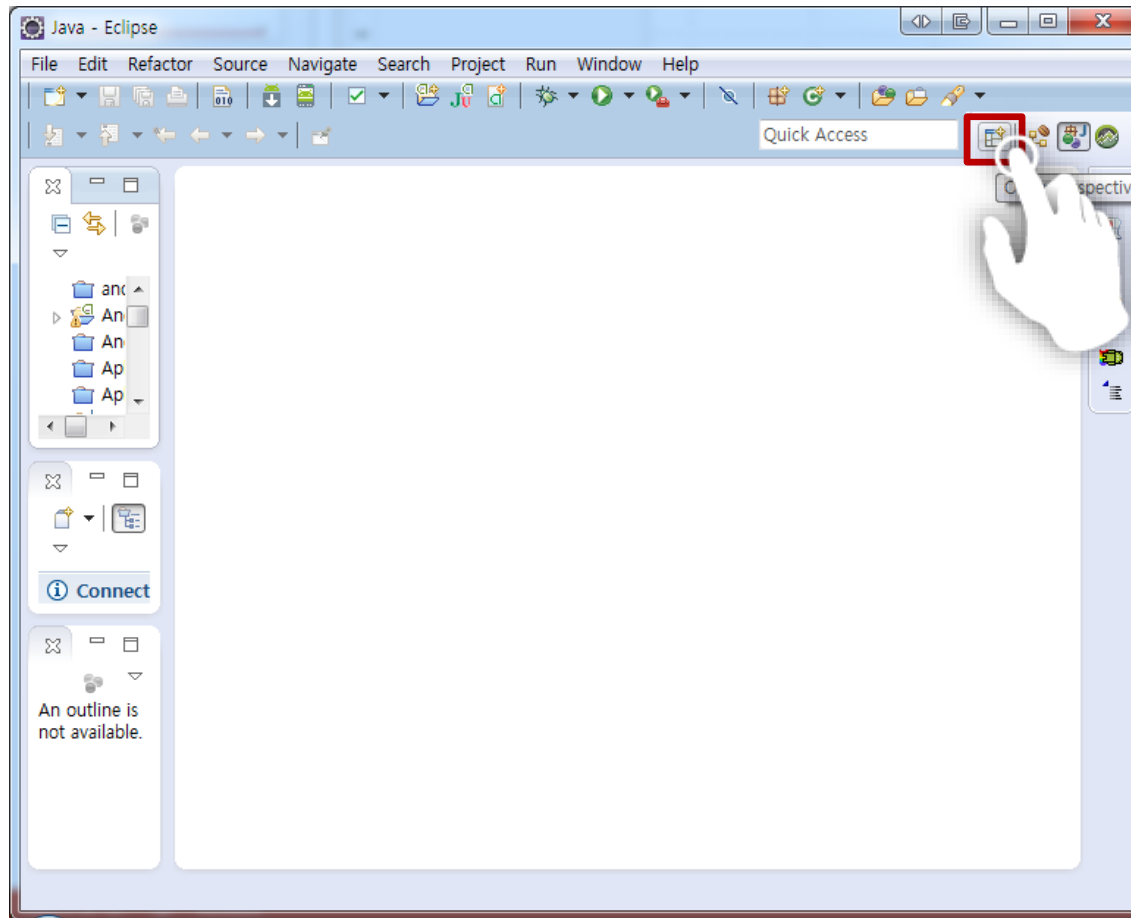
□ Subversive 설치방법

■ Next, 동의 후 설치



Eclipse addon - Subversive

- Subversive 사용
 - SVN 관련 Window 추가



Eclipse addon - Subversive

☐ Subversive 사용

- SVN 관련 Window 추가
 - ☐ SVN Repository Exploring
 - Repository 추가 용도
 - ☐ Team Synchronizing
 - Project Synchronize 용도

