도구 활용 가이드

SW중소기업을 위한 경량 개발 방법론



과학기술정보통신부
Ministry of Science and ICT

nipa 정보통신산업진흥원
National IT Industry Promotion Agency

경량 개발 방법론
도구 활용 가이드

nipa 정보통신산업진흥원 National IT Industry Promotion Agency
STA Global Testing Leader 테스팅컨설팅

발간사

2016년 세계경제포럼(WEF, World Economic Forum)은 전 세계가 당면하게 될 주요 이슈로 4차 산업혁명에 주목했으며, 최근 빅데이터, 클라우드, 인공지능, 사물인터넷 등 융합과 혁신이 빠르게 진행되고 있습니다.

이러한 흐름의 중심에는 소프트웨어가 있습니다. 최근의 소프트웨어는 이전과는 비교할 수 없는 빠른 속도로 변화하고 있습니다. 그러나 전통적 소프트웨어 개발 방법론은 이러한 변화를 빠르게 수용하기 어렵고, 이에 대한 대안으로 경량 개발 방법론이 주목 받고 있습니다.

한 조사에 따르면 해외의 소프트웨어 프로젝트 상당수가 경량 개발 방법론을 활용하고 있으며, 프로젝트 관리 지식 체계를 다루고 있는 PMBOK (Project Management Body of Knowledge)는 2018년 개정판에 경량 개발 방법론을 신규로 포함 시킬 것임을 발표하였습니다.

그러나 기존의 경량 개발 방법론 가이드들은 개념의 이해 위주여서 실제 적용 시 구체적인 실행 방안이 부족하며, 또한 방법론이 추구하는 수평적 조직 문화와 국내 조직 문화와의 이질성 등의 한계로 성공 사례가 적고, 경량 개발 방법론의 대표적인 애자일 및 데브옵스 도입률은 아시아태평양 지역 국가 가운데 가장 낮다는 조사결과도 발표되었습니다.

본 가이드는 기존 경량 개발 방법론의 장점을 모으고 국내 SW 기업의 특성을 반영한 경량 개발 방법론이며, 활용도를 높이기 위해 상세한 설명과, 다양한 사례 및 팁을 수록하였습니다.

아무쪼록 SW 기업들이 본 가이드를 활용하여 비즈니스의 변화에 민첩히 대응하고, 나아가 4차 산업혁명이라는 변화의 중심에 설 수 있기를 바랍니다. 끝으로 본 가이드를 개발하기까지 수고해 주신 관계자 분들께 감사드립니다.

정보통신산업진흥원

원장 윤 종 록

Contents

CHAPTER 01 Toolchains 소개 07

CHAPTER 02 JAVA JDK 11

2.1 다운로드 12
2.2 자바 환경변수 설정(JAVA_HOME, PATH) 17

CHAPTER 03 SVN(Subversion) 19

3.1 다운로드 21
3.2 설치 22
3.3 저장소 생성 28
3.4 저장소 설정 29
3.5 저장소 연결(체크아웃을 통한 확인) 31

CHAPTER 04 Git 33

4.1 Git 설치 34
1) Git 다운로드 34
2) Git 설치 36
4.2 Git 실행 43

CHAPTER 05 Maven 47

5.1 다운로드 48
5.2 설치 49
5.3 Maven 프로젝트 생성 50

CHAPTER 06 Apache Ant 53

6.1 Ant 설치 54
6.2 Ant 환경변수 55
6.3 build.xml 57

CHAPTER 07 Findbugs 61

7.1 설치 62
7.2 실행 63
7.3 분석 64

Contents

CHAPTER 08 Junit 67

8.1 Maven Junit 설정 69
8.2 Junit 사용하기 70

CHAPTER 09 Selenium 73

9.1 Selenium IDE 설치 75
9.2 Selenium IDE 사용법 79
1) 메뉴설명 79
2) Selenium IDE – Recording 79
3) Selenium IDE – Play 80
4) Selenium IDE – Export 80

CHAPTER 10 Fitnesse 81

10.1 설치 82
10.2. 설계 84
1) Fit Table 84
2) ColumnFixture 84
10.3. 실행 86

CHAPTER 11 Nexus 93

11.1 다운로드 94
11.2 설치 96
11.3 설정 98
11.4 Maven 저장소 99
1) 새로운 저장소 생성(Maven) 99
2) Maven 설정(Configuring Apache Maven) 100

CHAPTER 12 Jenkins 105

12.1 다운로드 106
12.2 설치 107
12.3 Jenkins PORT 변경 방법 112
12.4 Jenkins 접속 113
12.5 Jenkins 관리 114
1) 시스템 설정 115
2) 사용자 등록과 권한 설정 126
3) Global Tool Configuration 127
4) 플러그인 관리 128
12.6 Jenkins 신규 아이템 추가 및 설정 130
1) 소스 코드 관리 131
2) 빌드 유발 133
12.7 Jenkins에서 빌드 실행 139

CHAPTER 13 Toolchain Tutorials 141

13.1 Tutorial 1 : Jenkins, SVN, Maven, Nexus 142
1) Maven 프로젝트 생성 142
2) Maven 프로젝트 형상관리(SVN) 144
3) Jenkins 프로젝트 생성 및 SVN 연결 150
4) Nexus 배포를 위한 Maven 설정 154
5) 변경 사항 SVN 저장소에 반영하기 157
6) Toolchain 확인 158
13.2 Tutorial 2 : Jenkins, SVN, Ant, Selenium, Junit 161
1) 프로세스 161
2) Eclipse 162
3) SVN 167
4) Jenkins 188
5) Jenkins 문제해결 197
6) SVN 대신 Git 으로 연결하는 경우 197
13.3 Tutorial 3 : Jenkins(Tutorial 1), Findbugs 198
1) Jenkins에서 정적분석 도구 사용하기(Findbugs) 198
13.4 Tutorial 4 : Jenkins, Fitnesse 205



CHAPTER 01

Toolchains 소개



CHAPTER 01 Toolchains 소개

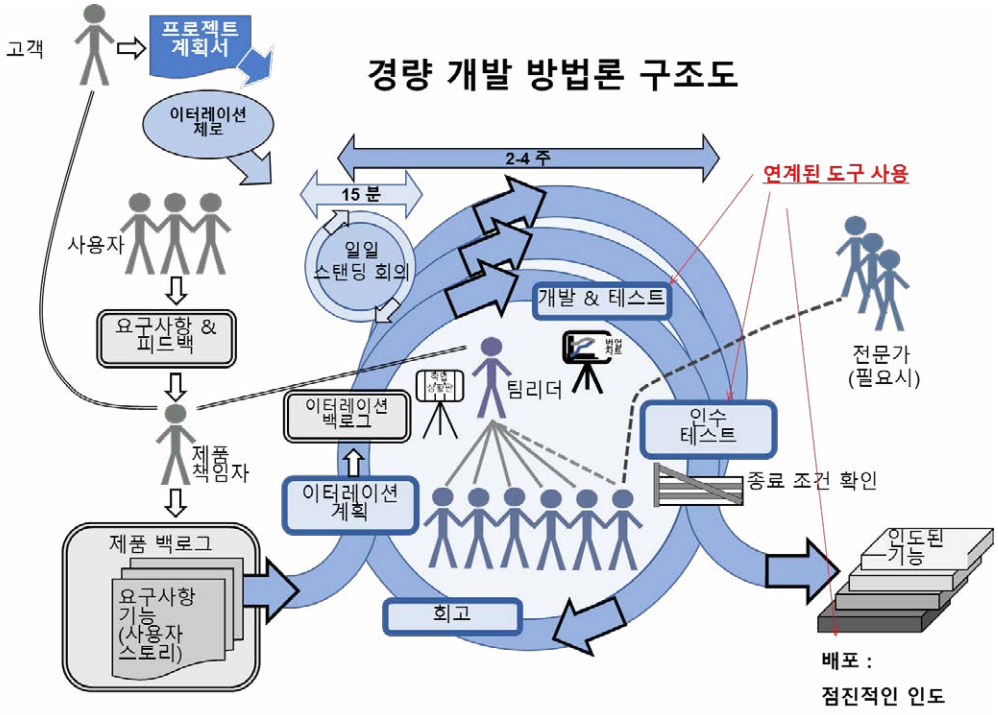
1.1. 도구 가이드 소개

본 도구 가이드는 정보통신산업진흥원에서 지원하는 사업, '중소기업을 위한 경량 개발 방법론'과 연계한 도구 가이드이다.

본 도구 가이드를 사용하는 목적은 아래와 같다.

- **반복적인 수작업을 줄여준다.**
  - 코드의 변경에 대해 자동으로 빌드하고 단위 테스트와 단위 회귀 테스트를 수행해 소프트웨어의 품질을 향상 시킬 수 있다.
- **언제 어느 때라도 배포할 수 있는 소프트웨어를 생성해낸다.**
  - 반복적인 빌드, 배포, 테스트로 소프트웨어 품질이 높아진다.
  - 문제 상황 발생시 이를 빠르게 피드백 받고 즉각 처리할 수 있는 환경이 구축된다.
- **프로젝트의 가시성을 좋게 한다.**

방법론의 구조도는 아래와 같다



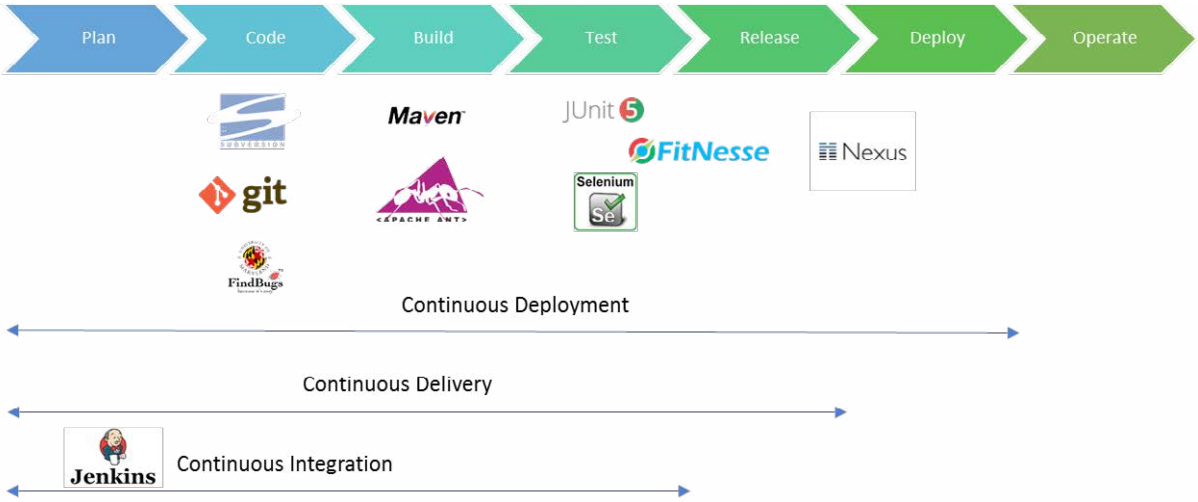
방법론과 연계된 기술 활동은 아래와 같다.

기술 활동	관계된 도구	설명
지속적 통합	Jenkins, SVN, Git, Maven, Ant, Findbugs,	코드 자동 빌드 및 통합
자동화된 빌드 & 테스트	Maven, Ant, Junit, Selenium, Findbugs	지속적 통합과 연계함
자동화된 단위 & 회귀 테스트	Junit, Selenium, Findbugs	코드가 수정될 때 회귀 또는 통합 문제를 방지하기 위해 단위 테스트 및 단위 회귀 테스트를 수행
지속적 전달 및 배포	Jenkins, Nexus	빌드 및 단위/회귀 테스트가 성공할 때마다 전체 회귀 테스트를 위해 코드가 테스트 서버에 전달되고 자동 배포함
인수 테스트 주도개발	Fitness	코드가 인수 테스트를 통과하도록 특별히 개발된 개발 접근 방법. 일반적으로 아래와 같이 특정 형식이 필요함. Given 어떤 상황이 주어지면 When 어떤 행동이 수행될 때 Then 이런 결과가 나타날 것이다.

본 가이드의 도구 선정 기준은 공개SW이면서 가장 널리 활용되는 도구 및 언어(JAVA)를 기준으로 선정했고 사용자는 프로젝트의 상황에 맞게 선택적으로 사용할 수 있다.

도구 이름	도구 분류	비고
SVN(Subversion)	형상관리 도구	소스 버전 관리 도구
Git	형상관리 도구	분산 버전 관리 도구
Maven	프로젝트 관리 및 빌드 도구	프로젝트 의존성 관리, 라이브러리 관리, 프로젝트 생명 주기 관리 기능 등을 제공
Ant	빌드 도구	JAVA 프로그래밍 언어에서 사용하는 자동화된 빌드 도구
Findbugs	정적분석 도구	자바 소스코드에 대한 잠재적인 문제에 대한 분석
Junit	단위 테스트 도구	JAVA 단위 테스트 프레임워크
Selenium	테스트 수행 자동화 도구	다양한 브라우저 지원 및 개발언어를 지원하는 웹애플리케이션 테스트 프레임워크
Fitnesse	인수 테스트 지원 도구	Fit에 위키를 결합하여 웹 서버 형태로 제공되는 인수 테스트 자동화 도구
Nexus	저장소	Maven에서 사용 할 수 있는 Repository
Jenkins	CI 도구	빌드 자동화를 통한 지속적 통합(Continuous Intergration) 도구

Toolchains의 구성은 아래와 같다.



본 가이드에서 사용하는 도구를 설치 및 사용을 할 때, IE8 이하에서는 정상적으로 설치 또는 사용이 되지 않을 수 있다. IE를 사용하려면 업그레이드를 하거나, 다른 브라우저를 설치하여 사용하여야 한다. 또한, 각 도구는 본 가이드 작성 시점에서 최신버전을 사용하려고 하였으나 호환성의 문제 등의 이유로 가장 최신버전이 아닌 경우도 있으며, 가이드 작성 이후 버전이 업데이트된 경우도 있다. 본 가이드에서 사용하고 있는 버전과 다른 버전을 사용할 시에 설치 또는 설정 시 문제가 발생하거나 원활한 진행이 되지 않을 수 있다는 것을 참고하길 바란다.

CHAPTER 02

JAVA JDK



2.1 다운로드 12
2.2 자바 환경변수 설정(JAVA_HOME, PATH) 17

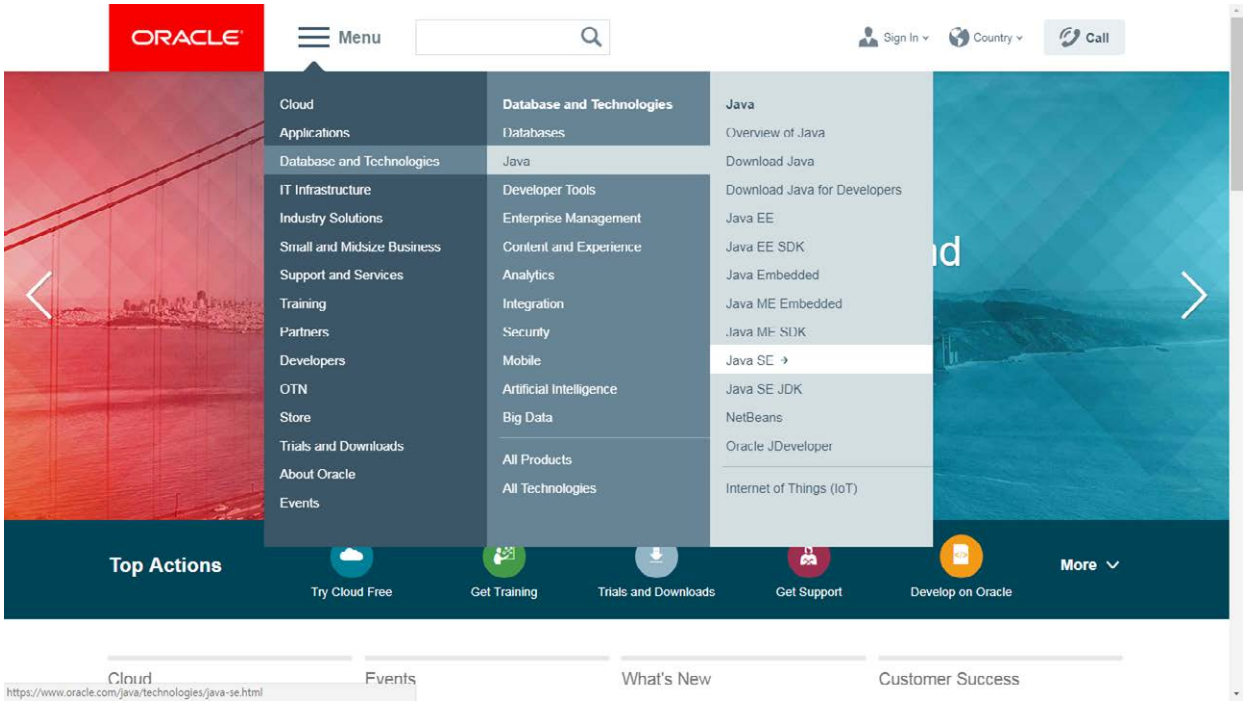
CHAPTER 02 JAVA JDK

이클립스(eclipse)로 JSP프로그래밍을 하거나 안드로이드 프로그래밍을 하기 위해서는 우선 JDK(Java Development Kit)라는 것을 설치해야 한다. 자바 개발 도구인 JDK를 설치를 해야 자바라는 언어로 개발을 할 수 있기 때문이다. (JDK는 단순한 자바실행환경인 JRE와는 구별이 된다. 참고로 JDK 〉 JRE 〉 JVM 이라고 할 수 있다.)

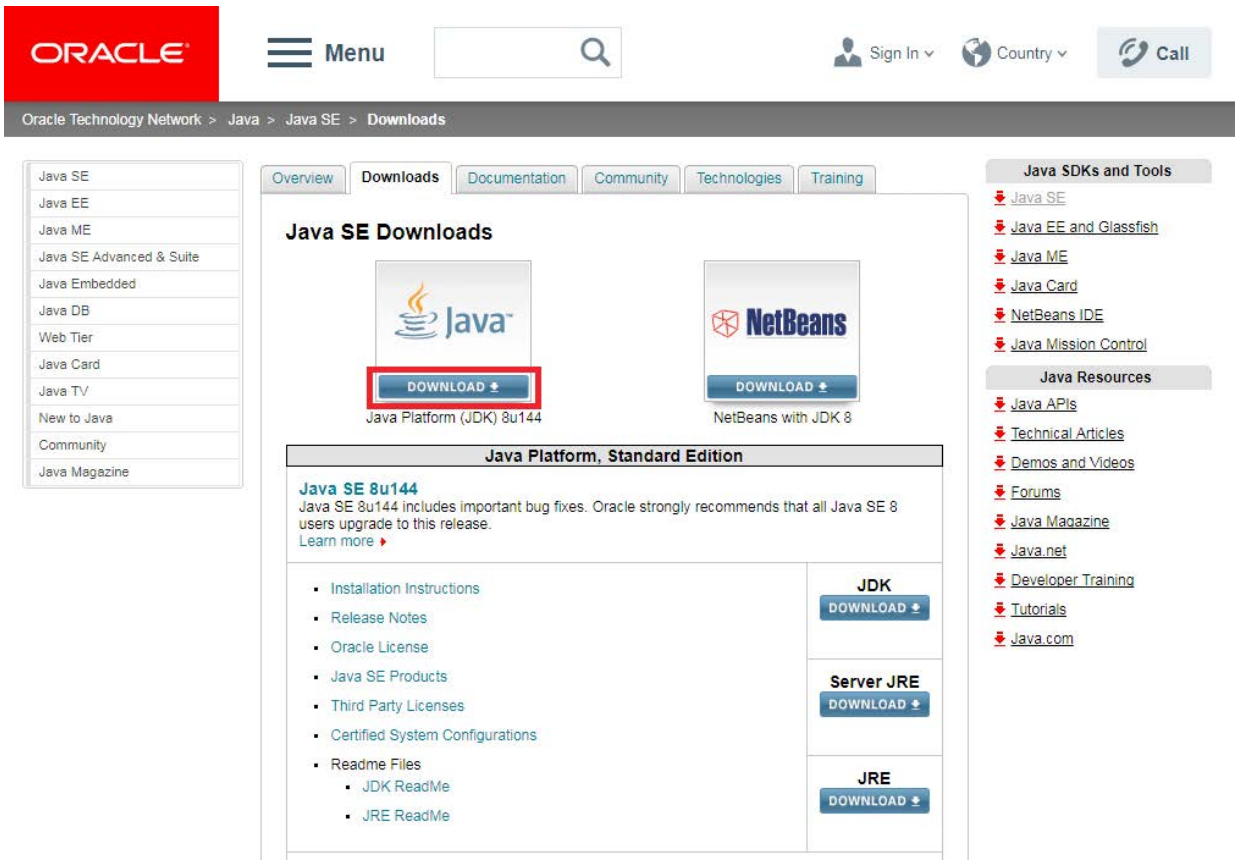
2.1. 다운로드

자바 JDK를 설치하기 위해서 자바 JDK를 다운을 받아야 한다.
다운로드는 Oracle 사이트에 접속하여 상단의 Menu 〉 Database and Technologies 〉 Java 〉 Java SE 선택 후 페이지 하단의 Get Started 아래 Download Java SE for Developers를 클릭한다. (다운로드 페이지로 바로 이동할 수 있다. 단, 페이지나 링크가 변경되어 접속이 되지 않을 수 있다.)

http://www.oracle.com
(http://www.oracle.com/technetwork/java/javase/downloads/index.html)



메뉴를 통해서 아래의 페이지로 접속을 하여 다운로드 링크를 선택한다. (상기 다운로드 링크로 접속을 하여도 된다.)



DOWNLOAD 버튼을 클릭하면 Java JDK 다운로드 링크를 선택하여 다운로드를 받는다.

앞서 소개에서도 도구들의 버전간 호환성 문제가 있을 수 있다고 언급한바 있다.
현재 Java JDK 최신 버전(9)으로 설치 시 Maven 빌드 시 오류가 발생할 수 있다. 오류 발생 시 해결하는 방법이 있지만, 본 가이드에서는 8버전으로 설치하여 진행하였다. 특별한 이유가 없다면 8버전으로 진행하는 것을 권장한다.

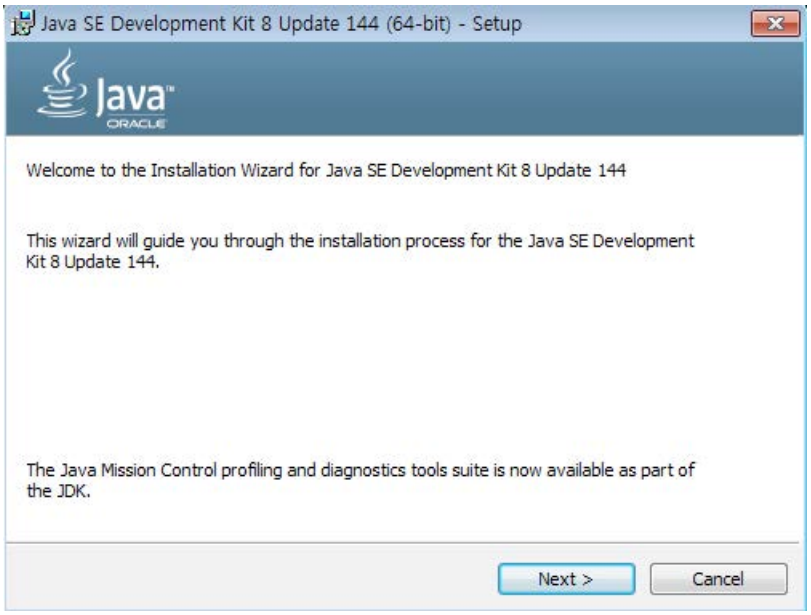
IE8.0에서는 Oracle 페이지의 메뉴 선택 시 Site Map 페이지로 이동하는 등의 문제가 발생하여 정상적인 진행이 어려울 수 있으니 IE8 이 후 버전으로 업데이트 하거나 Chrome 등의 브라우저를 사용하기를 권장한다.

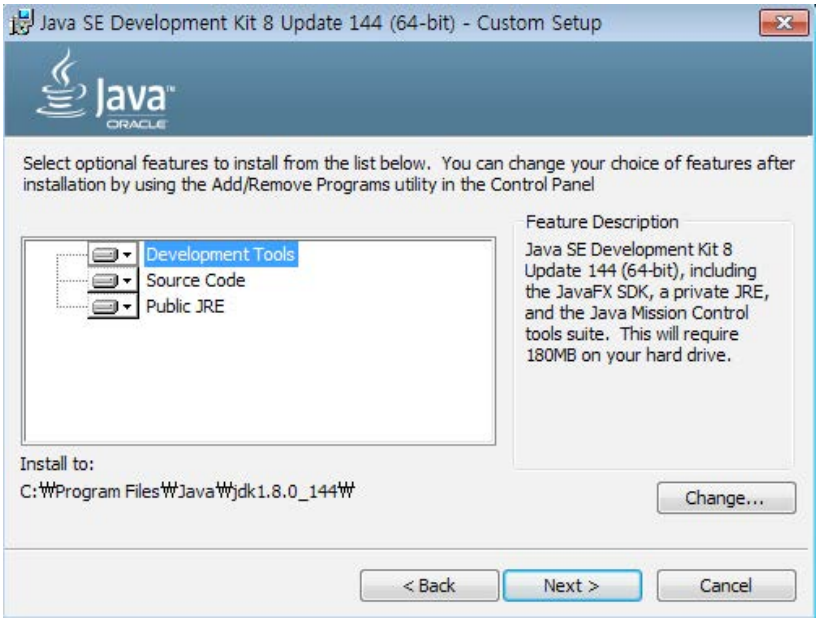


라이선스에 동의를 하고 아래에 있는 윈도우 버전의 JDK를 선택하여 다운받으면 된다.

다운받은 JDK 설치 파일을 실행하여 설치하면 된다. 별다른 설정이 없기 때문에 Next 버튼만 눌러도 된다.







자바 JDK가 설치될 경로를 직접 설정할 수 있는데 여기서는 그냥 기본경로로 설치를 한다.
만약, 컴퓨터에 자바실행환경 JRE(Java Runtime Environment)이 없다면 이것도 같이 설치가 된다.

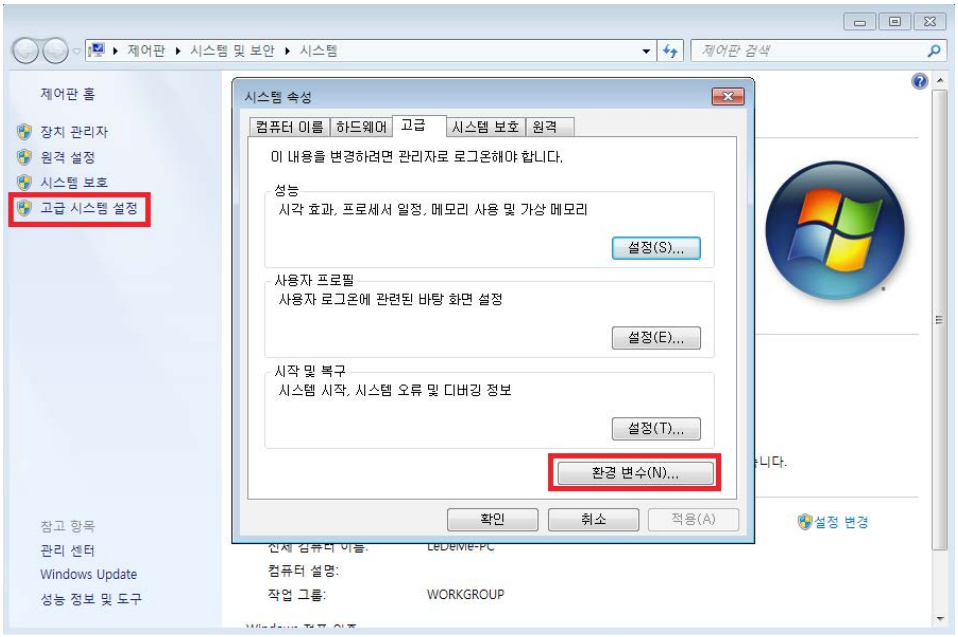
자바 JDK 설치가 완료되었으면 close 버튼을 눌러 설치를 종료한다.



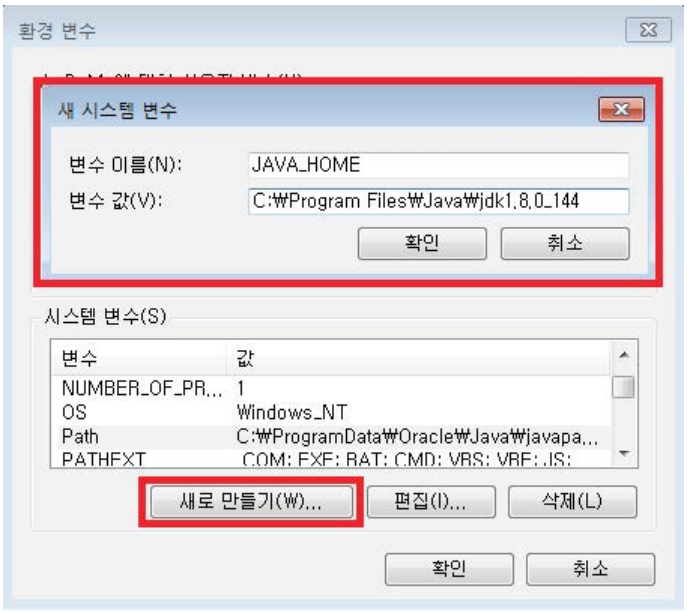
2.2. 자바 환경변수 설정(JAVA_HOME, PATH)

cmd 콘솔 명령 프롬프트에서 javac로 컴파일하거나, 다른 프로그램에서 자바 JDK를 참조하기 위해서는 윈도우 환경변수를 지정해야 한다.

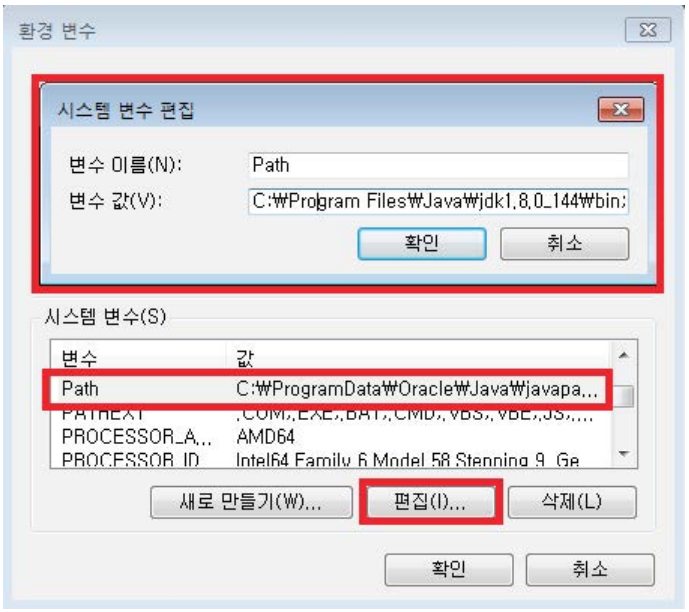
제어판 – 시스템 및 보안 – 시스템 – 고급 시스템 설정 – 고급 탭을 클릭하면 아래와 같은 화면이 뜬다. 여기서 환경변수를 클릭하여 자바 환경변수를 등록할 것 이다.



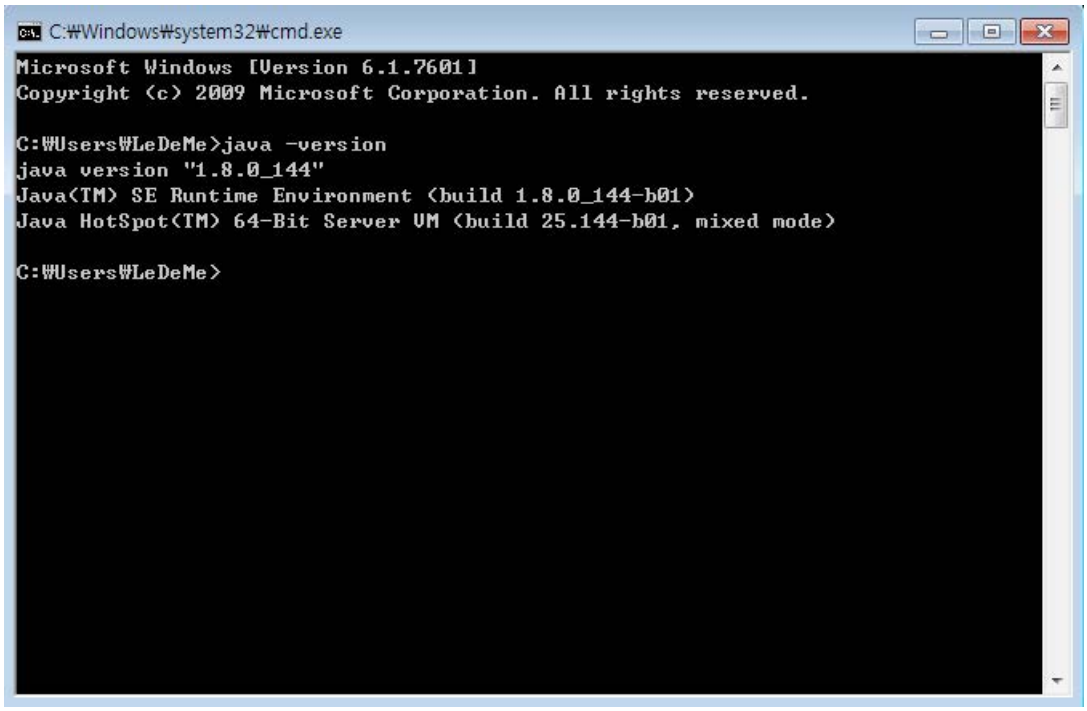
새로 만들기를 클릭하여 JAVA_HOME 환경변수의 값에 JDK가 설치된 디렉토리(폴더)를 지정하고 확인 버튼을 누른다.



PATH 변수를 클릭하고 편집을 버튼을 눌러 PATH 변수를 수정한다. JDK가 설치된 디렉토리의 bin까지 입력해야 제대로 동작한다.



자바 JDK를 설치하고, 환경변수까지 정확하게 지정했으면, cmd 콘솔 프로그램에서 아래 명령으로 자바 JDK가 제대로 설치되었는지 확인할 수 있다.



CHAPTER 03

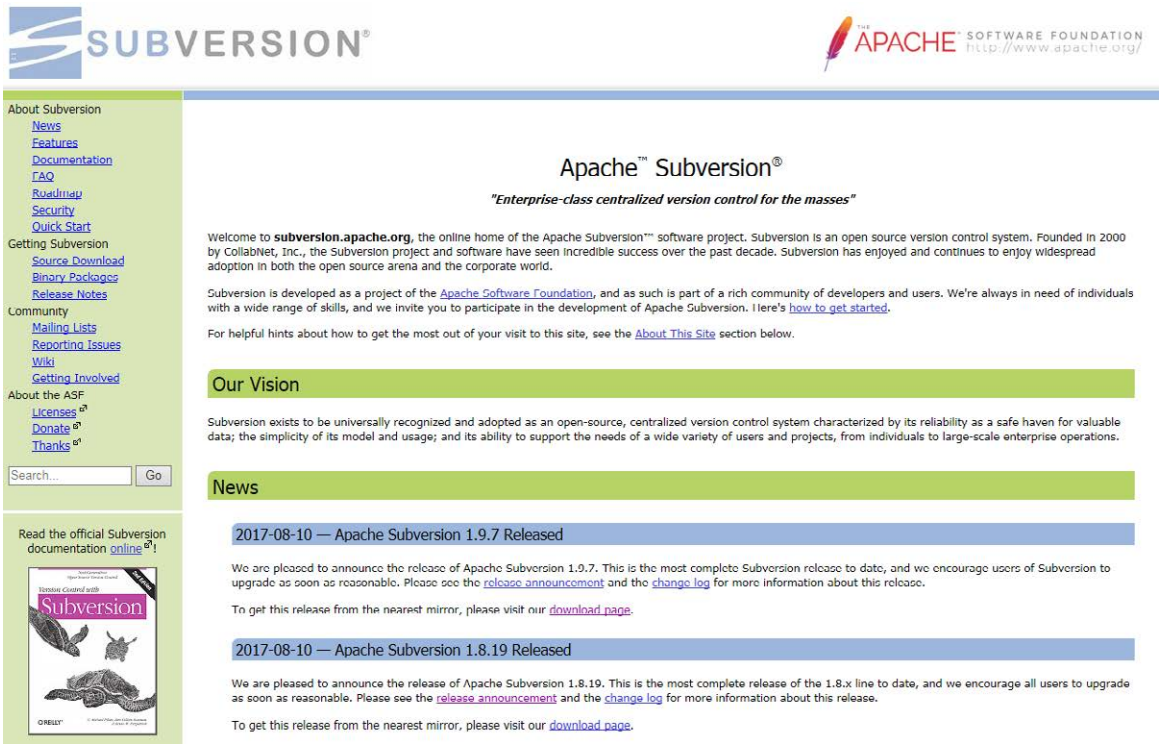
SVN(Subversion)

3.1 다운로드 21
3.2 설치 22
3.3 저장소 생성 28
3.4 저장소 설정 29
3.5 저장소 연결(체크아웃을 통한 확인) 31

CHAPTER 03 SVN(Subversion)

Subversion은 프리 소프트웨어 버전 관리 시스템이다. 인터페이스에서 사용하는 명령어를 따서 "SVN"이라고 줄여서 부르기도 한다. SVN을 사용하기 위하여 SVN Server를 설치할 것이다.

http://subversion.apache.org 에서 패키지 형태로 제공하고 있다. 서버는 소스만 받을 수 있고 설치파일을 받으려면 패키지로만 가능하다. 패키지 형태란 CollabNet 이나 VisualSVN 같은 3rd party 업체에서 서버+클라이언트를 제공하는 것을 말한다.



현재 1.9.7까지 나와 있고 CollabNet 이나 VisualSVN 에서 다운받아 설치 할 수 있다.

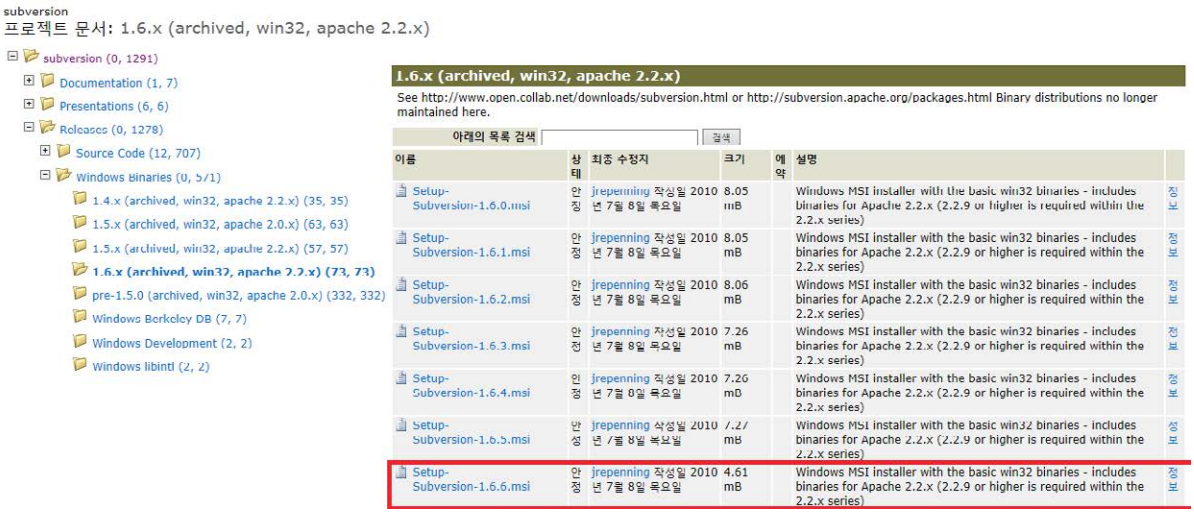


공식 사이트에서 제공하는 링크에서 다운받아 설치를 하여도 되지만, 바이너리 파일을 다운받아 설치 할 수 있는 다른 방법도 있다.

3.1 다운로드

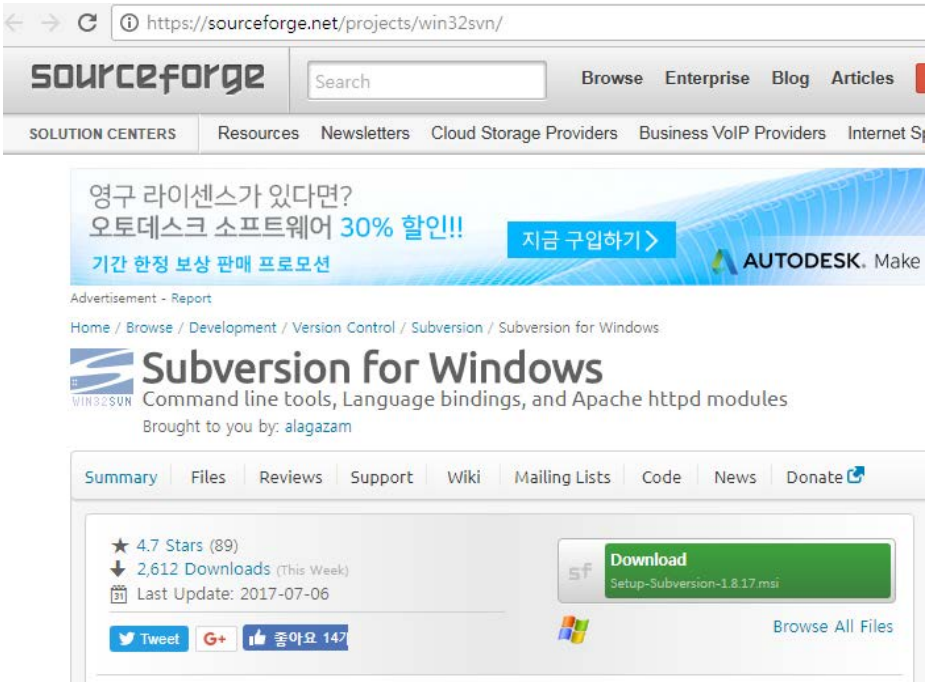
• 아래 링크에서 1.6.6 버전의 바이너리 파일을 다운받아 설치 할 수 있다.

http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=11151



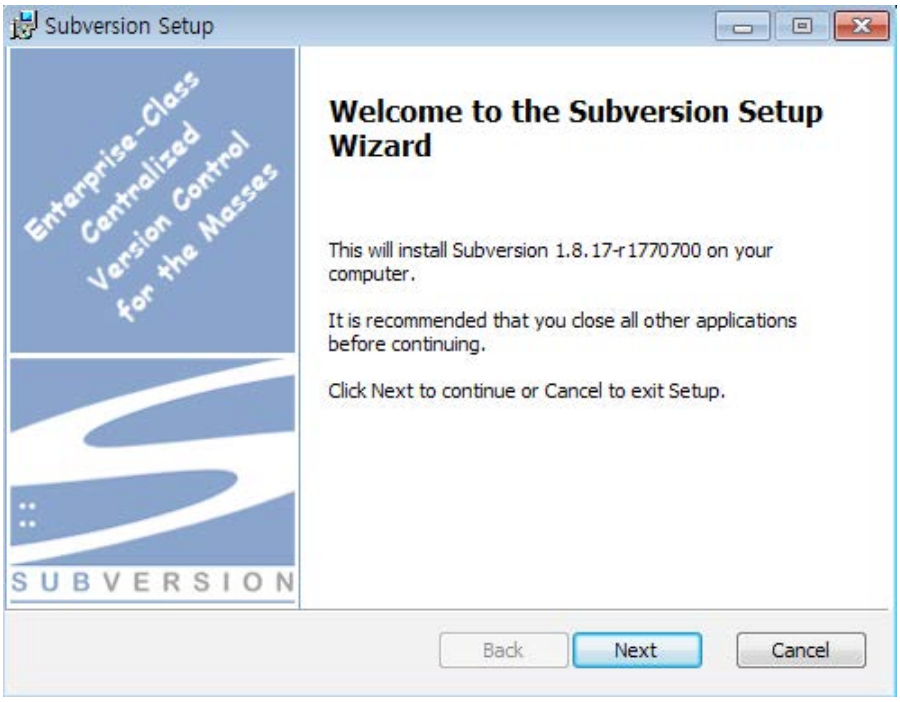
• Google에서 subversion으로 검색하면 sourceforge 에서 윈도우용 설치 파일을 찾을 수 있다. 아래 링크로 접속을 하여 설치 파일을 다운받을 수 있으며, 현재 1.8.17 버전이 올라와있다.

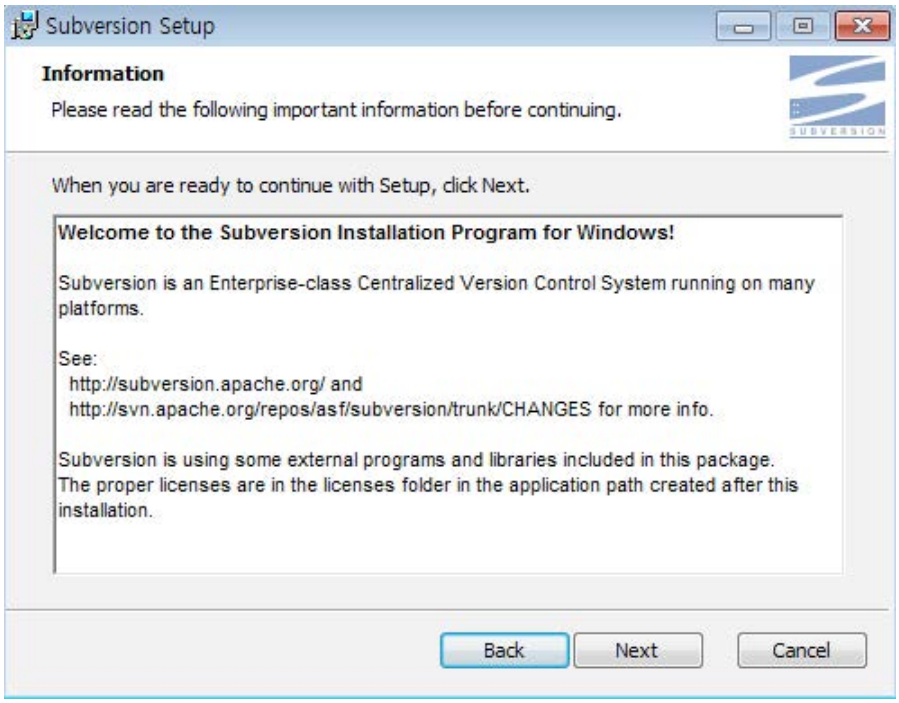
https://sourceforge.net/projects/win32svn/

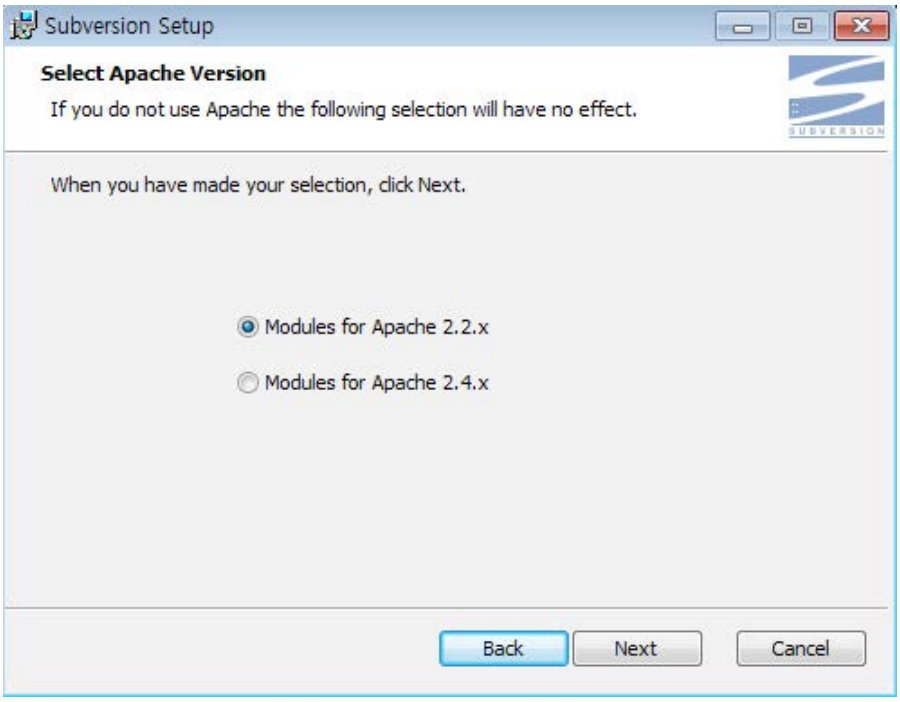


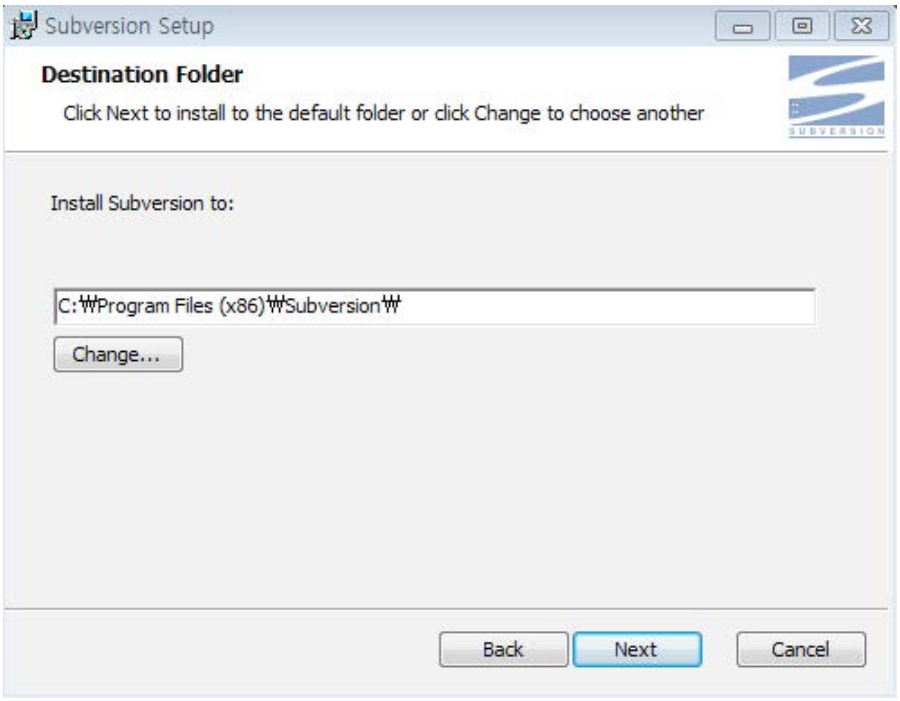
3.2 설치

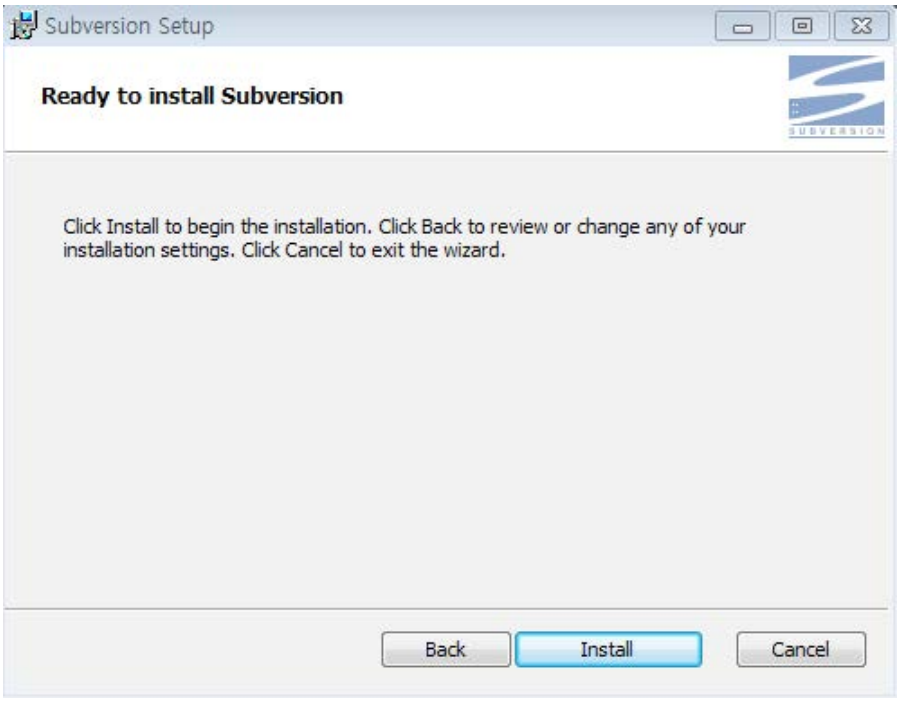
다운받은 파일을 실행한다. 계속 Next 버튼을 클릭하여 설치를 진행 한다.



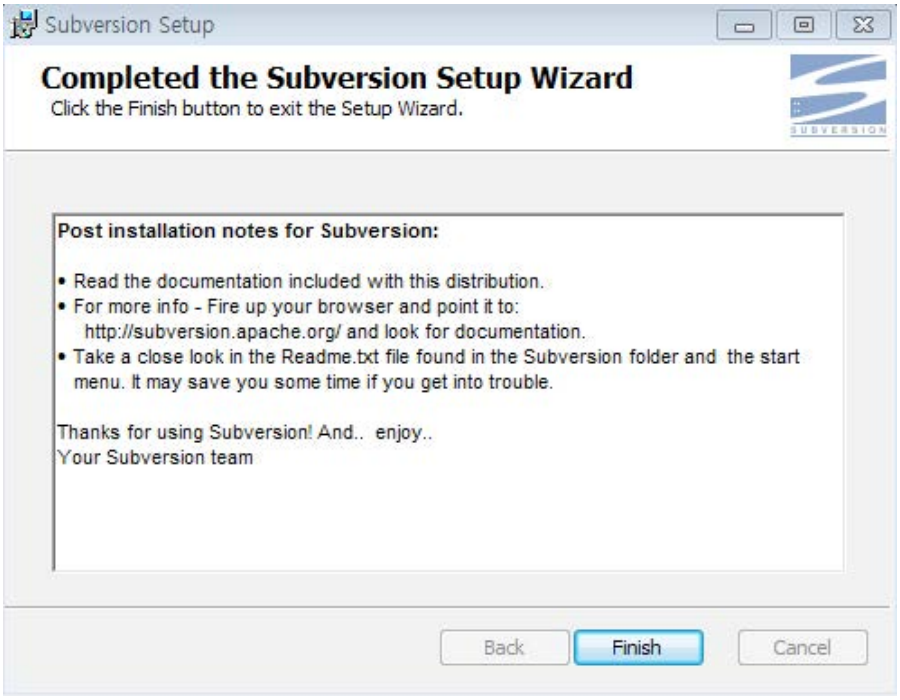



Subversion Setup
Select Apache Version
If you do not use Apache the following selection will have no effect.
When you have made your selection, click Next.
Modules for Apache 2.2.x
Modules for Apache 2.4.x
Back
Next
Cancel


Subversion Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another
Install Subversion to:
C:₩Program Files (x86)₩Subversion₩
Change...
Back
Next
Cancel

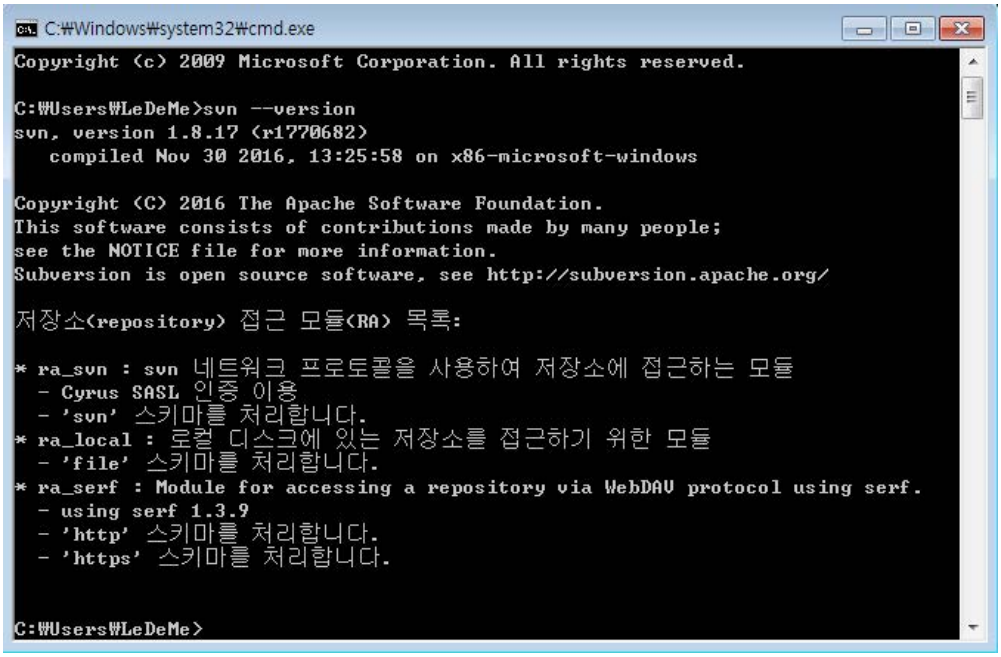


Install 를 선택한다.



설치가 완료 되고 cmd 명령프롬프트에서 아래 명령을 입력하여 설치 정보를 확인할 수 있다.

svn --version



svn이 설치 되었으면 컴퓨터(서버)를 재 시작할 때 자동으로 서비스를 시작해주기 위하여 서비스에 등록을 해 준다. 자동 시작이 필요하지 않다면 건너 뛰어도 된다.

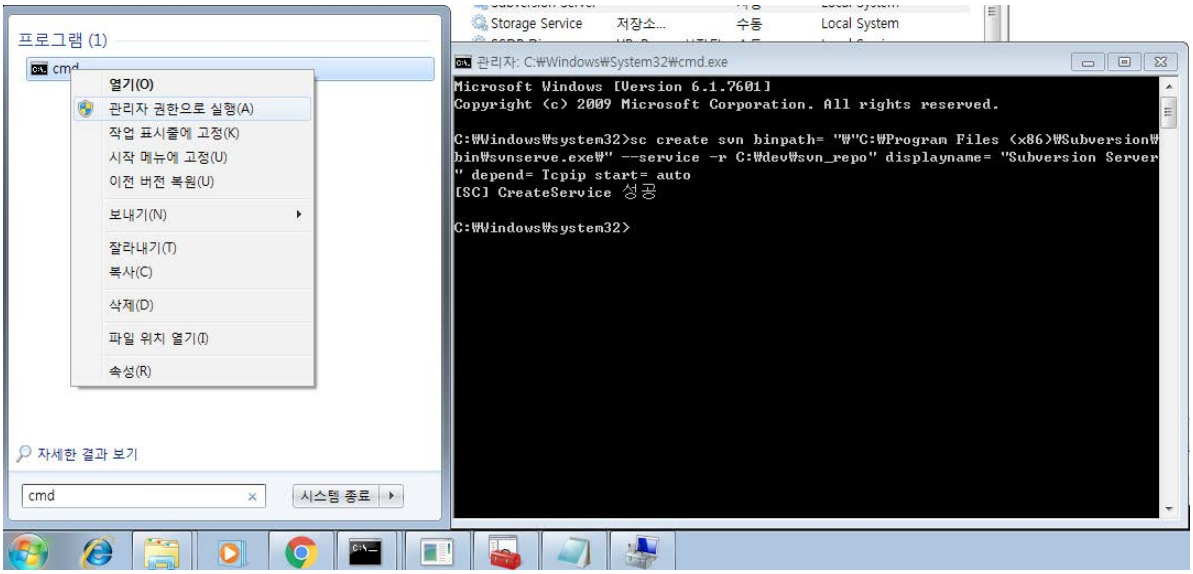
윈도우 버튼을 누르고 검색 창에 'cmd' 를 입력하고 cmd를 우클릭하여 관리자 권한으로 실행을 한다.

아래의 명령을 CMD 창에서 입력하고 실행한다.

앞서 소개에서도 도구들의 버전간 호환성 문제가 있을 수 있다고 언급한바 있다.
현재 Java JDK 최신 버전(9)으로 설치 시 Maven 빌드 시 오류가 발생할 수 있다. 오류 발생 시 해결하는 방법이 있지만, 본 가이드에서는 8버전으로 설치하여 진행하였다. 특별한 이유가 없다면 8버전으로 진행하는 것을 권장한다.

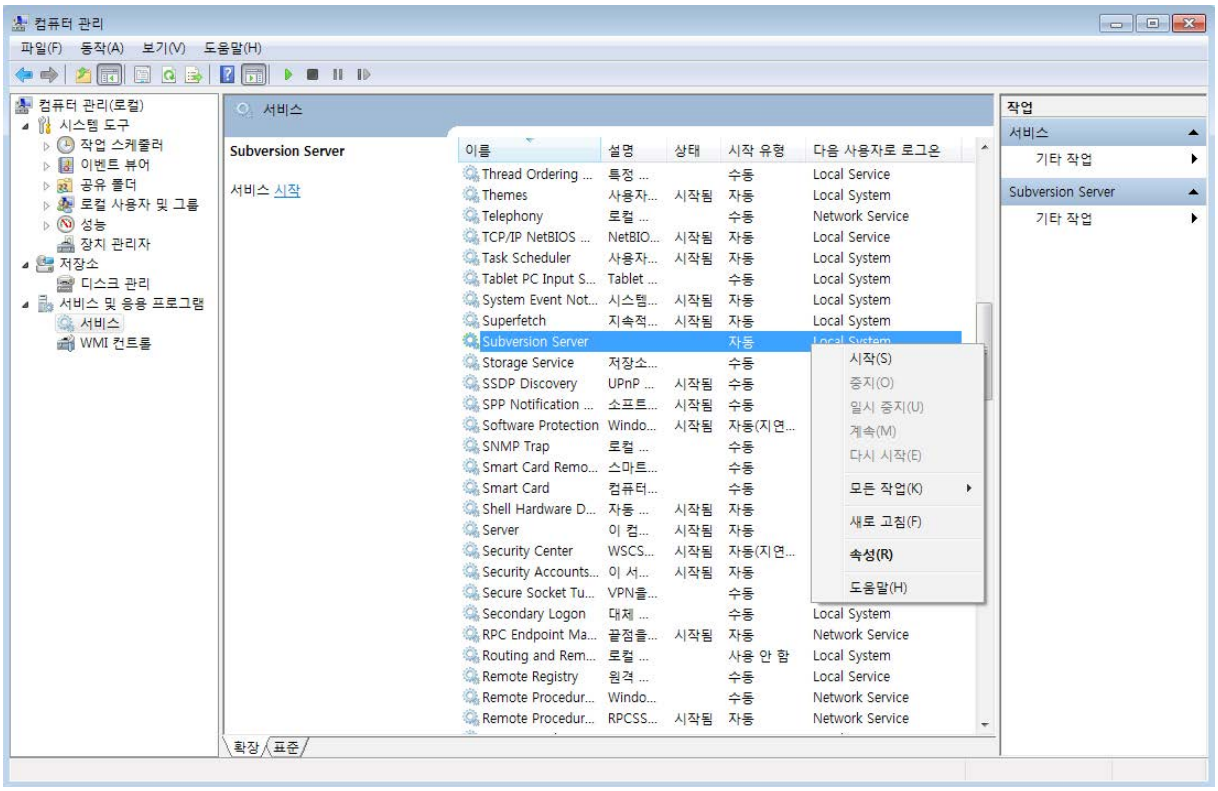
IE8.0에서는 Oracle 페이지의 메뉴 선택 시 Site Map 페이지로 이동하는 등의 문제가 발생하여 정상적인 진행이 어려울 수 있으니 IE8 이 후 버전으로 업데이트 하거나 Chrome 등의 브라우저를 사용하기를 권장한다.

```
sc create svn binpath= “\”C:\Program Files (x86)\Subversion\bin\svnserve.exe\” --service -r C:\dev\svn_repo” displayname= “Subversion Server” depend= Tcpip start= auto
```



위의 명령으로 서비스 등록이 되었다면 서비스 관리에서 윈도우 로고 – 컴퓨터에서 우클릭 – 관리 좌클릭 – 좌측 프레임에서 서비스 및 응용 프로그램 클릭 – 서비스 클릭 – 우측 프레임에서 subversion server 우클릭 하여 시작을 해준다.

최초 등록 후 시작이 되어있지 않았을 경우 시작을 해준다. 자동으로 시작되도록 옵션을 넣었으니 재부팅 시 자동으로 시작이 될 것이다.


컴퓨터 관리
파일(F) 동작(A) 보기(V) 도움말(H)
컴퓨터 관리(로컬)
시스템 도구
작업 스케줄러
이벤트 뷰어
공유 폴더
로컬 사용자 및 그룹
성능
장치 관리자
저장소
디스크 관리
서비스 및 응용 프로그램
서비스
WMI 컨트롤
서비스
Subversion Server
서비스 시작
이름 설명 상태 시작 유형 다음 사용자로 로그온
Thread Ordering ... 특정 ... 수동 Local Service
Themes 사용자... 시작됨 자동 Local System
Telephony 로컬 ... 수동 Network Service
TCP/IP NetBIOS ... NetBIO... 시작됨 자동 Local Service
Task Scheduler 사용자... 시작됨 자동 Local System
Tablet PC Input S... Tablet ... 수동 Local System
System Event Not... 시스템... 시작됨 자동 Local System
Superfetch 지속적... 시작됨 자동 Local System
Subversion Server 자동 Local System
Storage Service 저장소... 수동
SSDP Discovery UPnP ... 시작됨 수동
SPP Notification ... 소프트... 시작됨 수동
Software Protection Windo... 시작됨 자동(지연...
SNMP Trap 로컬 ... 수동
Smart Card Remo... 스마트... 수동
Smart Card 컴퓨터... 수동
Shell Hardware D... 자동 ... 시작됨 자동
Server 이 컴... 시작됨 자동
Security Center WSCS... 시작됨 자동(지연...
Security Accounts... 이 서... 시작됨 자동
Secure Socket Tu... VPN을... 수동
Secondary Logon 대체 ... 수동 Local System
RPC Endpoint Ma... 끝점을... 시작됨 자동 Network Service
Routing and Rem... 로컬 ... 사용 안 함 Local System
Remote Registry 원격 ... 수동 Local Service
Remote Procedur... Windo... 수동 Network Service
Remote Procedur... RPCSS... 시작됨 자동 Network Service
시작(S)
중지(O)
일시 중지(U)
계속(M)
다시 시작(E)
모든 작업(K)
새로 고침(F)
속성(R)
도움말(H)
확장 표준
작업
서비스
기타 작업
Subversion Server
기타 작업

3.3 저장소 생성

서버 1대에 저장소를 여러 개 생성할 수 있다. "c:\dev\svn_repo\" 아래에 저장소를 생성할 것이다. 이곳에서 저장소를 생성하여야 접근이 가능하다.

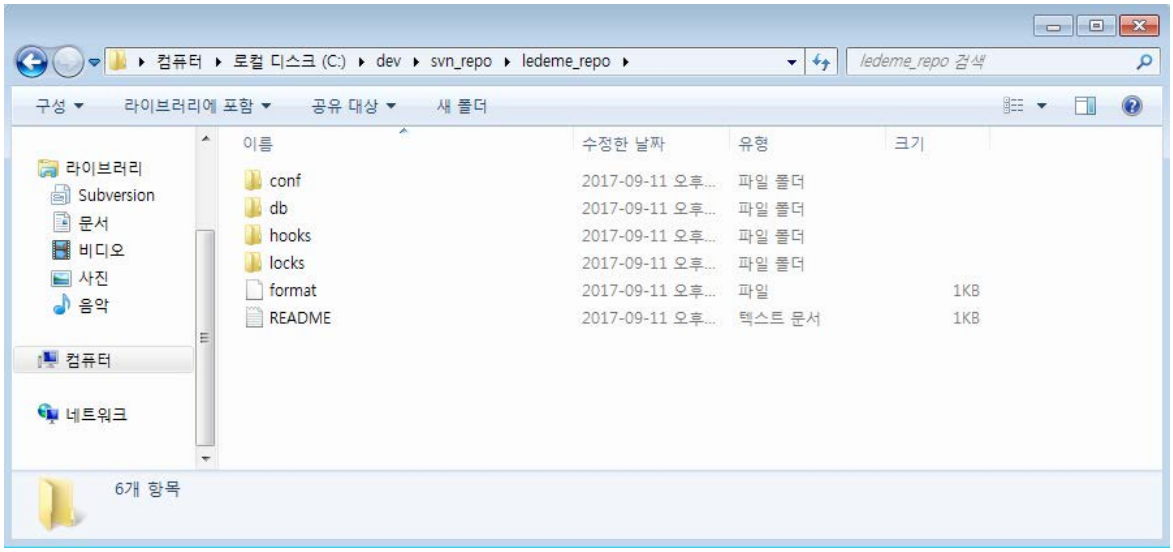
cmd 명령 프롬프트에서 아래 명령으로 저장소를 생성할 수 있다.

```
cd c:\dev\svn_repo\
svnadmin create --fs-type fsfs [저장소명]
```

```
C:\Users\LeDeMe>cd c:\dev\svn_repo

c:\dev\svn_repo>svnadmin create --fs-type fsfs ledeme_repo
```

위 명령을 실행하면 아래와 같이 저장소가 생성이 된다.

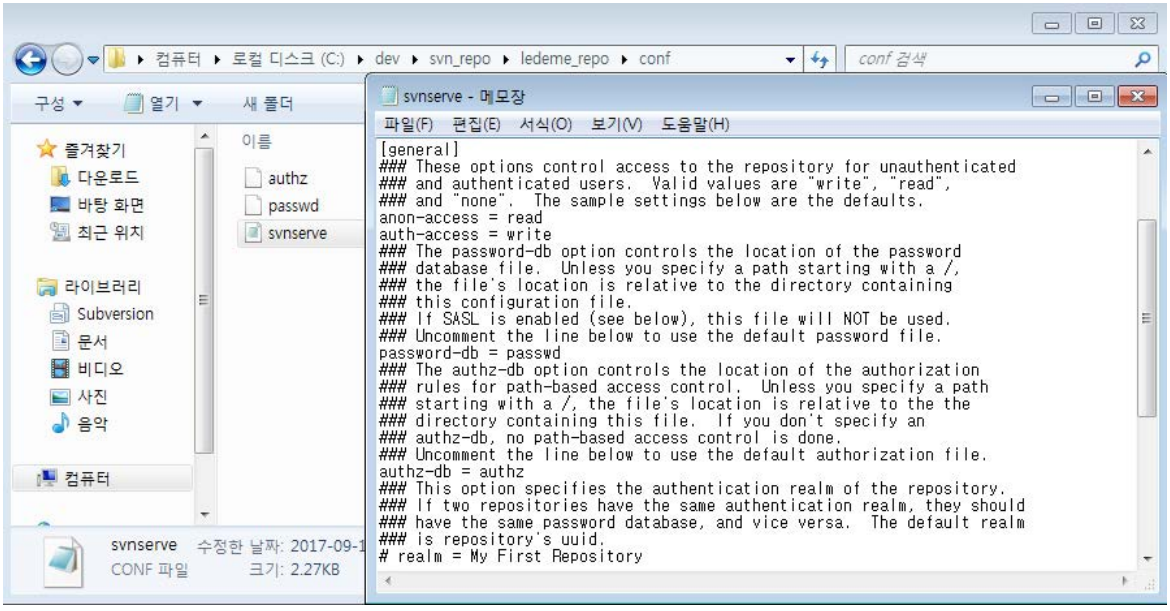


3.4 저장소 설정

다음은 생성된 저장소 설정을 할 것이다.
생성된 저장소로 이동을 하여 “conf/svnserve.conf” 파일에서 아래 주석을 풀어준다.

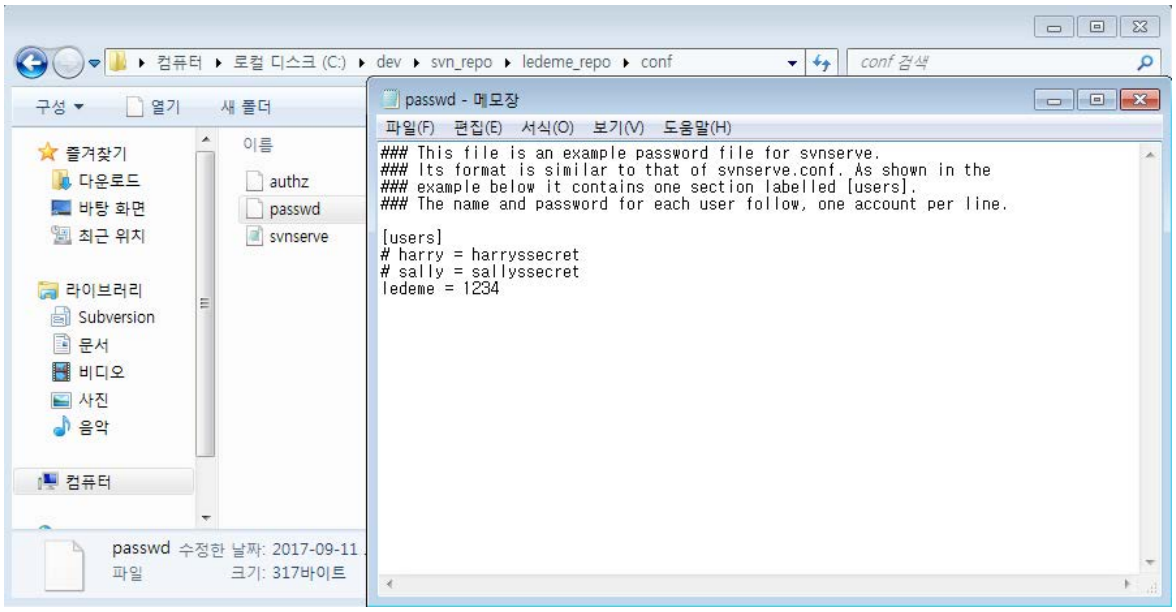
```
[general]
anon-access = none (인증 되지 않은 사용자 접근 거부. 읽기 권한은 read)
auth-access = write (인증된 사용자 쓰기 권한)
password-db = passwd (사용자에 대한 계정정보)
authz-db = authz (사용자에 대한 저장소 권한주기. optional)
```

주석을 제거한 경우 앞에 공백이 있을 경우 오류가 발생할 수 있으니 공백이 없도록 한다.



다음으로 저장소 사용자와 패스워드를 생성한다.
“conf/passwd” 파일에 사용자와 패스워드를 입력한다

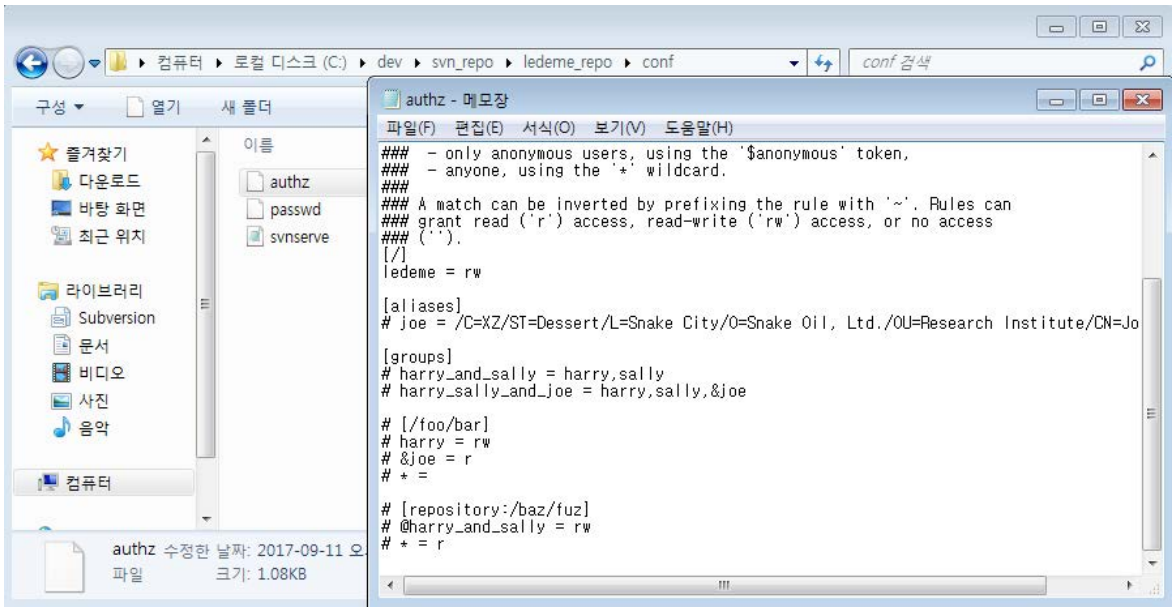
```
[users]
ledeme = 1234
```



다음으로 저장소에 대한 권한 설정을 한다.

“conf/authz” 파일을 열어서 위에서 추가한 사용자에 대한 권한을 입력한다.

```
[/]
ledeme = rw (읽기, 쓰기 권한)
```



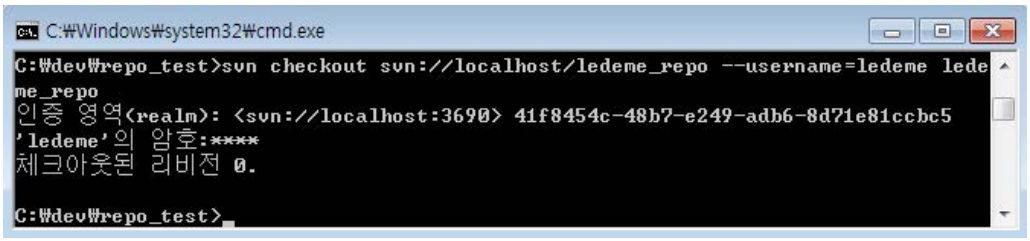
3.5 저장소 연결(체크아웃을 통한 확인)

위에서 생성한 저장소를 “c:\dev”에 “repo_test”라는 폴더를 생성하고 생성한 폴더에서 저장소 체크아웃을 하여 연결을 할 것이다.

“repo_test”라는 폴더를 생성하였으면, cmd 명령프롬프트를 열어서 아래의 명령을 실행한다.

```
cd c:\dev\repo_test
svn checkout svn://localhost/ledeme_repo --username=[사용자ID] [원격 저장소명]
```

명령을 실행하고 저장소에 설정한 계정의 암호를 물어보면, 설정한 계정의 암호를 입력한다.
아래와 같이 진행이 되었다면 정상적으로 저장소 연결이 이루어진 것이다.





CHAPTER 04

Git

4.1 Git 설치 34
4.2 Git 실행 43

CHAPTER 04 Git

Git은 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템이다.

Git의 작업 폴더는 모두 전체 기록과 각 기록을 추적할 수 있는 정보를 포함하고 있으며, 완전한 형태의 저장소이다. 네트워크에 접근하거나 중앙 서버에 의존하지 않는다.

Git은 GNU 일반 공중 사용 허가서 v2하에 배포되는 자유 소프트웨어이다.

Github 란?

Git을 사용하는 사람들을 위한 웹기반 호스팅 서비스로 저장소, 협업 공간을 제공한다. git 명령어를 사용하여 Github가 제공하는 원격저장소에 연결하고 관리한다

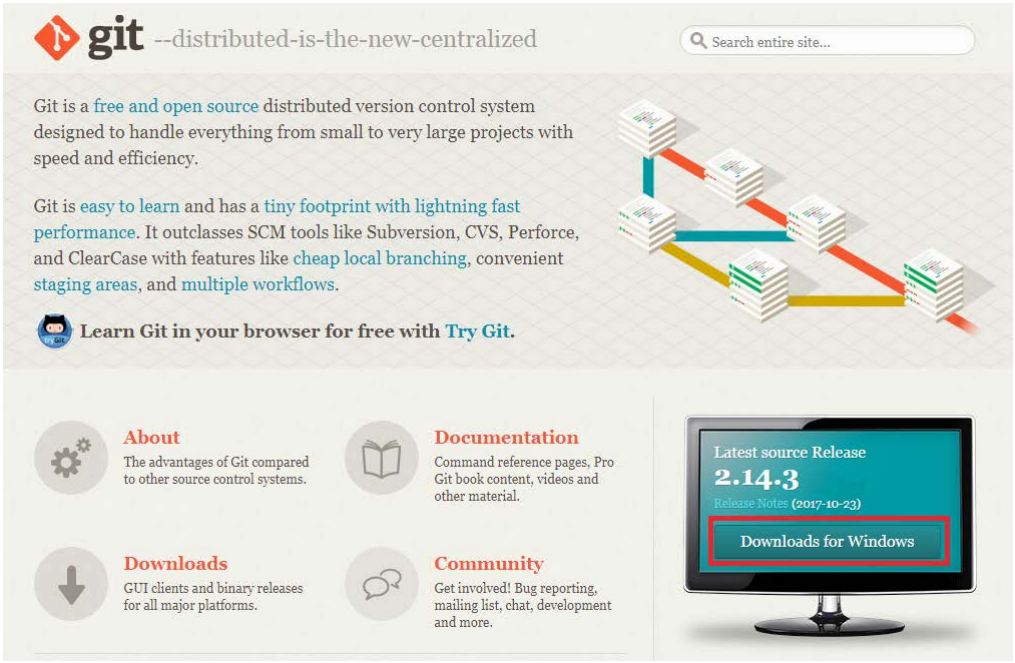
4.1. Git 설치

1) Git 다운로드

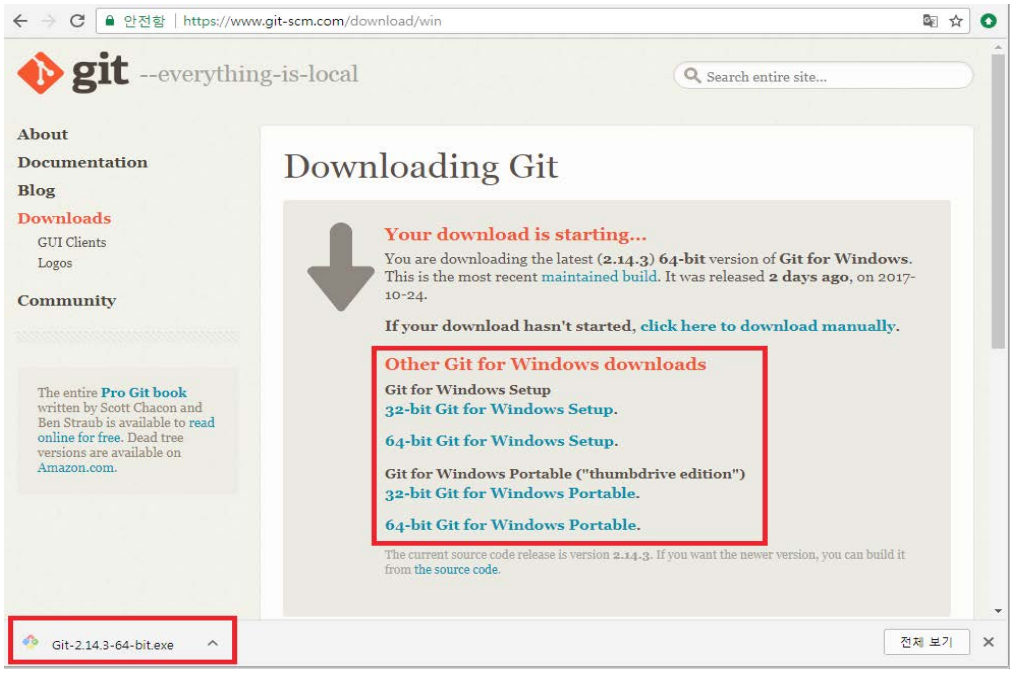
Git을 설치하기 위해서 아래 Git 홈페이지에 접속하여 파일을 다운받아 설치 할 수 있다.

https://www.git-scm.com/

Git 홈페이지에 접속하면 아래와 같은 화면이 보이며, 윈도우 설치 파일을 다운받을 수 있다.
'Downloads for Windows' 버튼을 클릭하여 다운로드를 받는다.

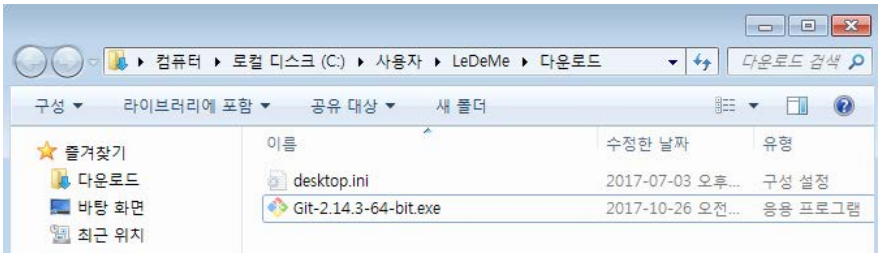


다운로드 버튼을 클릭하면 다운로드 페이지로 이동하면서 파일 다운로드를 시작한다.
다른 설치 파일을 받고자 한다면 다운로드 페이지에서 원하는 파일을 다운로드 받을 수 있다.

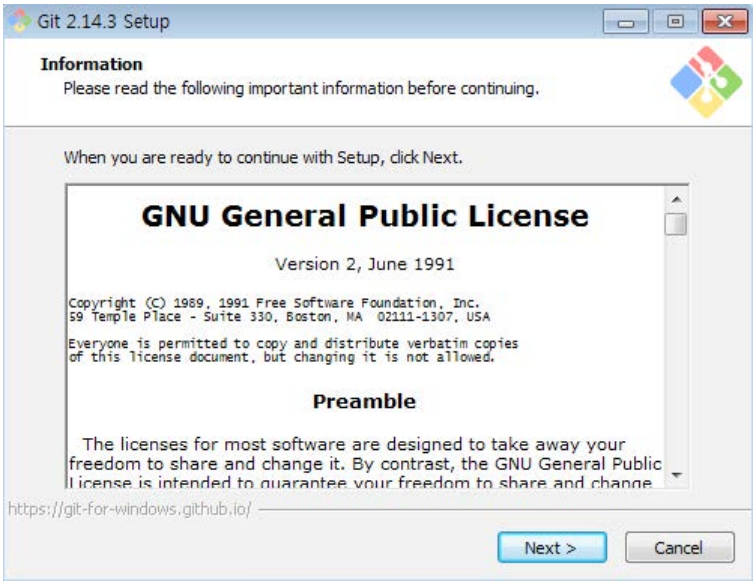


2) Git 설치

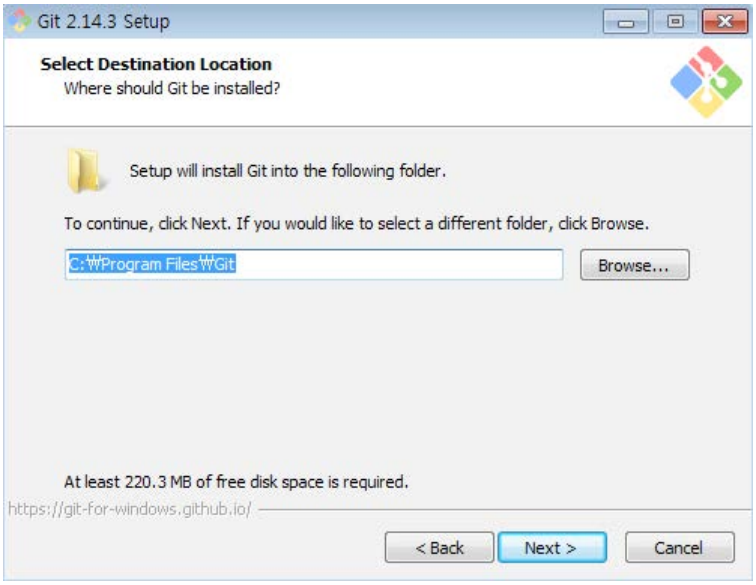
다운로드 받은 Git 설치 파일을 실행한다.



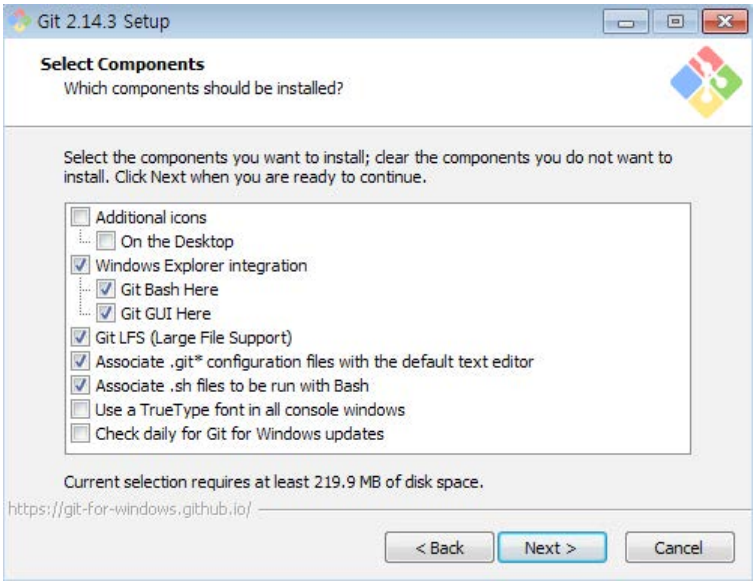
실행을 하면 아래와 같은 Setup 화면이 보인다.
라이선스 관련된 내용을 확인할 수 있으며, Next 버튼을 클릭하여 설치를 진행한다.



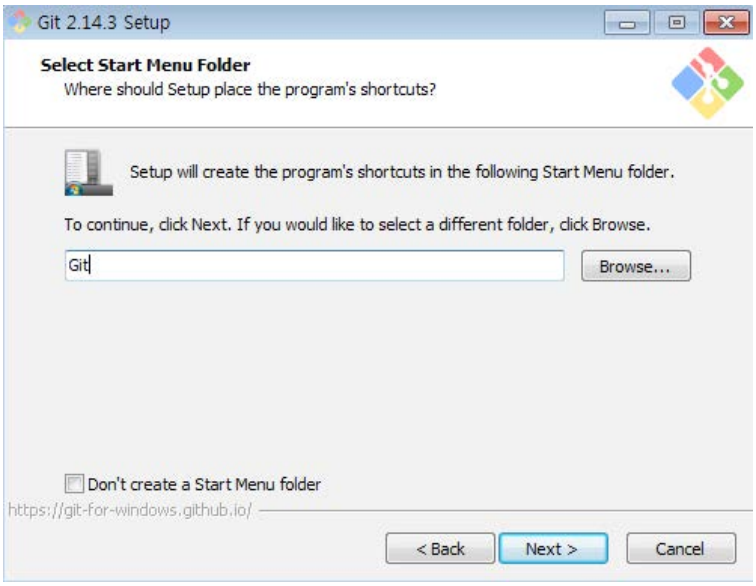
실행을 하면 아래와 같은 Setup 화면이 보인다.
라이선스 관련된 내용을 확인할 수 있으며, Next 버튼을 클릭하여 설치를 진행한다.



Components를 선택하는 화면이며, 기본으로 체크되어있는 상태로 Next를 클릭한다.
필요한 것이 있다면 체크를 하고 Next를 클릭한다.



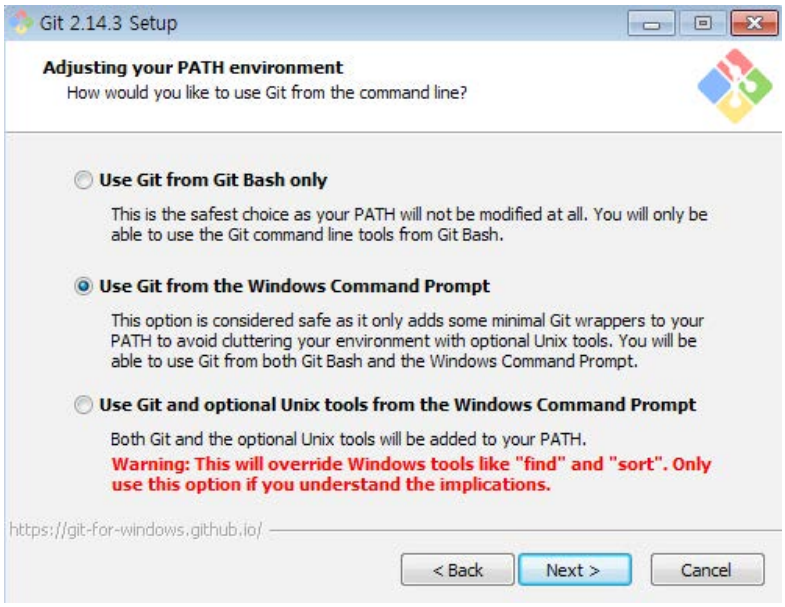
Git의 shortcut 생성 위치를 묻는 화면이며, Next 버튼을 클릭한다.



Git을 사용하기 위한 환경을 설정하는 화면이다.

- Use Git from Git Bash only
  해당 프로그램에서 제공해주는 Git Bash에서만 Git을 사용할 수 있도록 설치
- Use Git from the Windows Command Prompt
  윈도우에 명령 프롬프트에서도 Git을 사용할 수 있도록 설치
- Use Git and optional Unix tools from the Windows Command Prompt
  명령 프롬프트에서 몇 가지 유닉스 도구와 Git을 사용할 수 있도록 설치

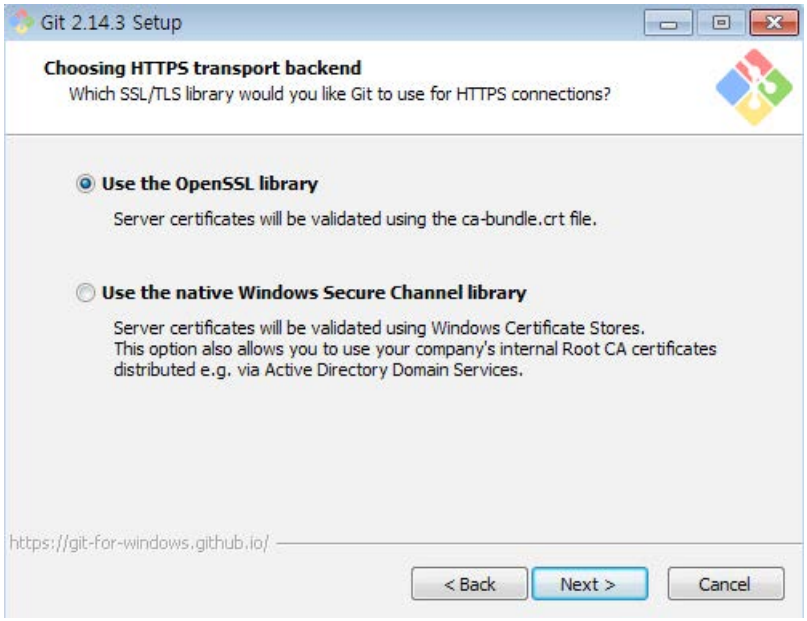
2번째 옵션을 체크하고 Next 버튼을 클릭한다.



다음은 인증서관련 설정이다.

- Use the OpenSSL library
  OpenSSL 라이브러리 사용. 사용인증서는 ca-bundle.crt 파일을 사용하여 유효성 검사
- Use the native Windows Secure Channel library
  Windows 인증서 저장소를 사용하여 서버 인증서의 유효성 검사. Active Directory 도메인 서버를 통해서 회사 내부 루트 CA인증서도 사용할 수 있음.

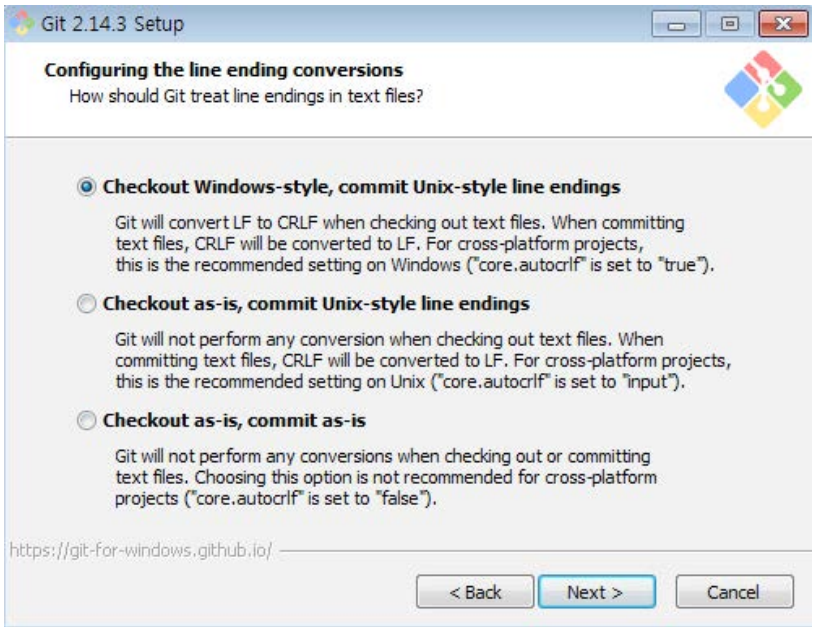
기본 선택된 Use the OpenSSL library를 선택하고 Next 버튼을 클릭한다.



다음은 checkout, commit 할 때 텍스트 엔딩을 선택하는 화면이다.

- Checkout Windows-style, commit Unix-style line endings
  Checkout 할 때는 윈도우 스타일, Commit 할 때는 유닉스 스타일로 적용
- Checkout as-is, commit Unix-style line endings
  Checkout 할 때는 스타일 변환 없고, Commit 할 때는 유닉스 스타일로 적용
- Checkout as-is, commit as-is
  Checkout, Commit 모두 스타일 변환 없음.

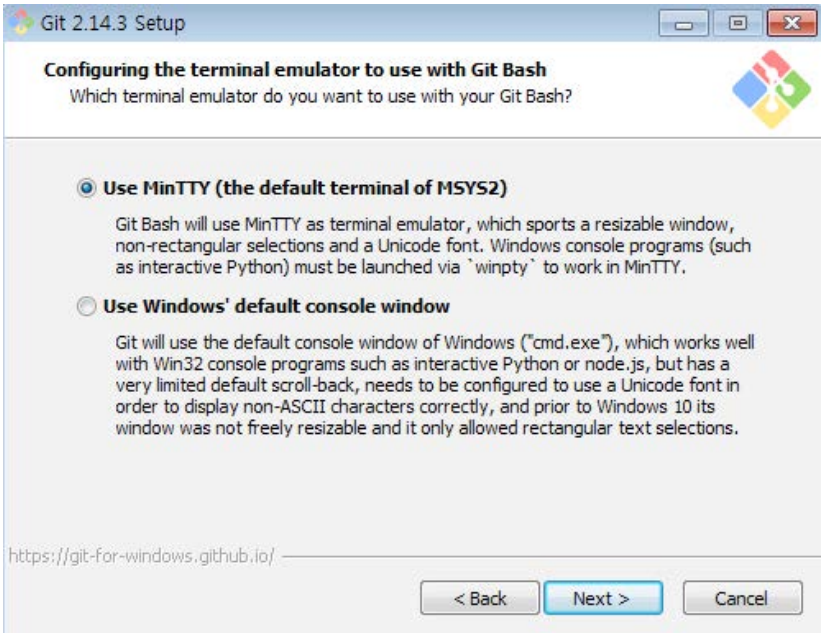
여기서도 기본 선택 되어있는 것을 변경 없이 Next 버튼을 클릭한다.



다음은 Git Bash 터미널 형식을 선택하는 화면이다.

- Use MinTTY(the default terminal of MSYS2)
- Use Windows default console window

Next 버튼을 클릭한다.



마지막으로 기타 옵션을 선택하는 화면이다.

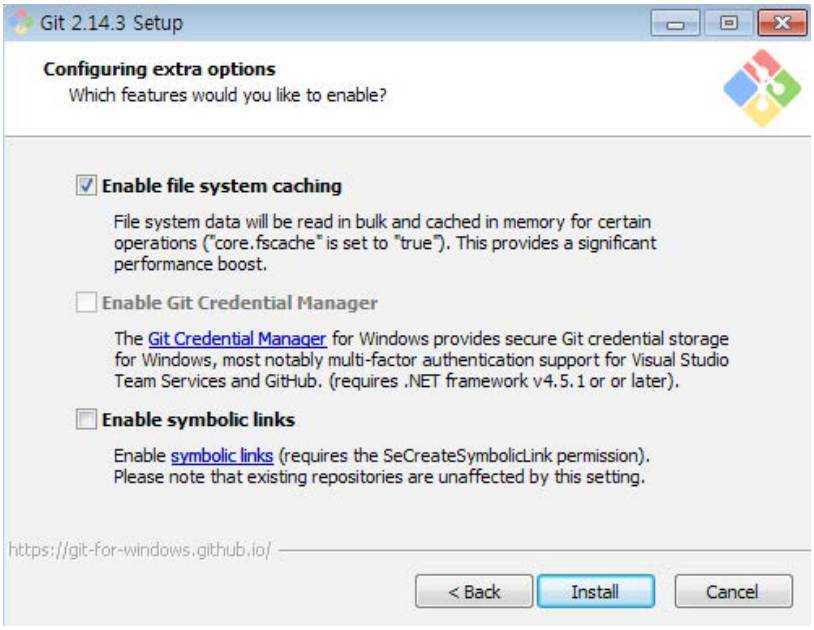
- Enable file system caching

  성능향상을 위해서 파일 시스템 캐싱 활성화
- Enable Git Credential Manager

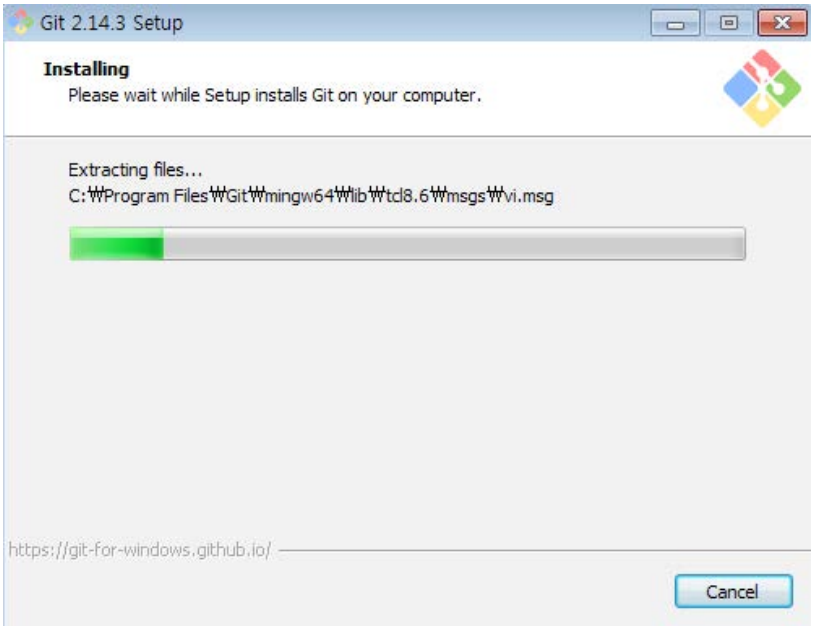
  Windows용 보안 Git 자격증명 저장소를 사용하기 위해 Git Credential Manager 활성화
- Enable symbolic links

  심볼릭 링크 활성화(기존 저장소는 영향을 받지 않음)

중복 체크가 가능하다. 기본으로 체크되어있는 상태로 Install 버튼을 클릭한다.



아래와 같이 설치가 진행된다.



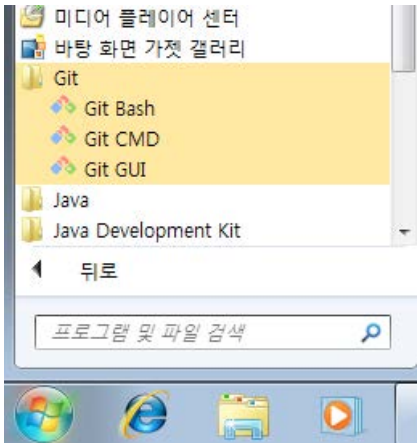
설치가 완료 되면 아래와 같은 화면이 보인다.

Finish 버튼을 클릭한다.

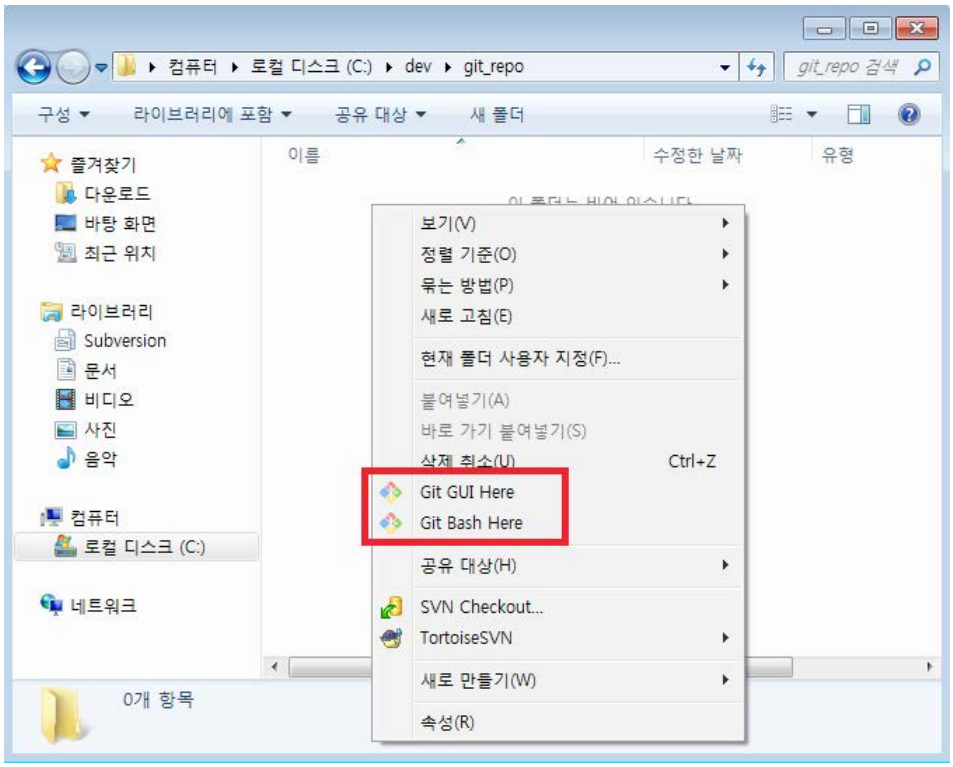


4.2 Git 실행

윈도우 시작메뉴를 확인하면 아래와 같이 3가지를 선택 실행 할 수 있다.
사용하기 편하거나 익숙한 것을 선택하여 사용하면 된다.



윈도우 폴더에서 우클릭을 하면 아래와 같이 2가지를 선택 실행 할 수 있다.
사용하기 편하거나 익숙한 것을 선택하여 사용하면 된다. (아래 화면의 경로는 git 저장소를 한곳에서 관리하기 위해서 생성한 임의의 폴더이다. 아래와 같은 경로를 동일하게 생성하여서 사용해도 되고 사용자가 원하는 경로의 폴더를 생성하여도 된다.)



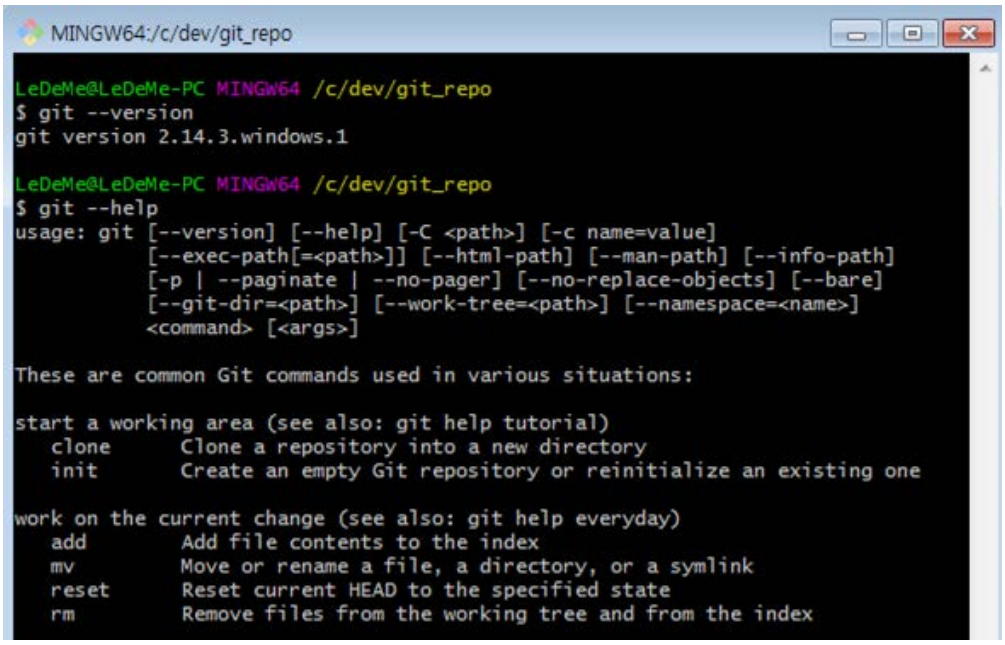
윈도우 시작메뉴를 기준으로 설명을 하겠다.

Git GUI를 클릭하면 아래와 같은 창이 나타나며 3가지 항목 중에 선택하여 사용할 수 있다.



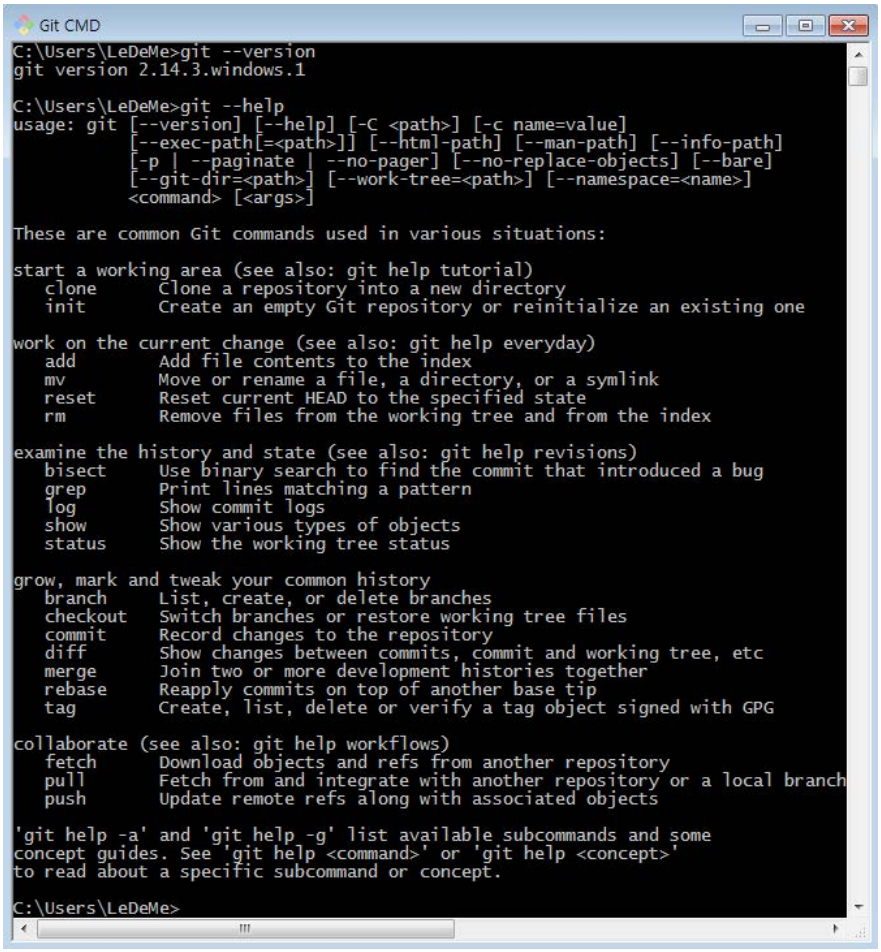
Git Bash를 클릭하면 아래와 같은 창이 나타나며 Git 을 사용할 수 있다.

아래는 Git 버전확인과 help 명령을 실행한 것이다.



Git CMD를 클릭하면 아래와 같은 창이 나타나며 Git 을 사용할 수 있다.

아래는 Git 버전확인과 help 명령을 실행한 것이다.





CHAPTER 05

Maven



5.1 다운로드 54

5.2 설치 55

5.3 Maven 프로젝트 생성 57

CHAPTER 05 Maven

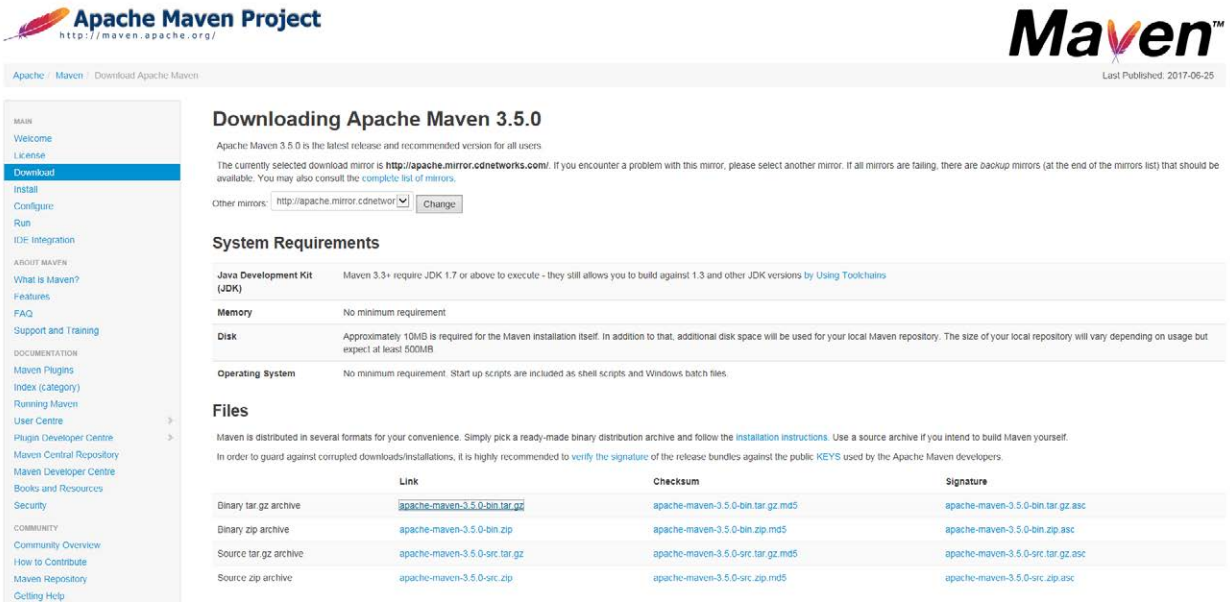
Maven은 프로젝트 객체 모델(Project Object Model)이라는 개념을 바탕으로 프로젝트 의존성 관리, 라이브러리 관리, 프로젝트 생명 주기 관리 기능 등을 제공하는 프로젝트 관리 도구이다. 또한 플러그인을 기반으로 소스 코드로부터 배포 가능한 산출물을 만들어 내는 빌드 기능 뿐만 아니라 레포팅 및 documentation 작성 기능 등을 제공한다.

5.1 다운로드

아래의 Maven 사이트에 접속을 하여 Maven을 다운받을 수 있다.

https://maven.apache.org/download.cgi#

다운로드 파일에서 Binary 아카이브 파일을 다운로드 받는다.(zip or tar.gz)



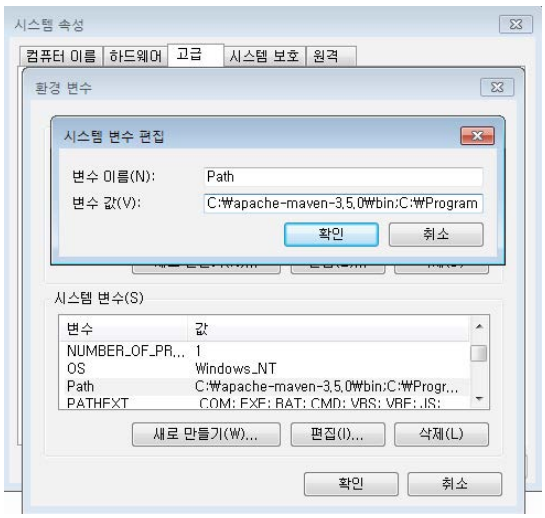
5.2 설치

다운받은 압축파일을 원하는 경로에 풀어 놓고, 해당 경로를 시스템 환경 변수에 “MAVEN_HOME”을 추가한다.

“C:\apache-maven-3.5.0” 에 압축을 풀어놓았다면, “MAVEN_HOME”을 해당 경로를 변수 값으로 입력 후 확인을 하여 저장한다.

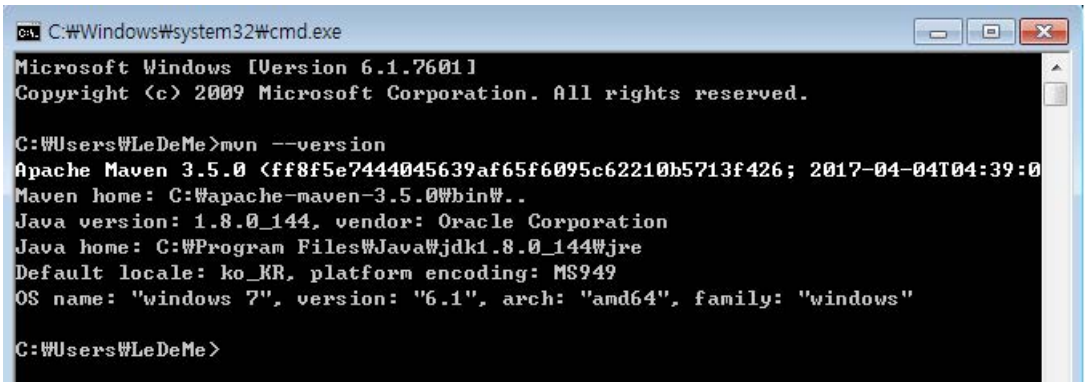
시스템 환경 변수 “PATH”에는 C:\apache-maven-3.5.0\bin 을 추가한다.(mvn 명령을 편하게 사용하기 위함)





cmd에서 아래 명령을 실행하여 maven 설정을 확인 할 수 있다.

```
mvn --version
```

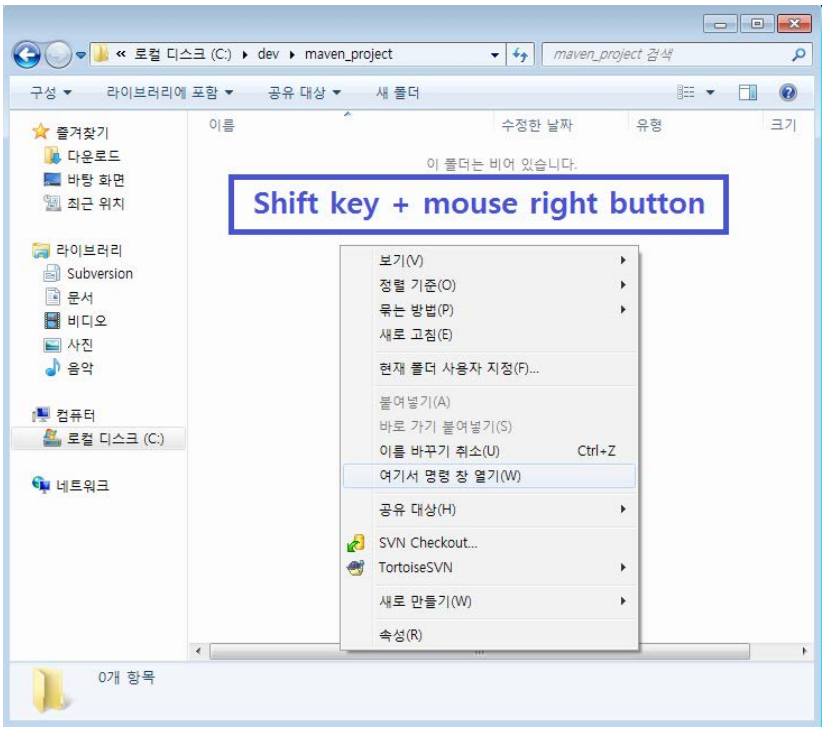


5.3 Maven 프로젝트 생성

Maven 프로젝트를 생성할 폴더를 생성하고, 생성한 폴더에서 명령을 실행하면 간단하게 생성할 수 있다. 본 가이드에서는 cmd에서 명령으로 프로젝트를 생성하는 것을 설명하지만, Eclipse 등에서도 쉽게 Maven 프로젝트를 생성할 수 있다.

먼저, Maven 프로젝트를 생성할 폴더를 생성할 것이다.
"C:\dev\" 에 "maven_project" 폴더를 생성하고, 생성한 폴더로 이동한다.
(maven_project 는 Maven 프로젝트를 한 곳에서 생성하기 위하여 임으로 생성한 폴더이며, 다른 경로 또는 원하는 곳에서 Maven 프로젝트를 생성할 수 있다.)

생성한 폴더로 이동을 하였다면, 이제 Maven 프로젝트를 생성할 것이다.
아래와 같이 이동한 폴더에서 "Shift key + 마우스 우클릭"을 하고, '여기서 명령 창 열기' 를 클릭하여 명령 창을 연다.



명령 창에서 다음의 명령을 실행하여 프로젝트를 생성할 수 있다. 아래의 명령으로 생성을 하면 Maven 프로젝트가 생성 되는 과정에서 groupId, artifactId 등을 물어보게 된다.

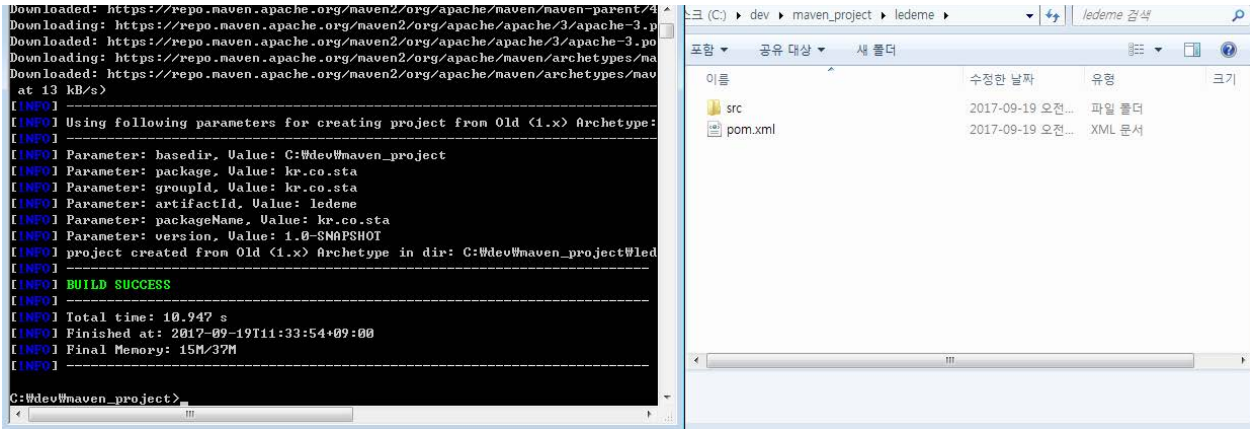
```
mvn archetype:generate
```

위의 명령을 실행하여 프로젝트를 생성 시 입력 과정을 지나치고 싶다면 생성 명령을 실행할 때 아래와 같이 미리 지정을 할 수 있다.

```
mvn archetype:generate -DgroupId=kr.co.sta -DartifactId=ledeme -DinteractiveMode=false
```

- **groupId** : 프로젝트에 속하는 그룹 식별 값. 회사, 본부, 또는 단체를 의미하는 값이 오며, 패키지 형식으로 계층을 표현한다.
- **artifactId** : 프로젝트 결과물의 식별 값. 프로젝트나 모듈을 의미하는 값이 온다.
- **version** : 결과물의 버전을 입력한다. 기본값은 1.0-SNAPSHOT 이다.
- **package** : 기본적으로 생성할 패키지를 입력한다. 별도로 입력하지 않을 경우 groupId와 동일한 구조의 패키지를 생성한다.
- **inferactive mode** : 프로젝트 생성 옵션을 묻는다. false 시 자동 생성한다.

명령을 실행을 하고 아래와 같이 정상적으로 종료가 되었다면, Maven 프로젝트가 생성될 것이다.





CHAPTER 06

Apache Ant



6.1 Ant 설치 54

6.2 Ant 환경변수 55

6.3 build.xml 57

CHAPTER 06 Apache Ant

Apache Ant는 자바 프로그래밍 언어에서 사용하는 자동화된 소프트웨어 빌드 도구이다. 유닉스나 리눅스에서 사용되는 make와 비슷하나 자바언어로 구현되어 있어 자바 실행환경이 필요하며 자바 프로젝트들을 빌드하는 표준으로 사용된다.

Make와 눈에 띄는 가장 다른 부분은 빌드를 위한 환경구성을 XML 파일을 사용한다는 점이다. 기본 빌드 파일명은 build.xml 이다.

로고는 이름(ANT)에 따라 개미 모양으로 만들어졌으나 이는 Another Neat Tool의 약어이다.

최초 빌드 도구로서 제작되었으나 점점 많은 기능이 추가되면서 빌드와 배포, 유닛 테스트 등을 포함하는 통합 툴로서 발전되고 있다.

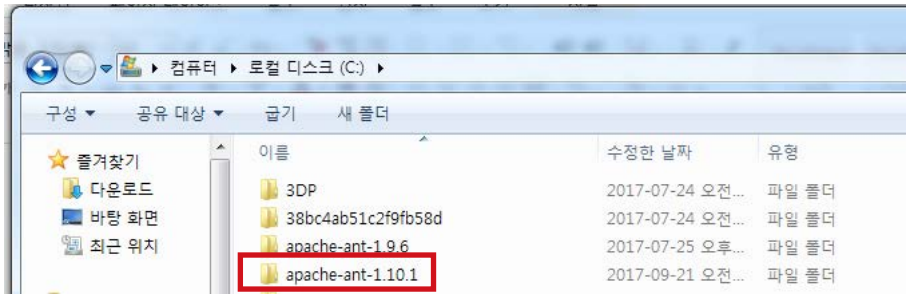
6.1 Ant 설치

Ant를 사용하려면 먼저 Ant를 컴퓨터에 설치해야 한다. 아래 URL에 접속하여 Ant를 다운받아야 한다.

http://ant.apache.org/bindownload.cgi?Prefer

- 1.10.1 .zip archive: apache-ant-1.10.1-bin.zip [PGP] [SHA1] [SHA512] [MD5]
- 1.9.9 .zip archive: apache-ant-1.9.9-bin.zip [PGP] [SHA1] [SHA512] [MD5]
- 1.10.1 .tar.gz archive: apache-ant-1.10.1-bin.tar.gz [PGP] [SHA1] [SHA512] [MD5]
- 1.9.9 .tar.gz archive: apache-ant-1.9.9-bin.tar.gz [PGP] [SHA1] [SHA512] [MD5]
- 1.10.1 .tar.bz2 archive: apache-ant-1.10.1-bin.tar.bz2 [PGP] [SHA1] [SHA512] [MD5]
- 1.9.9 .tar.bz2 archive: apache-ant-1.9.9-bin.tar.bz2 [PGP] [SHA1] [SHA512] [MD5]
- 1.10.1 .tar.xz archive: apache-ant-1.10.1-bin.tar.xz [PGP] [SHA1] [SHA512] [MD5]

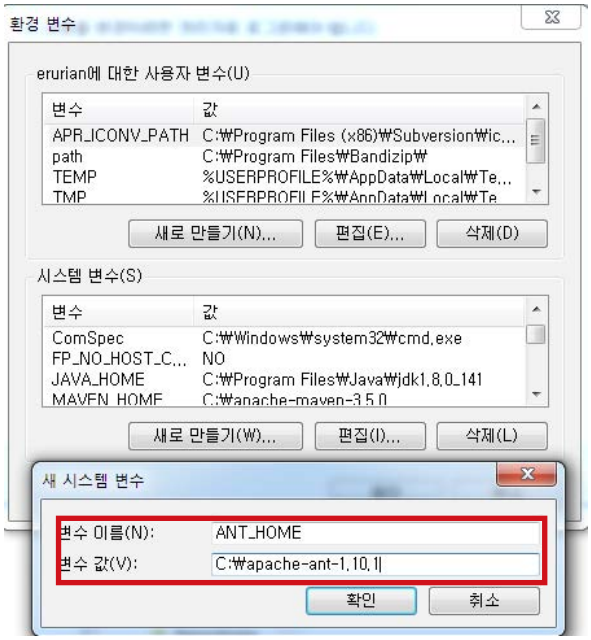
해당 파일을 다운받은 후 압축을 해제하고, 압축 해제된 파일을 임의의 위치에 이동한다.
편의상 C드라이브 하위에 이동하는걸 추천한다.



6.2 Ant 환경변수

Ant의 설치가 완료되었으면, Ant가 정상적으로 실행되도록 환경변수를 지정해 주어야 한다.
순서는 아래와 같다.
"내컴퓨터 마우스 우클릭 〉 속성클릭 〉 고급 시스템 설정 클릭 〉 고급 탭 선택 〉 환경변수" 클릭
환경변수 팝업이 발생하면 아래 시스템 변수에서 새로 만들기 버튼을 클릭하고 아래와 같이 입력 뒤 확인 버튼을 클릭한다.

변수 이름 : ANT_HOME
변수 값 : C:\apache-ant-1.10.1 (Ant가 설치된 경로)

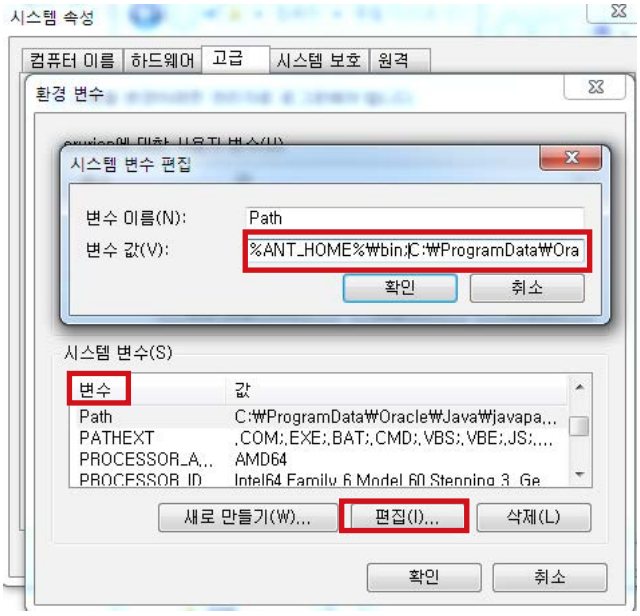


ANT_HOME 추가가 완료 되었으면 아래의 경로를 Path에 추가 해야 한다.

시스템변수의 Path 항목 선택 후 편집버튼을 클릭 후 Path 항목에 아래 경로를 추가한다

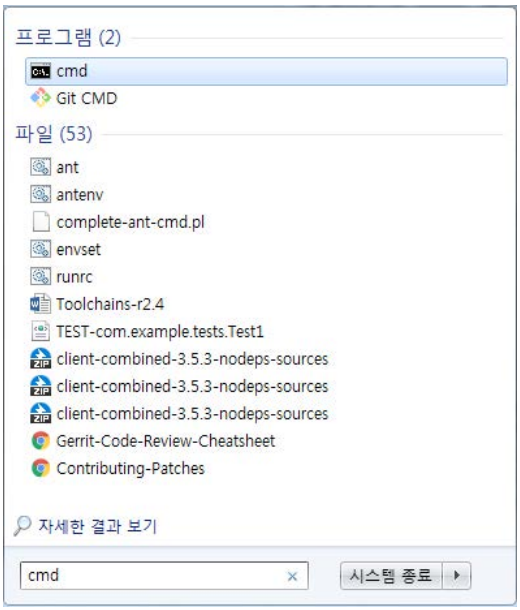
```
%ANT_HOME%\bin;
```

기존 Path에 추가된 환경변수를 삭제하지 않고, Home 키를 눌러 가장 앞에 위 환경변수를 추가해 주면 된다.
추가 후 확인 버튼을 클릭하여 환경변수 설정을 마무리한다.

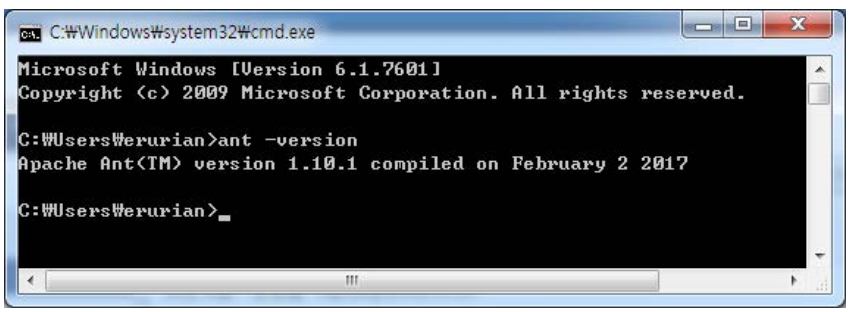


위 설치가 완료되면 아래와 같이 Command 창에서 확인이 가능하다.

시작(or 윈도우키)를 누른뒤 검색창에 cmd 를 검색하여 Command 창을 활성화 한다.



ant -version 입력 후 엔터키를 눌러 아래와 같이 Ant version 이 표시되면 정상적으로 설치가 완료된다.



6.3 build.xml

Apache Ant로 빌드할 때 필요한 필수파일로, 빌드에 대한 내용을 넣어주는 "build.xml" 파일 작성 방법을 알아보자.

〈 build.xml 예제 〉

```
<?xml version="1.0"?>
<!-- project -->
<project name="JUNIT" default="main" basedir=".">

<!-- property -->
    <property name="test.class.dir" location="bin" />
    <property name="test.report.dir" location="C:\Users\testreport" />

<!-- path -->
    <path id="junit.class.path">
        <fileset dir="selenium-java-3.5.3">
        <include name="**/*.jar"/>
        </fileset>
    </path>
<!-- target -->
    <target name="makedir">
        <mkdir dir="${test.report.dir}" />
</target>
<target name="junit" depends="makedir">
    <junit printsummary="on" fork="true" haltonfailure="no" failureproperty="test.failed">
        <classpath refid="junit.class.path" />
        <classpath>
            <pathelement location="${test.class.dir}"/>
        </classpath>
        <formatter type="xml" />
        <batchtest todir="${test.report.dir}">
            <fileset dir="${test.class.dir}">
                <include name="**/**.class" />
            </fileset>
        </batchtest>
    </junit>
</target>
<target name="main" depends="junit">
        <description>Main target</description>
    </target>
</project>
```

위 예제 파일을 보면 xml 파일로 쓰여지며, 하나의 project와 하나 이상의 target으로 구성된다. Target은 업무단위라고 볼 수 있는데, 세부 내용으로 다양한 task 가 들어갈 수 있으며 각각의 업무가 수행되기 위한 선행업무 (target) 도 지정할 수 있다.

#project

build file에 반드시 들어가는 필수단위로, 단 하나만 존재할 수 있다.

속성값	설 명
Name	프로젝트 이름. Eclipse 내의 ant view에서는 해당 이름으로 표시되어서 각각의 빌드 파일을 구분할 수 있음
Default	target 지정이 되지 않은 경우 기본으로 수행되는 target 값
basedir	모든 경로를 계산하기 위한 기본 경로. property에서 basedir 속성 설정하면 오버라이드 되지만, 없는 경우 project의 basedir 속성을 참조하게 됨. 상대경로는 빌드파일이 들어있는 폴더를 기준으로 결정

#target

빌드에 필요한 단계별 과정에 해당되는 단위. 각 과정은 상세한 수행내역을 가지고 있다.
각 과정은 다른 과정과의 의존성을 가지고 있어서 이 의존성에 의거하여 수행순서가 정해진다.
이 의존성을 나타내는 속성이 depends 이다

```
〈target name="A"/〉
〈target name="B" depends="A"/〉
〈target name="C" depends="B"/〉
〈target name="D" depends="C,B,A"/〉
```

Target D를 실행시킨다고 가정했을 경우에, 의존성 관계에 따라 C가 먼저 수행된다.
(depends 대상이 여러 개인 경우, 왼쪽에서 오른쪽 순서로 수행됨)
C가 실행되려면 B가 먼저 선행되어야 하고, B를 실행하기 위해서는 A가 선행되어야 한다.
Target D에서 선언한 depends의 C, B, A가 아니라 각 target의 선행단계가 우선적으로 수행되므로 결국 A-〉 B -〉 C -〉 D의 순서로 target이 실행된다.
수행 순서를 결정하기 위한 다양함 attribute가 있지만 보편적으로 depends를 많이 사용한다.

#task

task는 target에 속하는 더 작은 일의 단위 이다. (project 〉 target 〉 task)
task가 될 수 있는 것은 다양하게 있는데, 위의 예시에서 사용하는 내용을 인용하면 clean 업무를 하고자 하는 경우 delete를 사용해서 기존의 classes 파일을 지우도록 상세 업무를 주고 compile을 할 때는 지웠던 classes 폴더를 다시 만들어 준 후에, javac를 이용하여 compile을 수행한다.

다양한 task중 가장 많이 사용하는 목록은 아래와 같다.

Task 명령어	설 명
copy	복사를 위한 명령어. fileset 설정을 통해 복사할 대상 선택
delete	삭제를 위한 명령어. fileset 설정을 통해 삭제할 대상 선택
javac	java compile 명령어
javadoc	javadoc을 사용한 API 문서 설정
mkdir	신규 폴더 생성
jar (or tar, zip)	Jar 또는 tar, zip을 통한 압축파일 생성

예시에서 보다시피 다른 task 명령어들은 직관적인 동작만 수행하고 있으므로 fileset 사용법만 안다면 쉽게 사용이 가능하지만, javac의 경우에는 attribute가 많아 따로 설명이 필요하다.

#javac

java source tree를 compile 하는 명령어로, 주요 속성은 아래와 같다.

속성명	설 명
srcdir	java 파일이 위치한 경로를 지정
destdir	컴파일 된 class 파일이 위치할 경로
encoding	소스 파일의 인코딩 설정
debug	컴파일 시 디버그 정보를 보여줄 것인지 설정하는 것 (기본값 off)
optimize	컴파일 시 소스 최적화 여부 설정 (기본 값 off) JDK 1.3 이후로는 해당 플래그가 필요 없음
deprecation	JDK 버전 업데이트 등의 이유로 사용 중단된 소스 코드(deprecation)를 컴파일 할 것인지를 물어보는 것
verbose	컴파일 시 상세한 진행상황을 보여줄 것인지를 설정 (기본값 no)
includeAntRuntime	Classpath에 ant의 runtime 라이브러리를 포함시킬 것인지를 확인 build.sysclasspath가 설정되어있지 않으면 기본값이 yes apache ant 매뉴얼에서는 false 설정을 권장
failonerror	빌드에 실패하는 경우 컴파일 에러를 표시할 것인지 묻는 것 (기본값 true)



CHAPTER 07

Findbugs



7.1 설치 62

7.2 실행 63

7.3 분석 64

CHAPTER 07 Findbugs

Findbugs는 Java로 작성한 프로그램의 잠재적인 결함을 찾아주는 정적 분석 도구이다.

라이선스는 GNU(Lesser GNU Public License) 그리고 로고는 매릴랜드 대학(The University of Maryland)이 상표권을 가지고 있다고 한다.

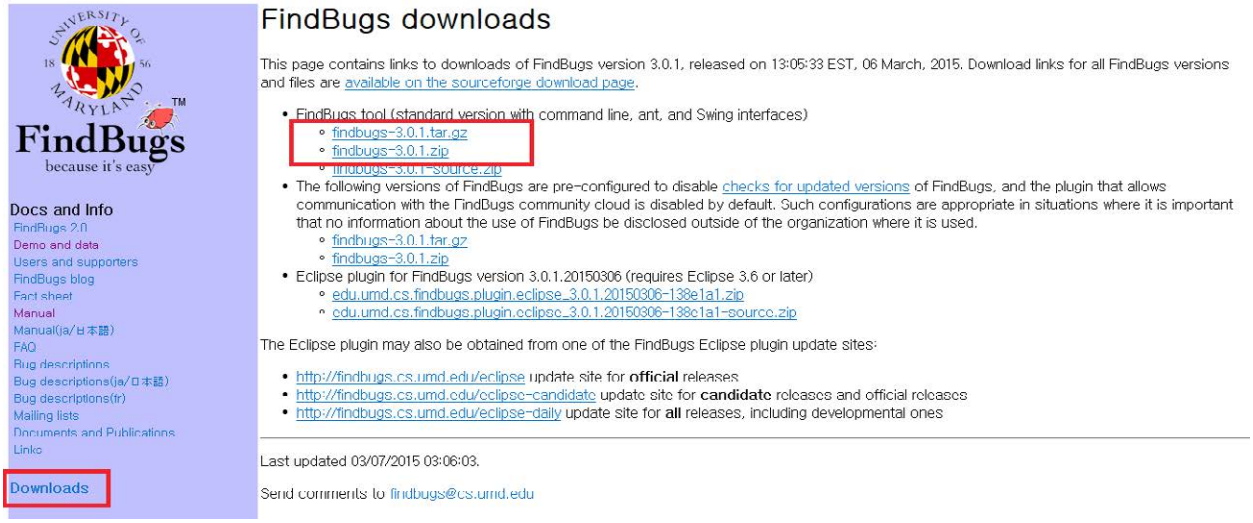
직접 프로그램을 설치 할 수도 있고, 이클립스 플러그인을 설치하여 사용할 수 있다.

7.1 설치

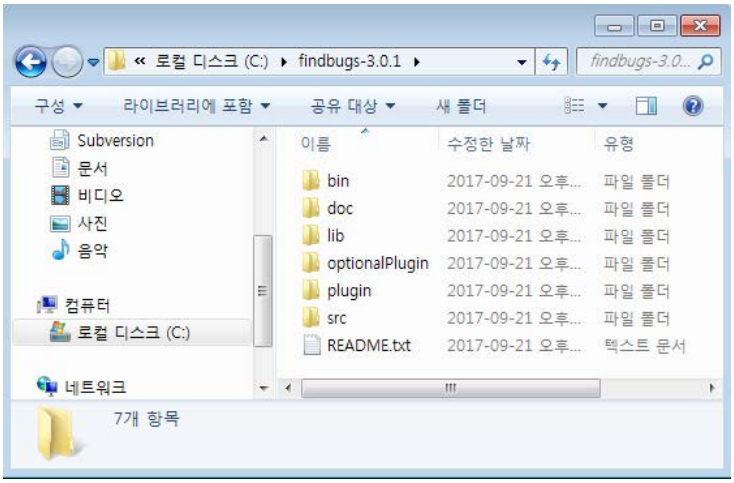
아래의 사이트에 가면 Download 받을 수 있다. 바이너리 파일(zip or tar.gz)을 다운로드를 받고 원하는 디렉토리에 풀면 되며, 별다른 설치과정은 없다.

http://findbugs.sourceforge.net

현재 최신 버전은 3.0.1 이며, JRE(or JDK) 1.7.0 이상이 필요하다.

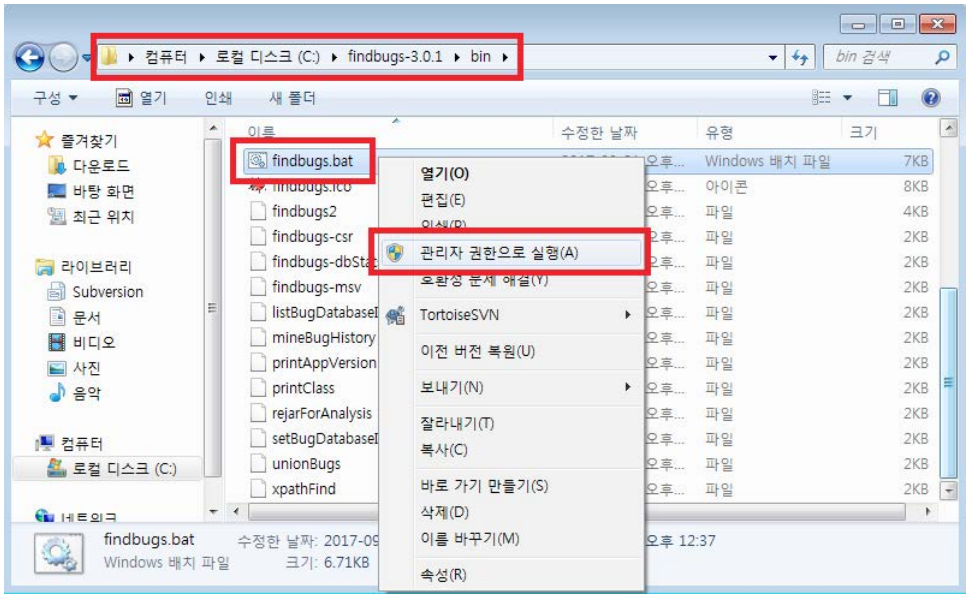


"findbugs-3.0.1.zip" 을 다운로드 받고 아래와 같이 압축을 풀면 끝이 난다.

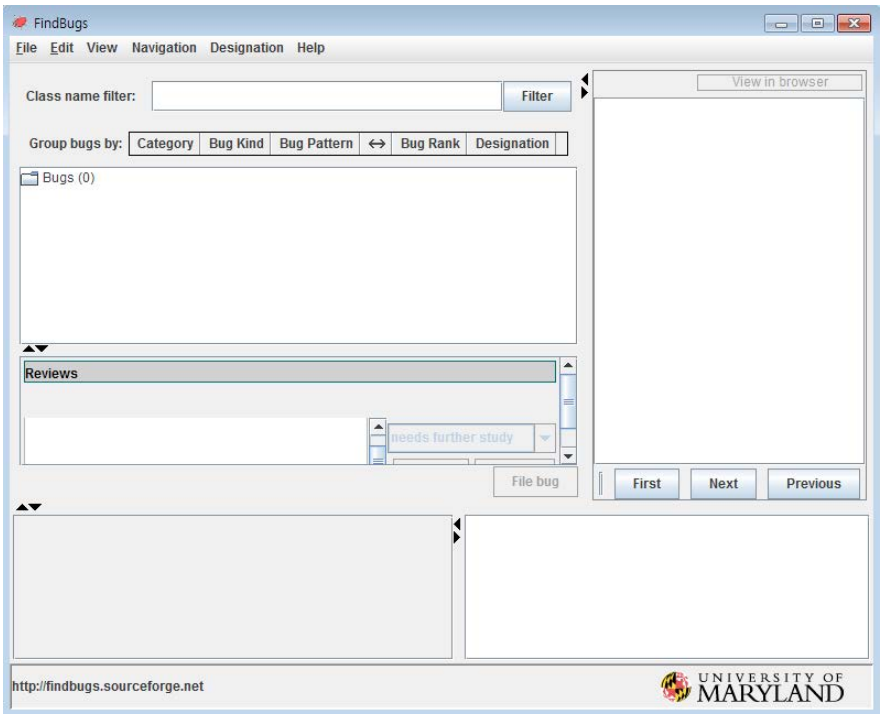


7.2 실행

실행하기 위해서는 Ja3va 2 Standard Edition 1.5 이상이 필요하다. GNU/Linux, Windows, and MacOS X platforms.에서 사용할 수 있으며, 512MB 이상의 메모리가 필요하다.
설치한 디렉토리의 bin 디렉토리에 가면 findbugs.bat(windows)가 있다. 이를 우클릭하여 관리자권한으로 실행한다. 관리자 권한으로 실행을 하지 않으면 경고(warning)이 발생할 수 있다.



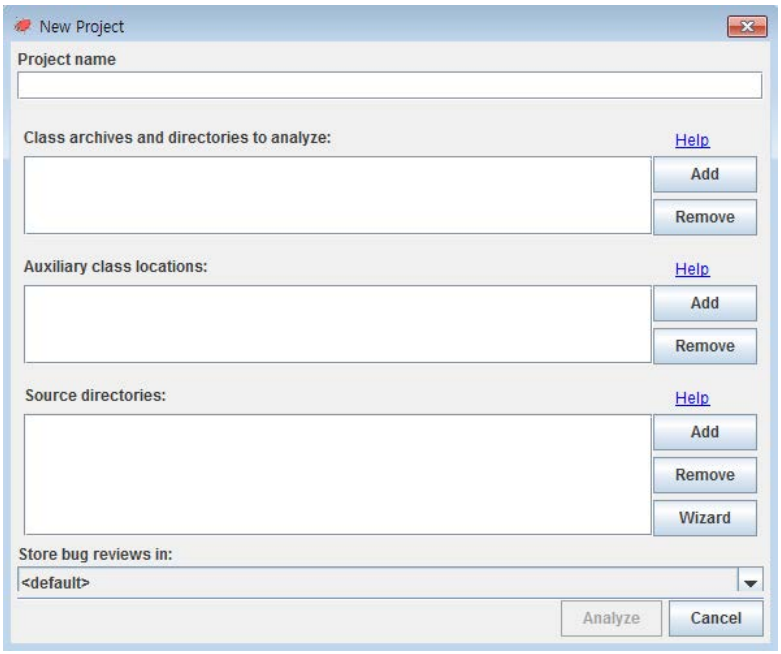
"findbugs.bat"를 관리자 권한으로 실행하면 아래와 같은 화면이 나온다.



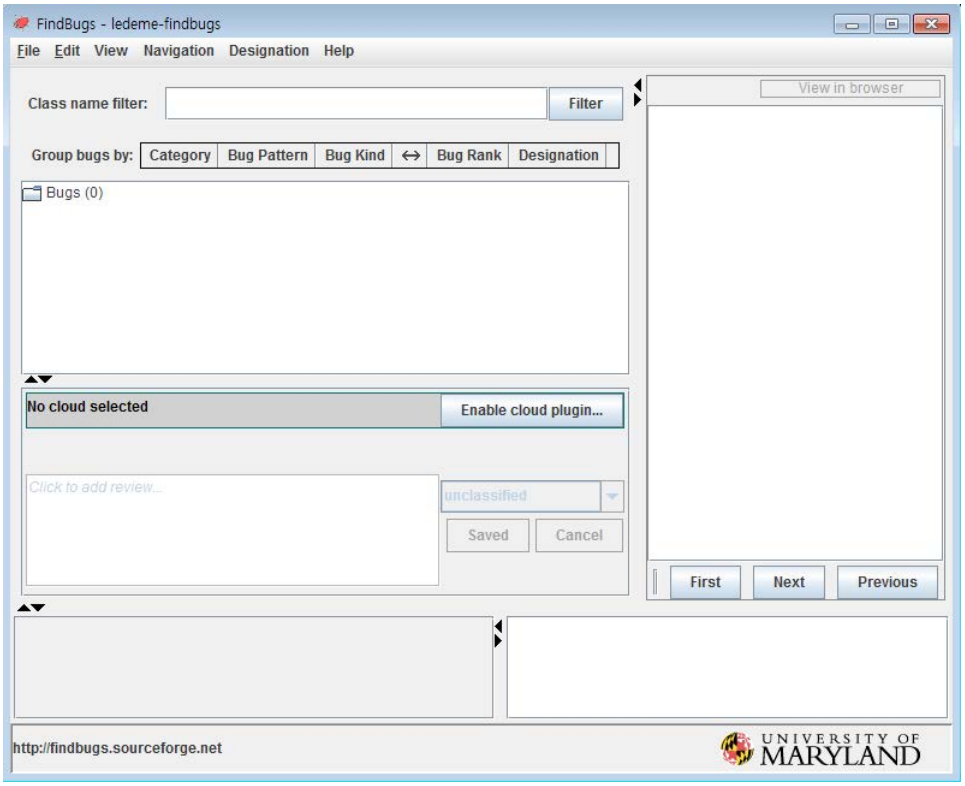
7.3 분석

먼저 프로젝트를 만들어야 한다. "File 〉 New Project" menu 를 선택하면 프로젝트 생성을 위한 창을 볼 수 있다. 아래 항목에 필요한 것들을 지정한다.

- Class archives and directories to analyze : 분석할 클래스가 있는 아카이브(jar)나 디렉토리
- Auxiliary class locations : 분석 클래스가 사용하는 클래스의 위치(jar 또는 디렉토리) – 분석대상은 아님
- Source directories : 분석 클래스의 소스가 있는 디렉토리(분석결과를 소스와 보고 싶다면 지정)

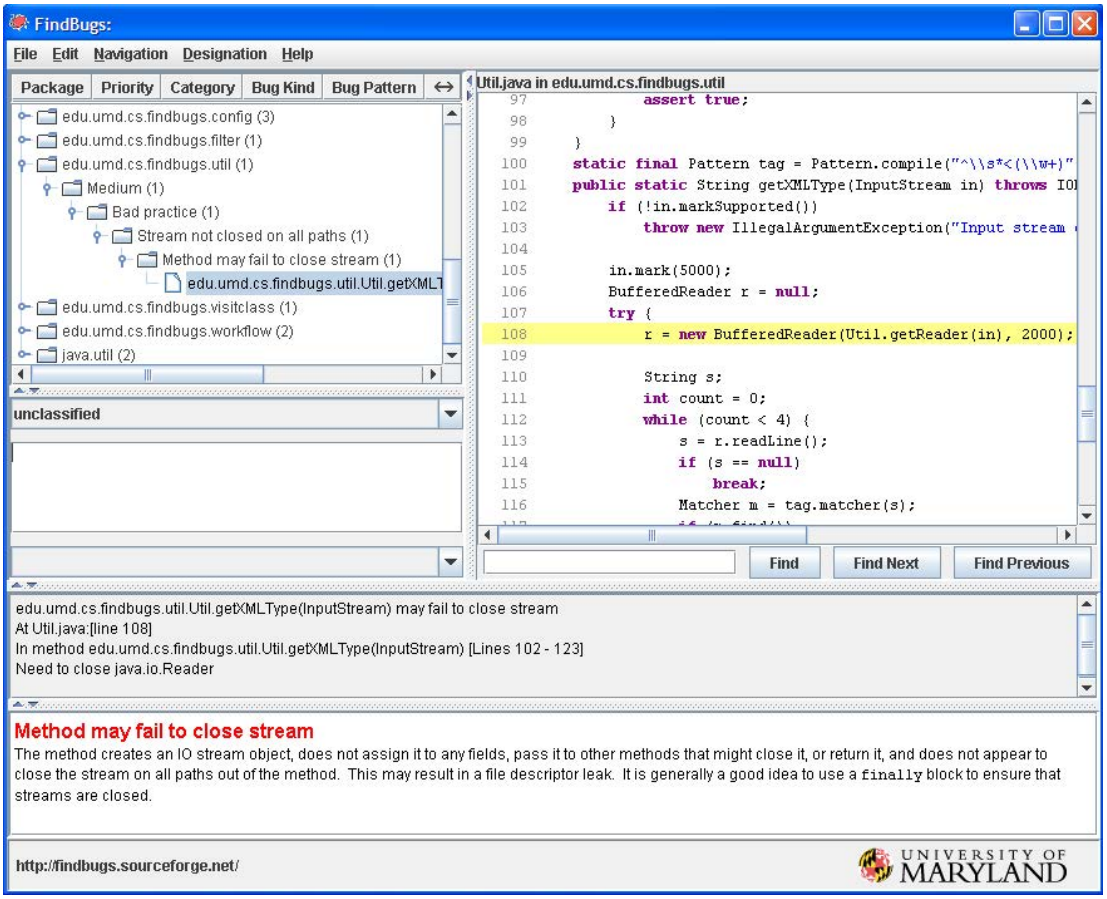


"Analyze" 버튼을 클릭하면 분석이 시작되고 분석 결과가 나타난다.



검출한 버그가 있다면 버그를 선택하면 버그 가능성이 있는 부분의 소스 위치 그리고 버그 가능성에 대한 설명을 볼 수 있다.

아래의 이미지는 Findbugs 메뉴얼에 나와있는 버그 발견 시 보여지는 내용이다.



CHAPTER **08**

Junit



8.1 Maven Junit 설정 **69**

8.2 Junit 사용하기 **70**

CHAPTER 08 Junit

Junit은 자바용 단위 테스트 도구이다. 오픈 소스이며, Maven, Eclipse등에 기본 포함되어 있기 때문에 별다른 추가 설치 없이 사용할 수 있다.

Junit 공식 사이트는 http://junit.org/ 이며, Junit 관련된 정보를 얻을 수 있다.

Junit.은 기본적으로 소스 코드의 메소드를 대상으로 테스트한다. 메소드라는 하나의 단위(Unit)를 대상으로 테스트하기 때문에, 이런 테스트를 보통 단위 테스트(Unit Test)라고 한다. 단위 테스트는 주로 개발자가 자기 코드를 검증하는 데 사용한다.

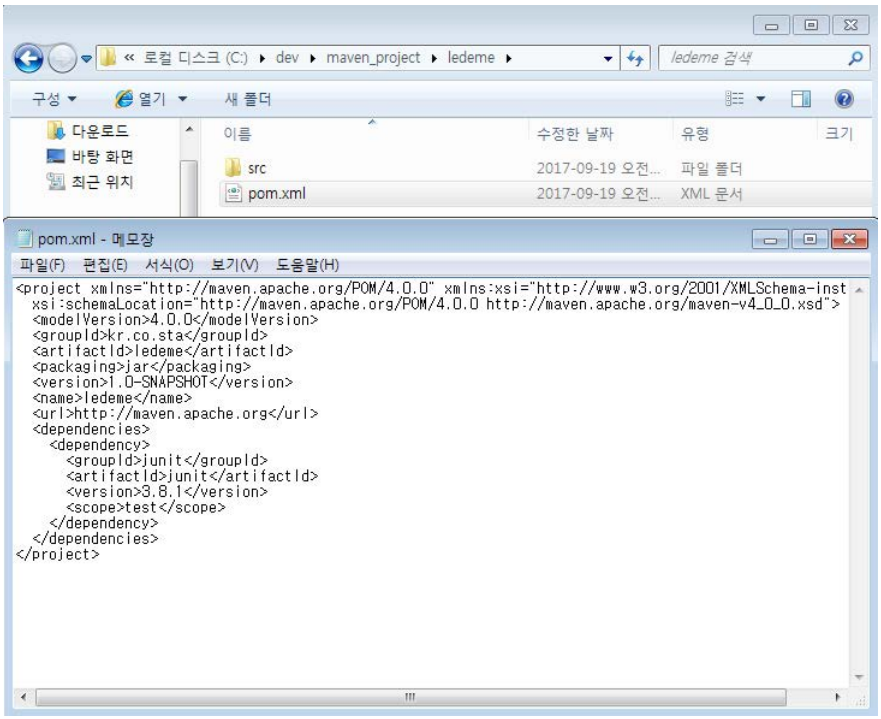
본 가이드에서는 Maven 프로젝트에서 Junit을 사용하는 방법에 대하여 설명할 것이다.

8.1 Maven Junit 설정

Maven에서 Maven 프로젝트를 생성하고, 생성된 프로젝트의 pom.xml 파일 수정하여 설정을 한다.

Maven 프로젝트를 새로 생성하지 않고, 앞서 생성한 Maven 프로젝트로 설명을 하겠다.

Maven 프로젝트 폴더로 이동을 하고, "pom.xml" 파일을 연다.(앞에서 생성한 Maven 프로젝트 경로 C:\dev\maven_project\ledeme)



"pom.xml"의 dependencies 에 Junit 이 이미 추가 되어있는 것을 확인할 수 있다.

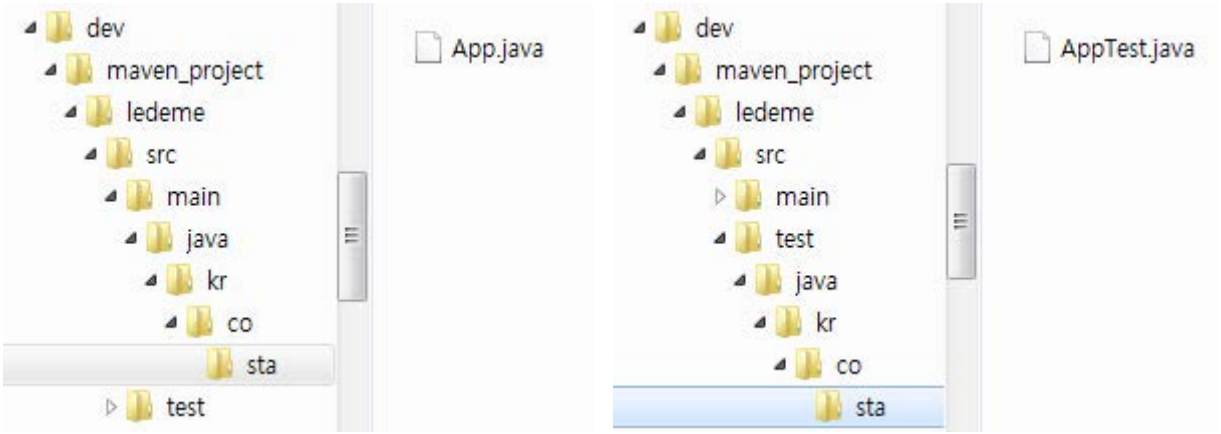
만약, Junit이 추가되어있지 않다면 Junit을 dependencies 에 추가해 준다.

```
〈dependency〉
    〈groupId〉junit〈/groupId〉
    〈artifactId〉junit〈/artifactId〉
    〈version〉3.8.1〈/version〉
    〈scope〉test〈/scope〉
〈/dependency〉
```

기본으로 설정되어 있는 버전으로 실행하면 Juint4 테스트 코드로 작성된 테스트 파일을 실행하면 오류가 발생할 수 있다. Juint4 테스트 코드로 작성되었다면 pom.xml 의 Junit 버전을 4.X 등의 버전으로 변경하여 실행을 하도록 한다.

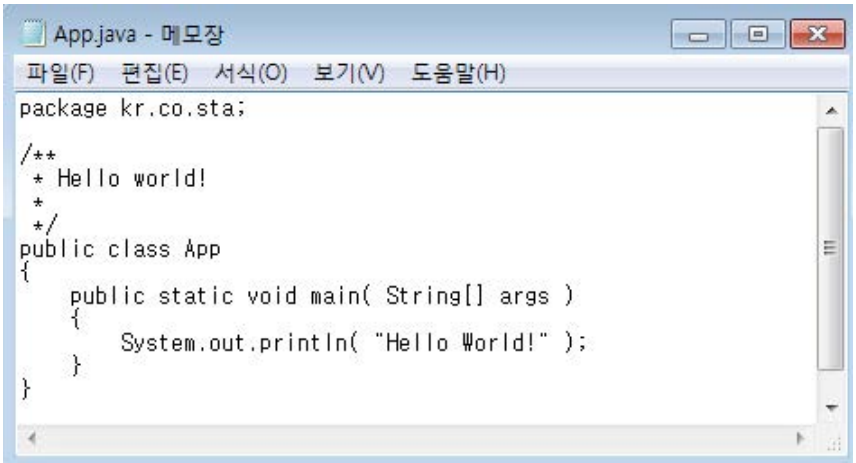
8.2. Junit 사용하기

앞서 생성한 Maven 프로젝트를 보면 아래와 같이 “src/main”과 “src/test”로 구성되어있는 것을 확인할 수 있다.
실제 코드는 “src/main”에 작성되고, 테스트 코드(테스트 케이스)는 ”src/test”에 작성되어 진다.

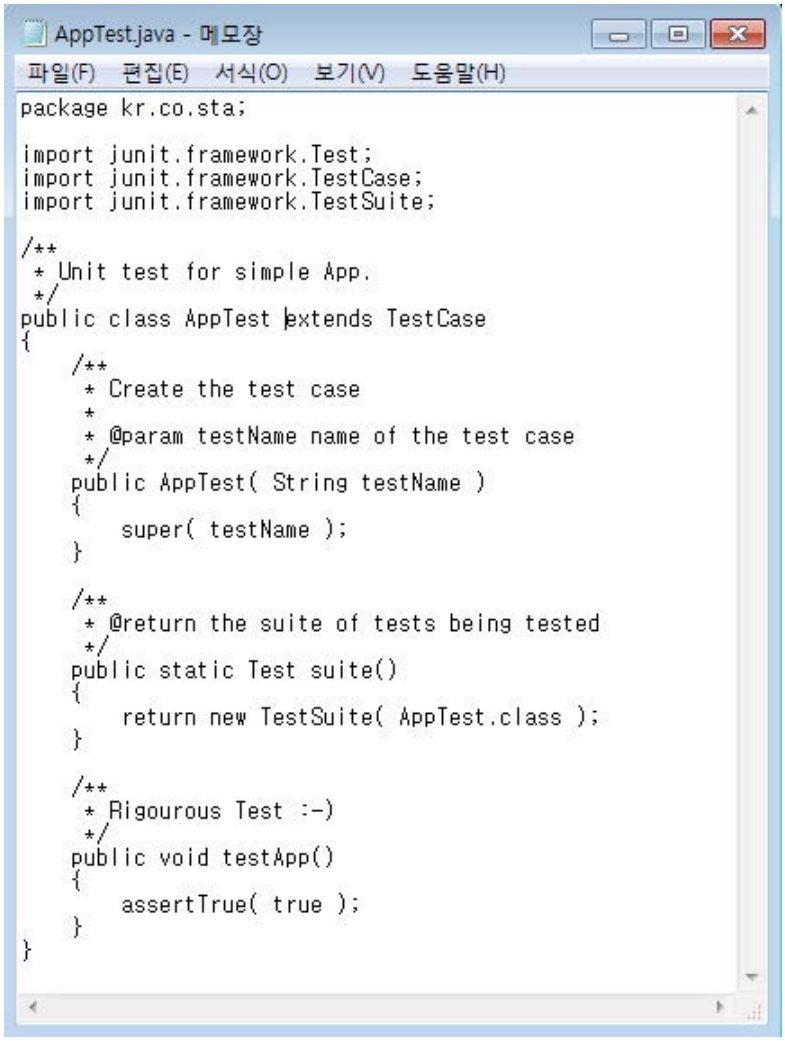


자동으로 생성되는 App.java 와 AppTest.java 파일이 있는 것을 확인할 수 있다.
두 파일을 열어서 보면 아래와 같이 간단하게 작성되어있는 것을 볼 수 있다.

코드를 보면 알 수 있겠지만 App.java 파일은 실행 시 Hello World! 를 출력하는 익숙한 예제 코드이다.



AppTest.java 파일은 Junit을 import 하여 사용하는 것을 확인할 수 있다.



작성된 예제를 Maven에서 Junit 테스트 실행을 해보겠다.

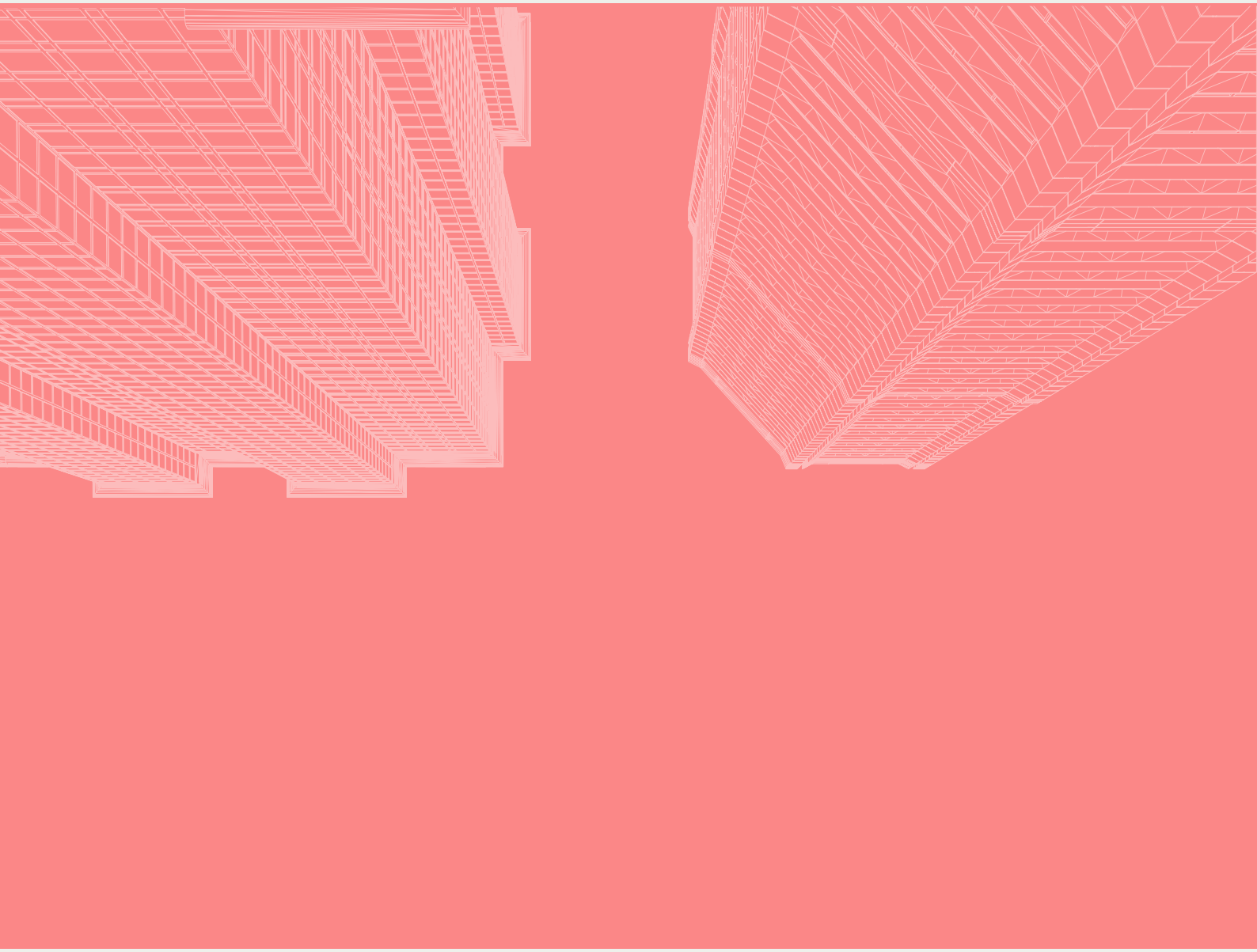
Maven default 라이프 사이클의 단계에서 test 단계부터 테스트를 수행한다. Maven의 경우 Junit에 대한 설정이 기본 포함되어있고, 위에서 이미 설정을 하였기에 "pom.xml"이 있는 프로젝트 폴더로 이동을 하고, 다음 명령으로 테스트를 실행할 수 있다.

mvn test

프로젝트 폴더로 이동을 한다.

```
C:₩dev₩maven_project>cd C:₩dev₩maven_project₩ledeme

C:₩dev₩maven_project₩ledeme>
```



CHAPTER 09

Selenium



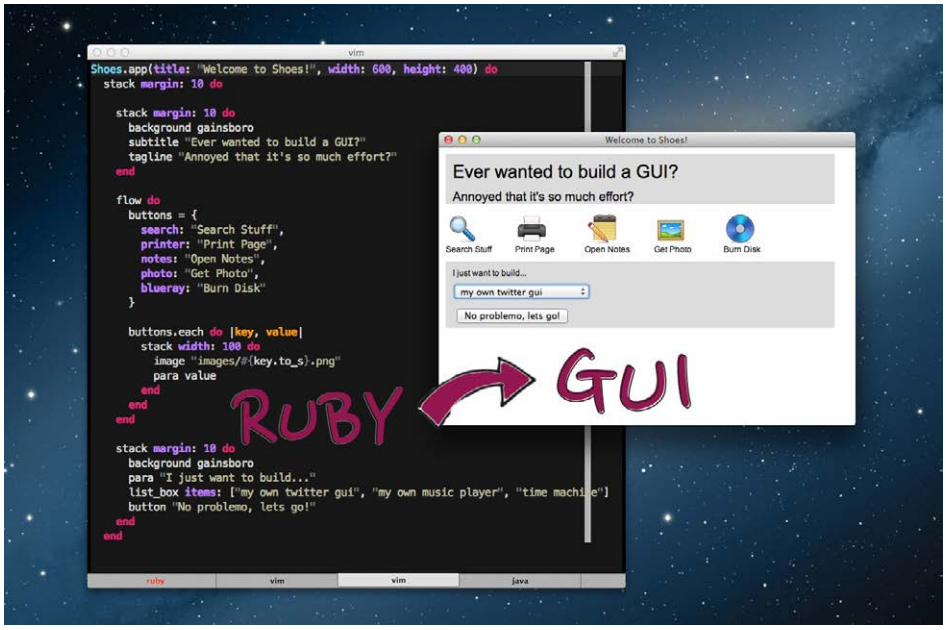
9.1 Selenium IDE 설치 75
9.2 Selenium IDE 사용법 79

CHAPTER 09 Selenium

자동화 테스트에서는 2가지 방식이 있다. 흔히 GUI (Graphical User Interface) 방식과, API (Application Programming Interface) 방식으로 Selenium 은 2가지 모두 지원하고 있다.

GUI 란? 아래와 같이 정의 하고 있다.

컴퓨터를 사용하면서, 그림으로 된 화면 위의 물체나 틀, 색상과 같은 그래픽 요소들을 어떠한 기능과 용도를 나타내기 위해 고안된 사용자를 위한 컴퓨터 인터페이스 (아래 그림 참조)



그럼 API 란 GUI 와 반대 되는 개념으로 코드기반으로 프로그래밍적인 요소가 적용된 인터페이스라고 생각하면 된다.

Selenium 의 장점은 아래와 같다.

- Web 기반 응용프로그램에 대한UI 테스트 자동화 지원
- Web 기반 응용프로그램을 테스트하는데 필요한 다양한 기능 제공
- UI Element에 대한 유연하고, 다양한 옵션 제공
- Cross Browser 테스트 가능 (FireFox, Chrome, Internet Explorer, Android 등)
- 여러 언어로 Scripting 가능 (Java, C#, PHP, Python, Ruby 등)
- 테스트 결과에 대한 다양한 Assertion 처리 기능 제공

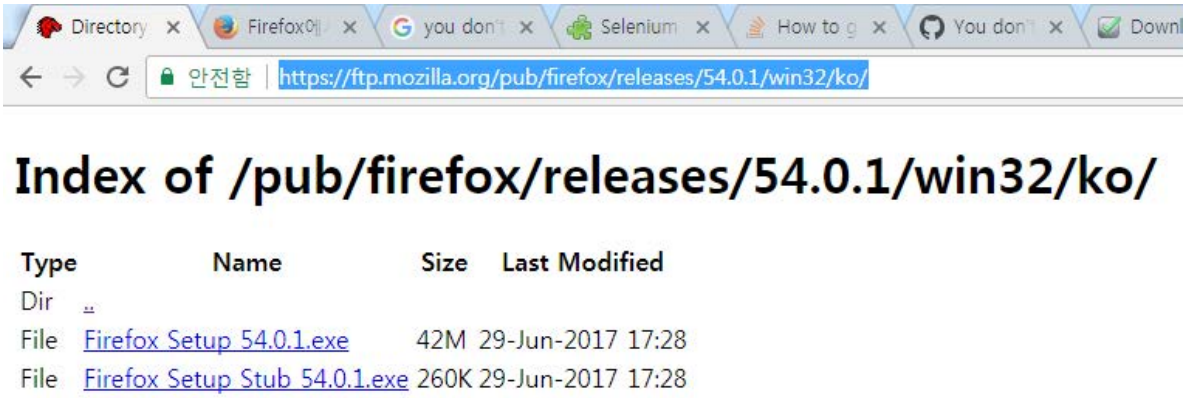
9.1 Selenium IDE 설치

Fire Fox 에서 지원하는 Plugin 으로, 쉽게 Test Case를 Recording 및 Replay 할 수 있다. (다른 Browser 에서는 사용할 수 없다.)
일단 Fire Fox 브라우저를 설치 해야 하는데, 현 시점(2017년 9월 15일 기준) Fire fox 55.* 버전에서 Selenium IDE를 사용할 수 없어 (Selenium IDE Update 필요) Fire Fox 54.* 버전을 이용하기를 권장한다.

해당 버전은 아래 URL 에서 다운 받아 설치 할 수 있다.

https://ftp.mozilla.org/pub/firefox/releases/54.0/win64/ko/

(설치된 PC의 버전에 따라 다른 버전을 다운받아 설치)



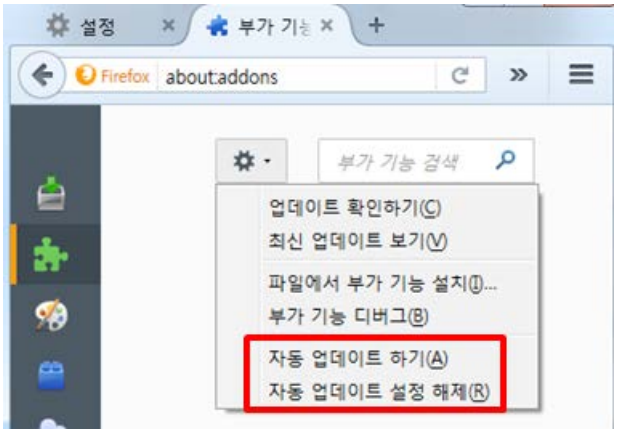
설치 후 우측 Fire Fox의 자동 업데이트가 되지 않도록 아래와 같이 설정을 변경한다.

① 우측 상단에 위치한 메뉴 버튼(≡)을 클릭하고 설정 버튼을 선택
② 고급 아이콘 선택
③ 업데이트 탭 선택
④ 업데이트 확인 안함(추천 안함: 보안 위협) 에 체크 하고 '업데이트 설치 시 백그라운드 작업하기', '검색 사이트' 체크 해제 (필수)



Fire Fox 브라우저 플러그인 설치시 자동으로 Fire Fox 업데이트 되므로, 추후 Selenium IDE 가 업데이트 되기 전까지는 업데이트가 되지 않도록 해야 한다.

① 우측 상단에 위치한 메뉴 버튼(≡)을 클릭하고 부가기능 을 클릭
② 부가 기능 관리자 탭에서 확장 기능 항목을 선택
③ 탭의 상단에서, 모든 부가 기능에 대한 설정 메뉴에서 자동 업데이트 하기의 체크를 해제하고, 자동 업데이트 설정 해제를 선택 (필수)



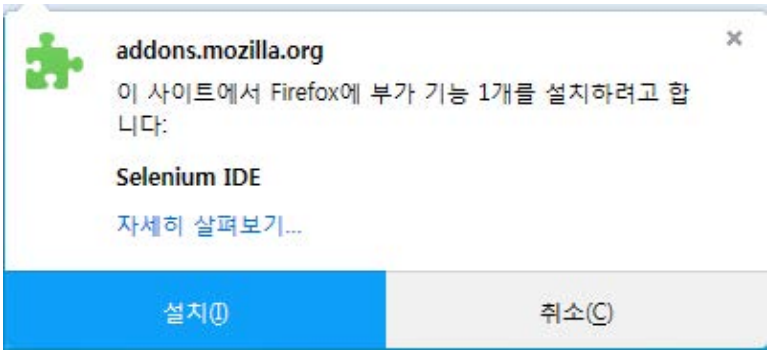
추가로 Fire Fox Plugin 설치시 자동으로 Fire fox가 업데이트 되므로, Fire fox 가 설치된 폴더의 update.exe 파일을 삭제한다.
ex) C:\Program Files\Mozilla Firefox 폴더안의 updater.exe 삭제

Fire Fox 브라우저가 설치 되었으면 Selenium IDE를 설치해야 한다. Selenium IDE는 UI 자동화를 위한 통합 개발환경을 제공하고, 다양한 포맷으로 테스트케이스를 저장할 수 있다. (HTML, Ruby, Java등) 해당 Plug in 을 설치 하기 위해서는 Fire Fox Browser 를 설치한 뒤 아래 URL 에 접속하여 IDE Plugin 을 실행한다.

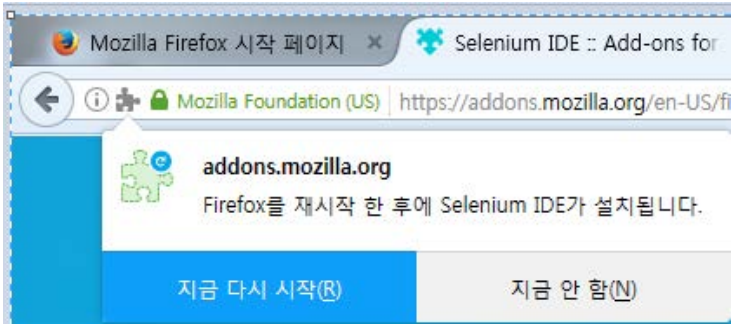
https://addons.mozilla.org/en-US/firefox/addon/selenium-ide



Add to Firefox 버튼을 클릭하여 설치하면 설치 여부를 묻는 팝업이 발생하면 설치 클릭

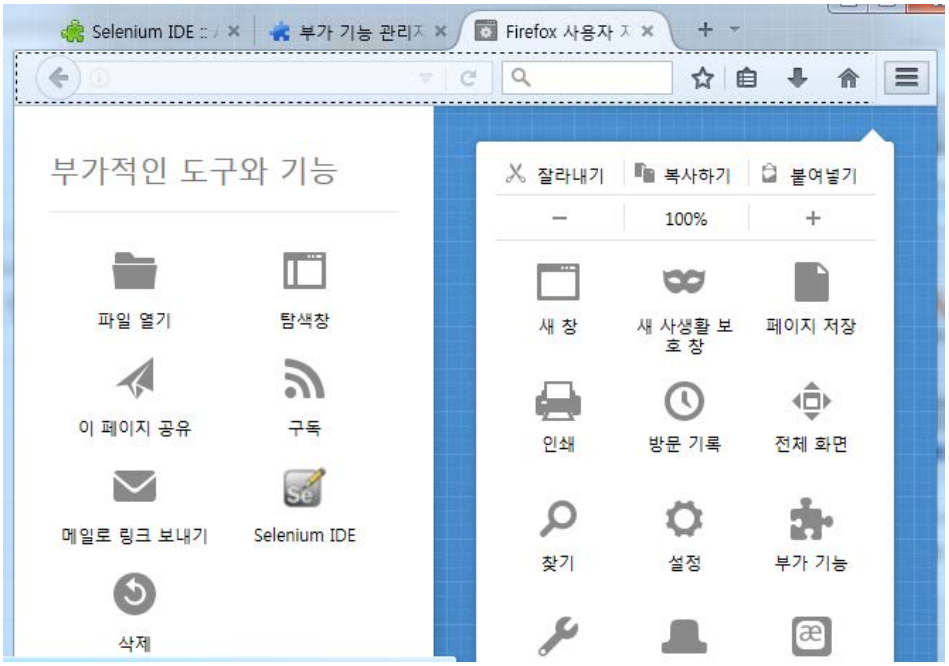


설치가 완료되면 아래 지금 다시 시작 클릭



설치가 완료되면 소프트웨어 업데이트 실패 팝업이 발생한다. 이는 Fire Fox 의 업데이트를 강제적으로 막아 Selenium IDE 를 실행하기 위한 방법으로 해당 팝업은 무시해도 된다.

설치가 완료되면 Selenium IDE를 사용하기 위해 브라우저 상단의 도구모음에 추가한다.
① 우측 상단에 위치한 메뉴 버튼(≡)을 클릭하고 사용자 설정 클릭
② 부가적인 도구와 기능에서 Selenium IDE 클릭후 상단 도구모음으로 드래그 이동

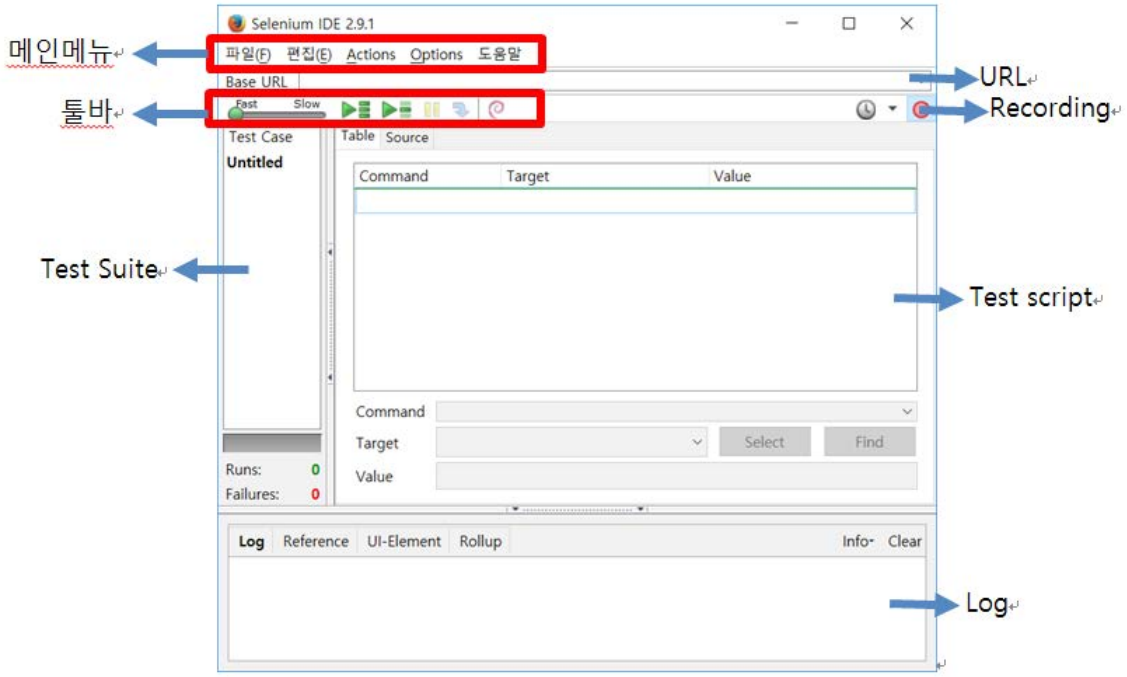


모든 작업이 완료되면 Fire Fox 상단에 아래와 같이 표시된다.



9.2 Selenium IDE 사용법

1) 메뉴설명



- 메인메뉴 : Selenium IDE 의 Export, 편집등 기능을 담당한다.
- 툴바 : Recording 된 테스트 스크립트를 수행하는 도구
- URL : 테스트에 사용되는 URL 정보가 저장된다.
- Recording : 해당 버튼 클릭 후 URL 에서 발생하는 이벤트(행위)를 저장 하여 Test script 에 저장
- Test script : Recording된 내용을 Script 언어로 저장
- Test Suite : 여러 개의 Test script 의 집합
- Log : 테스트 수행시 에러 메시지나, 수행 결과를 Log 에 표시됨

2) Selenium IDE - Recording

테스트 스크립트를 수행하기 위해서는 Test case 가 있어야 하고, 해당 Test case를 Selenium IDE 로 사용하기 위해서는 Recording 방법을 통하여 Test script 로 변경되어야 한다.

해당 방법을 수행하는 방법은 아래와 같다.

① Test case 작성

② Fire Fox Browser 오픈

③ Test URL 이동 (ex : www.naver.com)

④ Selenium IDE 실행

⑤ Recording 버튼 클릭
⑥ 테스트 케이스 순서대로 수행 (클릭 및 내용 입력)
⑦ Recording 버튼 클릭

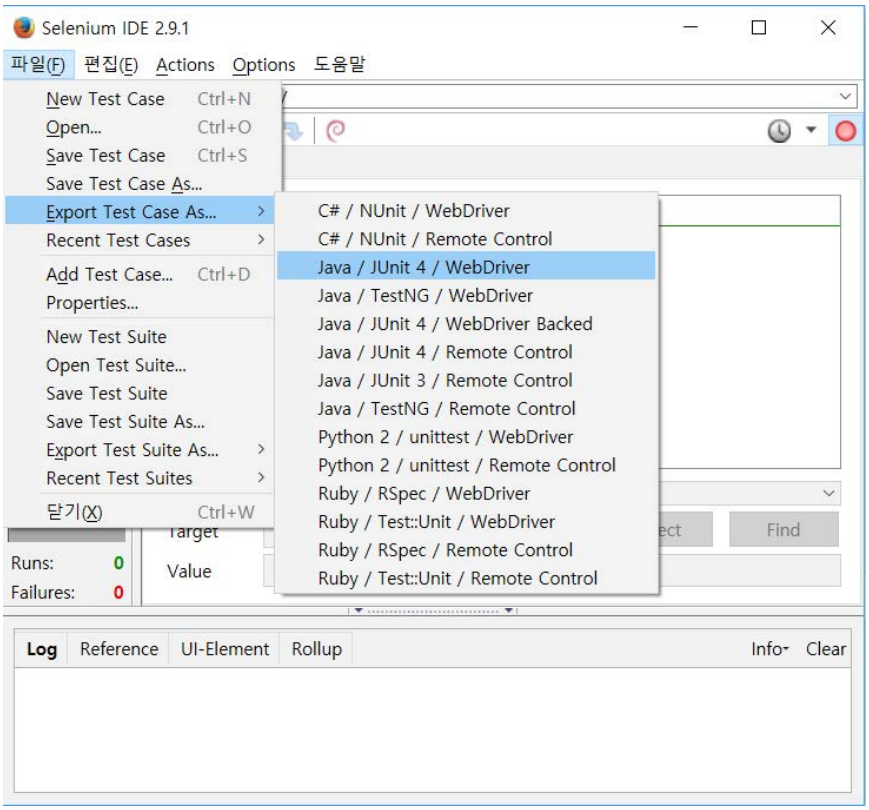
3) Selenium IDE - Play

작성된 Test script를 수행하기 위해서는 아래의 순서로 진행한다.
① 수행을 원하는 Test case를 Test Suite 에서 선택
② 툴바에서 Play 버튼 클릭
한 개의 Test case 및 Test suite 에 있는 Test case 를 모두 수행 할 수 있다.

4) Selenium IDE - Export

Selenium IDE는 FireFox만 지원하기 때문에 특정 언어로 Export 하여, 다른 Browser 사용이 가능하도록 Web driver 를 연동하여 사용할 수 있다.
변환이 가능한 언어는 아래와 같다.
- C# ,Java, Python, Ruby

아래와 같이 여러 가지 언어와 다양한 방식으로 Export 받을 수 있다.
그 중 Junit 을 기반으로 테스트가 가능한 Java / Junit 4 / WebDriver 를 선택하여 파일을 저장한다.



CHAPTER 10

Fitnesse



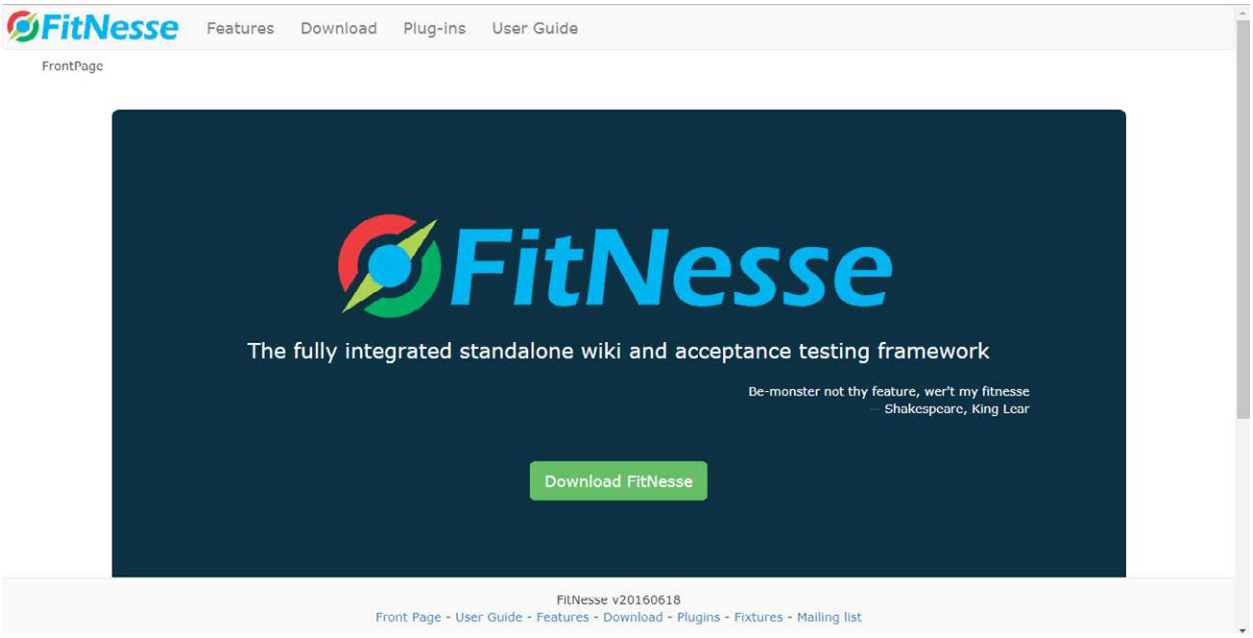
10.1 설치 82
10.2. 설계 84
10.3. 실행 86

CHAPTER 10 Fitnesse

FitNesse는 Fit에 위키를 결합하여 웹 서버 형태로 제공되는 오픈소스 인수 테스트 자동화 도구이다. 공식 홈페이지에서 jar 파일을 내려 받아 설치나 설정 없이 간편하게 바로 웹 서버를 실행하여 사용할 수 있다.

10.1 설치

FitNesse 공식 홈페이지(http://fitnesse.org)에 접속하여 Download FitNesse 페이지로 이동을 한다.



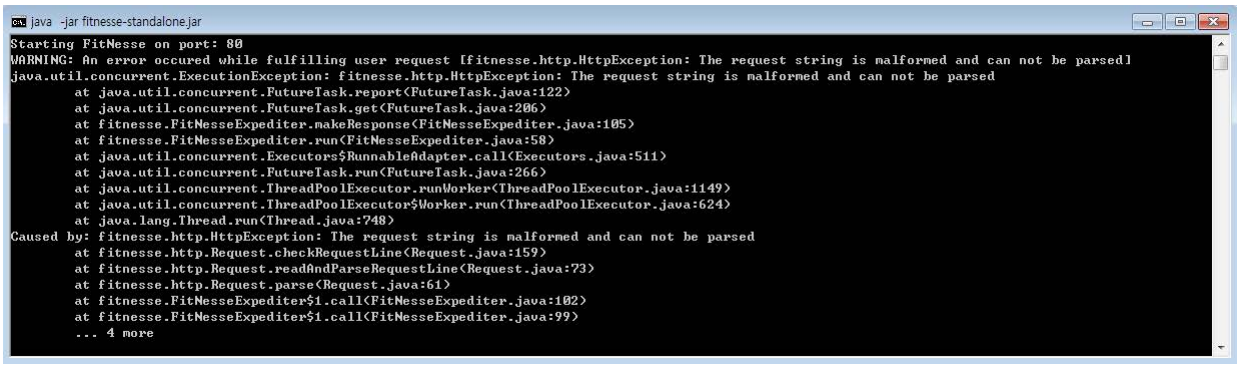
아래의 다운로드 페이지에서 fitnesse-standalone.jar 파일을 다운로드하여 적당한 폴더에 넣는다.

> 본 가이드에서는 C:\ledeme 경로의 폴더를 생성하고 이곳에 FitNesse를 설치 할 것이다. 필요에 따라 사용자가 원하는 경로의 폴더를 생성하여 사용하여도 된다. 단, 이 후 설명에서 해당 경로의 폴더를 사용하거나 언급할 수 있으므로 관련 내용을 숙지하길 바란다.

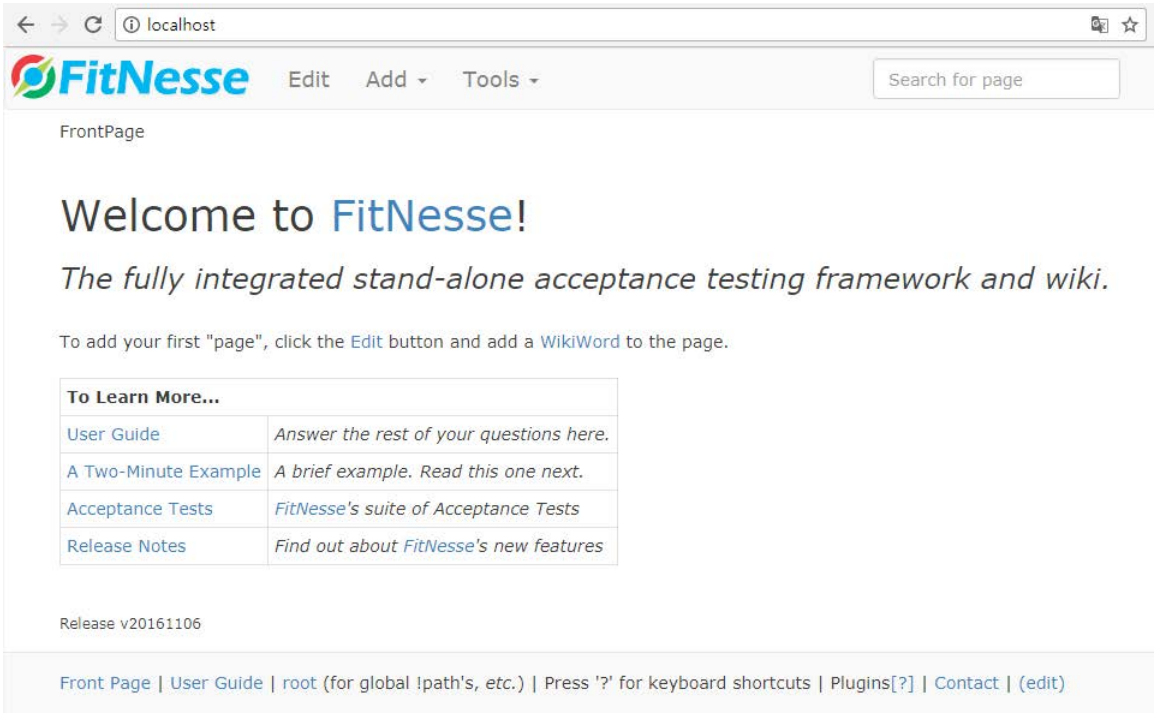


다운로드 받은 FitNesse 서버를 실행하기 위하여 해당 폴더에서 커맨드 창을 열고 다음과 같이 입력한다. FitNesse의 기본 포트는 80이다. FitNesse를 실행하는 서버에 80포트를 사용하고 있는 서비스가 있다면, 옵션을 주어 다른 포트로 실행 할 수 있다. 기본 포트로 사용하고자 한다면 포트 옵션을 넣지 않고 명령을 실행 하면 된다.

java –jar fitnesse–standalone.jar –p 포트번호



실행이 정상적으로 되면 웹브라우저 창에 “http://localhost:포트번호”를 입력하면 접속이 되는 것을 확인할 수 있다. 기본 포트로 실행을 하였다면 포트번호를 입력하지 않아도 된다.



10.2 설계

1) Fit Table

Fitnesse에서 설계란 Fit Table 설계를 말한다. 테스트하고자 하는 모듈 또는 클래스를 어떻게 테스트할 것인지에 대한 것을 Table 형태로 작성을 한다.
Fit Table을 해석하는 로직으로 Fit Fixture가 있으며, 각 Fixture 마다 Table Format 과 Fixture class 작성하는 방법이 다르다.
보다 자세한 내용은 아래의 사용자 가이드에서 확인 할 수 있다.

http://fitnesse.org/FitNesse.FullReferenceGuide.UserGuide.FixtureGallery

2) ColumnFixture

FixtureGallery 에서 Basic Fit fixtures에 6가지가 있다. 이 중에서 ColumnFixture 가이드에 나와있는 내용을 살펴 보겠다.

ColumnFixture경우 Table 은 아래와 같은 형식으로 엑셀 등에서 작성을 할 수 있다.

테스트 클래스명	
변수명	메소드명[() 또는 ?]
변수값	결과값

엑셀에서 작성된 Table을 Fitnesse에서 작성된 테이블을 복사, 붙여 넣기 하여 아래와 같은 Fit Table으로 형태로 바꿀 수 있다.

```
!|테스트 클래스명|
|변수명|메소드명[() 또는 ?]|
|변수값|결과값
```

아래는 FitNesse 사용자 가이드에 있는 ColumnFixture 의 예제이다.
테이블의 첫번째 줄은 테스트 클래스 이름, 두 번째 줄은 변수명 또는 메소드명을 적는다.

output values는 변수 또는 메소드명 뒤에 ? 또는 () 을 붙여주어야 한다.

Table Format

```
!|info.fitnesse.fixturegallery.ColumnFixtureTest|
|firstPart|secondPart|together?|totalLength?|
|Hello|World|Hello, World|10|
|Houston|We Have a Problem|Houston, We Have a Problem|24|
```

Fixture class 에서 fit.ColumnFixture 를 extend 해야 하고, 테이블 두 번째 줄의 변수와 메소드를 public으로 해야 한다.

Java Source Code

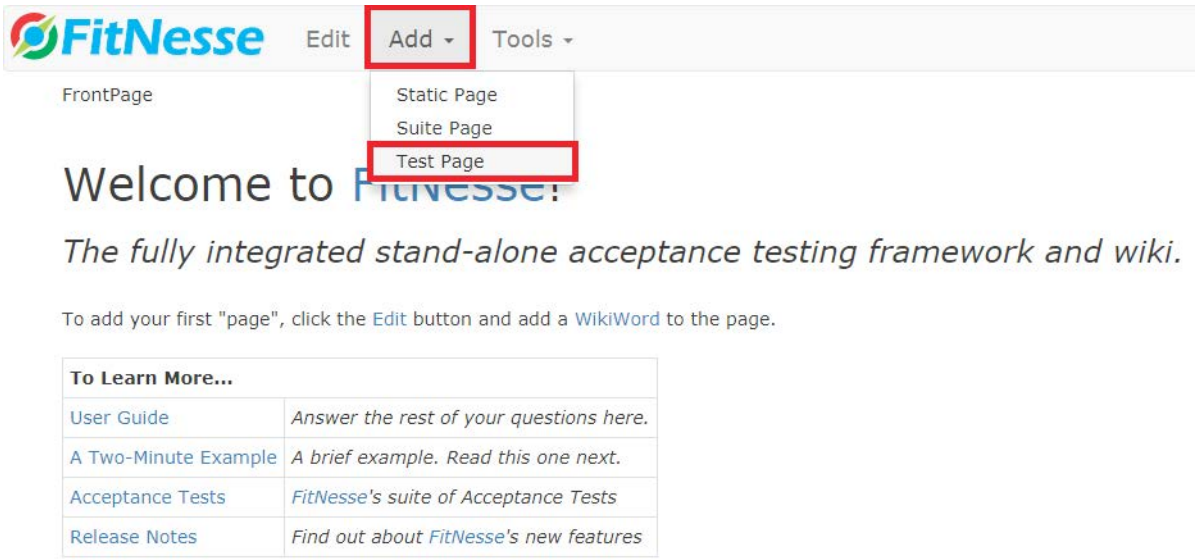
```
package info.fitnesse.fixturegallery;

import fit.ColumnFixture;

public class ColumnFixtureTest extends ColumnFixture {
        public String firstPart;
        public String secondPart;
        private int length;
        public String together(){
                length=firstPart.length()+secondPart.length();
                return firstPart+ “, “+secondPart;
        }
        public int totalLength(){
                return length;
        }
}
```

10.3 실행

설계에서 설명한 예제를 FitNesse 에서 어떻게 작성하고 실행을 하는 방법을 설명한다.
브라우저에서 FitNesse 접속(http://localhost)을 하고 상단 메뉴에 Add-〉 Test Page 를 클릭한다.



페이지 추가화면에서 Page name과 내용을 입력하고, Save 버튼을 클릭한다.

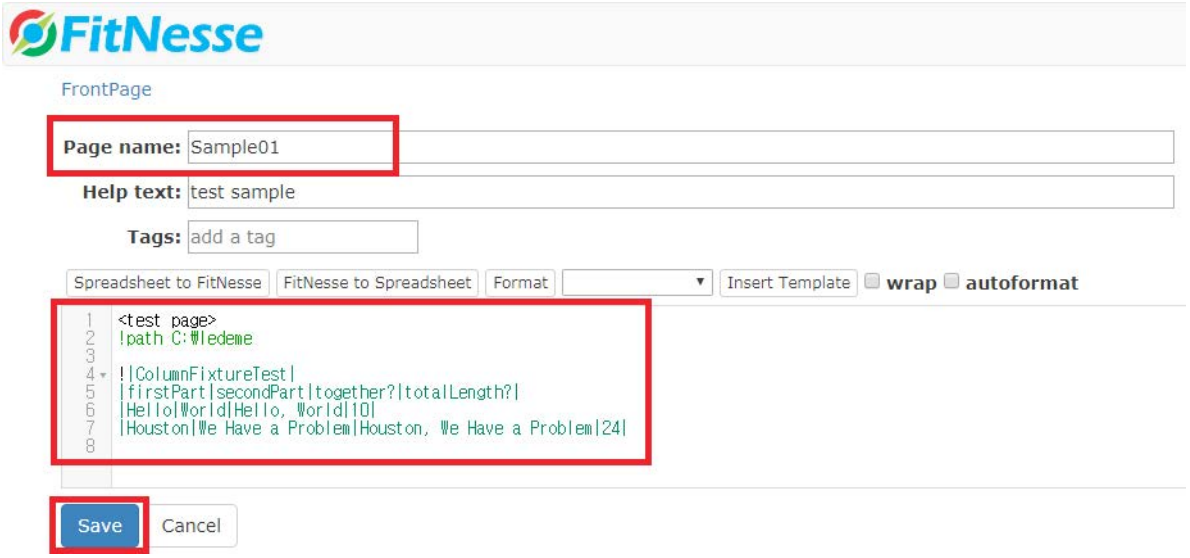
Page name은 Sample01 으로 작성하고, 내용은 아래와 같이 작성한다.

```
!path C:\ledeme

!|ColumnFixtureTest|
|firstPart|secondPart|together?|totalLength?|
|Hello|World|Hello, World|10|
|Houston|We Have a Problem|Houston, We Have a Problem|24|
```

!path 는 FitNesse 와 테스트 클래스가 있는 경로를 모두 작성하여야 한다.
FitNesse 가 있는 같은 경로에 테스트 클래스 파일을 생성할 것이라 경로 하나만 작성하였다.
이전 예제에서는 Fit Table 에서 패키지(info.fitnesse.fixturegallery)가 명시 되어있었으나 패키지로 빌드를 할 것이 아니라서 클래스명만 작성하였다.

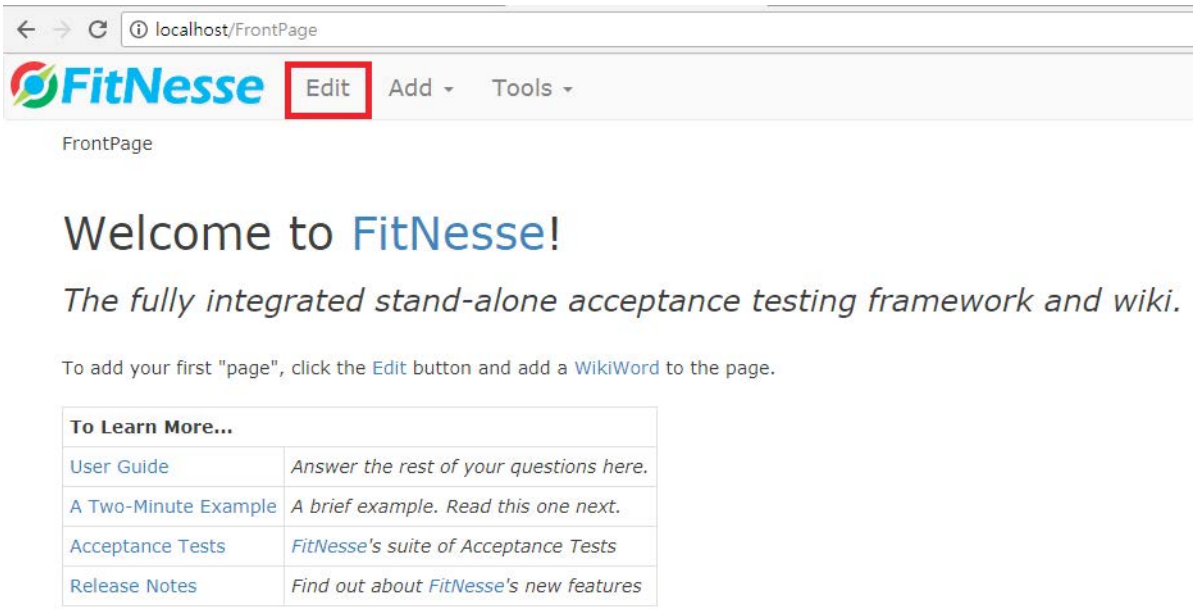
작성을 하였으면 하단의 Save 버튼을 클릭한다.



작성한 페이지가 저장되면 메인 화면(FrontPage)으로 이동을 하게 된다. 메인 화면에서는 방금 작성한 페이지가 보이지 않을 것이다.
작성한 페이지로 이동을 하기 위해서 주소 창에 직접 경로를 입력하여도 되나, 메인 화면에서 링크를 추가하여 사용할 수 있다. 앞에서 추가한 페이지 경로는 아래와 같다.

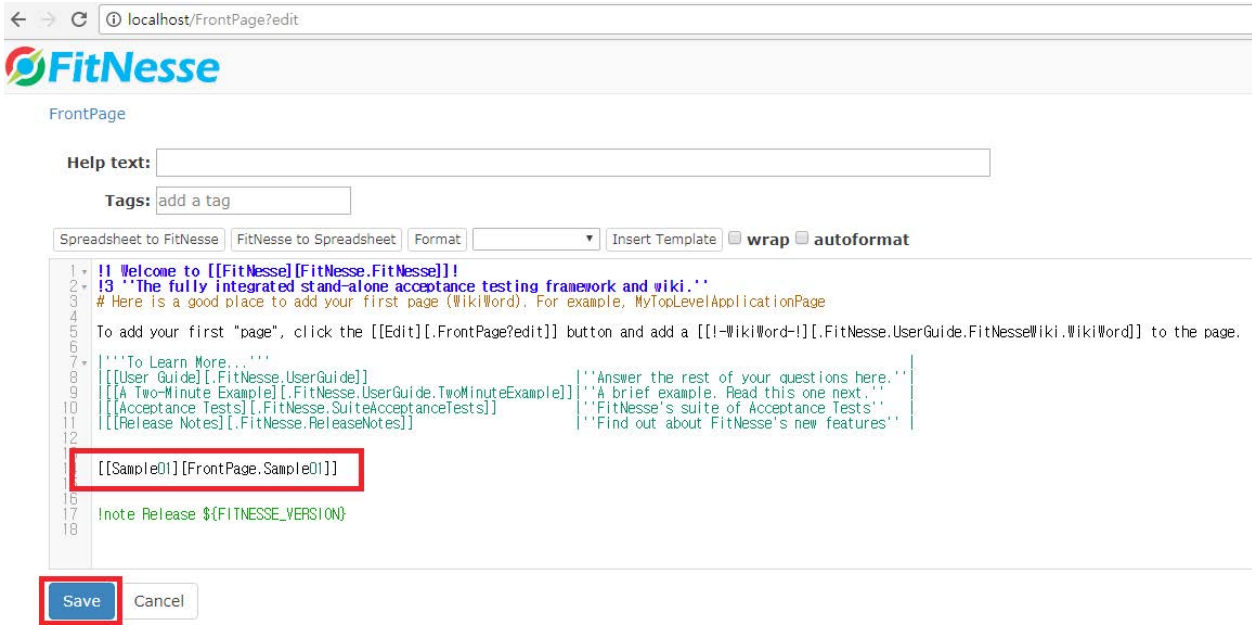
http://localhost/FrontPage.Sample01

메인 화면에서 링크를 추가하려면 메인 화면의 상단 메뉴 중 Edit 를 클릭한다.



Edit 화면의 내용에서 적당한 위치에 아래와 같이 작성하고 하단의 Save 버튼을 클릭한다.

〈 [Sample01][FrontPage.Sample01] 〉

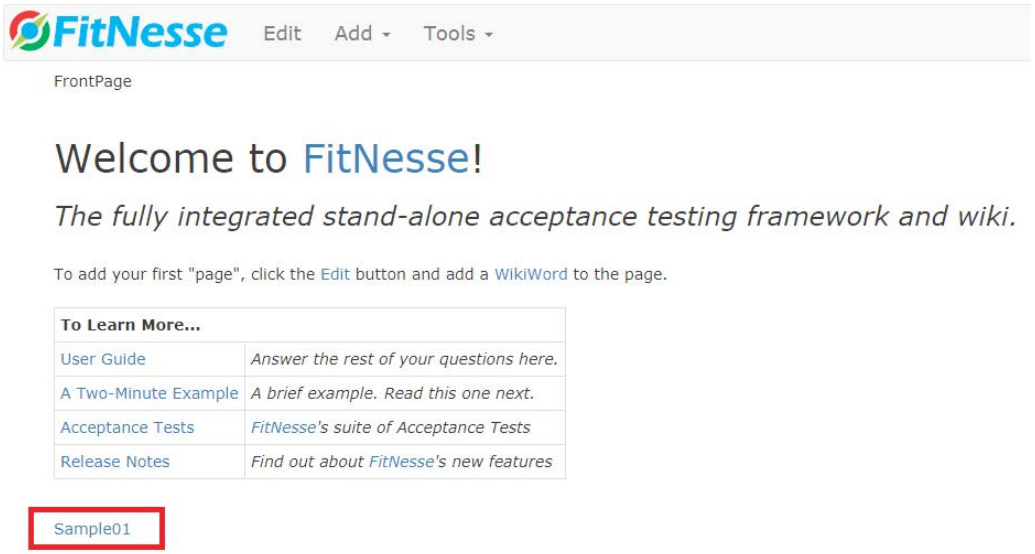


작성한 텍스트는 앞에서 추가한 페이지 이름인 Sample01 텍스트에 FrontPage.Sample01 링크를 추가하겠다는 것을 의미한다.

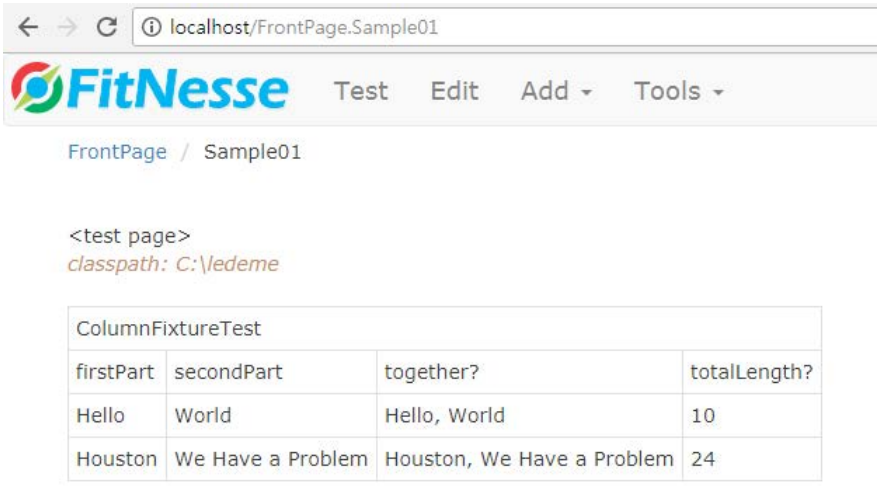
Fitnesse 사용 및 페이지 작성 방법은 아래의 가이드를 참고 하면 된다.

http://www.fitnesse.org/FitNesse.UserGuide.QuickReferenceGuide

수정을 완료하면 아래와 같이 메인 화면에 링크가 나타난다.



추가된 링크를 클릭하면 아래와 같이 새로 추가하였던 페이지로 이동을 한다.



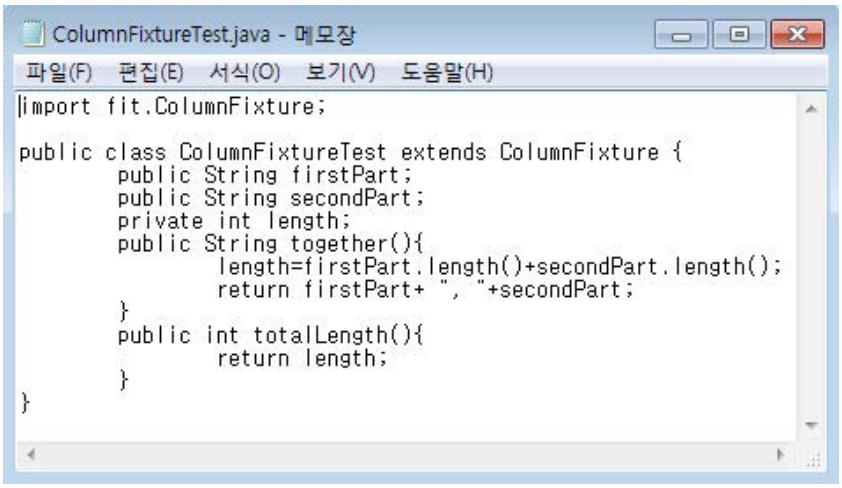
다음으로 Fit fixtures 예제를 작성하고 빌드하여 FitNesse에서 사용할 수 있도록 할 것이다.

아래와 같이 FitNesse 파일이 있는 경로(C:\ledeme)에 ColumnFixtureTest.java 파일을 작성하여 저장을 한다. 앞서 패키지 정보는 삭제를 하였으니 여기에서도 패키지 정보를 삭제하고 아래와 같이 작성하면 된다.

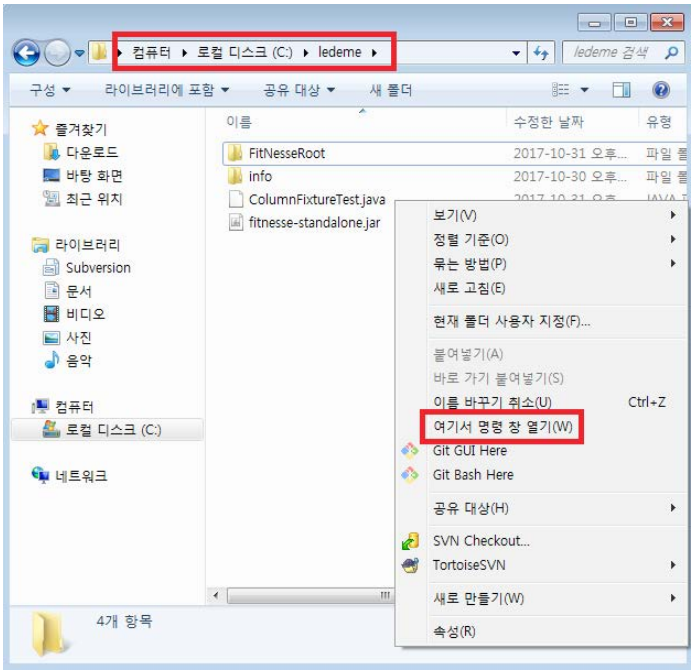
Java Source Code

```
import fit.ColumnFixture;

public class ColumnFixtureTest extends ColumnFixture {
        public String firstPart;
        public String secondPart;
        private int length;
        public String together(){
                length=firstPart.length()+secondPart.length();
                return firstPart+ “, “+secondPart;
        }
        public int totalLength(){
                return length;
        }
}
```



폴더에서 “Shift key + 마우스 우클릭”을 하고, ‘여기서 명령 창 열기’ 를 클릭하여 명령 창을 연다

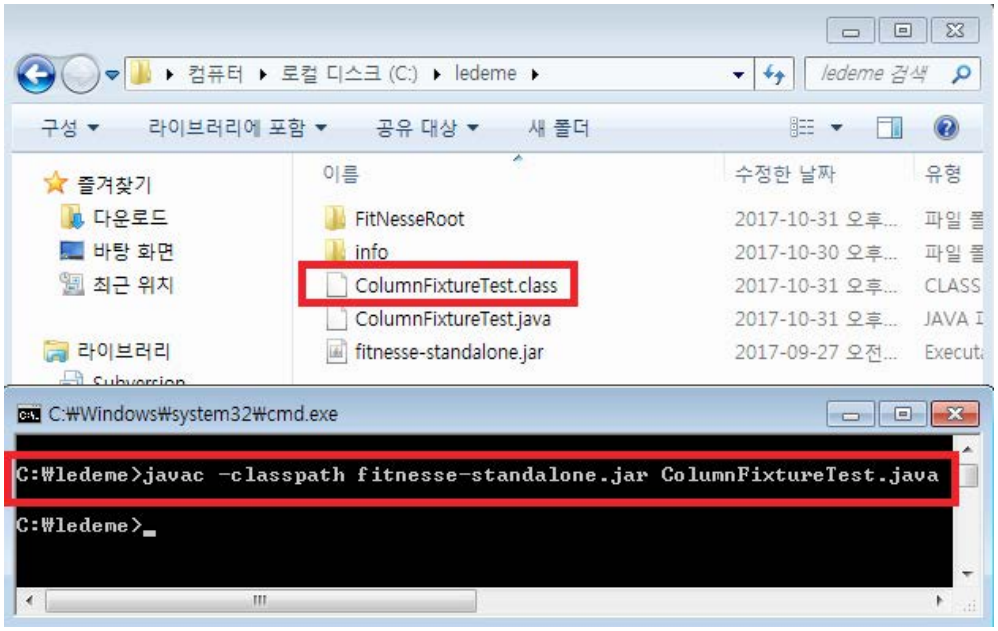


명령 창에서 아래 빌드 명령을 실행한다.

```
javac -classpath fitnesse-standalone.jar ColumnFixtureTest.java
```

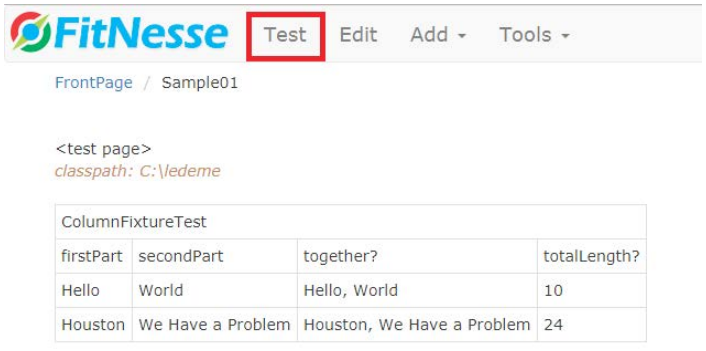
import한 Fit Fixtures 를 사용하기 위해서는 위의 빌드 명령에서 처럼 classpath 를 꼭 지정해 주어야 한다. 지정하지 않으면 fit 빌드 에러가 발생한다.

위의 명령으로 빌드를 실행하여 성공하면 아래와 같이 class 파일이 생성이 된다.

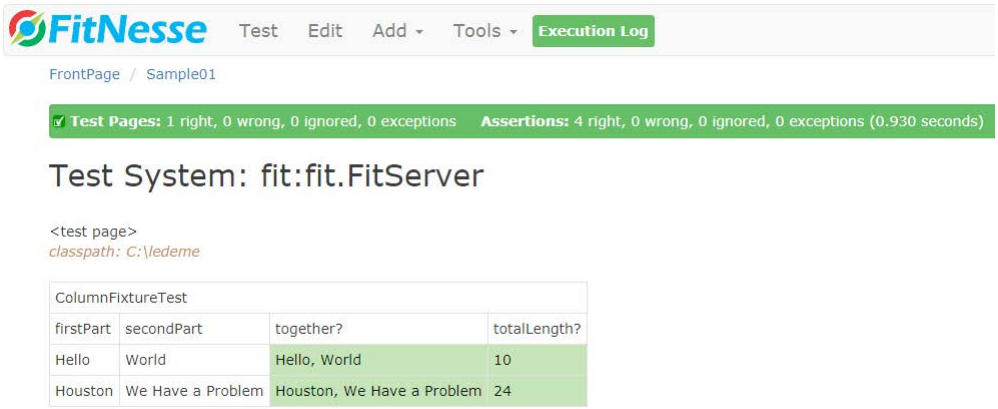


이제 FitNesse Test 를 하기 위한 준비는 모두 끝났다.
이제 FitNesse에서 Test를 실행하고 실행 결과를 보도록 하겠다.

FitNesse 메인 화면에서 앞서 추가를 하였던 페이지로 접속을 하면 아래와 같은 화면이 보이게 된다.
화면 상단 Test라는 메뉴가 있으며, 이 Test를 클릭하면 Test가 진행된다.



Test 메뉴를 클릭하여 Test를 실행하면 Test가 진행이 되며, Test 결과를 아래와 같이 보여준다.





CHAPTER 11

Nexus

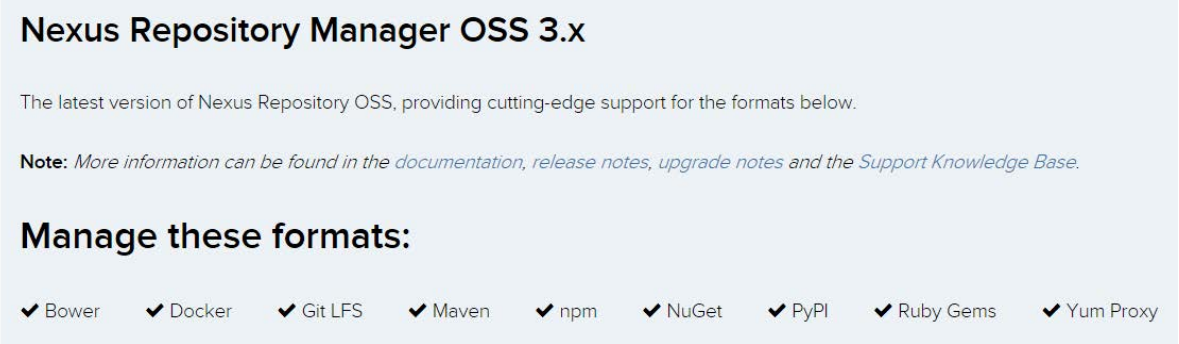


11.1 다운로드 94
11.2 설치 96
11.3 설정 98
11.4 Maven 저장소 99

CHAPTER 11 Nexus

Nexus는 Maven에서 사용 할 수 있는 Repository 이며, 현재 Nexus3가 최신 버전이다.

3. x 버전으로 업그레이드 되면서 Maven 뿐만 아니라 다양한 포맷을 지원하는 저장소 관리자로 사용할 수 있다.



11.1 다운로드

아래의 공식 사이트에서 Nexus를 다운로드 받을 수 있다.

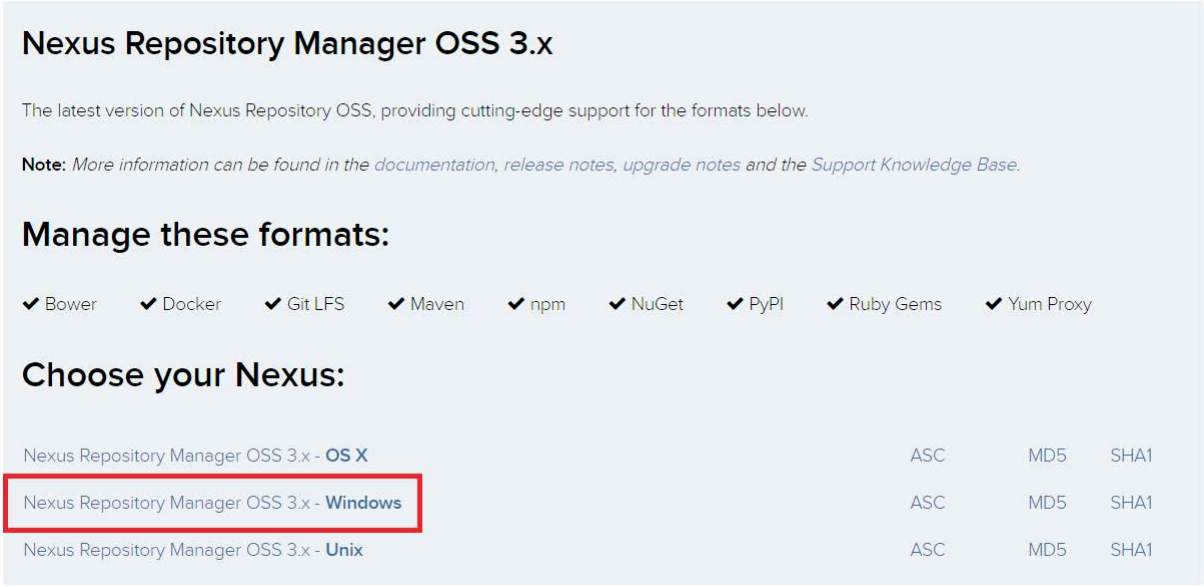
http://www.sonatype.org/nexus



Nexus는 무료 버전(OSS)과 상용 버전(Pro)이 있다. 상용 버전(Pro)의 경우 Artifact들의 라이선스로 인한 문제가 되는게 있는지 확인하거나 취약점이 있는지 찾아서 보고해주는 기능 등의 유용한 기능과 기술지원을 해준다. 무료 버전(OSS)에서 기본적인 기능과 대부분의 필요한 기능을 사용 할 수 있다. 본 가이드에서는 OSS 버전을 설치 할 것이다.

> #IE 8 이하 버전은 지원을 하지 않으므로, IE를 업그레이드 하거나 다른 브라우저를 사용하여야 정상적으로 사용할 수 있다.

상단의 "DOWNLOADS 〉 NEXUS REPOSITORY MANAGER OSS" 를 클릭하여 다운로드 페이지로 이동을 한다.



"Nexus Repository Manager OSS 3.x – Windows" 를 클릭하여 Nexus를 다운로드 받는다.

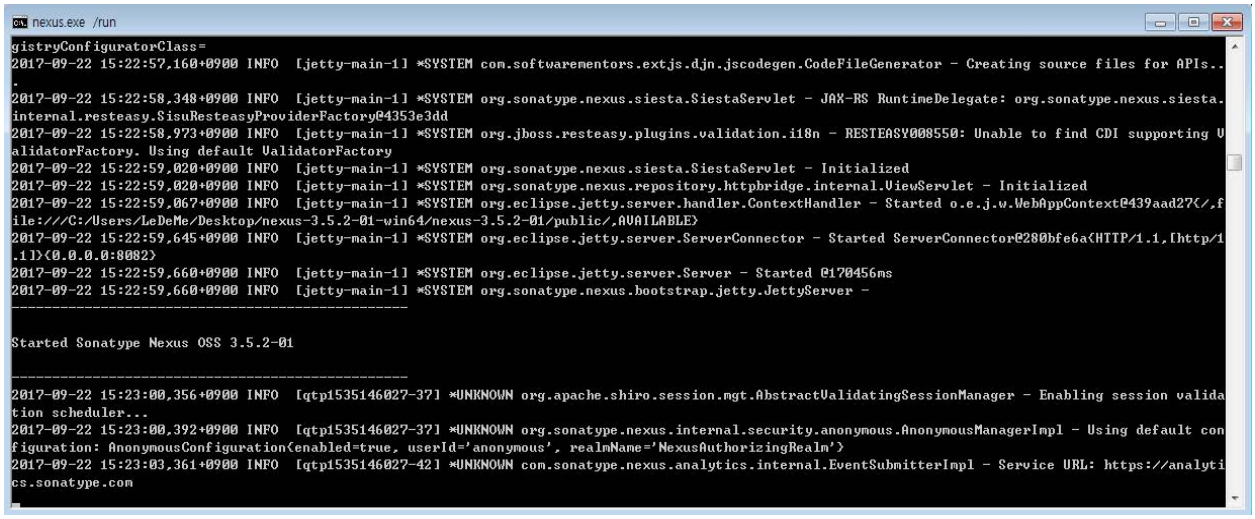
11.2 설치

Nexus는 별다른 설치 과정은 없으며, 다운로드 받은 파일을 원하는 곳에 압축을 푸는 것으로 설치가 끝난다. 압축을 풀면 2개의 폴더가 보이며, "nexus-3.x.x" 폴더는 Nexus Repository Manager 어플리케이션 폴더이고 "sonatype-work" 폴더는 Data-store로 저장소, 설정, 캐시 등의 모든 데이터가 저장된다.

"nexus-3.x.x" 폴더의 bin 폴더는 Nexus 구동에 관련한 스크립트가 있다.

Nexus 실행은 bin 폴더의 nexus.exe. 를 CMD참에서 아래의 명령으로 실행하면 된다.

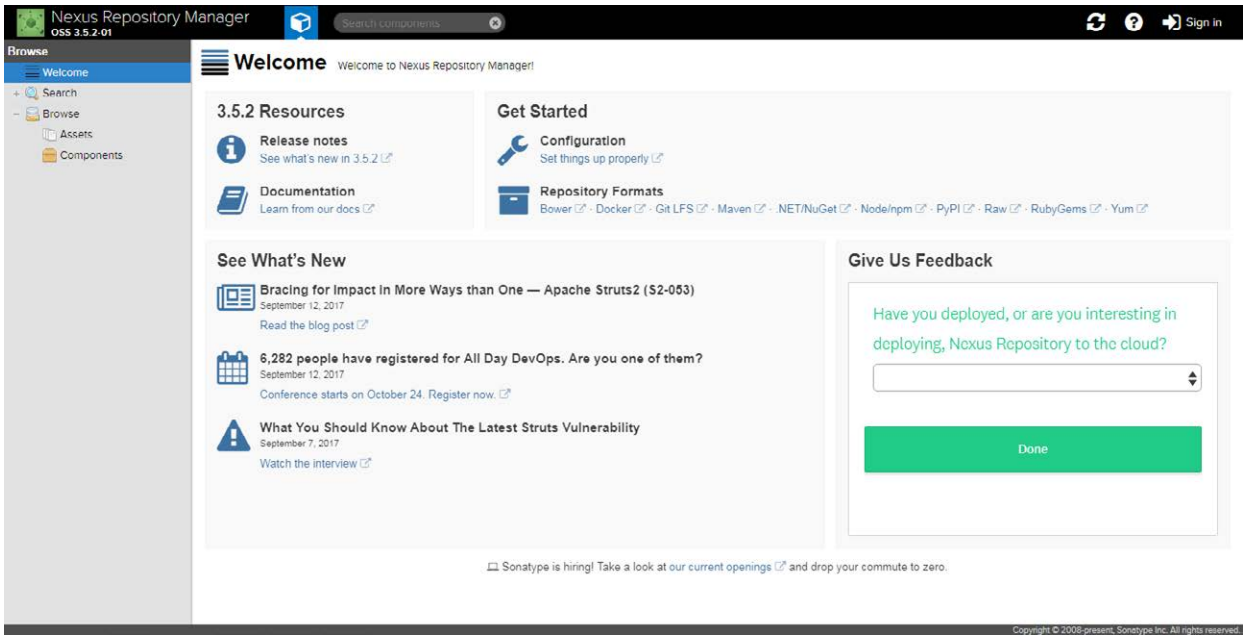
```
nexus.exe /run
```



실행이 완료되면 Nexus 화면이 웹 브라우저에서 아래와 같이 보여진다.

아래의 화면이 나오지 않는다면, 브라우저를 실행하여 "http://localhost:8081"으로 접속을 하면 된다.

최소 생성시 "ID/PW"는 "admin/admin123" 으로 기본값이 설정되어 있다.



• Window Service 등록(Running as a Service on Windows)

Windows Service 등록을 하려면 콘솔에서 아래의 명령을 실행하면 된다.

```
nexus.exe /install
```

• Nexus Port 변경

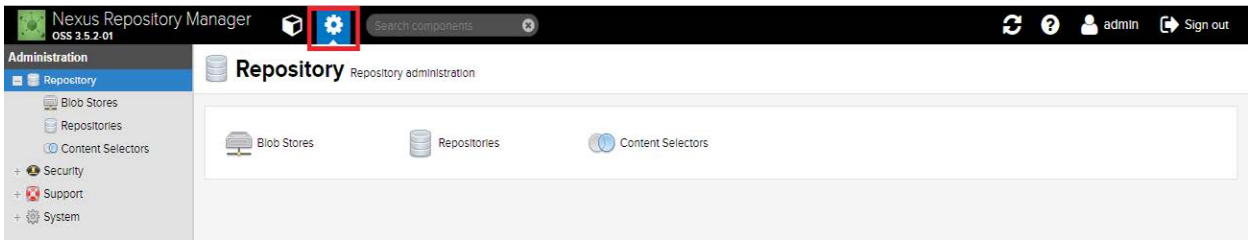
기본포트는 '8081'이며, 포트 정보 등 변경은 "/etc/nexus-default.properties" 에서 할 수 있다

```
# Jetty section
application-port=8081
application-host=0.0.0.0
```

11.3 설정

웹 브라우저에서 http://localhost:8081 으로 Nexus에 접속을 하고, 관리자 계정으로 로그인을 하면 상단에 설정 이미지 버튼이 보인다.

설정 버튼을 클릭하면 아래와 같이 저장소 등 설정을 할 수 있는 메뉴가 보인다.



좌측 메뉴를 보면 Repository, Security, Support, System 이 있으며 각각 필요한 설정들을 할 수 있도록 제공하고 있다.
관리자 설정에서 필요한 기능이 있다면, 아래 공식 가이드 링크를 참고하여 설정하면 된다.

https://books.sonatype.com/nexus-book/3.0/reference/admin.html

우리가 주로 사용할 Repository(저장소)에 대해서 간단하게 알아보면 다음과 같다.

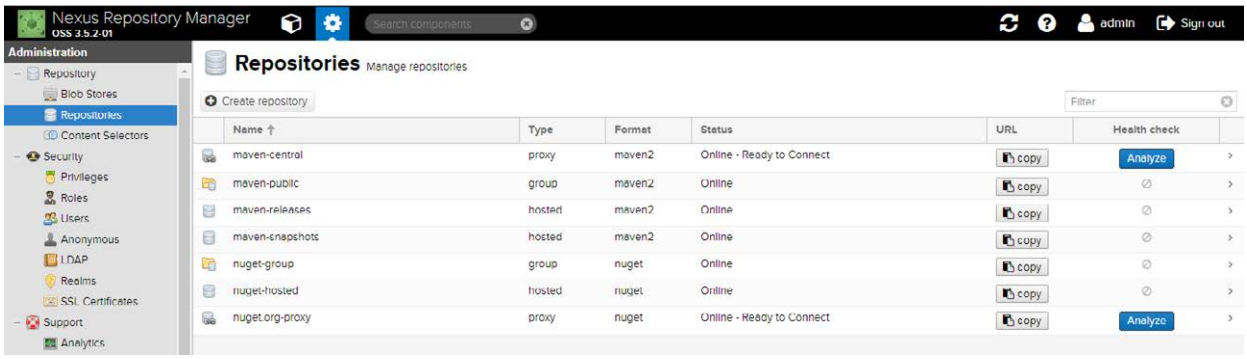
• Blob Stores

모든 저장소는 Blob Store를 지정해 줘야 한다. Blob Store는 실제 데이터가 저장되는 장소이며, sonatype-work 하위 폴더와 연결된다.

• Repository Type

모든 Format의 저장소에는 3가지 Type이 있다.

- proxy : 외부의 다른 경로를 proxy하는 저장소. proxy를 통해 외부의 데이터를 캐시 할 수 있다. (samples : maven-central, nugget.org-proxy)
- hosted : 자체 모듈 저장소(samples : maven-releases, maven-snapshots, nugget-hosted)
- group : proxy와 hosted 등을 묶을 수 있는 단위 집단, 그룹에 저장소를 나열하는 순서가 그룹의 라이브러리를 탐색 우선순위가 된다.(samples : maven-public, nugget-group)



11.4 Maven 저장소

1) 새로운 저장소 생성(Maven)

Nexus에서 Maven 저장소 생성 시 최소 설정 절차는 Nexus 가이드에서 다음과 같이 기술하고 있다.

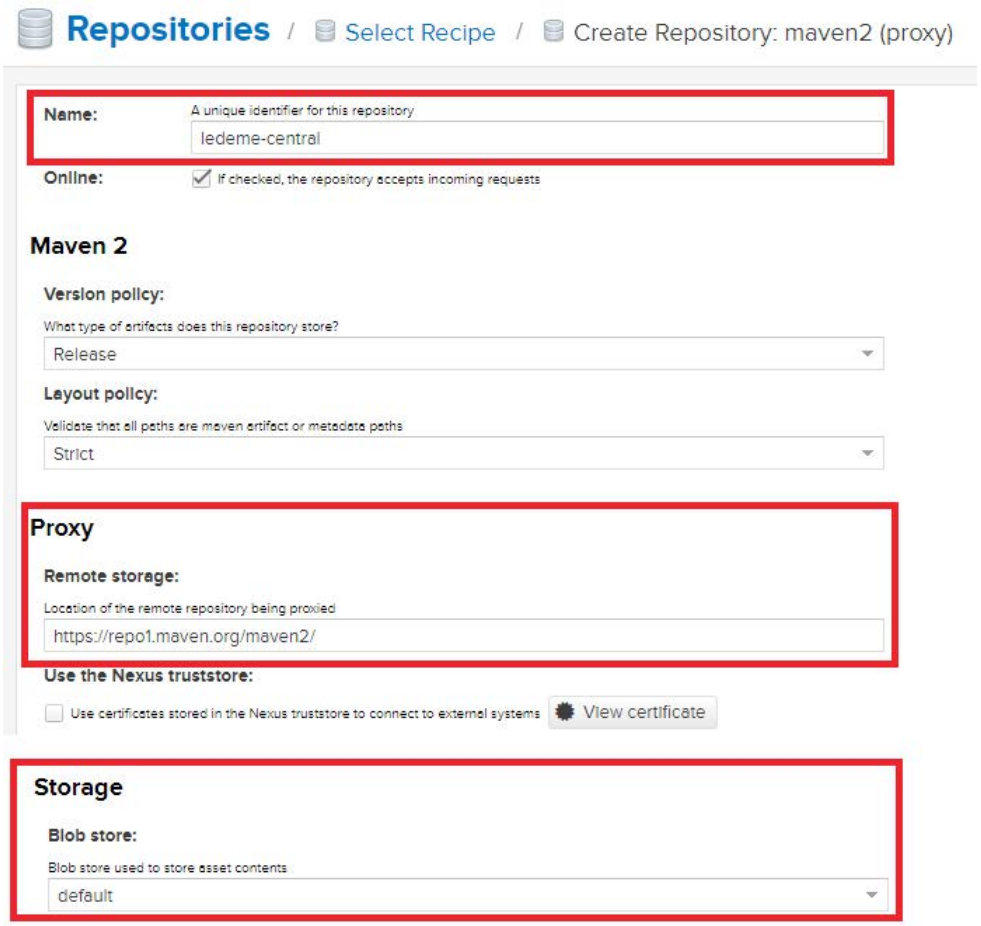
- **Proxying Maven Repositories**
  - Define Name : 저장소명 입력
  - Define URL for Remote storage e.g. https://repo1.maven.org/maven2/ : 원격 저장소 URL 입력
  - Select Blob store for Storage : 저장소에서 Blob store를 선택

- **Hosting Maven Repositories**
  - Define Name : 저장소명 입력
  - Select Blob store for Storage : 저장소에서 Blob store 선택

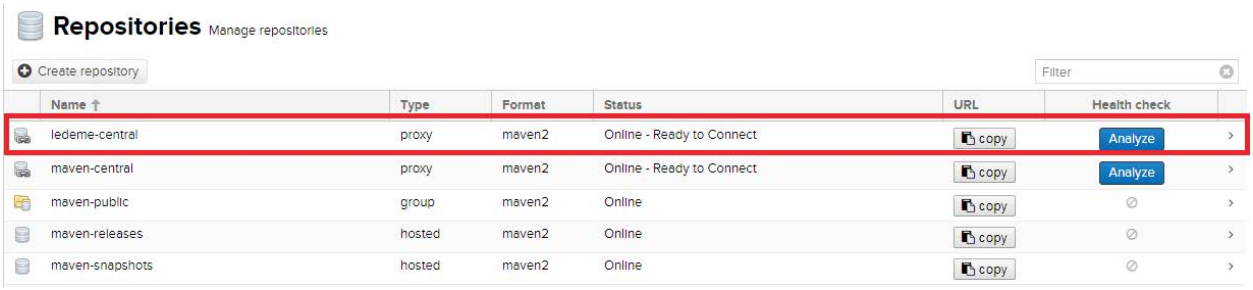
- **Grouping Maven Repositories**
  - Define Name : 저장소명 입력
  - Select Blob store for Storage : 저장소에서 Blob store 선택
  - Add Maven repositories to the Member list in the desired order : 목록에서 Maven 저장소 추가

Maven Proxy 저장소를 생성해 볼 것이다.

① "Create repository" 버튼을 클릭한다.
② maven2(proxy)를 클릭한다.
③ Name에 "ledeme-central" 를 입력한다.
④ Remote storage에 "https://repo1.maven.org/maven2/" 를 입력한다.
⑤ 아래의 "Create repository" 버튼을 클릭한다.



저장이 완료 되면 저장소 목록에 추가된 저장소가 보인다.



2) Maven 설정(Configuring Apache Maven)

Maven의 setting.xml 에 아래의 설정을 추가 한다.

Maven 설치를 본 가이드를 따라 하였다면, setting.xml 은 “C:\apache-maven-3.5.0\conf\”에 있다.

• Nexus 정보를 설정한다.

```
<settings>
....
  <servers>
    <server>
      <id>nexus</id>
      <username>admin</username> <!--nexus 관리자 계정-->
      <password>admin123</password> <!--nexus 관리자 비밀번호-->
    </server>
  </servers>
```

• Nexus 저장소 정보를 설정한다.

```
<settings>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <!--Proxy로 설정된 repository(외부 Repository 접근가능하도록)-->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://localhost:8081/repository/maven-central/</url>
          <releases><enabled>true</enabled></releases>
```

```
        <snapshots><enabled>true</enabled></snapshots>
      </repository>
    </repositories>
    <!--Maven Plugin에 대한 Repository 설정-->
    <pluginRepositories>
      <pluginRepository>
        <id>central</id>
        <url> http://localhost:8081/repository/maven-central/</url>
        <releases><enabled>true</enabled></releases>
        <snapshots><enabled>true</enabled></snapshots>
      </pluginRepository>
    </pluginRepositories>
  </profile>
</profiles>

<!--위에 설정한 Profile을 사용하겠다고 선언 -->
  <activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
</settings>
```

• **Maven Project의 pom.xml에 아래의 설정을 추가 한다.**

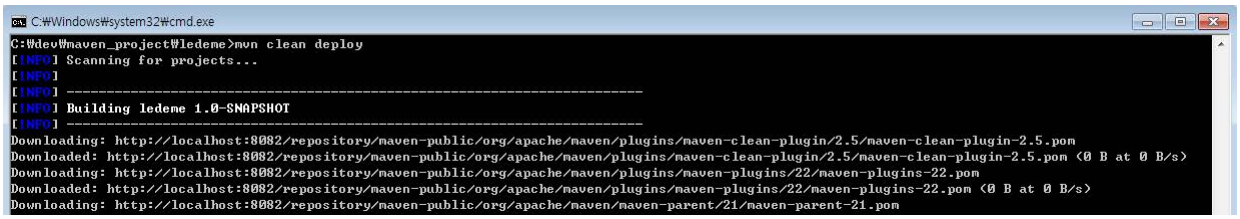
```
<project>
...
<distributionManagement>
    <repository>
      <id>nexus</id>
      <name>Releases</name>
      <url>http://localhost:8081/repository/maven-central</url>
    </repository>
    <snapshotRepository>
      <id>nexus</id>
```

```
    <name>Snapshot</name>
    <url>http://localhost:8081/repository/maven-snapshots</url>
  </snapshotRepository>
</distributionManagement>
```

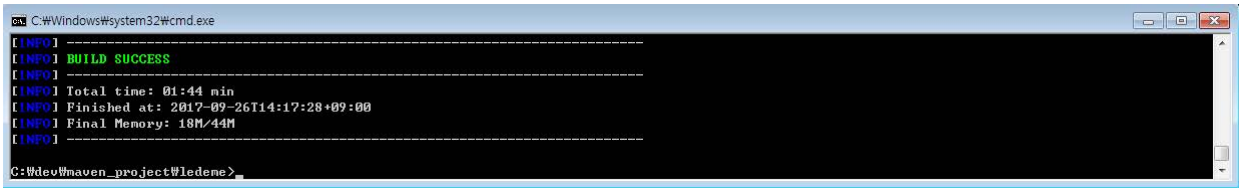
```
# setting.xml 파일에 설정한 <server><id> 와 pom.xml 에 설정한 <repository><id> 값이 동일하여야 정상적으로 배포가 된다.
```

모든 설정이 되었다면 콘솔에서 아래의 명령을 실행하여 배포할 수 있다.

mvn clean deploy

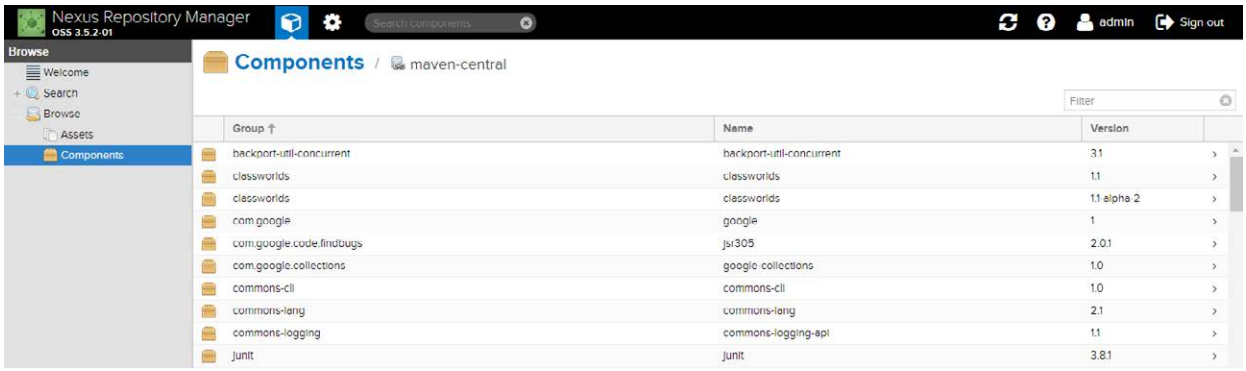


명령을 실행하고 정상적으로 빌드가 성공하면 아래와 같이 출력되며 종료한다.

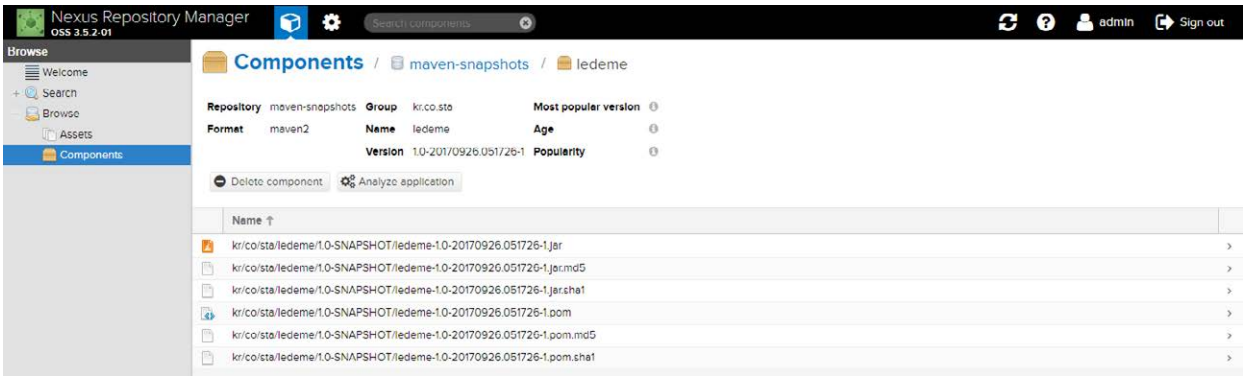


빌드가 정상적으로 되었다면, Nexus의 저장소에 배포가 되었는지 확인해 본다.
예제의 설정에서 “maven-central” 으로 releases, “maven-snapshots”에 Snapshot 저장소로 설정이 되어있다.

아래 화면에서 “maven-central” 저장소에 라이브러리들이 올라와 있는 것을 볼 수 있다.



아래 화면에서 “maven-snapshots” 저장소에 Maven 빌드한 “ledeme”프로젝트의 snapshot 파일이 올라와 있는 것을 볼 수 있다.



CHAPTER 12

Jenkins



12.1 다운로드 106
12.2 설치 107
12.3 Jenkins PORT 변경 방법 112
12.4 Jenkins 접속 113
12.5 Jenkins 관리 114
12.6 Jenkins 신규 아이템 추가 및 설정 130
12.7 Jenkins에서 빌드 실행 139

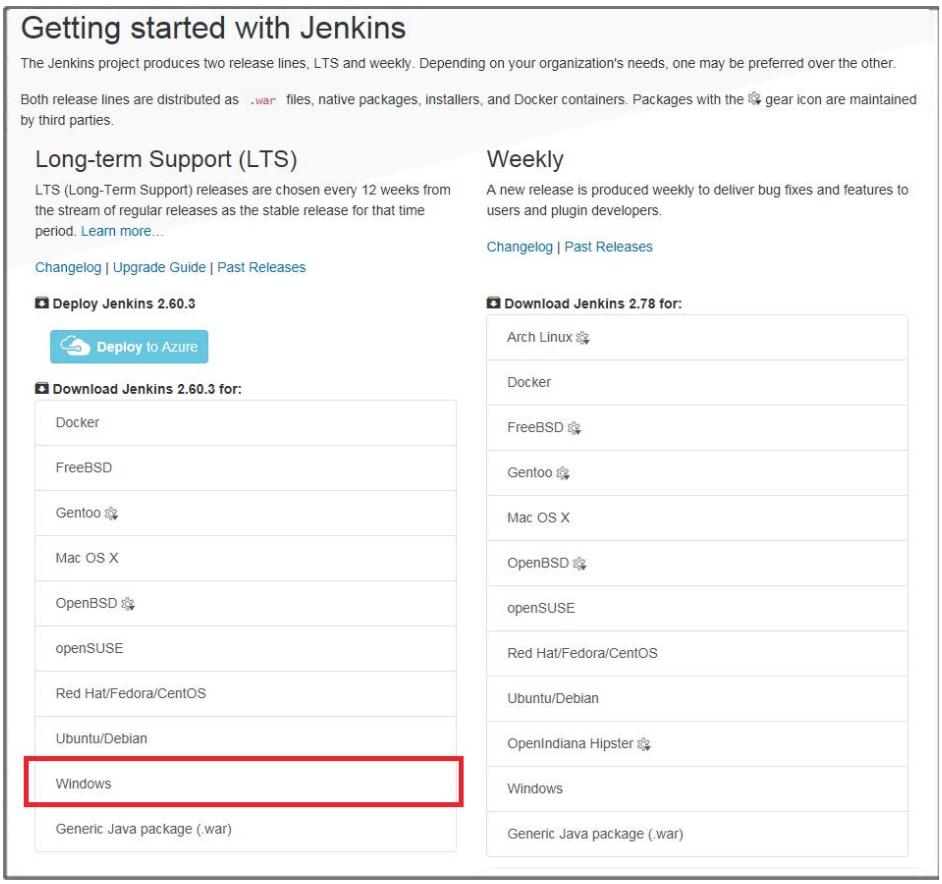
CHAPTER 12 Jenkins

Jenkins는 Java로 개발되었으며 서블릿 컨테이너 위에서 구동된다. Java로 개발 되었으므로 JRE 또는 JDK가 설치되어있어야 한다.

12.1 다운로드

CI 도구인 Jenkins 설치는 https://jenkins.io/download/ 에서 다운로드를 받을 수 있으며, 해당 OS에 맞는 패키지를 다운로드 받아서 설치하면 된다.
여러 OS에서 설치하는 방법은 아래 링크를 참고하여 설치 할 수 있습니다.

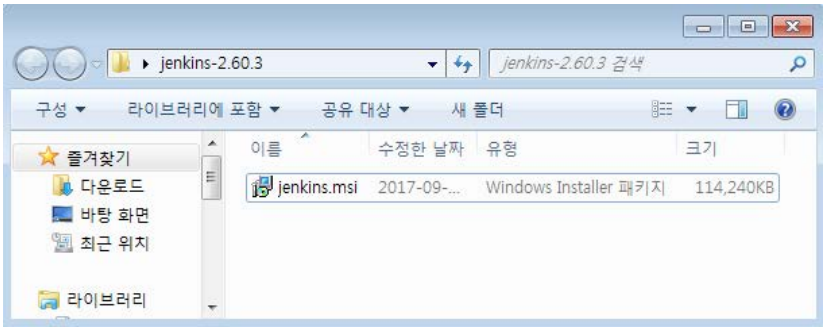
https://jenkins.io/doc/book/getting-started/installing/



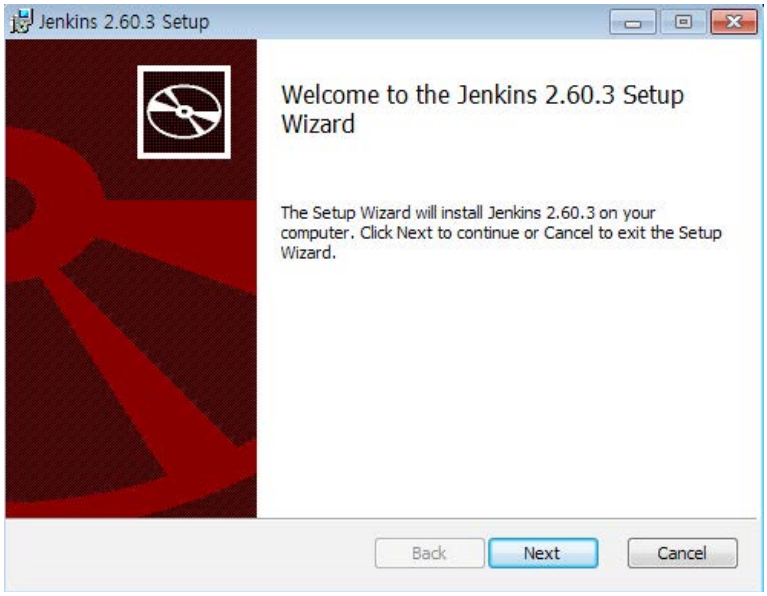
매주 버그를 수정하고 새 기능을 추가한 버전(Weekly)과 안정 버전인 LTS(Long-Term Support)가 있으며, 원하는 버전을 선택하여 설치를 할 수 있다.

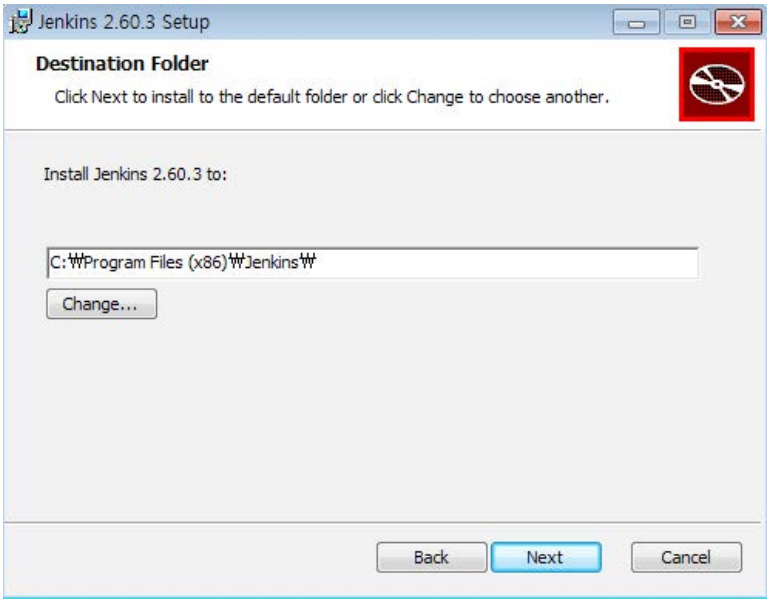
12.2 설치

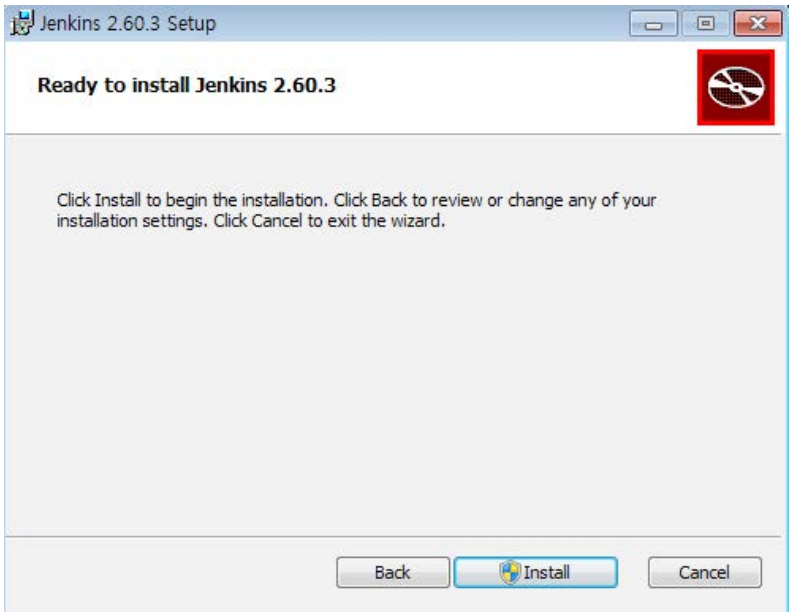
다운받은 파일의 압축을 풀고, Jenkins Windows 설치 파일을 실행한다.



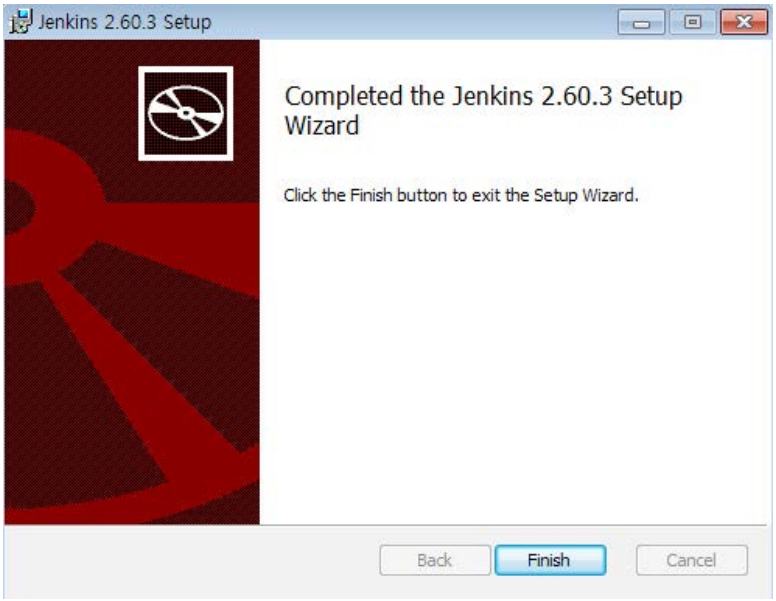
계속 Next를 클릭하여 설치를 진행한다.







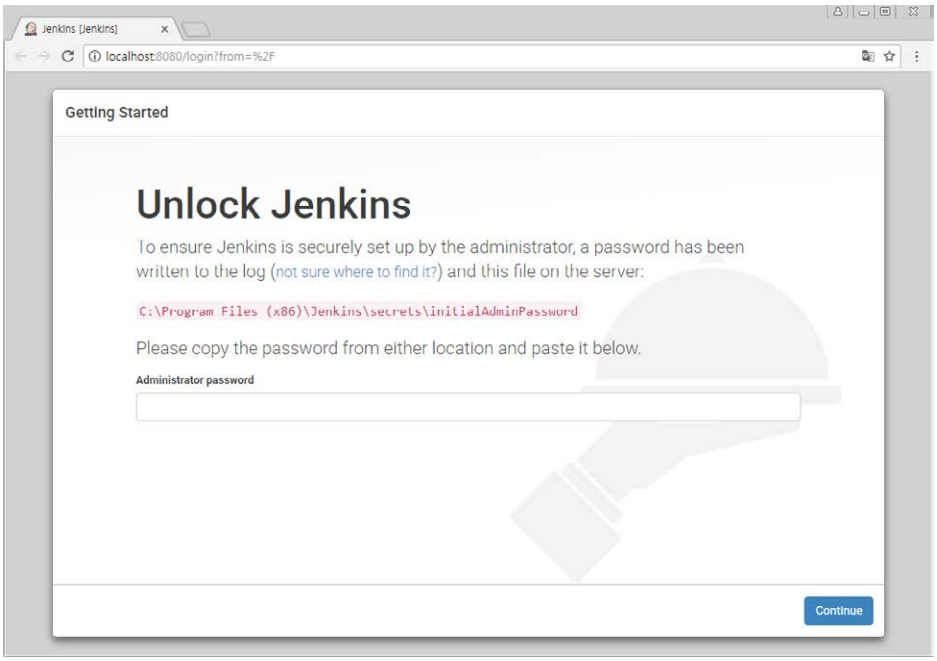
Install 버튼을 클릭 한다.



Finish 버튼을 클릭한다.

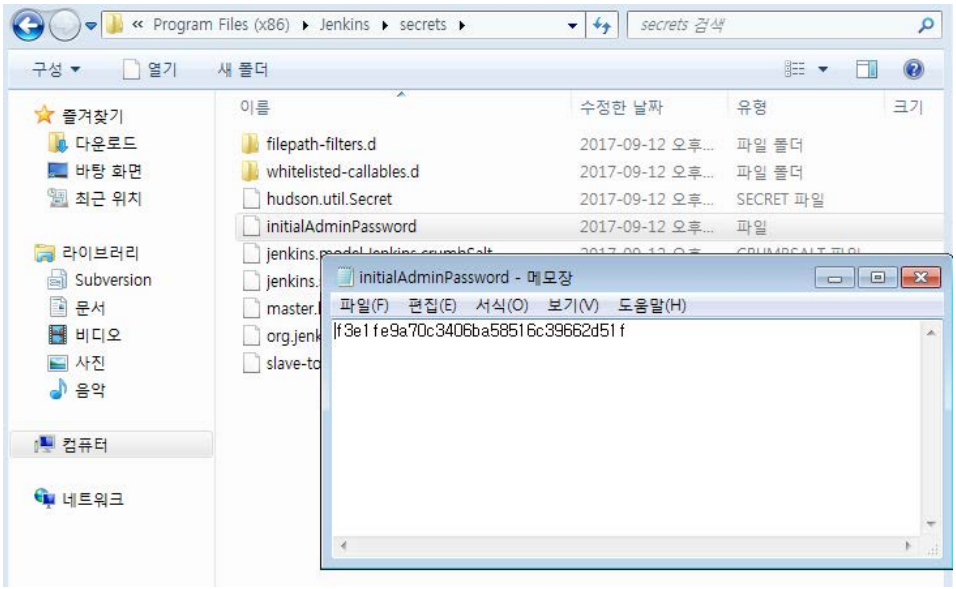
Jenkins 설치가 완료되면 웹 브라우저에서 설치된 Jenkins로 접속(http://localhost:8080)이 된다.

IE8이하 버전에서 설치진행이 되지 않을 수 있다. 원활한 Jenkins 설치를 하기 위해서는 크롬, 파이어폭스 설치 또는 IE 버전 업그레이드를 하여 설치를 진행하여야 한다.



붉은색의 경로에서 암호를 확인하여 로그인을 하여야 한다.
“c:₩Program Files (x86)₩Jenkins₩secrets₩initialAdminPassword”

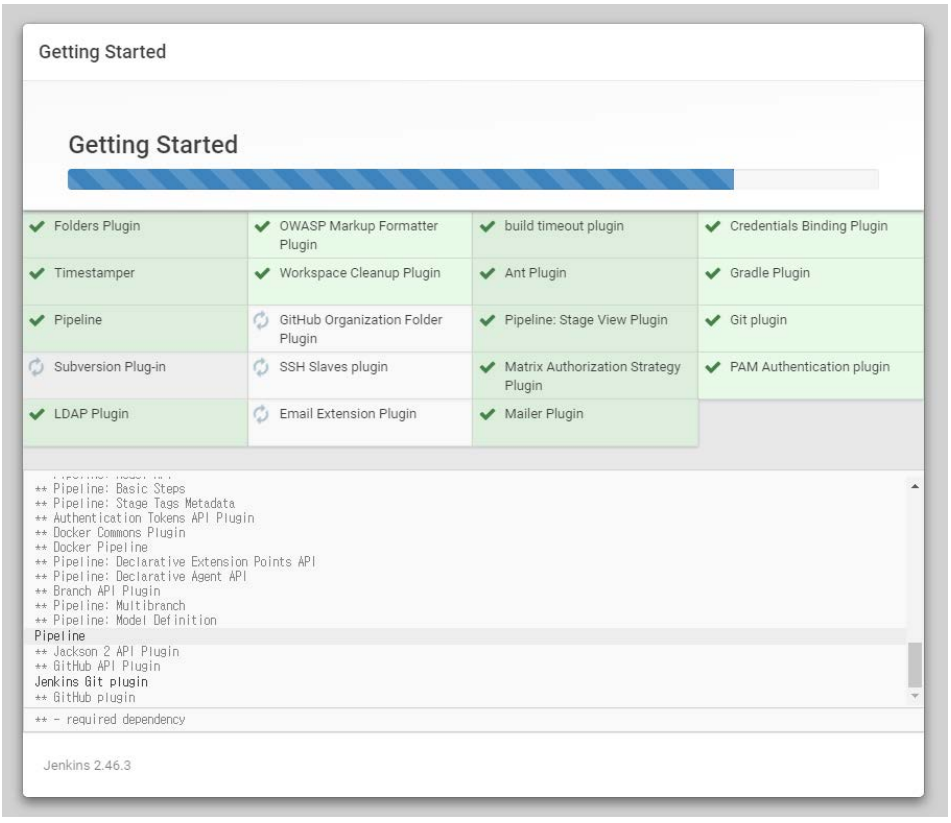
위의 파일을 메모장 같은 편집기로 열어 확인된 암호를 ‘Administrator password’ 에 입력하고 “Continue” 버튼을 선택한다.



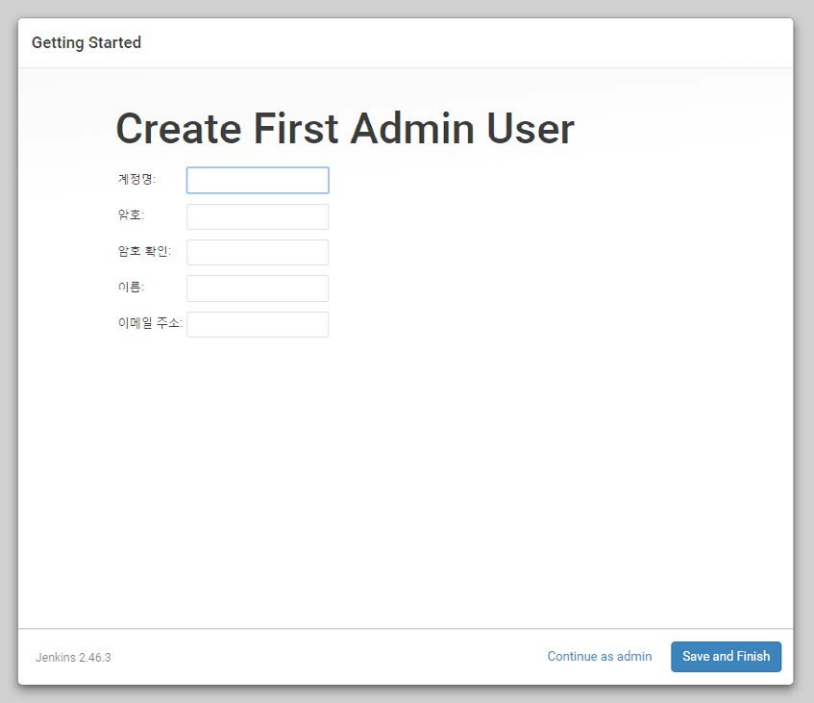
다음화면에는 옵션 설정을 할 수 있으며 기본적인 설치인 왼쪽을 선택한다.



아래 화면과 같이 설치가 진행중인 것을 확인 할 수 있다.

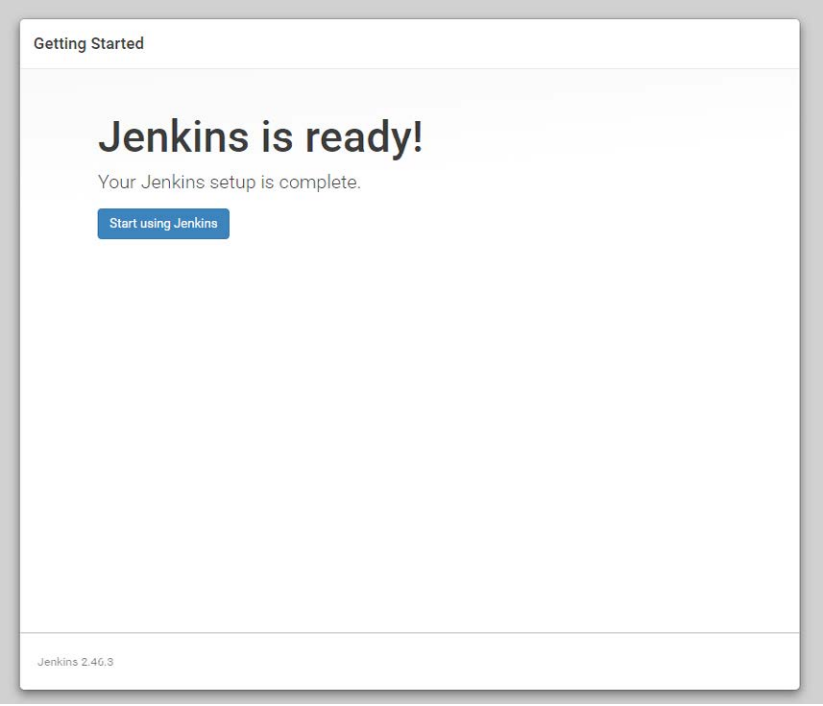


설치가 완료 되면 관리자 계정 생성 화면이 나타난다.



각 항목을 입력하고 Save and finish 를 선택한다.

정상적으로 완료가 되면 아래와 같은 화면이 나타난다.

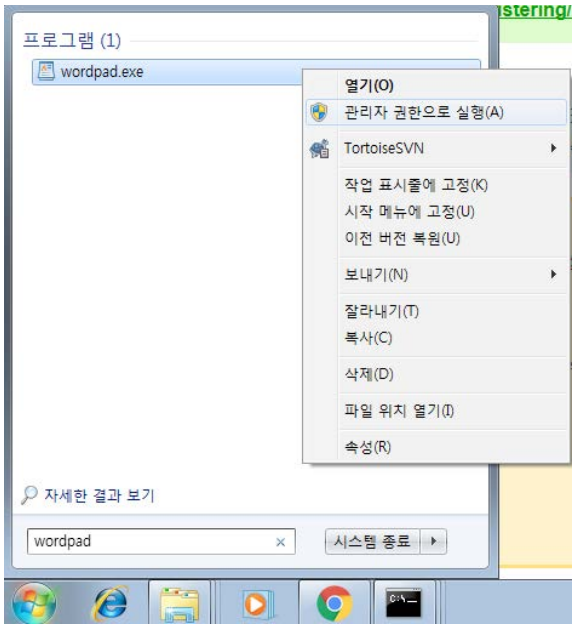


12.3 Jenkins PORT 변경 방법

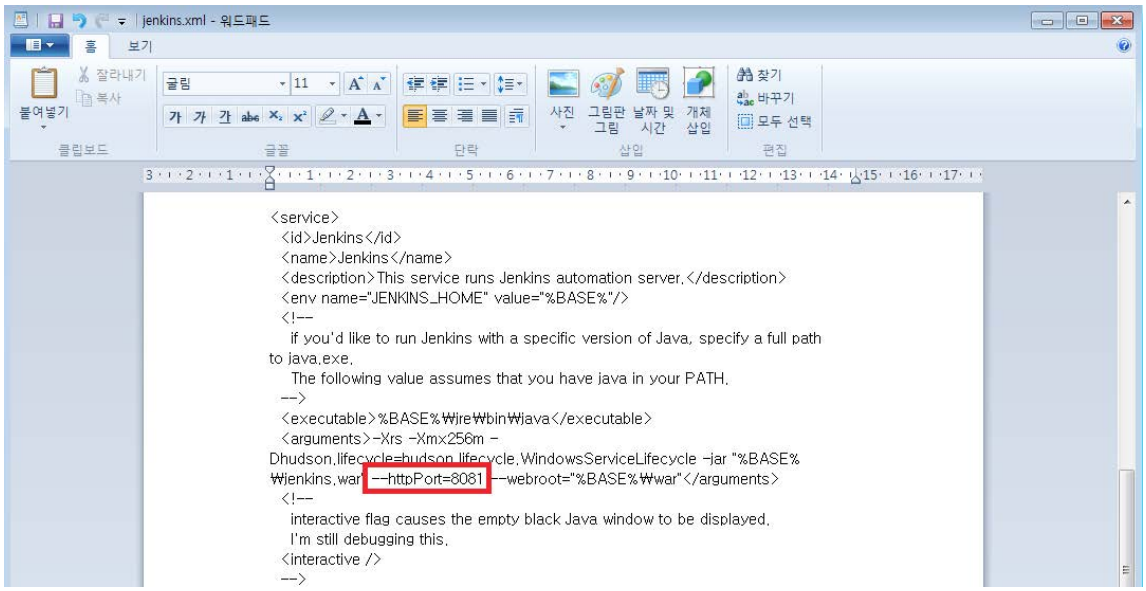
본 가이드에서 Tomcat을 사용하지 않고 있어 기본 포트를 사용한다. 환경에 따라 필요할 경우 포트 변경을 하도록 한다. 포트를 변경할 필요가 없다면 포트 변경 방법은 참고만 하고 넘어가도 된다.

Jenkins를 설치하는 Server에 Tomcat이 설치 되어있다면, Jenkins와 port(8080)가 같아서 같이 사용할 수 없게 된다. Tomcat이 설치 되어있고, Jenkins와 동일한 port(8080)을 사용하고 있다면 둘 중 하나의 port를 변경해 주어야 한다.
jenkins의 port 변경은 "Jenkins.xml"(C:₩Program Files (x86)₩Jenkins₩jenkins.xml) 파일에서 port를 변경하고 jenkins를 재시작 해주면 된다.

윈도우키를 누르고 검색창에 wordpad 검색어를 입력하여 wordpad에서 우클릭하여 관리자 권한으로 실행한다.(관리자 권한으로 실행을 하지 않으면 편집은 되나 저장이 되지 않는다.)



편집기에서 Jenkins.xml 파일을 열어서 --httpPort=8080을 원하는 port 번호로 변경하여 저장한다.



Jenkins 설치가 완료되고 port 변경 후 재시작을 하고, 웹 브라우저에서 변경된 port로 Jenkins 접속(http://localhost:변경한port)을 할 수 있다.

12.4 Jenkins 접속

Jenkins의 기본적인 설치가 끝나고 http://localhost:8080 로 접속(port를 변경하였다면 변경된 port로 접속)하여 로그인을 하면 아래와 같은 Jenkins main 화면이 나타난다.



12.5 Jenkins 관리

좌측 메뉴에서 Jenkins 관리를 선택하면 아래와 같이 Jenkins 관리 메뉴들이 나타난다.



주로 사용하게 되는 메뉴는 다음과 같다.

메뉴	용도
시스템 설정	환경변수 및 경로 정보등을 설정하는 메뉴로 Jenkins 동작에 필요한 주요 설정을 한다.
Configure Global Security	권한 및 접근 통제를 설정해서 Jenkins 보안을 강화하는 메뉴이다.
Global Tool Configuration	전역으로 사용되는 도구들의 설정을 할 수 있는 메뉴이다.
Reload Configuration from Disk	메모리에 로딩된 모든 데이터를 버리고 파일 시스템에서 새로 로딩한다. 관리 메뉴에서 설정을 변경하지 않고 설정 xml을 에디터로 직접 수정했을 때 사용할 수 있지만 설정 xml을 직접 수정하는건 권장하지 않는다.
플러그인 관리	Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 업데이트, 사용, 미사용으로 설정할 수 있다.

1) 시스템 설정

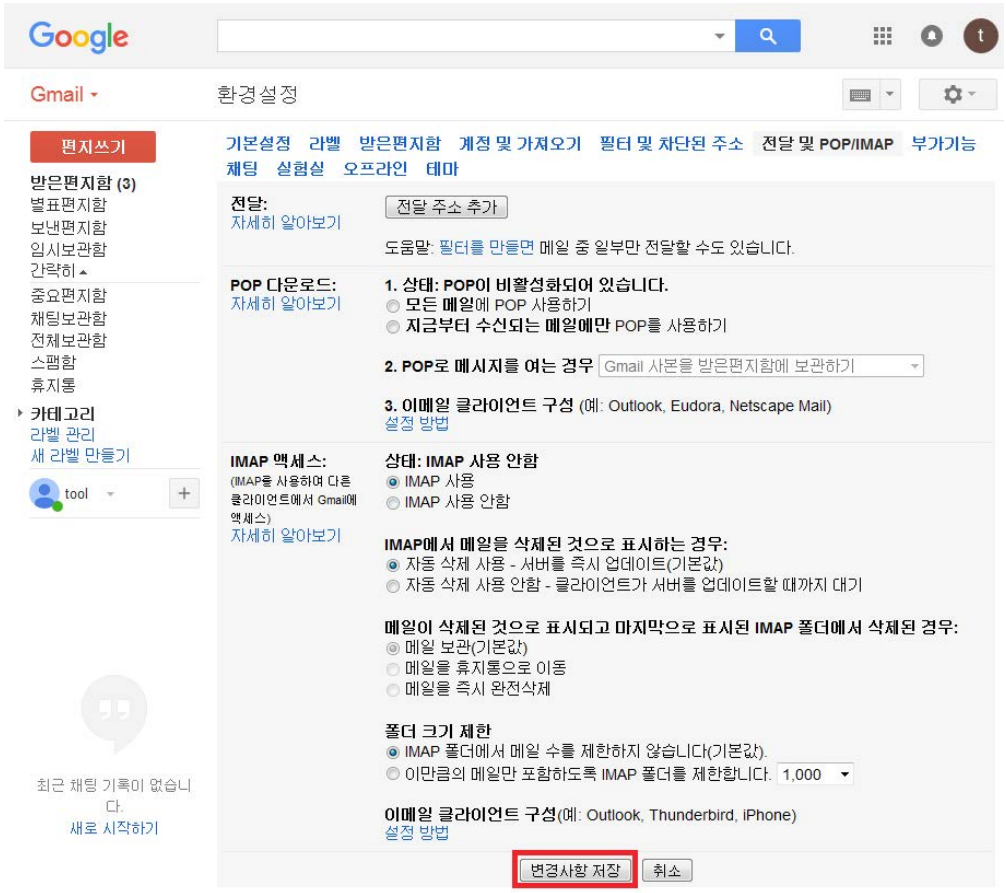
1) E-mail로 알려줌(Notification)

Jenkins에서 E-mail이 발송 될 수 있도록 설정할 수 있다.

메일 설정은 Gmail 을 사용하여 설정하는 방법을 설명할 것이다. 내부적으로 사용하는 메일 서버가 있다면 해당 메일 서버 정보로 설정을 하면 된다.

gmail 을 보내는 메일 서버로 사용하기 위해서는 gmail환경설정에서 IMAP을 사용으로 설정해야 한다.





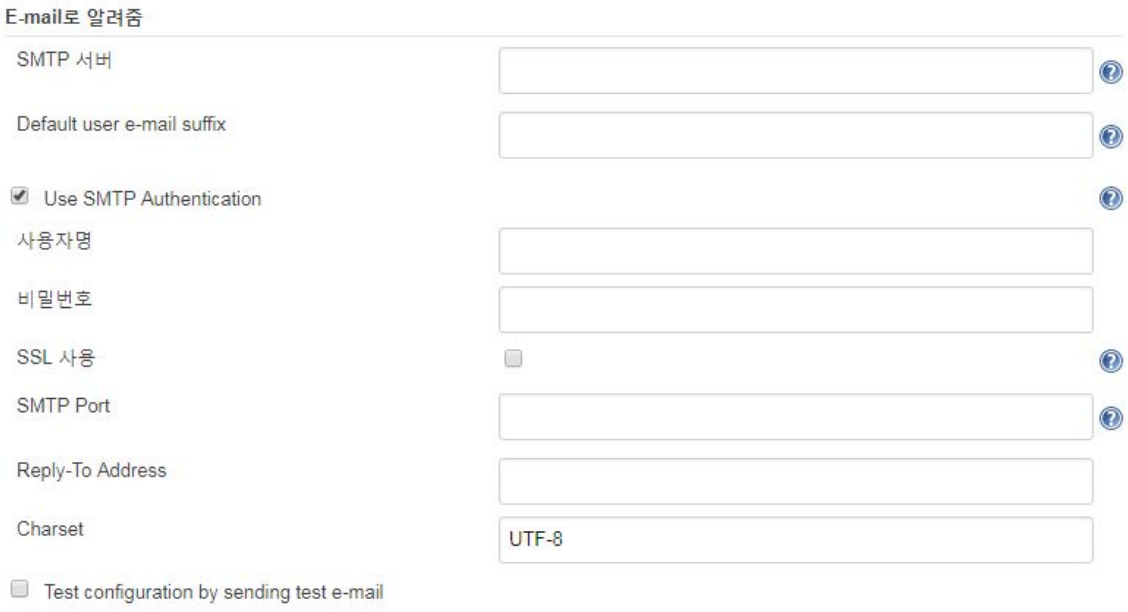
Jenkins에서는 Jenkins 관리 > 시스템 설정 에서 설정할 수 있다. 시스템 설정 화면의 마지막에 'E-mail로 알려줌' 이라는 항목이 있다.



Jenkins아래 설정 항목에서 고급 버튼을 클릭한다.



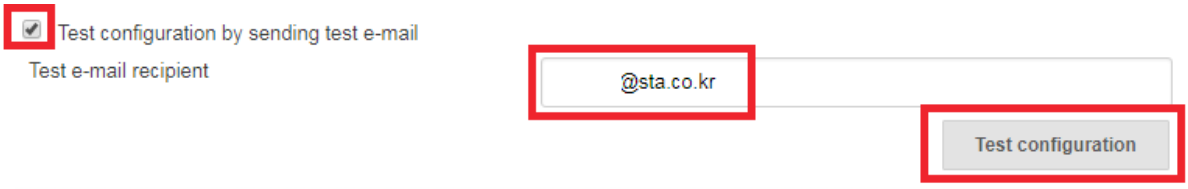
고급 버튼을 클릭하면 아래와 같이 추가 입력항목이 나타나며 서버 계정 정보를 입력하기 위하여 Use SMTP Authentication 을 체크해 준다.



입력하여야 하는 항목은 아래와 같다.

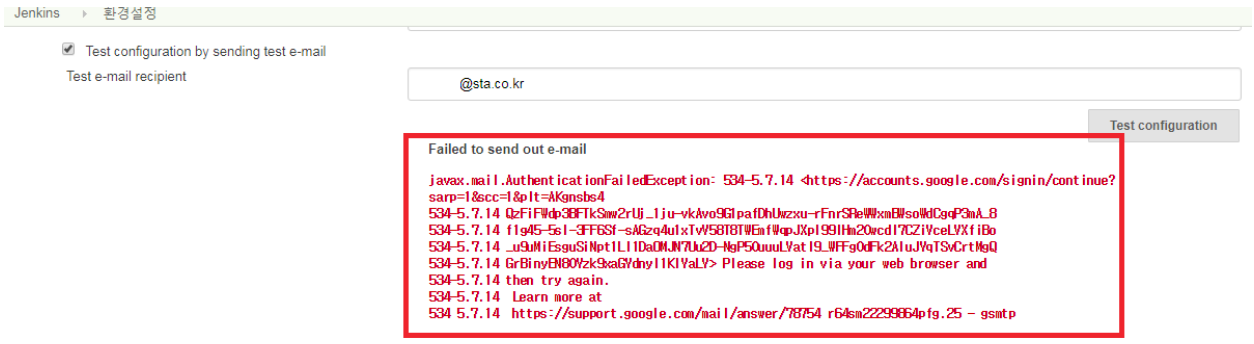
- SMTP 서버 : smtp.gmail.com
- Default user e-mail suffix : 메일 주소(@gmail.com 등)를 기본으로 사용하려면 입력.
- 사용자명 : gmail 사용자 정보
- 비밀번호 : gmail 사용자 정보
- SSL 사용 : 체크(사용함)
- SMTP Port : 465

Test configuration by sending test e-mail 을 체크하면 입력한 값으로 메일 테스트를 할 수 있다.



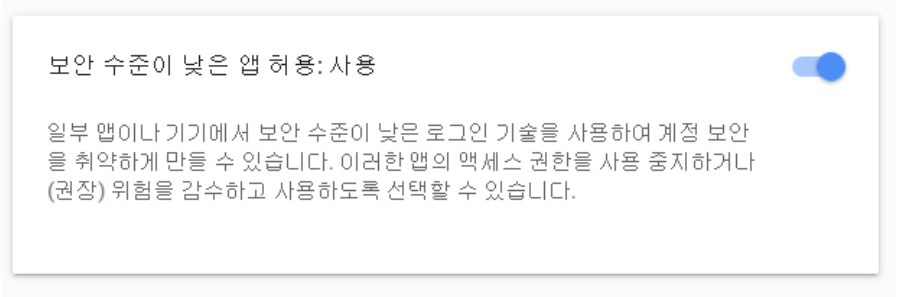
테스트 메일을 수신할 메일계정을 입력하고 Test configuration 버튼을 클릭하면 테스트 메일을 발송하게 된다.

만약, 아래처럼 에러가 발생을 한다면 gmail에서 보안 관련 설정을 해주어야 한다.

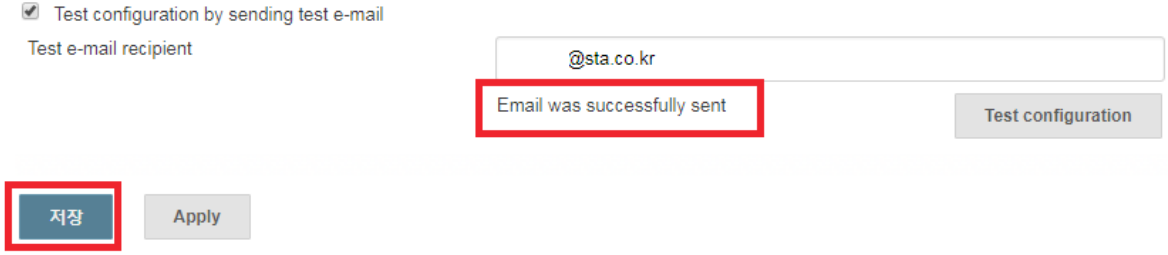


gmail에서 로그인 및 보안 메뉴에서 보안 수준이 낮은 앱 허용 항목을 사용으로 변경해 주어야 한다.





gmail에서 설정을 변경한 후 다시 Test configuration 버튼을 클릭하면 아래와 같이 메일 전송이 되었다는 메시지를 확인 할 수 있다.
저장 버튼을 클릭하면 설정이 저장된다.



2) Extended E-mail Notification

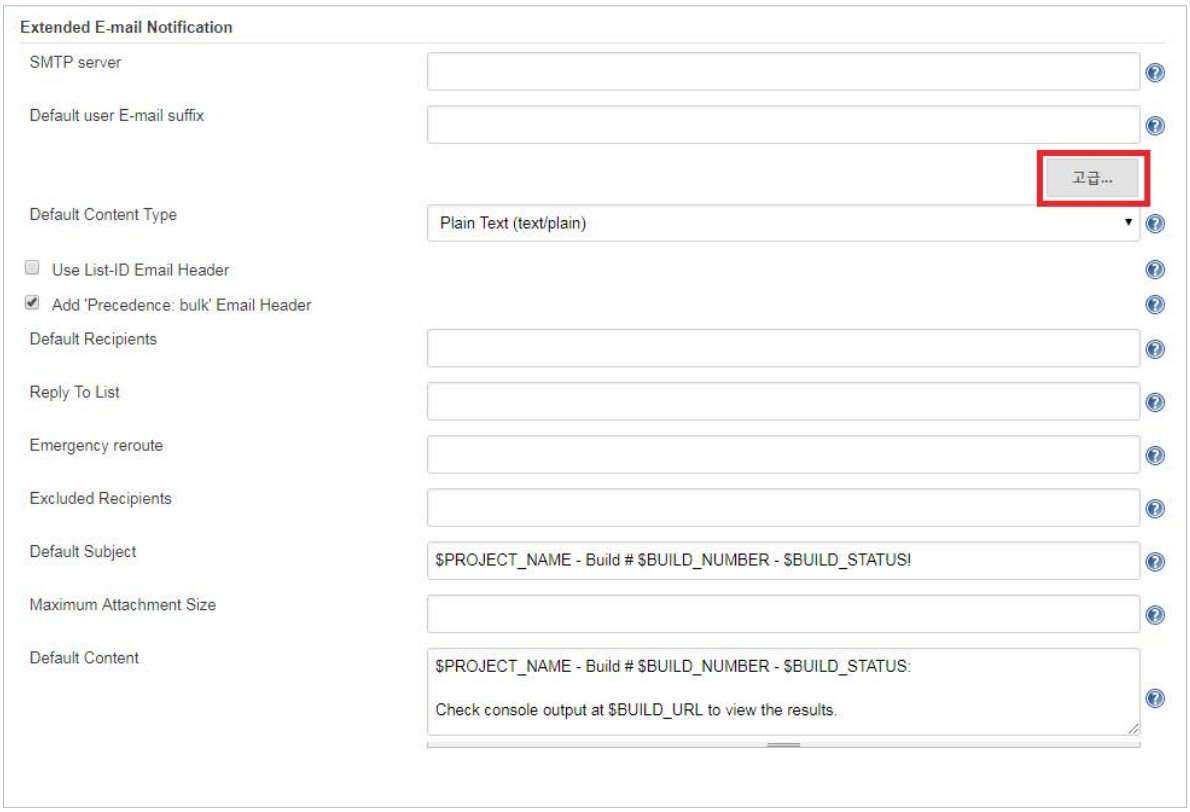
Jenkins에서 빌드 결과를 다양한 조건과 보낼 정보를 가공하고 수신자를 설정할 수 있으며, 파라미터와 변수를 지원하여 동적으로 설정하여 받아 볼 수 있다. 이와 같은 설정은 Jenkins 관리 > 시스템 설정의 "Extended E-mail Notification" 에서 할 수 있다. 여기서 설정하는 것은 전역 설정이며, 각 Job(프로젝트) 구성에서 이곳 설정을 기본으로 사용을 하게 된다.



설정 항목은 아래와 같으며, 필요한 항목을 확인하고 입력하면 된다.

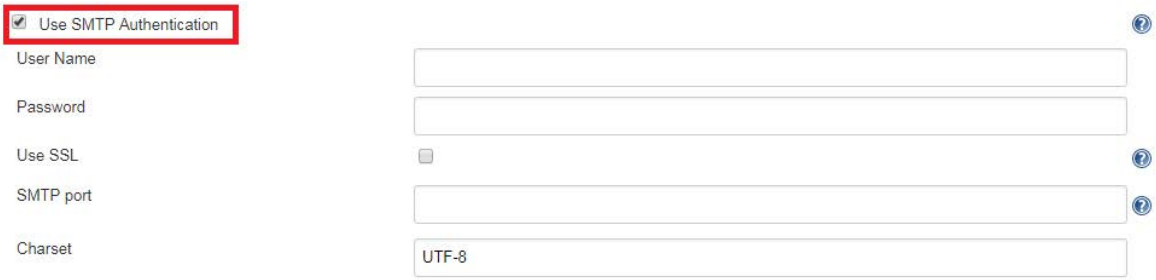
- SMTP server : SMTP 서버
- Default user E-mail suffix : 기본 이메일 사용자
- (고급버튼 클릭)
  - Use SMTP Authentication : 체크하게 되면 SMTP 사용자 계정정보 입력항목이 나옴
  - Use SSL : SSL 사용
  - SMTP port : SMTP 포트
  - Charset : 문자셋
- Default Content Type : Plain Text(text/plan) - 일반텍스트 / HTML(text/html) - HTML텍스트
- Use List-ID Email Header : 목록 ID 이메일 헤더 사용
- Add 'Precedence: bulk' Email Header : 모든 이메일에 'Precedence: bulk'를 붙여서 송신. 이 헤더가 있으면 송신자가 휴가나 부재중이라 자동 응답 메시지로 설정 했을 경우 송신자에게 가는 것을 방지해 준다.
- Default Recipients : 기본 수신자. 콤마를 구분자로 여러 주소 추가 가능.
- Reply To List : 수신자가 답장할 경우 응답 받을 메일 주소를 설정
- Emergency reroute : 긴급 재 전송 라우터
- Excludes Recipients : 제외된 수신자(필터링으로 수신자를 걸러냄)
- Default Subject : 기본 제목
- Maximum Attachment Size : 최대 첨부파일 크기
- Default Content : 기본 내용
- Default Pre-send Script : 미리 보내기 스크립트
- Default Post-send Script : 전송 후 스크립트
- Additional groovy classpath : 추가 그루비 클래스 패스

- Enable Debug Mode : 디버그 모드 활성화
- Require Administrator for Template Testing : 관리자 권한 있는 사용자에게 이메일 템플릿 링크 표시
- Enable watching for jobs : 감시 기능
- Default Triggers : 기본 트리거



SMTP server 사용을 할 때 로그인 정보 등 추가 입력사항이 있다면 고급 버튼을 클릭한다.

고급 버튼을 클릭하면 아래와 같은 추가 항목이 나타나며, 'Use SMTP Authentication' 을 체크하면 로그인 계정 입력항목(User Name, Password)이 나타난다.
gmail을 SMTP server로 사용할 경우 gamil 계정 정보와 Use SSL을 사용으로 체크하고 SMTP port 번호도 입력해 주어야 한다.

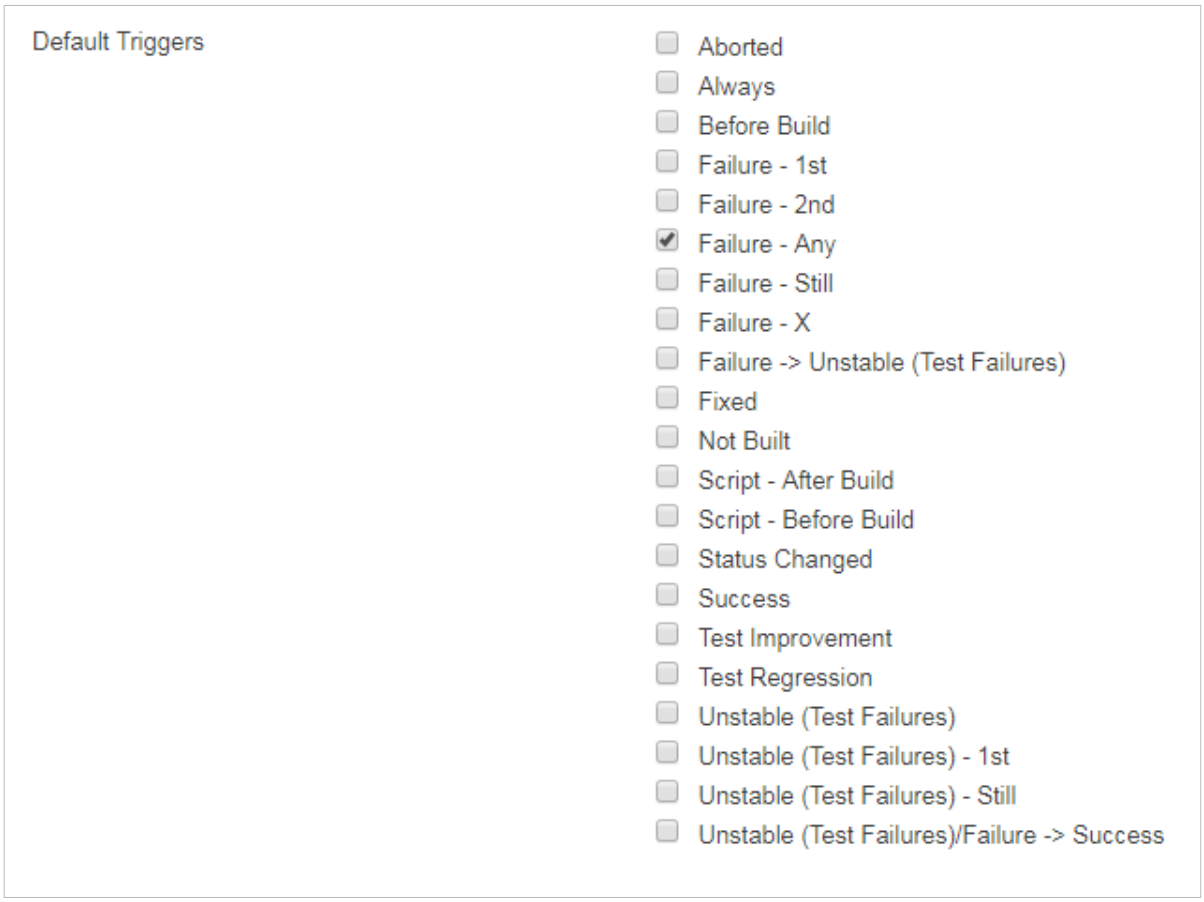


설정 아래 부분의 항목은 필요한 것이 있다면 입력을 한다.

메일 알림을 빌드 성공 할 때 또는 실패 할 때 아니면 항상 보낼지를 설정을 하려면 'Default Triggers' 버튼을 클릭하여 설정할 수 있다.



메일을 받고자 하는 시점을 아래의 목록에서 선택하여 설정한다.
다중선택이 되므로 필요한 항목을 체크하여 사용 하면 된다. 해당 설정은 전역으로 사용되는 기본 설정이기에 공통으로 사용할 항목에 체크하는 것이 좋을 것이다. 각 프로젝트에서 수정하여 설정 가능하다.



■ **프로젝트(Job)에서 사용하기**

전역 설정이 끝났다면 Job(프로젝트)별로 설정하는 방법을 알아보겠다.

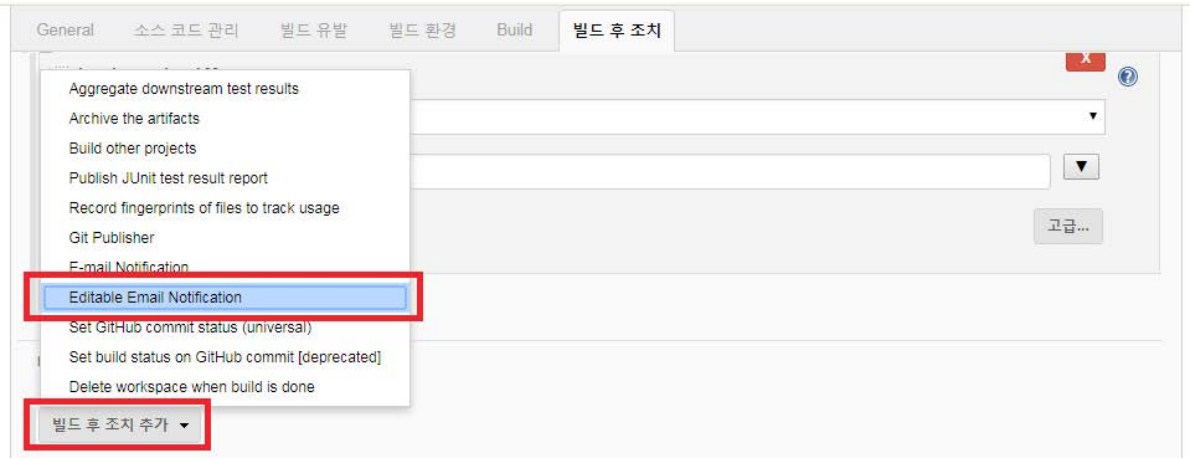
프로젝트를 선택 > 구성을 선택하여 설정 화면으로 이동을 한다.



상단의 빌드 후 조치를 클릭한다.



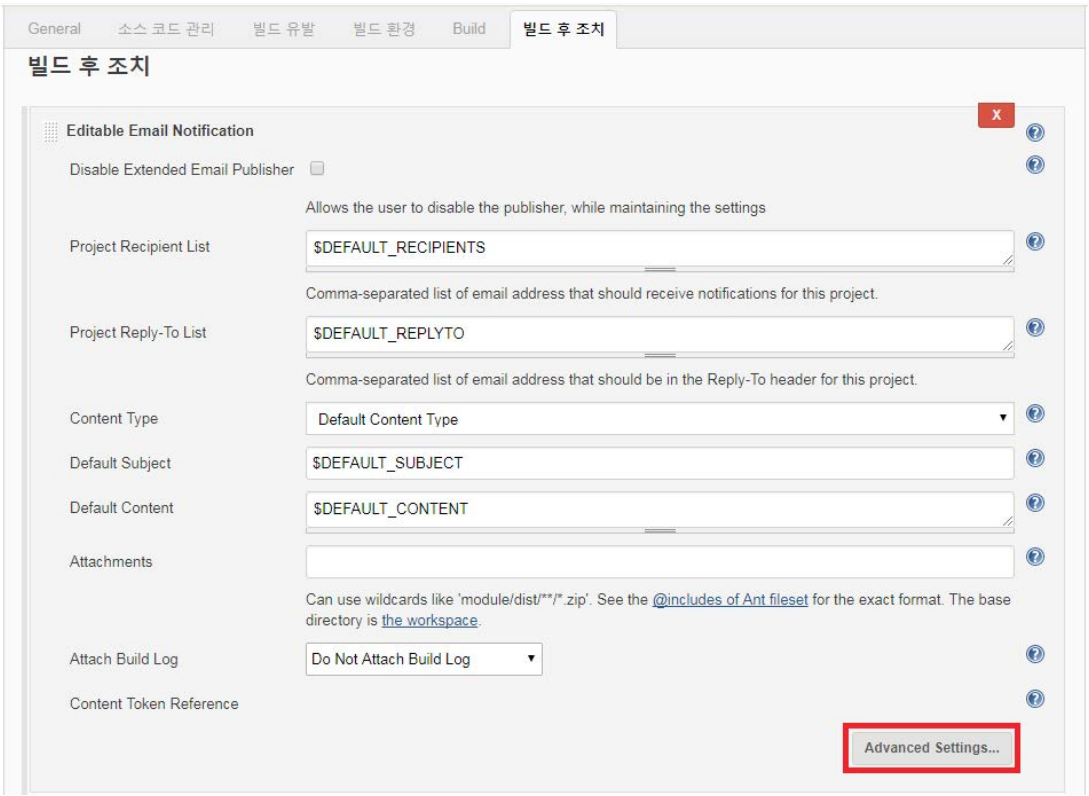
빌드 후 조치 추가 콤보 박스를 누르고 “Editable Email Notification”을 선택한다.



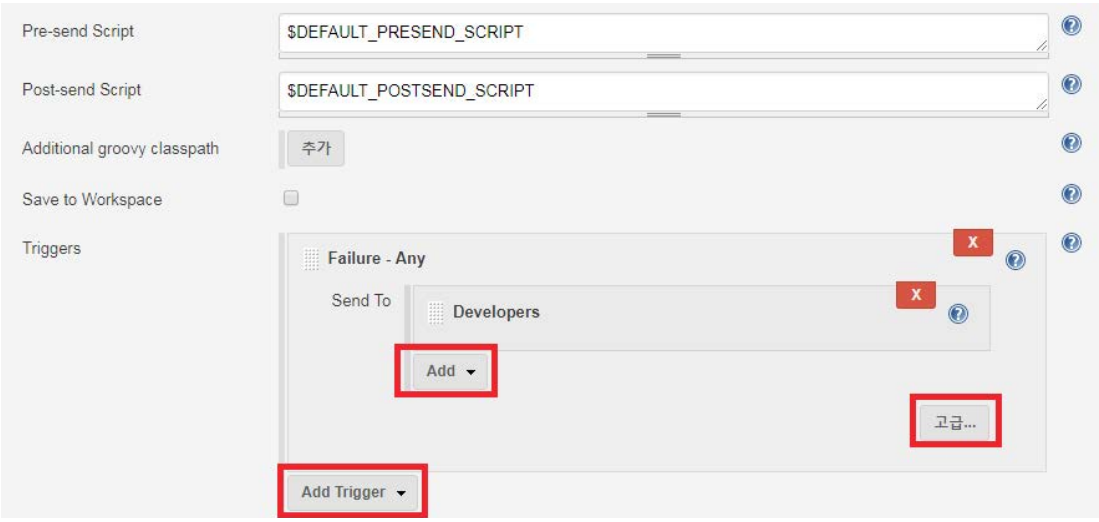
설정 항목은 아래와 같다.

- Project Recipient List : 메일 받을 주소를 설정한다. 콤마를 구분자로 여러 주소를 추가할 수 있다.
- Project Reply-To List : 수신자가 답변 메일을 보낼 주소를 설정
- Content Type : Content 형식을 설정
- Default Subject : 제목으로 전역 설정이 기본 값
- Default Content : 본문 내용으로 전역 설정이 기본 값
- Attachments : 첨부 파일 설정
- Attach Build Log : 빌드 로그를 이메일에 첨부할지 여부
- Pre-send Script : 이메일 전송전 실행할 스크립트
- Post-send Script : 이메일 전송후 실행할 스크립트
- Save to Workspace : 생성된 이메일을 작업공간에 저장
- Triggers : 어떤 이벤트시 누구에게 메일을 보낼 지 설정하는 부분이다. 다양한 조합으로 여러 가지를 설정할 수 있다.

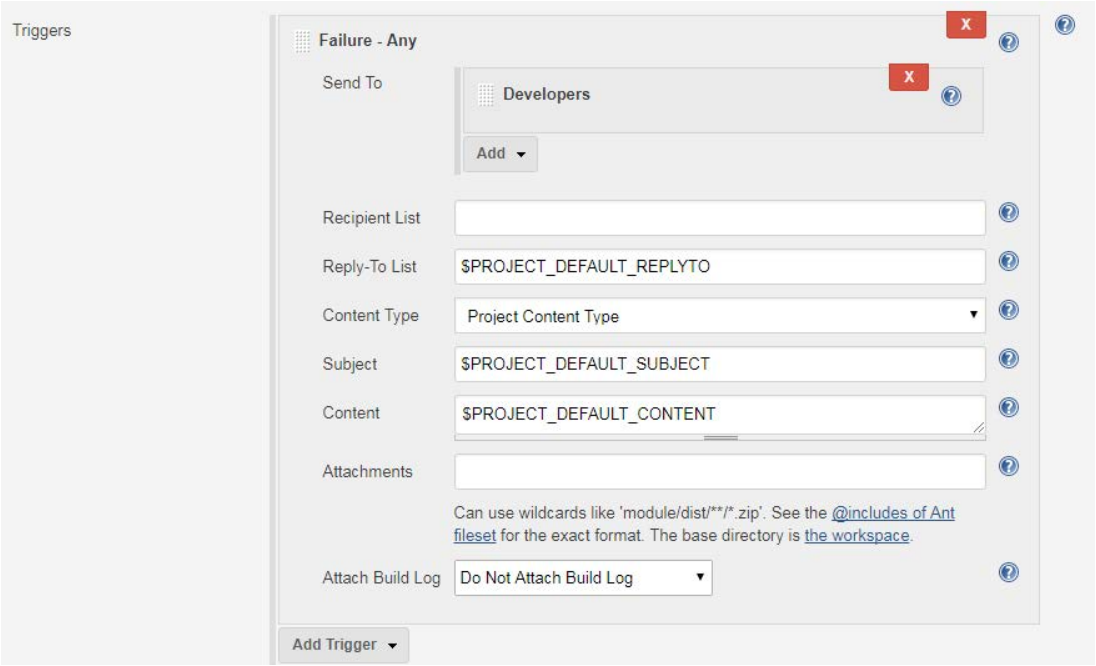
대부분 앞서 설정한 기본 값을 사용하도록 되어있다. 각 항목을 해당 프로젝트 별도로 수정 또는 추가하여 사용할 수 있다.



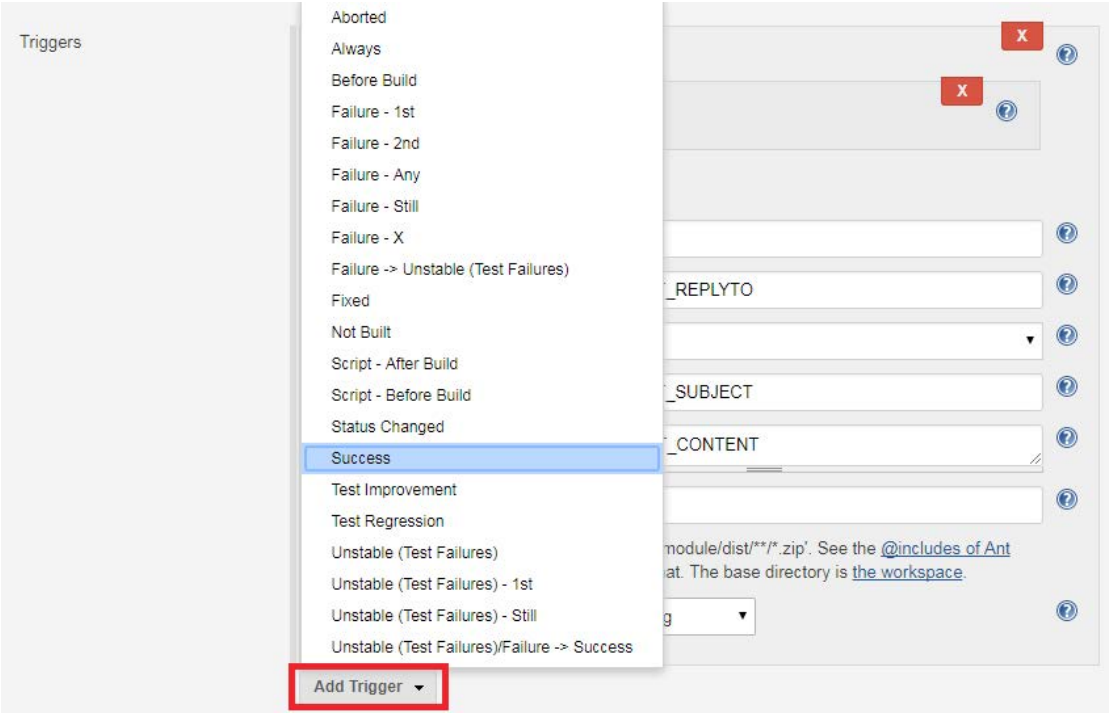
'Advanced Settings' 버튼을 클릭하여 추가 항목을 설정할 수 있다. 추가 항목에 Triggers 을 설정할 수 있는 항목이 있다.
Triggers 항목은 앞서 설정한 기본 값이 보여 지며 'Add Trigger' 버튼을 클릭하여 Trigger를 추가할 수 있으며, Trigger의 고급 버튼을 클릭하여 상세한 설정을 할 수 있다.



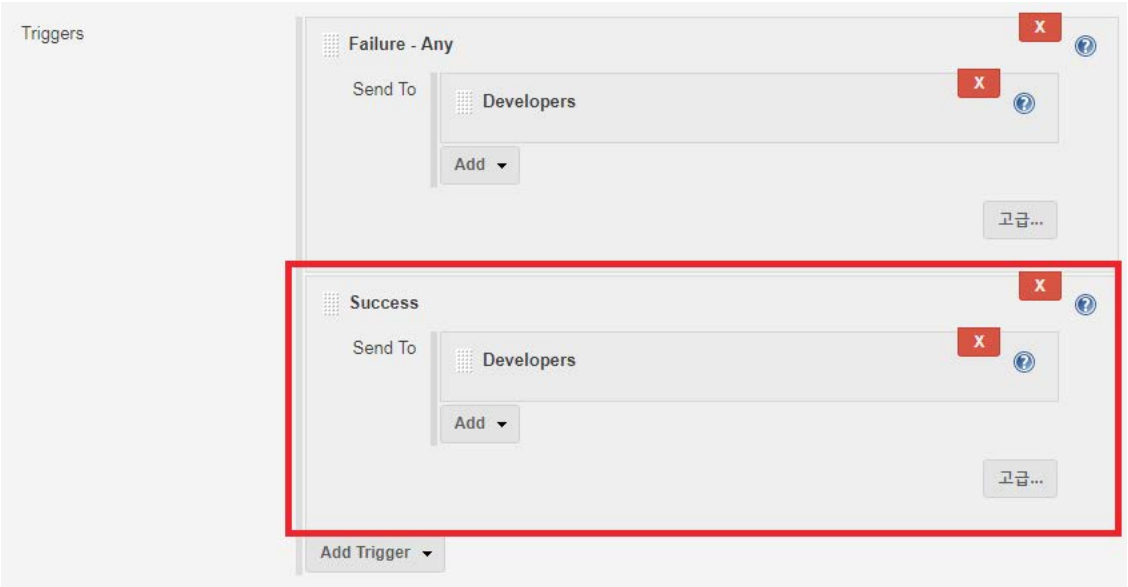
고급 버튼을 클릭하면 아래와 같이 입력항목이 나오며 해당 Trigger에서 수신자, 제목, 내용 등을 수정할 수 있다.



'Add Trigger' 를 클릭하면 추가 할 수 있는 항목들이 보인다.
추가를 원하는 항목을 선택한다.

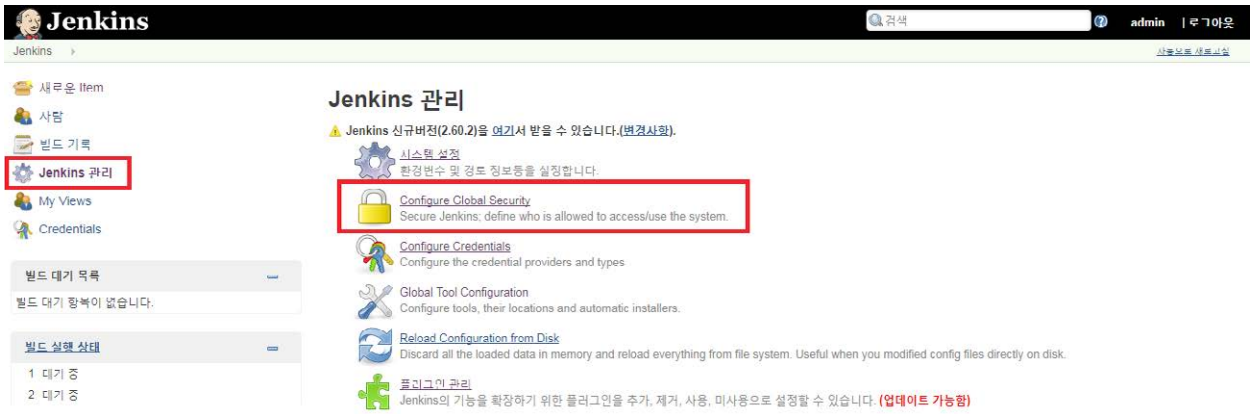


트리거 추가에서 Success를 선택하면 아래와 같이 Trigger가 추가되는 것을 확인할 수 있다.

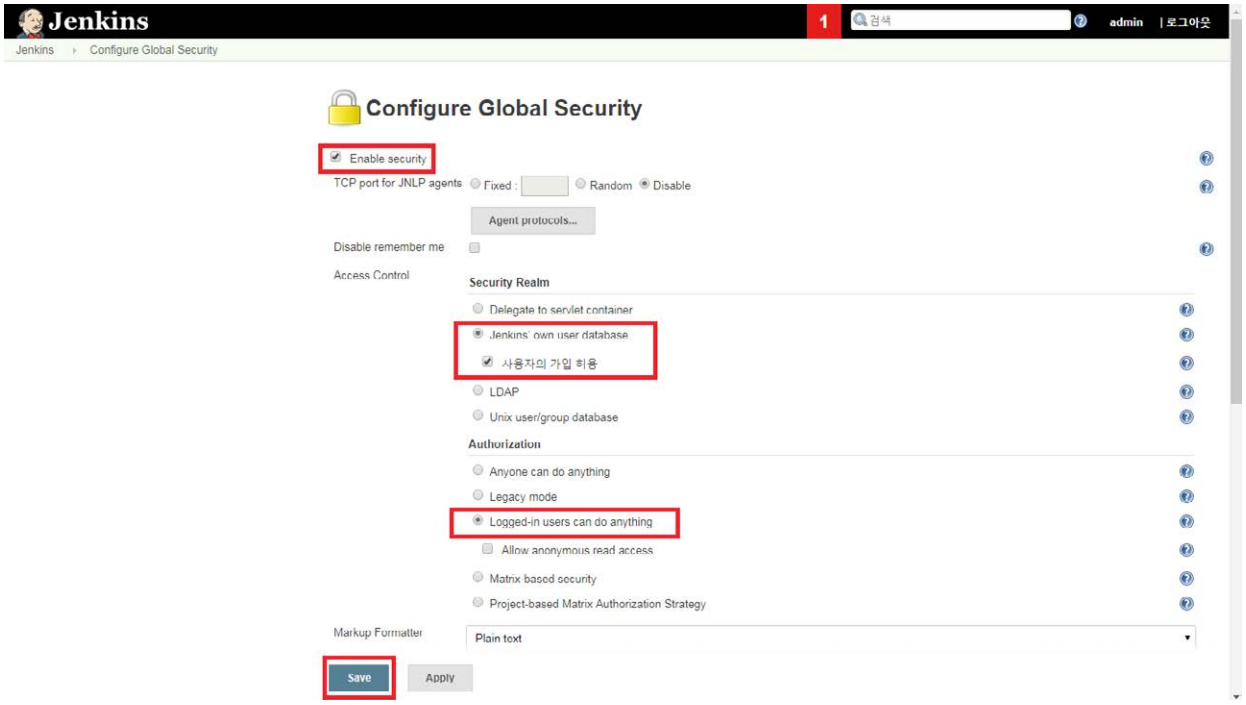


2) 사용자 등록과 권한 설정

"Jenkins 관리 → Configure Global Security" 를 선택한다.



사용자와 권한을 설정할 수 있는 화면이 표시된다.



설정화면에서 Enable security를 체크한다. "Security Realm"에서 "Jenkins' own user database"를 선택해서 Jenkins 내부 DB를 사용하도록 하고, 사용자의 가입 허용을 선택한다.
Authorization에서 "Logged-in users can do anything"을 선택하고 "Save"클릭한다.

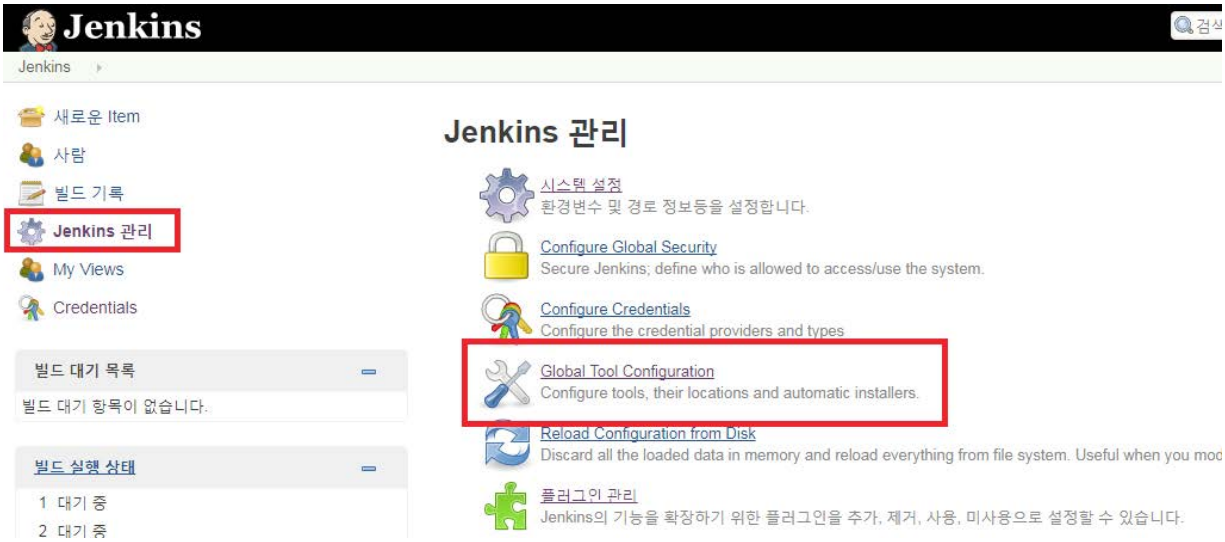
로그인 화면에서 계정 생성을 선택한다.



하여 사용자 정보를 입력한 후 “Sign up” 버튼을 선택하면 계정이 생성된다.

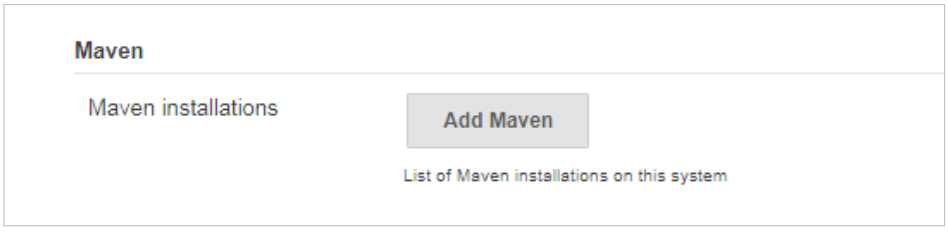
3) Global Tool Configuration

“Jenkins 관리 → Global Tool Configuration” 를 선택

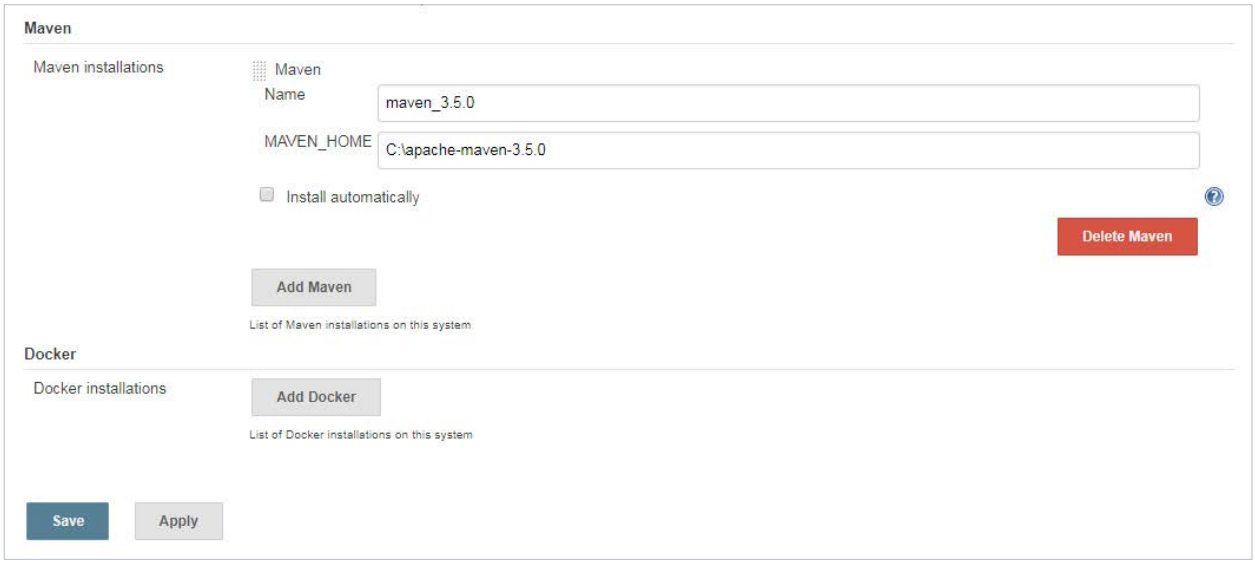


“Global Tool Configuration” 에서 Maven, JDK, Git, Gradle, Ant, Docker 를 설치하거나 설정할 수 있다.
이미 따로 설치가 되어있다면, 설치된 정보를 입력하면 되고 설치가 필요한 것이 있다면 “Install automatically”을 체크하여 설치 할 수 있다.

여기서 이미 설치를 하였던 Maven을 설정해보도록 할 것이다.
Add Maven 버튼을 클릭한다.



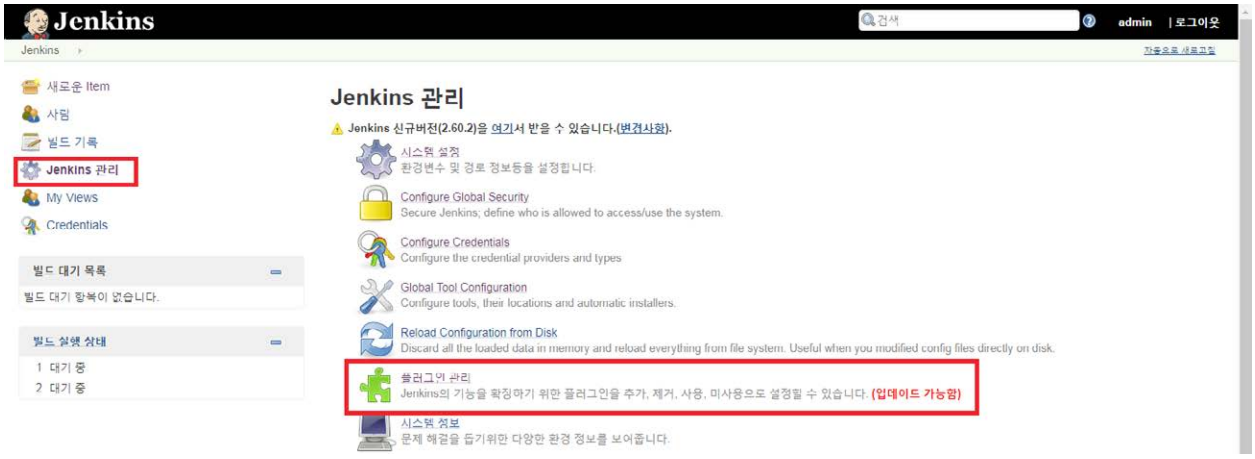
이미 설치를 하였던 Maven을 설정할 것 이라 “Install automatically”을 체크를 해제한다.
그리고 Name, MAVEN_HOME 항목에 설치 정보를 입력하면 된다.
Name은 알아볼 수 있도록 임의로 입력하면 되며, 여기서는 “maven_3.5.0” 이라고 입력을 한다.
“MAVEN_HOME”은 압축을 풀었던 maven 경로를 입력하고 하단의 Save 버튼을 클릭하면 설정이 저장된다.



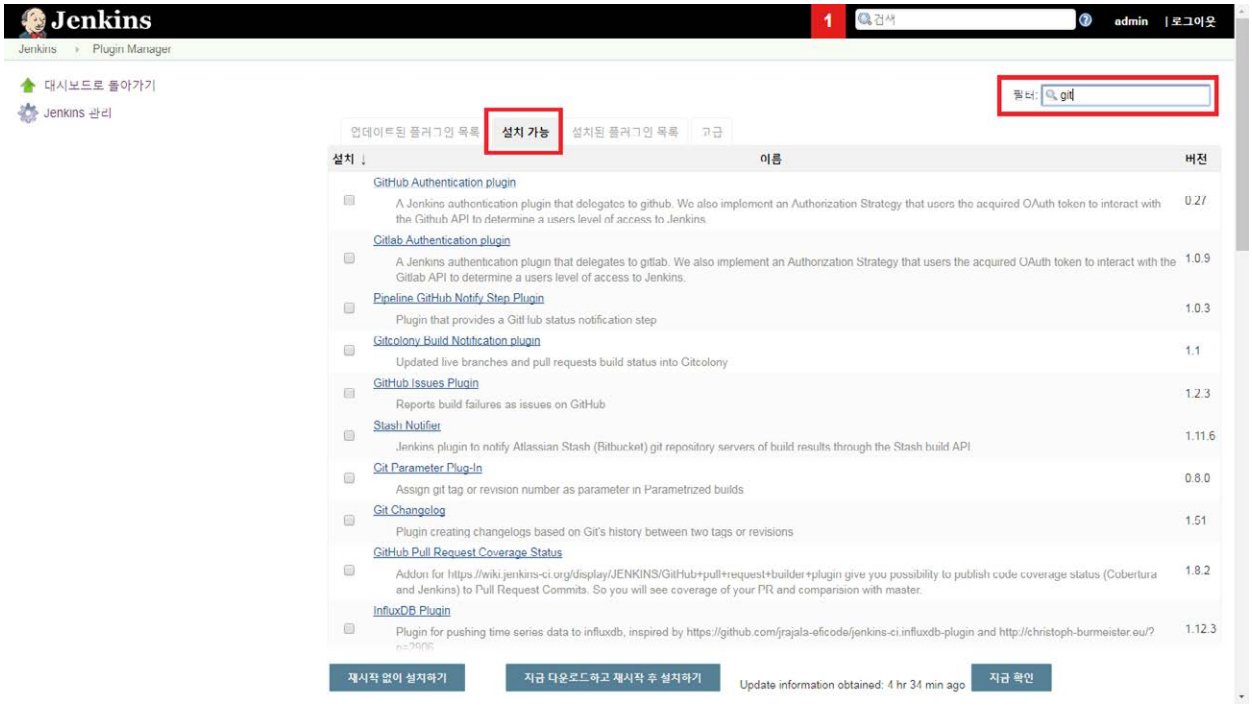
Maven을 더 추가 하고 싶다면 “Add Maven” 버튼을 클릭하여 다른 버전을 설치할 수도 있다.
Git, Gradle, Ant 등 설치 또는 설정도 위와 같은 방법으로 할 수 있다.

4) 플러그인 관리

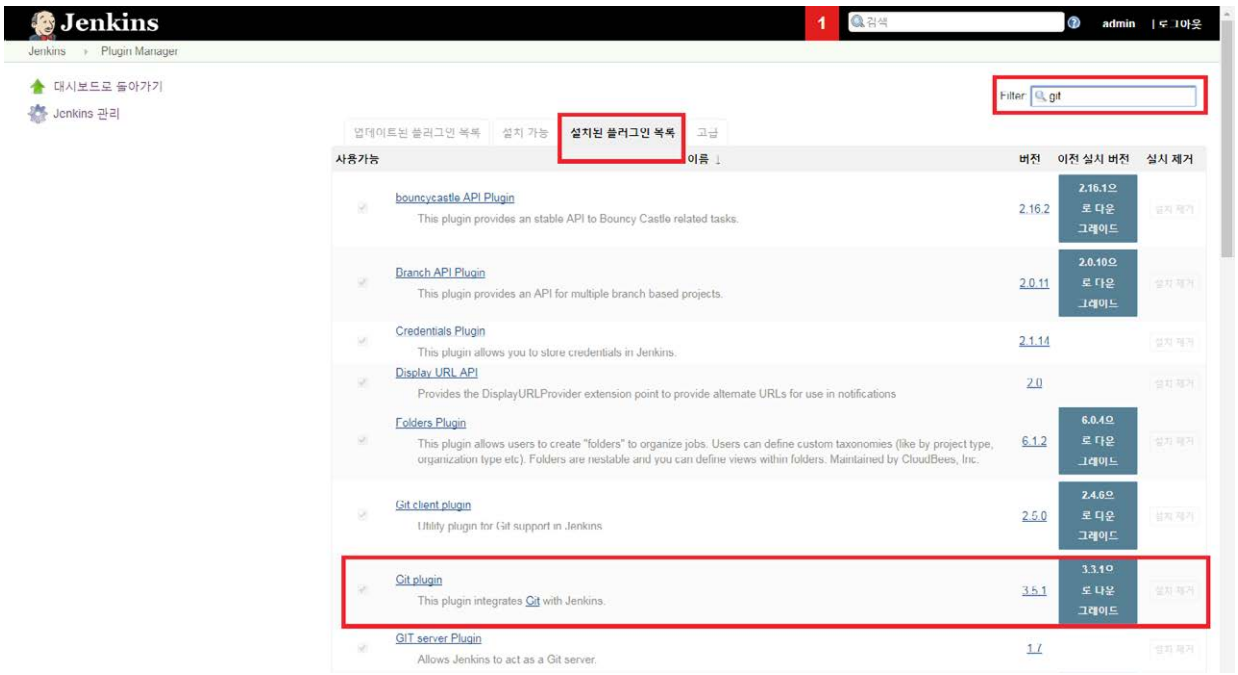
Jenkins에서 플러그인을 통하여 다양한 기능을 추가하여 사용할 수 있다. 플러그인 설치는 “Jenkins관리 → 플러그인 관리” 에서 설치 할 수 있다. jenkins를 설치하면 형상관리 관련 플러그인 등 많이 사용하는 플러그인이 이미 설치가 되어있으며, 플러그인 관리에서 설치된 플러그인과 플러그인 업데이트 확인 등을 할 수 있다.



설치할 플러그인은 설치 가능 탭을 선택하고 상단의 필터로 검색을 할 수 있으며, 검색된 목록에서 설치할 플러그인을 체크하고 설치를 하면 된다.

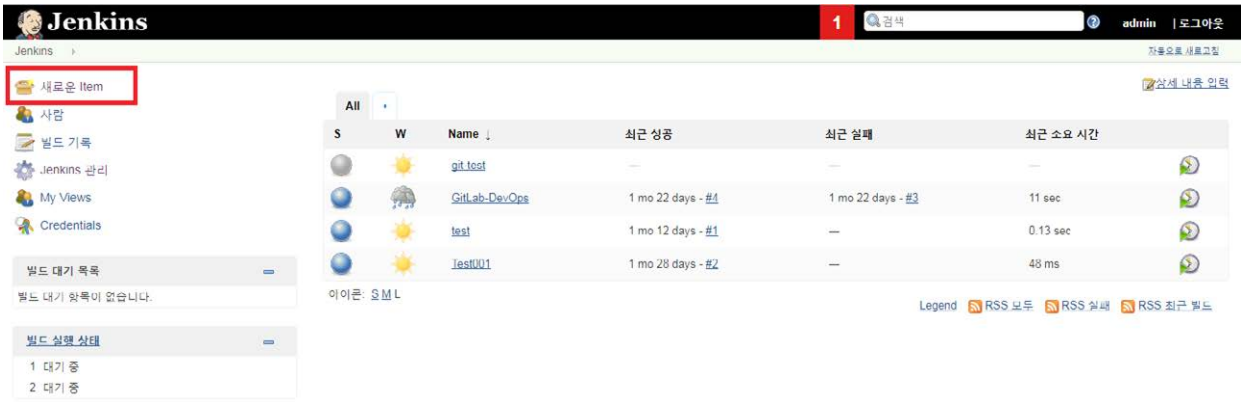


설치된 플러그인은 설치된 플러그인 목록에서 확인할 수 있으며, 설치 제거 및 다운그레이드를 할 수 있다.

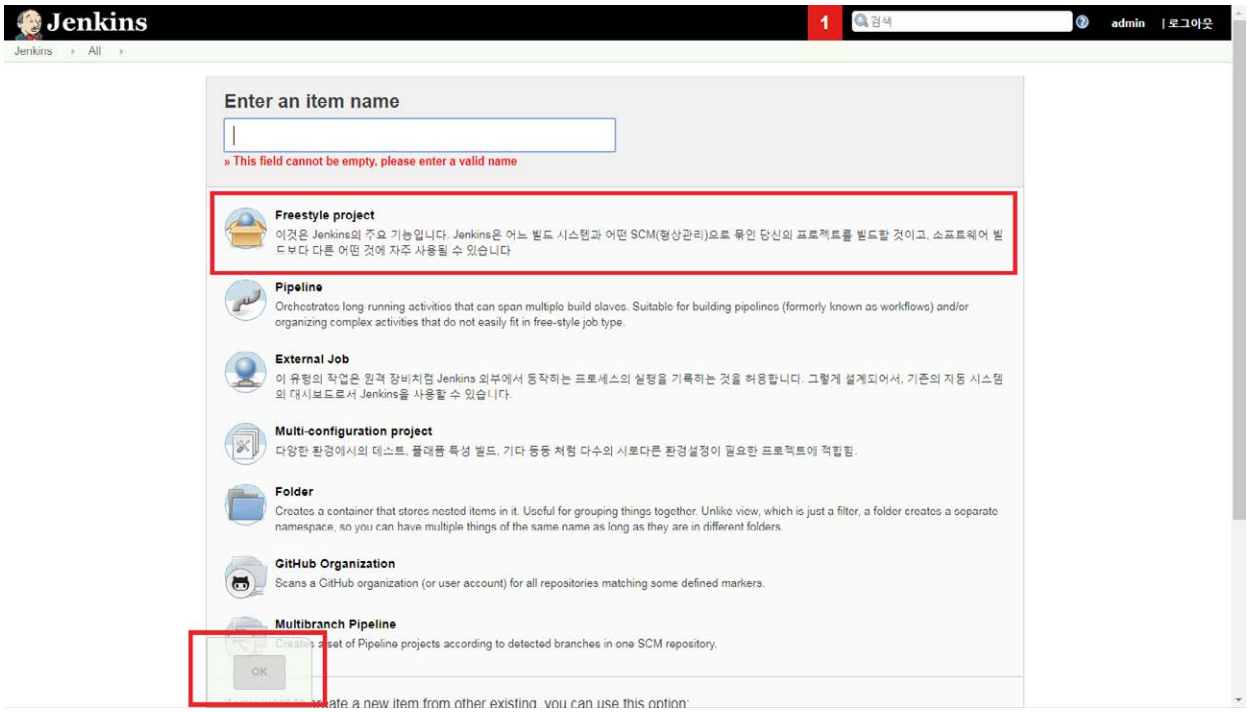


12.6 Jenkins 신규 아이템 추가 및 설정

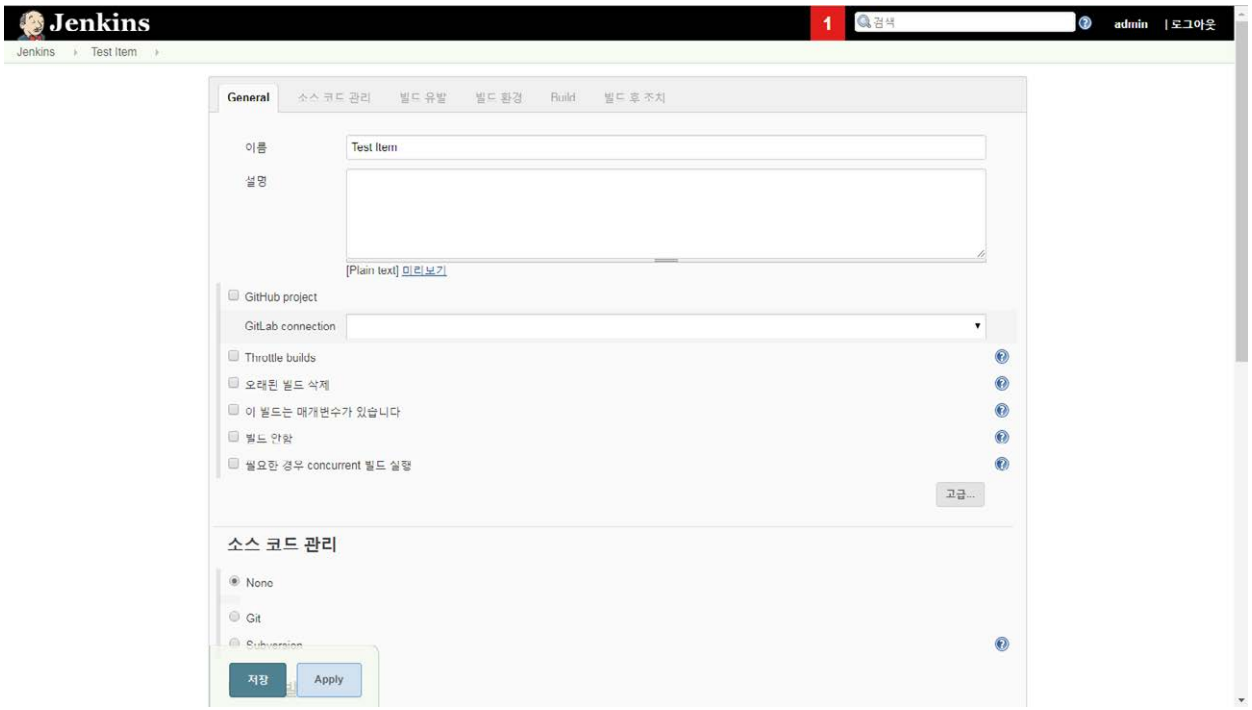
Jenkins 초기 화면에서 '새로운 Item'을 선택하여 신규 아이템(프로젝트)를 생성할 수 있다.



"Enter an Item name"을 입력하고 "Freestyle project"를 선택하고 아래 "OK" 버튼을 선택 한다.

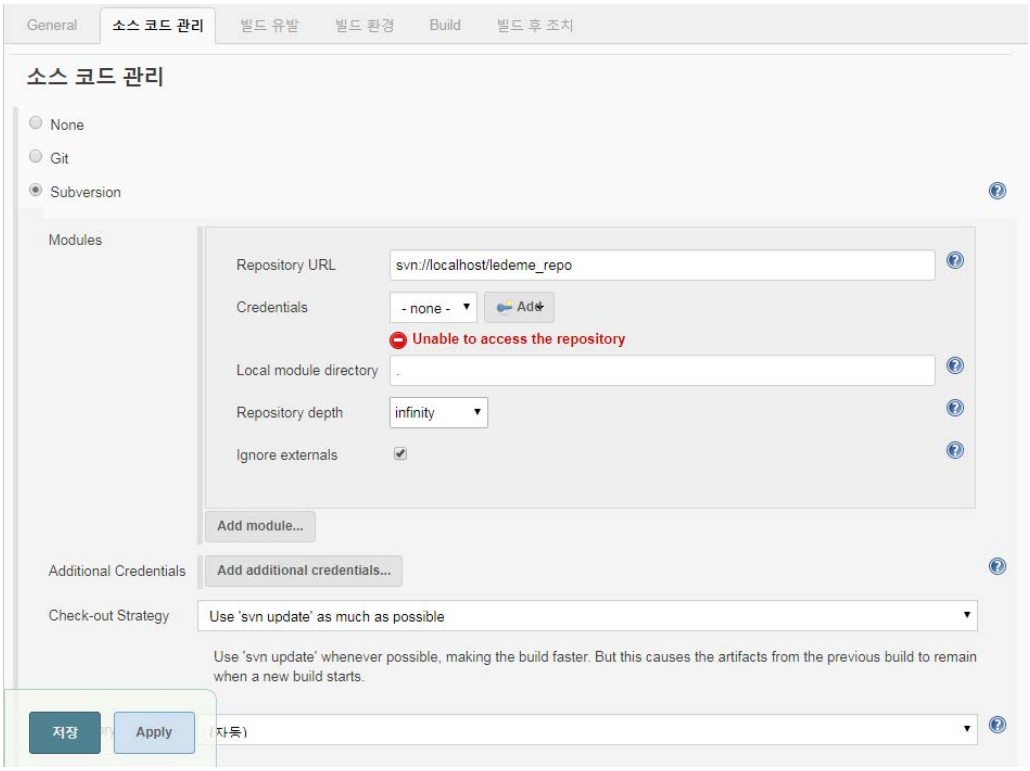


새로운 item을 생성하면 아래와 같은 설정 화면에서 기본설정(General), 소스코드관리, 빌드 유발, 빌드 환경, Build, 빌드 후 조치 대한 설정을 할 수 있다.

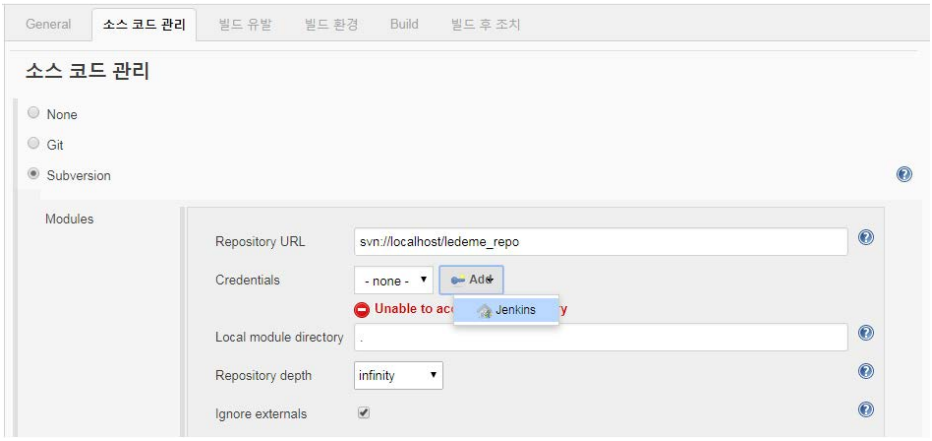


1) 소스 코드 관리

소스코드 관리 설정에서 Git 저장소 또는 Subversion 저장소와 연결 설정을 할 수 있다.



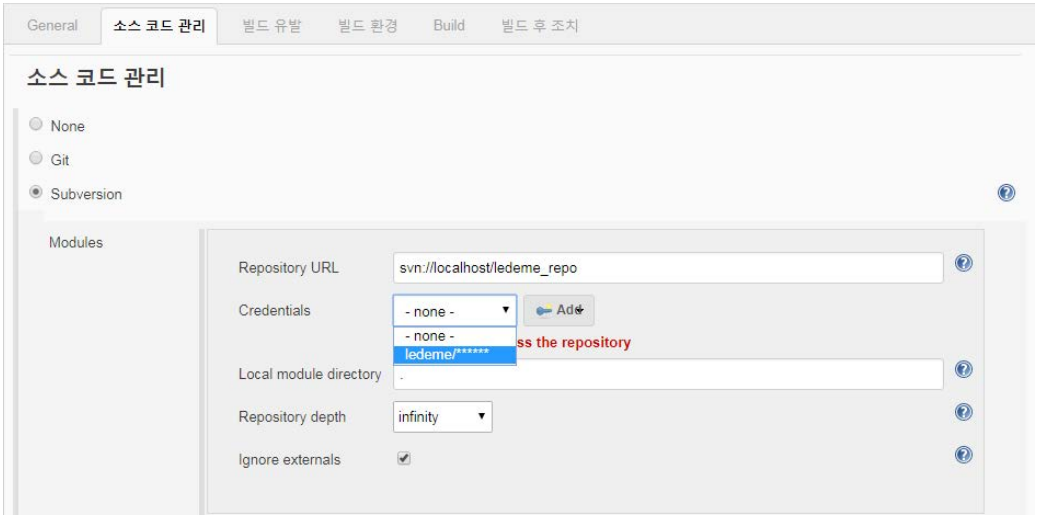
소스 코드 관리에서 “Repository URL” 을 입력하고, “Credentials” 에서 접속 계정을 선택하여주면 된다. 등록된 접속 계정이 없다면 “Add “ 버튼을 클릭하여 계정을 추가해 준다.



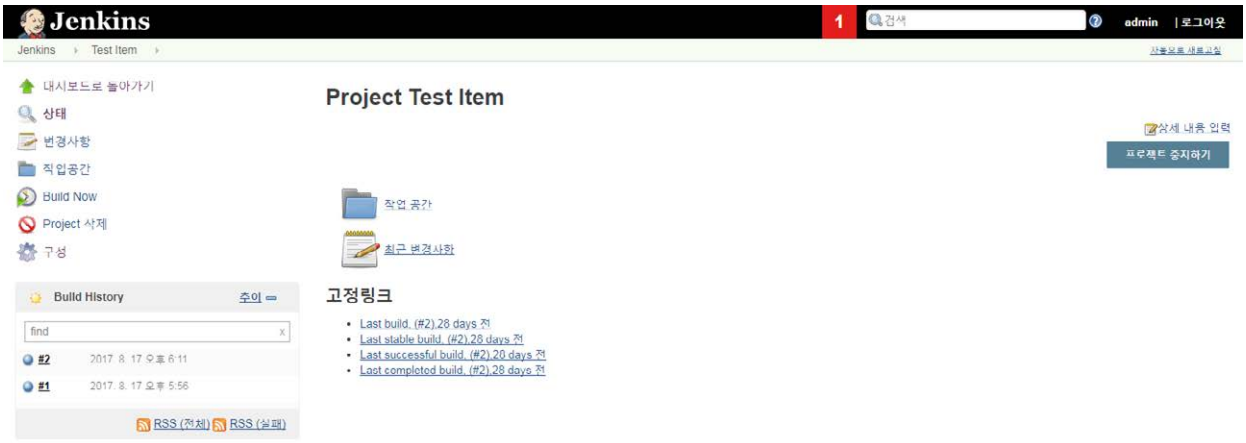
"Kind"에서 "Username with password"를 선택하고 앞서 설정한 SVN에서 설정한 계정 정보(Username, Password)를 입력한다. ID는 Jenkins에서 계정을 관리하기 위한 ID이며 중복으로 사용할 수 없으니 사용자가 알아볼 수 있도록 입력을 하면 된다.
입력을 하였으면 "Add" 버튼을 클릭하여 저장을 한다.



"combo box"에 등록한 계정이 추가되어있을 것이다. 해당 계정을 선택한다.



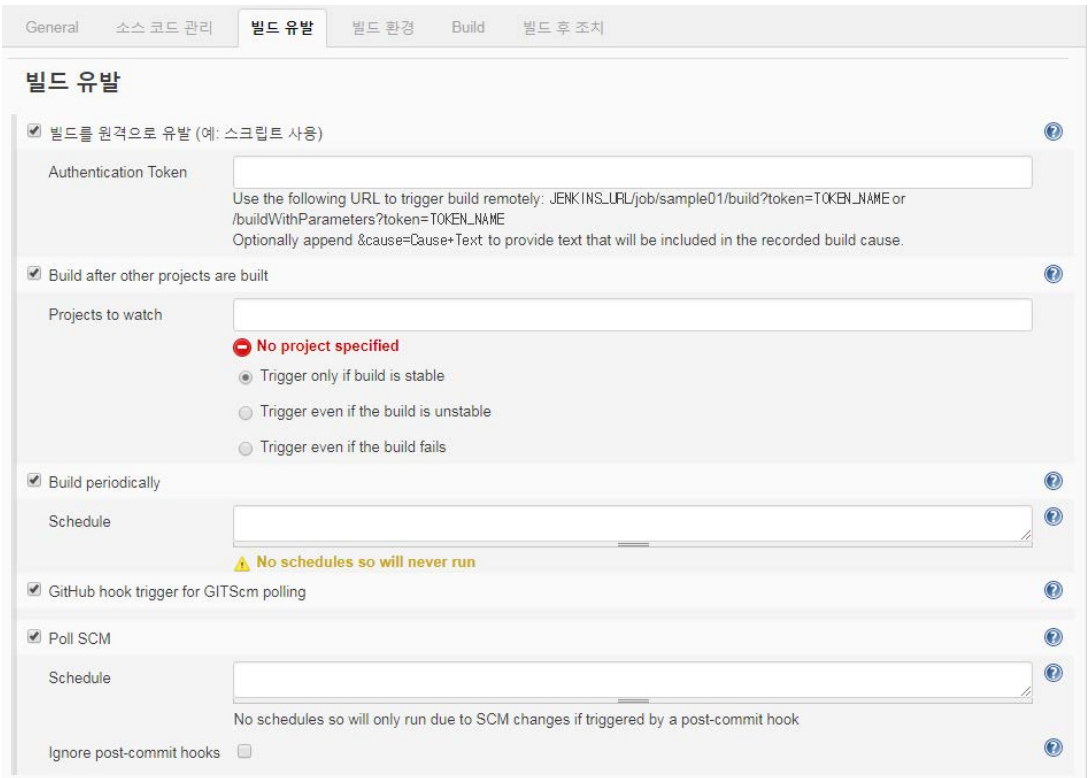
아래에 있는 저장 버튼을 클릭하면, 설정이 저장되고 프로젝트 메인 화면으로 이동을 한다.



2) 빌드 유발

프로젝트 구성에서 빌드 유발 항목을 선택한다.

빌드 유발 항목은 기본적으로 5가지를 선택하여 사용할 수 있으며, 중복으로 선택이 가능하다.



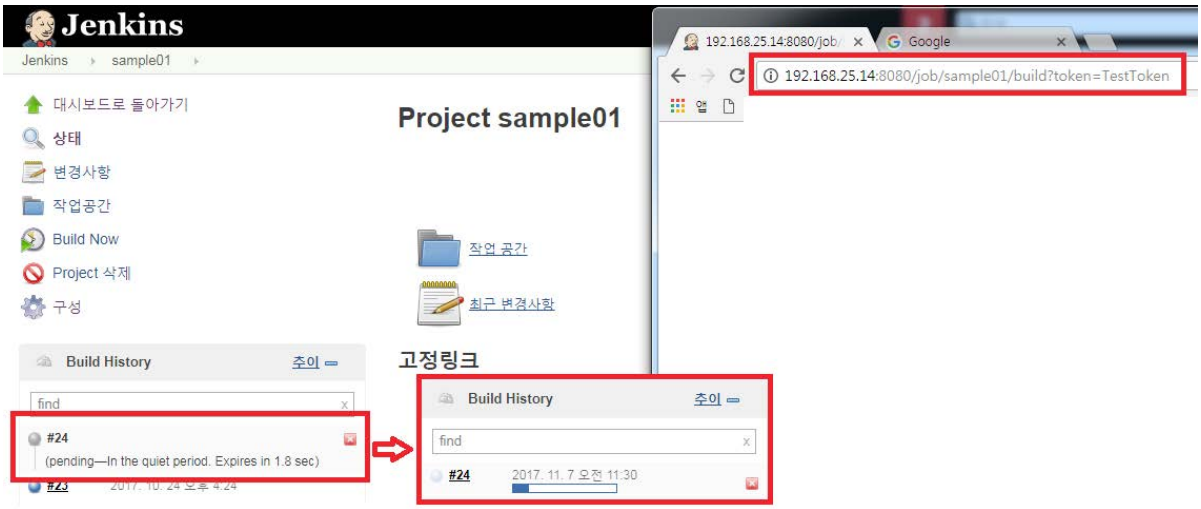
가) 빌드를 원격으로 유발(예: 스크립트 사용)

빌드를 원격으로 유발(예: 스크립트 사용)을 선택하면 Authentication Token 입력 항목이 보인다. 해당 항목을 입력하면 해당 token을 이용하여 원격으로 빌드를 할 수 있다.

입력한 토큰으로 원격에서 아래와 같이 빌드를 실행 할 수 있다.



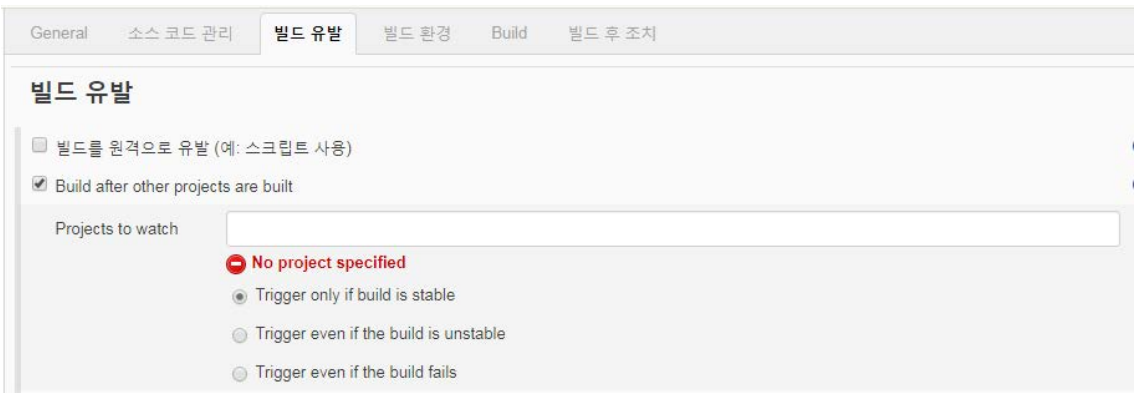
http://JENKINS_URL/job/sample01/build?token=TOKEN_NAME



나) Build after other projects are built

다른 프로젝트를 빌드한 후 이어서 현재 프로젝트를 빌드하는 설정이다.

Project to watch에는 다른 프로젝트의 Job 이름을 입력하면 된다.



해당 설정은 3가지 옵션을 제공한다.

- Trigger only if build is stable : 다른 프로젝트 빌드 결과가 정상인 경우 빌드를 진행
- Trigger even if the build is unstable : 빌드 결과가 정상이 아닌 경우에도 빌드를 진행
- Trigger even if the build fails : 빌드 경과가 실패가 났을때도 해당 빌드를 진행

여기서 unstable은 빌드는 성공했지만 테스트 결과가 실패하여 현재 시스템에 문제가 있는 상황을 말하며 fails 상태는 빌드가 완전히 실패했을 경우를 의미한다.

다) Build periodically

http://khie74.tistory.com/1169521477

빌드를 배치 프로세스 처럼 설정해 놓은 시간에 자동으로 빌드를 진행하도록 하는 설정이다.

시간 설정 값은 crontab와 비슷하게 설정을 하면 된다. 아래와 같이 5개의 필드를 공백으로 구분하여 작성하면 된다.

분(minute) 시간(hour) 일(day of month) 월(month) 주(week)

- 분(minute) : 0부터 59까지의 숫자 입력
- 시간(hour) : 0부터 23까지의 숫자 입력
- 일(day of month) : 1부터 31까지의 숫자 입력
- 월(month) : 1부터 12까지의 숫자 입력
- 주(week) : 0부터 7까지의 숫자를 입력하며 0과 7은 일요일을 의미

한 필드에서 여러 개의 값을 지정하고 싶은 경우 아래와 같은 오퍼레이터를 사용하면 된다.

- * : 모든 가능한 값
- M-N : 범위를 지정
- M-N/X 또는 */X : X라는 간격을 지정
- A,B,C, .. ,Z : 여러 개의 값을 열거

주기적으로 실행되는 Job이 여러 개가 있을 경우 Jenkins는 'H' 키워드를 사용하길 권장하고 있다. 여러 개의 Job을 같은 빌드 스케쥴 시간을 설정해 놓는다면 같은 시간에 모든 Job의 빌드를 실행하게 되어 Jenkins 서버에 굉장히 많은 부하가 발생하게 된다.

그래서 설정 값을 'H * * * *'로 등록하고 사용하게 되면 Jenkins가 시간을 임의로 지정하여 부하가 적은 시간에 빌드를 진행 시킨다. H도 여러 개의 값을 지정할 수 있다.

아래와 같이 시간 설정을 할 수 있다.

- 0 * * * * : 매시각 정각에 빌드
- H * * * * : 1시간 간격으로 빌드
- H H * * * : 하루 간격으로 빌드

- H/30 * * * * : 30분 간격으로 빌드
- H(0-29)/10 * * * * : 매 시간 전반 30분에 10분 간격으로 빌드
- H 9-16/2 * * 1-5 : 주말을 제외한 주중에 9시부터 16시 사이에 2시간 간격으로 빌드
- H H 1,15 1-11 * : 12월 달은 제외하고 매달 1일과 15일에 하루 한번 빌드

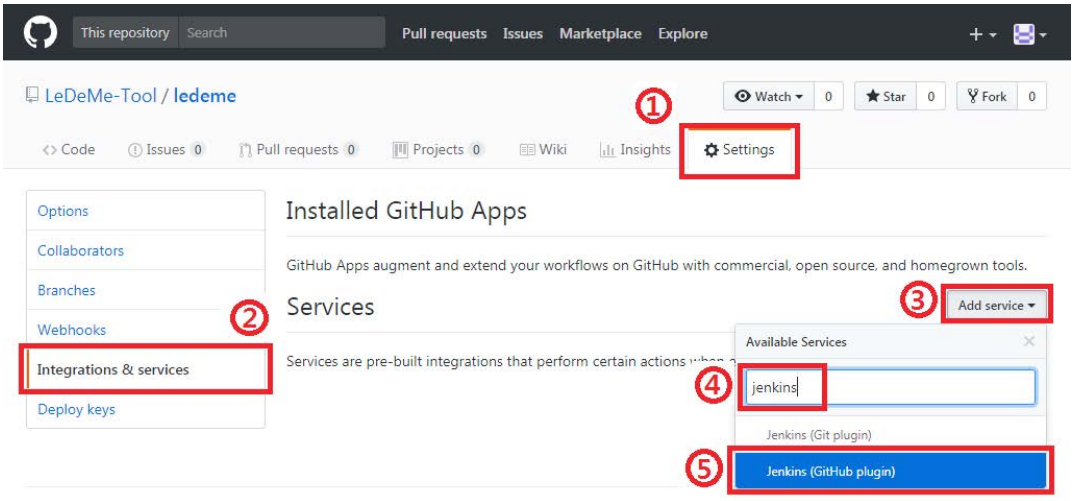
라) GitHub hook trigger for GITScm polling

Github 저장소와 Jenkins 프로젝트 연동을 통해 빌드 환경을 구성 한 후, Github Webhook을 이용하여 Github 저장소에 push 된 것이 있다면 그 때 Jenkins가 polling하여 빌드를 실행한다.

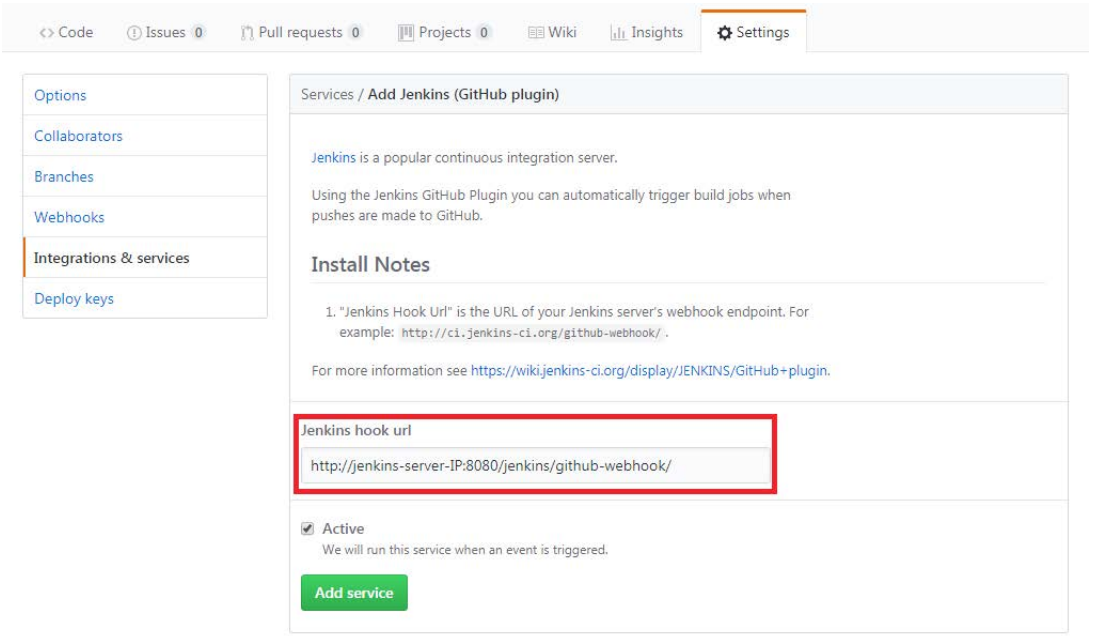
당연한 이야기 이겠지만 GitHub hook trigger for GITScm polling을 사용하려면 Github 프로젝트가 있어야 허며 본 가이드에서는 Github 프로젝트를 사용하고 있다는 전제하에 설명을 할 것이다. 그러므로 Github 프로젝트와 저장소를 만드는 방법에 대해서는 기술하지 않는다.

Github 프로젝트에 Jenkins 서비스를 추가 하여야 한다.

Github 프로젝트에 로그인을 하고 Setting을 클릭한다. 왼쪽 메뉴의 Integrations&services 를 클릭하면 우측에 서비스를 추가할 수 있는 버튼이 나온다.
Add service 버튼을 클릭하고 Jenkins 로 검색을 하여 Jenkins(Github plugin)이 나오면 클릭한다.

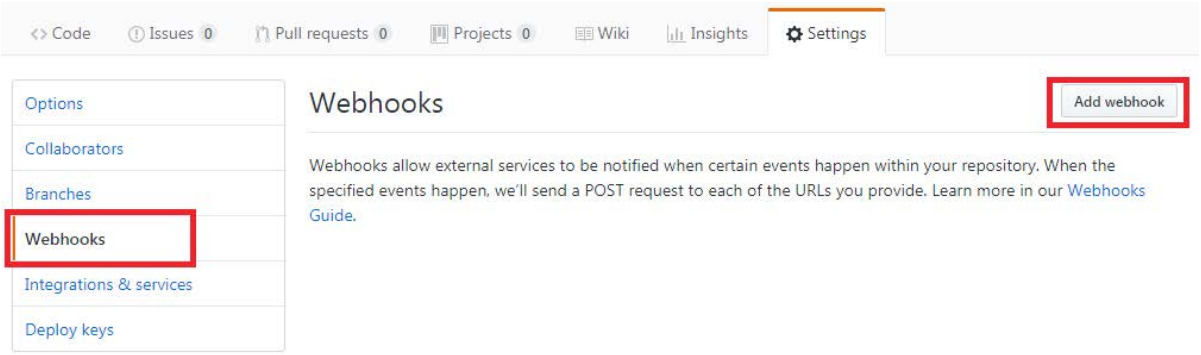


아래와 같은 화면이 나오면 Jenkins hook url 입력 항목에 아래의 예와 같이 입력을 하고 Add service 버튼을 클릭한다.

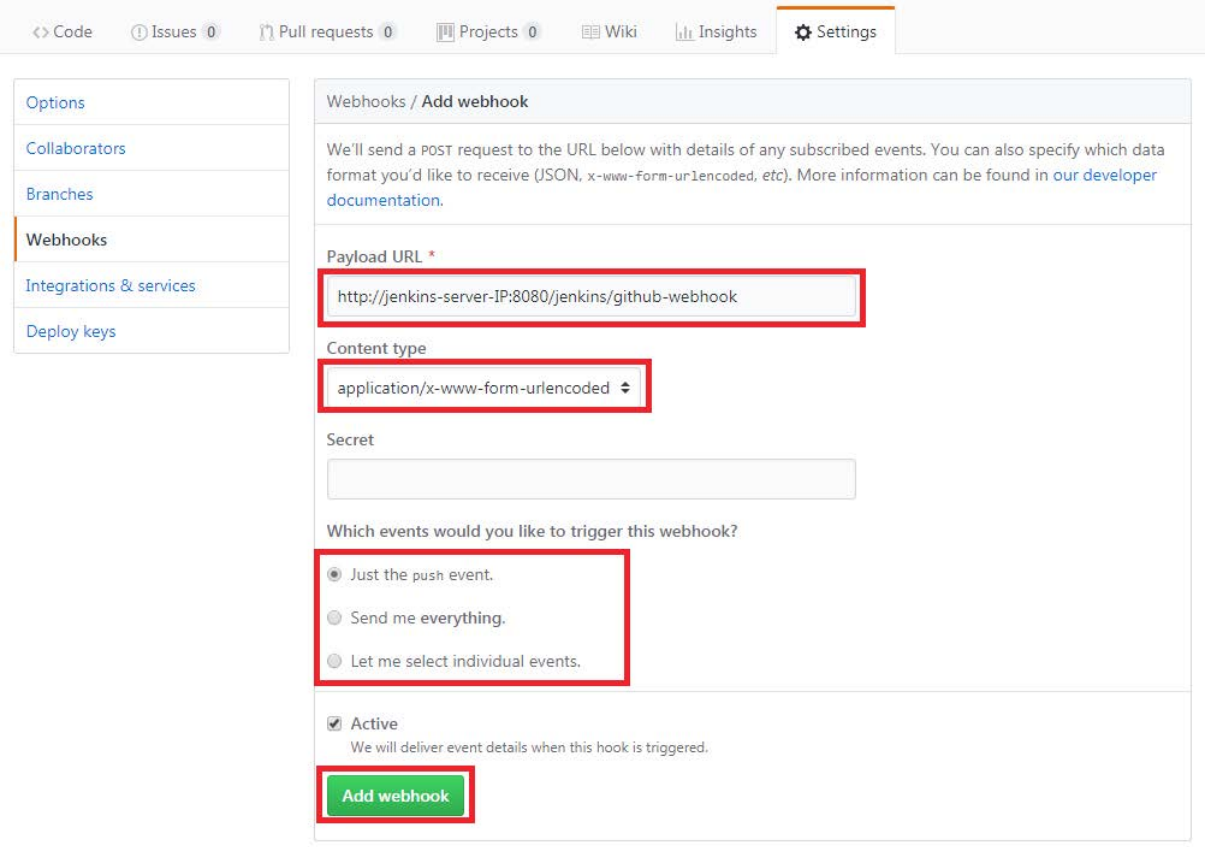


Jenkins hook url 은 Jenkins server가 외부에서 IP 또는 도메인으로 접속이 가능하여야 한다.

다음으로 Github 프로젝트의 Setting의 좌측 메뉴에서 Webhooks 메뉴를 클릭한다.
우측에 Add webhook 버튼을 클릭한다.

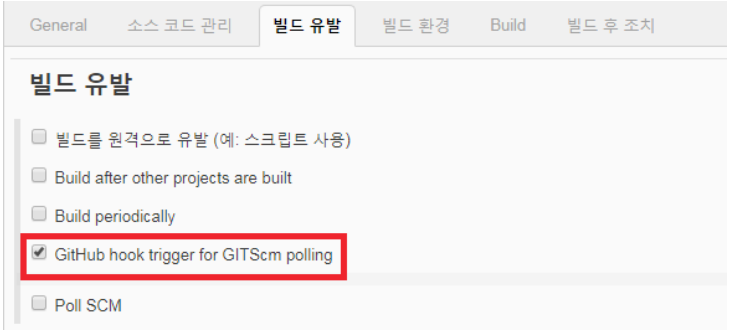


Payload URL에 앞서 입력하였던 URL을 입력하고 원하는 Content type과 trigger를 선택하고 Add webhook 버튼을 클릭한다.



Github 프로젝트에서 설정이 끝났다면 Jenkins 프로젝트 구성에서 빌드 유발을 선택하고 하위 목록 중 'GitHub hook trigger for GitScm polling' 을 체크하면 된다.

소스 코드 관리 탭에서 연동한 Github 저장소에서 push에 의한 hook 이벤트가 발생할 경우 저장소를 polling해서 빌드를 실행할 것이다.



라) Poll SCM

형상관리 서버에 주기적으로 감시하여 변경된 사항이 존재할 때 빌드를 수행하는 설정이다.

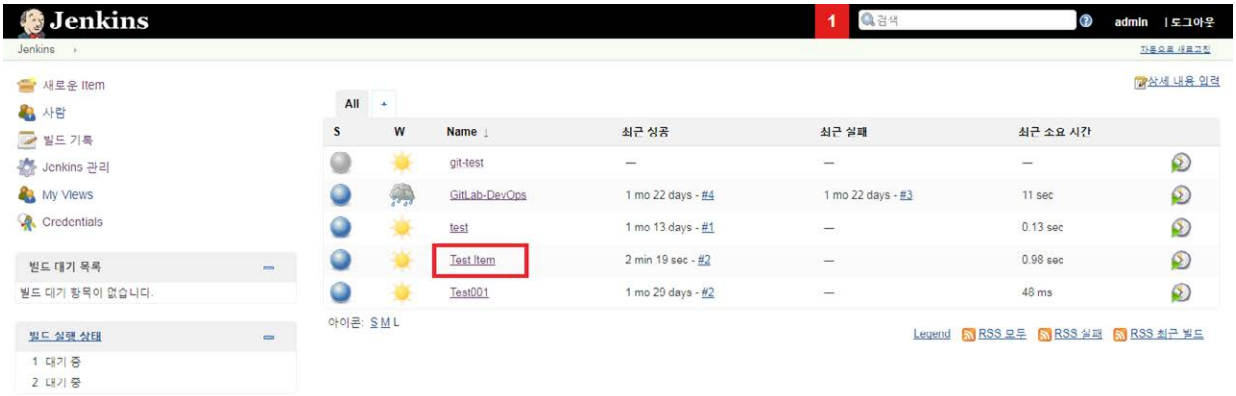
주기적으로 감시를 하기 위해서 설정하는 시간 값을 등록하게 되며, 해당 시간에 형상관리 서버를 확인하여 변경된 사항이 있다면 빌드를 수행한다.

시간 설정하는 방법은 'Build periodically' 에서 설명을 참고하여 설정하면 된다.

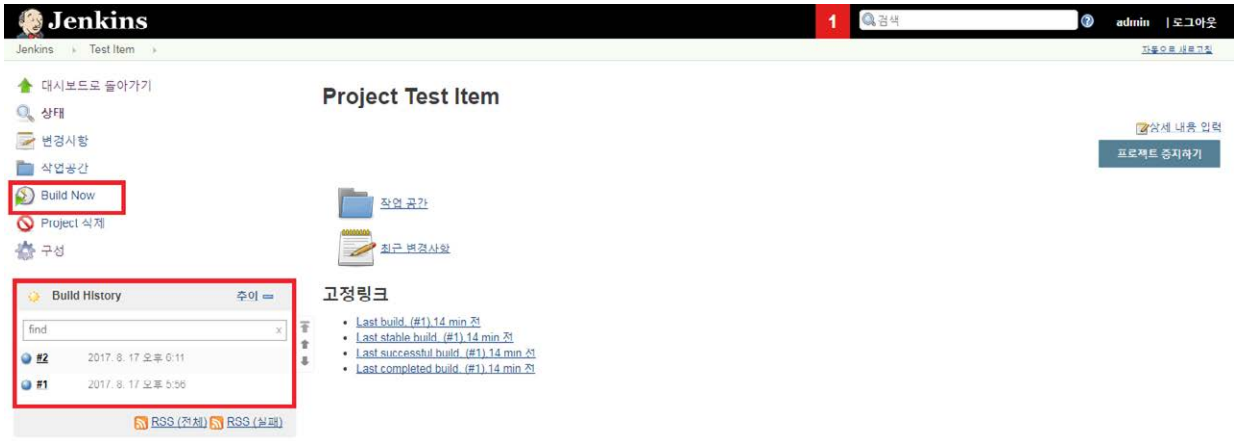
12.7 Jenkins에서 빌드 실행

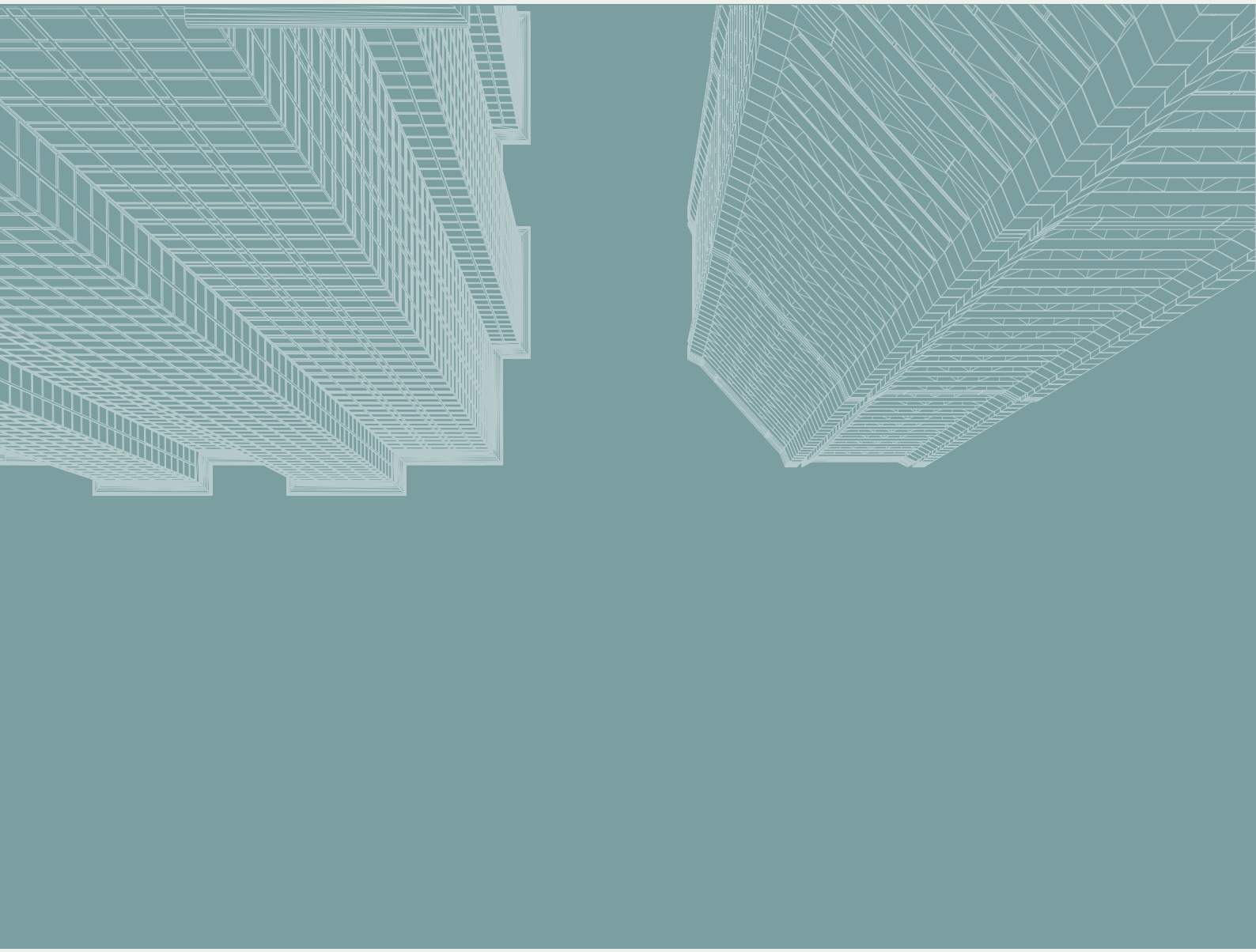
Jenkins에서 프로젝트 빌드 실행을 할 수 있다.

Jenkins의 프로젝트 목록에서 빌드를 원하는 프로젝트를 선택한다.



프로젝트를 선택하면 우측에 “Build Now”를 선택하면 빌드를 실행하며, “Build History”에서 빌드 내역과 성공, 실패를 확인할 수 있다.





CHAPTER 13

Toolchain Tutorials



13.1 Tutorial 1 : Jenkins, SVN, Maven, Nexus 142
13.2 Tutorial 2 : Jenkins, SVN, Ant, Selenium, Junit 161
13.3 Tutorial 3 : Jenkins(Tutorial 1), Findbugs 198
13.4 Tutorial 4 : Jenkins, Fitnesse 205

CHAPTER 13 Toolchain Tutorials

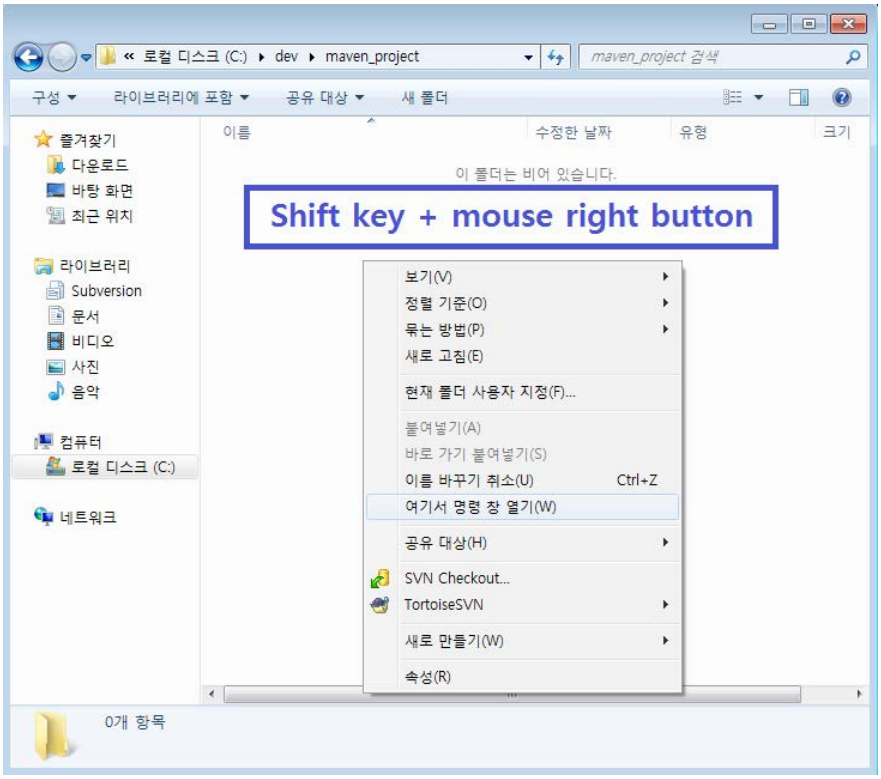
앞서 설치한 도구를 활용하여 Toolchain 실습을 할 것이다.
실습은 JAVA로 계발된 Web 프로젝트에서 간단하게 개발 소스형상관리, 빌드, 테스트, 배포의 과정을 어떻게 Toolchain으로 사용할 수 있는지 실습을 통하여 알아 보도록 하겠다.

13.1 Tutorial 1 : Jenkins, SVN, Maven, Nexus

본 가이드에서 제안하는 Toolchain의 기본적인 구성을 위하여 필요한 도구를 사용하여 구성하는 방법을 설명할 것이다. Maven 프로젝트를 구성하고 해당 소스를 SVN으로 형상관리를 할 수 있도록 하며, Nexus로 배포할 수 있도록 Jenkins로 CI를 구성하도록 한다.

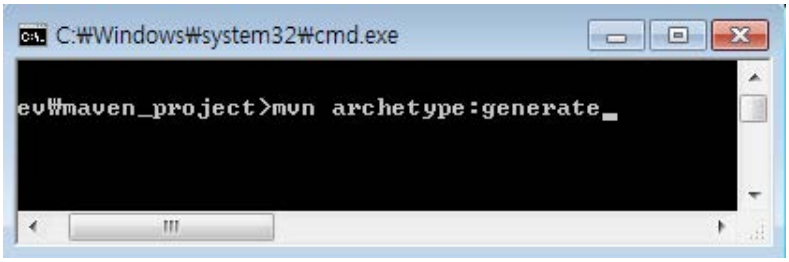
1) Maven 프로젝트 생성

앞서 설치한 Maven으로 기존의 Maven 프로젝트 디렉토리의 경로에 새 프로젝트를 생성할 것이다.
새 프로젝트를 생성할 디렉토리로 이동을 한다.
경로는 C:\dev\maven_project 이다. 사용자가 따로 사용하는 경로가 있다면 해당 경로로 이동을 하면 된다.
아래와 같이 이동한 폴더에서 “Shift key + 마우스 우클릭”을 하고, ‘여기서 명령 창 열기’ 를 클릭하여 명령 창을 연다.



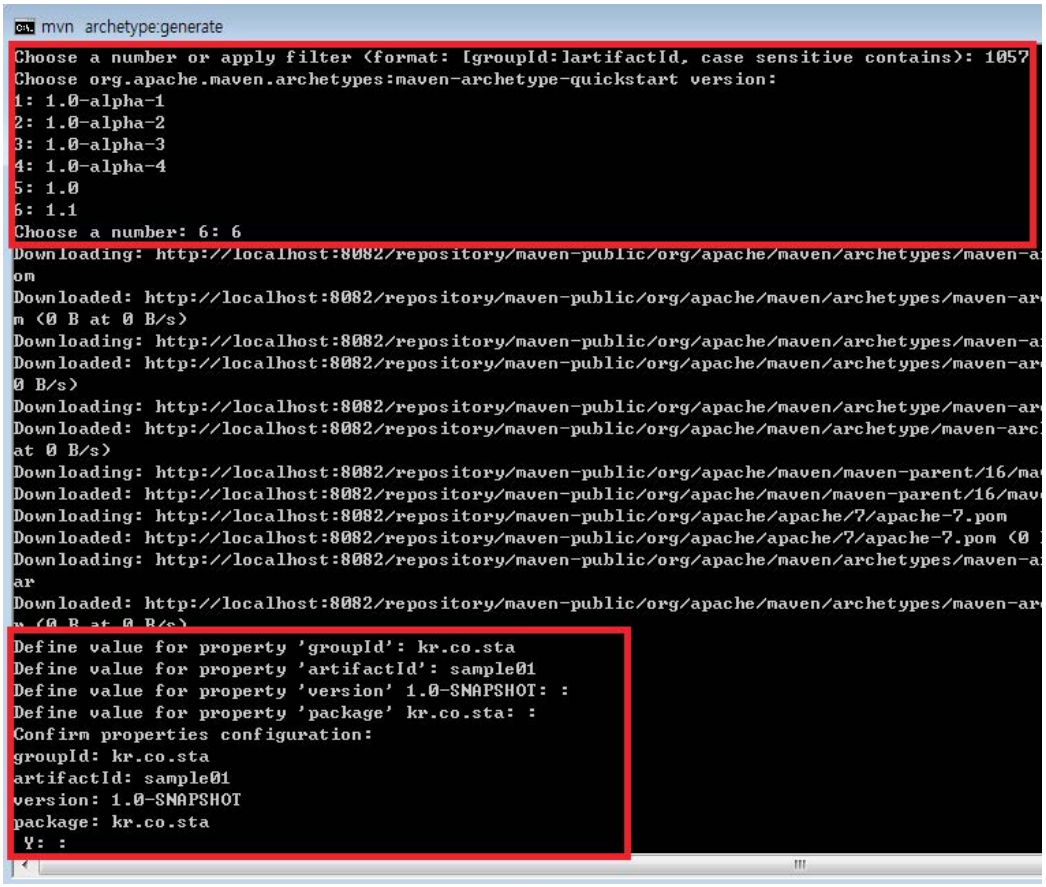
명령 창에서 다음의 명령을 실행한다.

```
mvn archetype:generate
```

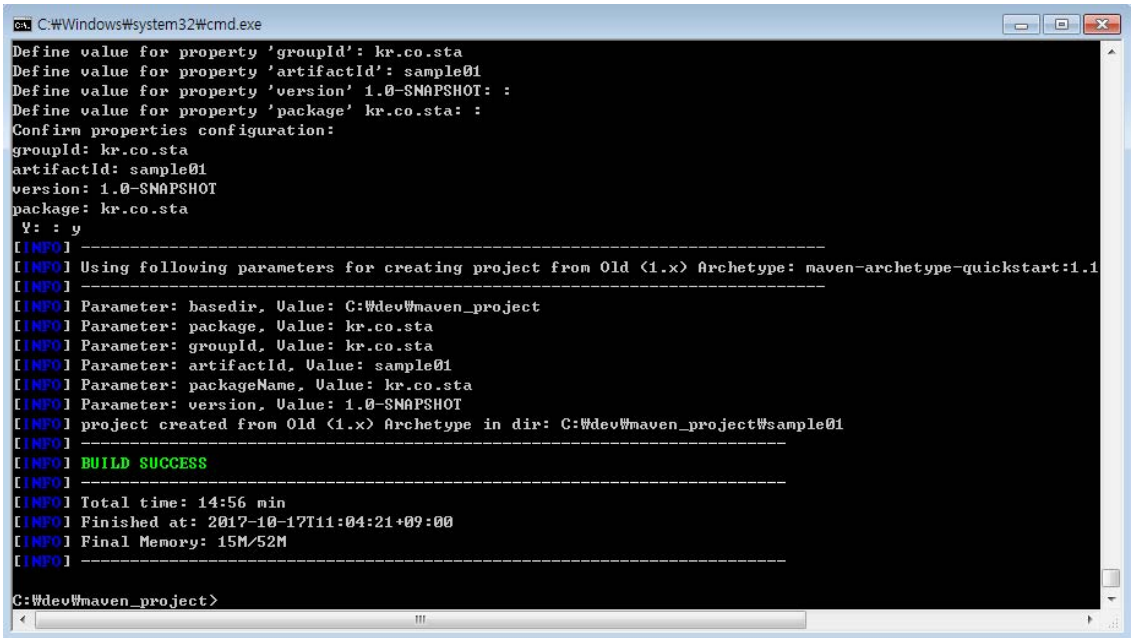


명령을 실행하면 Maven archetype version, groupId, artifactId 등을 묻는다.
각 항목을 입력하고 엔터를 하면, 입력한 항목에 대한 확인을 하고 정상적으로 입력을 하였다면 'Y' 를 입력하면 새 프로젝트가 생성된다.
예제에서 입력한 입력한 값은 다음과 같다.

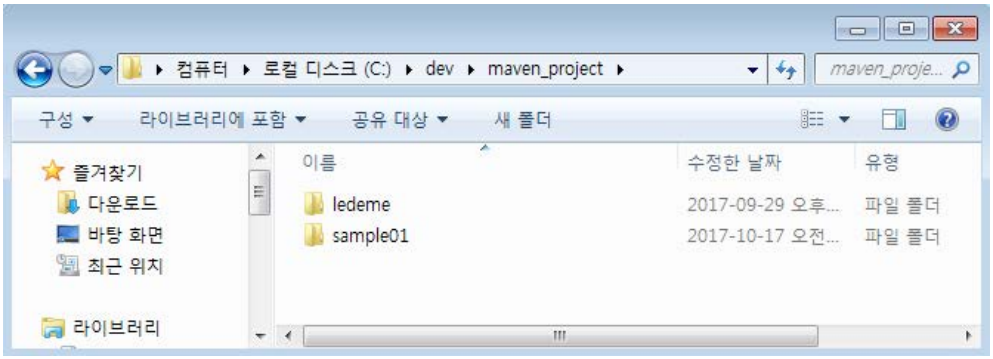
- maven archetype version : 입력 없이 엔터(기본값은 6)
- groupId : kr.co.sta
- artifactId : sample01
- version : 입력 없이 엔터(기본값 1.0-SNAPSHOT)
- package : 입력 없이 엔터(기본값 groupId 입력값)



프로젝트가 정상적으로 생성 되었다면 아래와 같은 화면이 보인다.



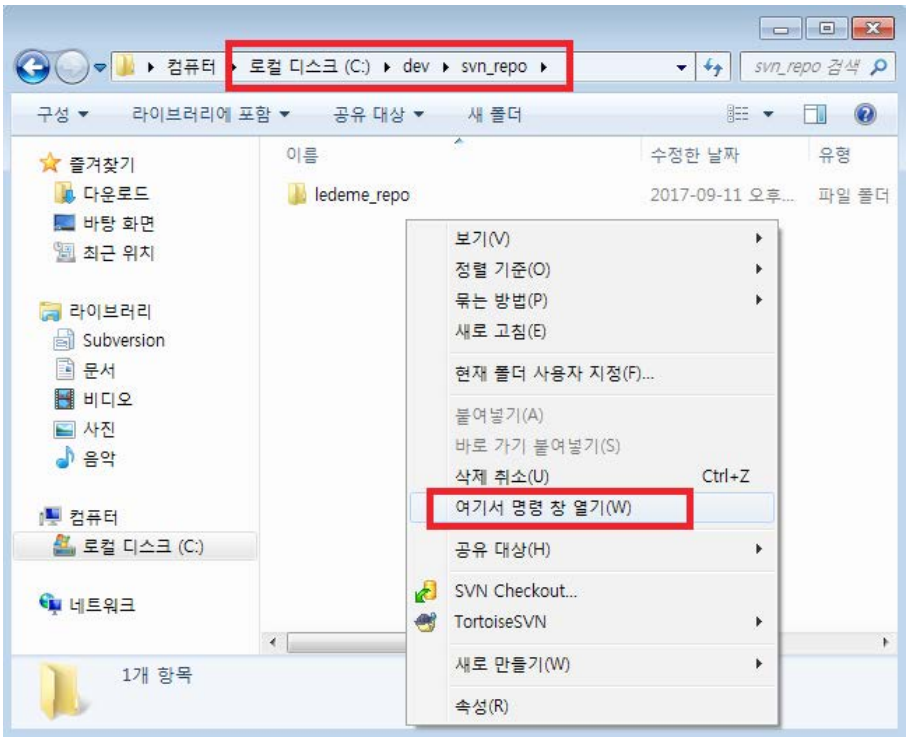
Maven 프로젝트 디렉토리를 확인 하면 방금 생성한 sample01 프로젝트 디렉토리가 생성되어있는 것을 확인 할 수 있다.



2) Maven 프로젝트 형상관리(SVN)

앞에서 생성한 Maven 프로젝트를 SVN으로 형상관리 할 수 있도록 할 것이다.
SVN 기본 저장소 경로에서 새 SVN 저장소를 생성하여야 외부에서 해당 저장소로 접근이 가능하다.
외부에서도 접근이 가능하도록 하기 위하여 기본저장소 경로에 새 SVN 저장소를 생성하고, SVN 저장소에 Maven 프로젝트를 복사하여 사용할 것이다.
본 가이드의 SVN 설치에서 기본 저장소 위치를 c:\dev\svn_repo 으로 설정하였다.

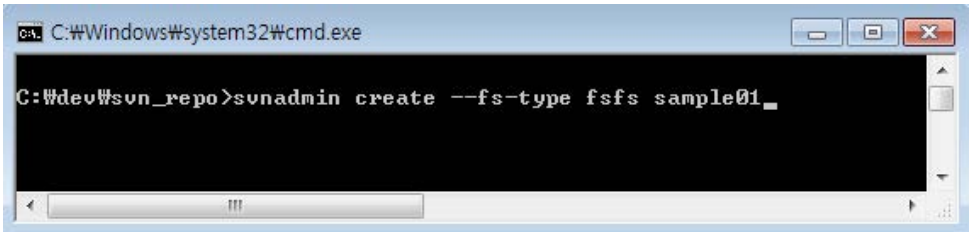
Maven 프로젝트 생성할 때와 동일하게 SVN 기본 저장소 폴더로 이동을 하고, “Shift key + 마우스 우클릭” 을 하고, ‘여기서 명령 창 열기’ 를 클릭하여 명령 창을 연다



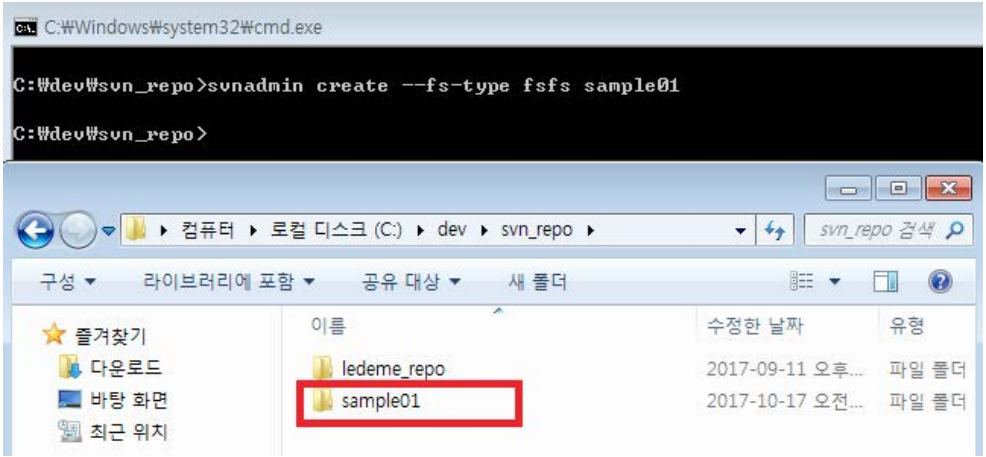
명령 창에서 다음의 명령을 실행한다.

```
svnadmin create --fs-type fsfs [저장소명]
```

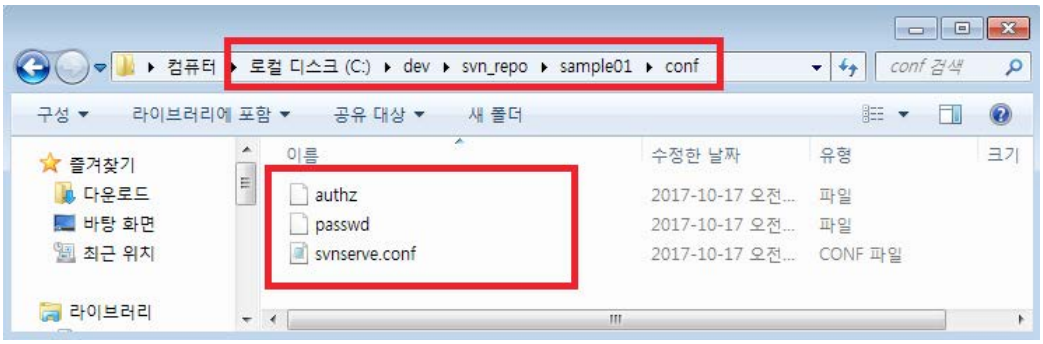
여기서 저장소명은 Maven 프로젝트 artifactId 와 동일한 sample01 으로 하겠다.



명령을 실행하면 아래와 같이 새 SVN 저장소가 생성된 것을 확인할 수 있다.

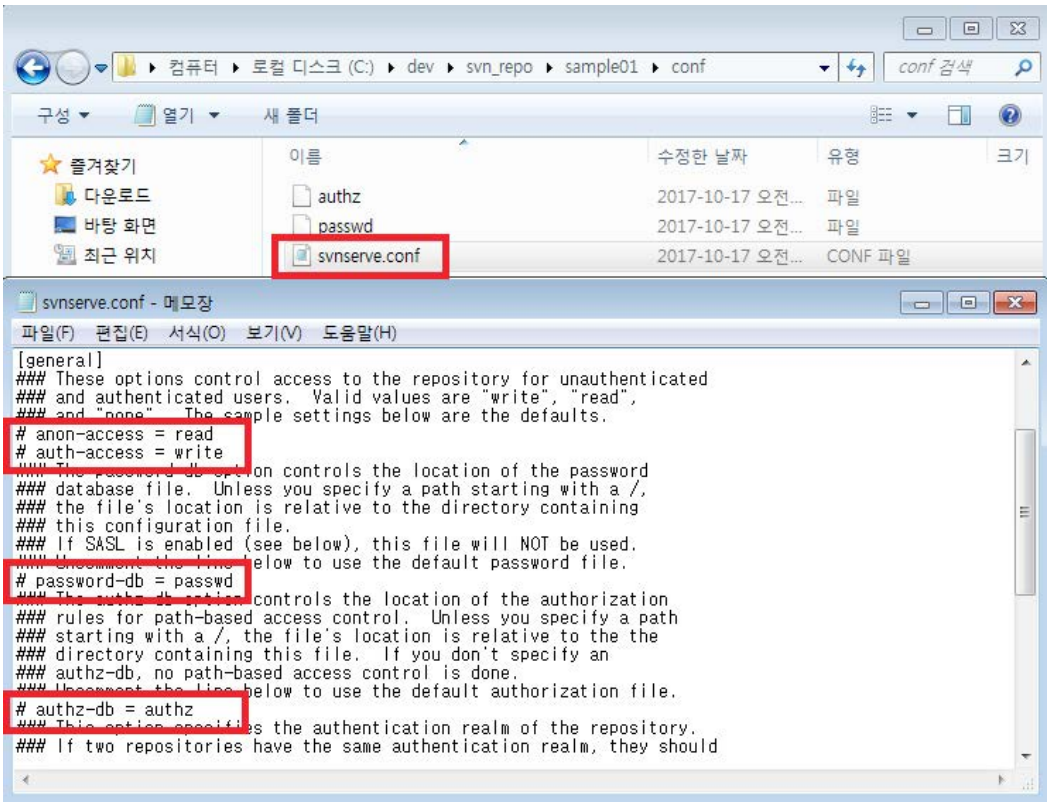


새 SVN 저장소를 생성하였다면 저장소에 접근할 수 있는 계정 등 설정을 하여야 한다.
설정은 저장소의 conf 경로의 파일을 수정하면 된다.



생성된 저장소로 이동을 하여 "conf/svnserve.conf" 파일에서 아래 주석을 풀어준다.

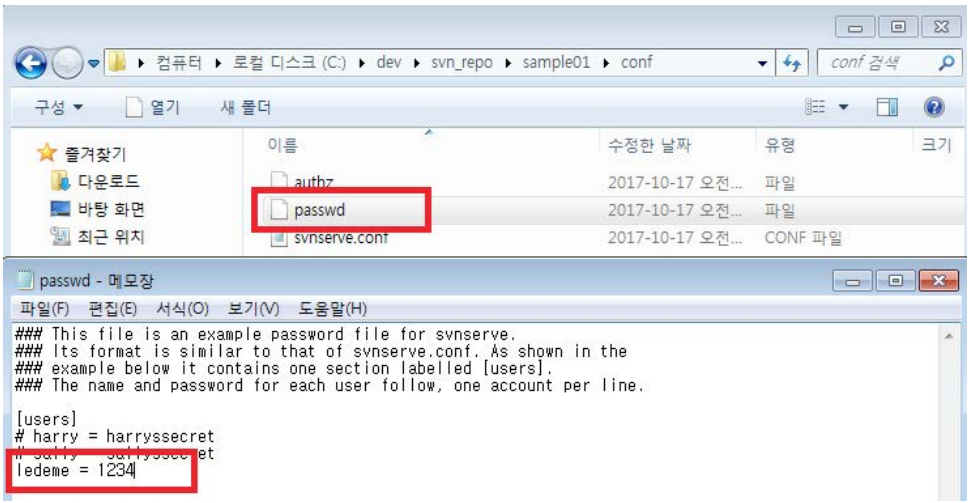
[general]
anon-access = read #(인증 되지 않은 사용자 접근 거부는 read를 none으로 변경)
auth-access = write #(인증된 사용자 쓰기 권한)
password-db = passwd #(사용자에 대한 계정정보)
authz-db = authz #(사용자에 대한 저장소 권한주기. optional)



다음으로 저장소 사용자와 패스워드를 생성한다.

“conf/passwd” 파일에 사용자와 패스워드를 입력한다.

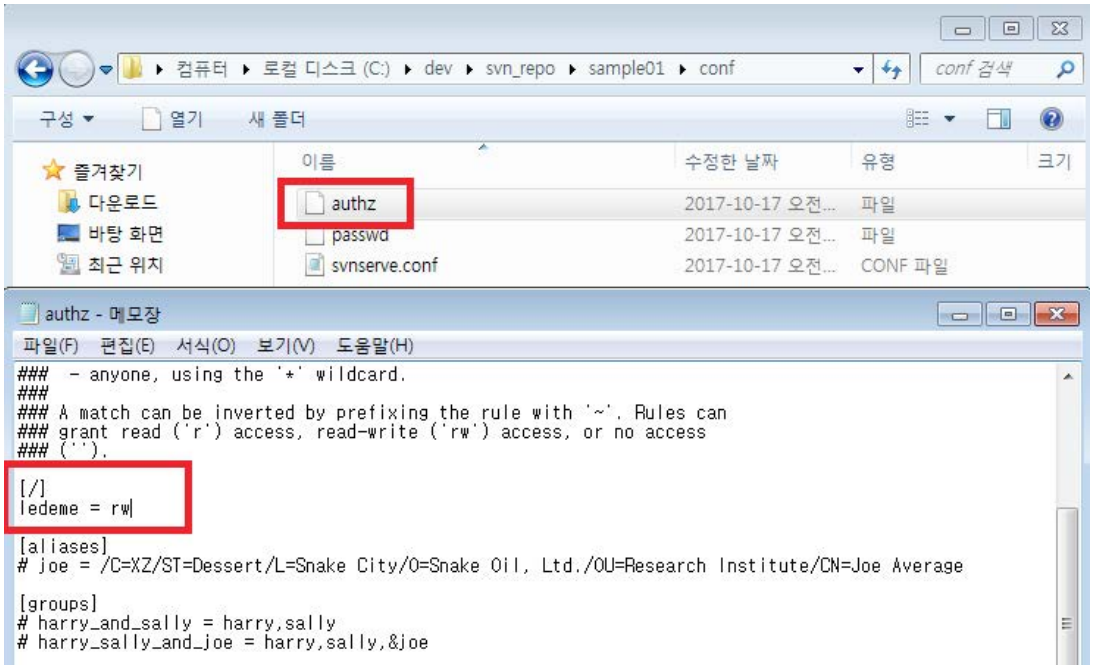
```
ledeme = 1234 #(패스워드는 외부에서 알 수 없도록 설정)
```



다음으로 저장소에 대한 권한 설정을 한다.

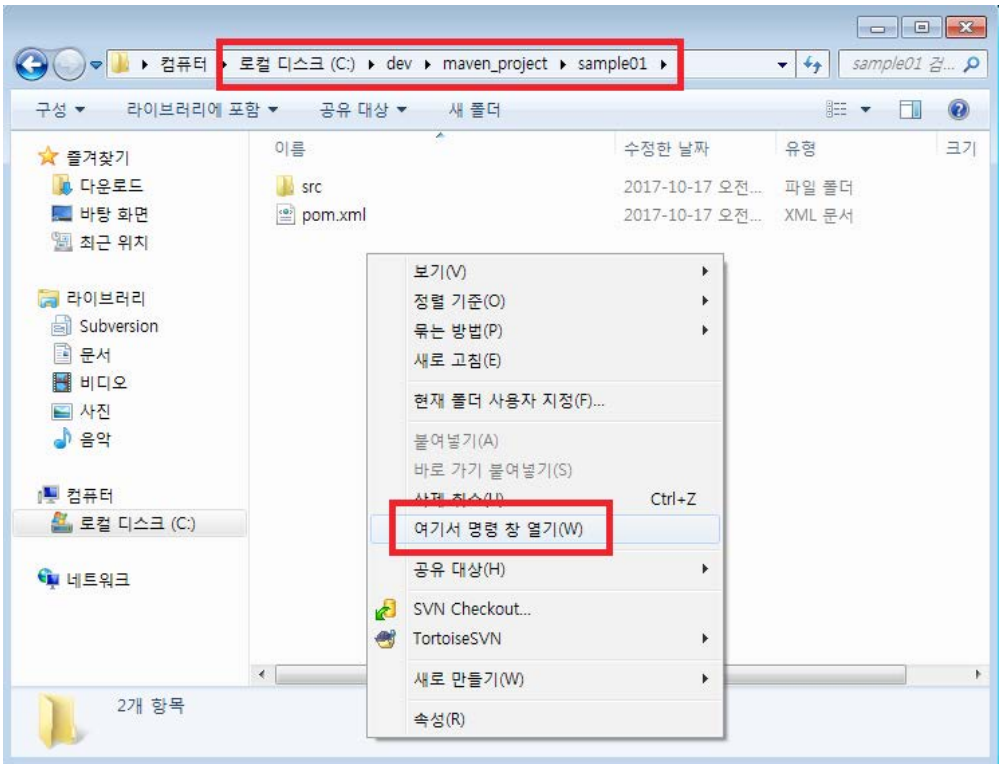
“conf/authz” 파일을 열어서 위에서 추가한 사용자에 대한 권한을 입력한다.

```
[/]
ledeme = rw #(읽기, 쓰기 권한)
```



새 SVN 저장소 설정이 끝났으면, Maven 프로젝트를 SVN 저장소에 연결하고 commit 하여 형상관리 할 수 있도록 하겠다.

앞서 생성한 Maven 프로젝트로 이동을 한다. 경로는 C:₩dev₩maven_project₩smaple01 이다.
이동한 폴더에서 "Shift key + 마우스 우클릭"을 하고, '여기서 명령 창 열기' 를 클릭하여 명령 창을 연다



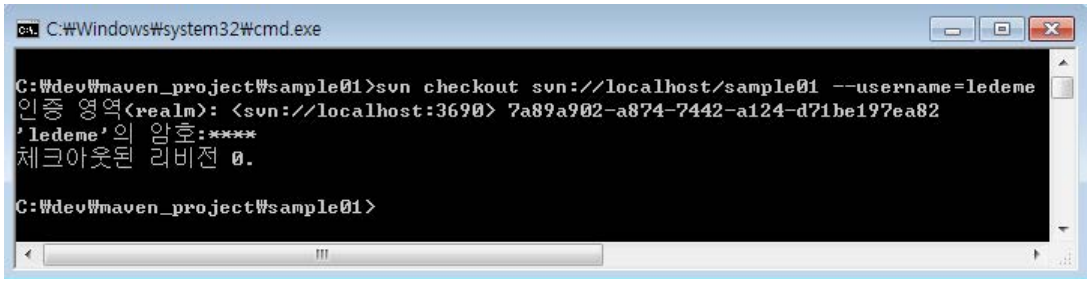
명령 창에서 아래의 명령을 실행한다.

```
svn checkout svn://localhost/sample01 --username=[사용자ID] [원격 저장소명]
```

여기서 사용자 ID는 'ledeme' 이고, 원경 저장소명은 현재 경로 './' 또는 'C:\dev\maven_project\smaple01' 으로 입력하면 된다.

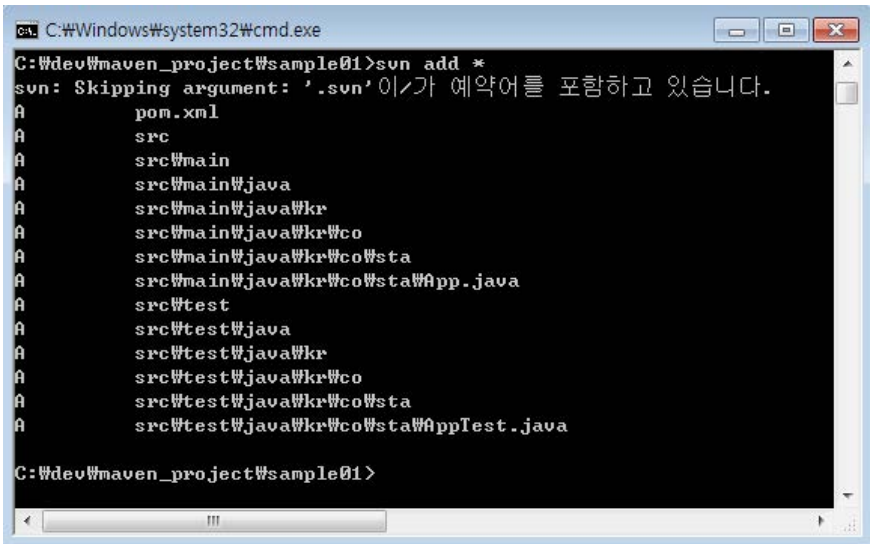
```
svn checkout svn://localhost/sample01 --username=ledeme ./
```

명령을 실행하면 암호를 묻는다. 위에서 설정한 암호를 입력하면 된다.
아래의 화면 처럼 체크아웃이 되면 저장소 연결된 것이다.



이제 SVN 저장소에 파일을 올릴 것이다.
SVN 저장소에 올릴 파일을 아래의 명령으로 추가 한다.

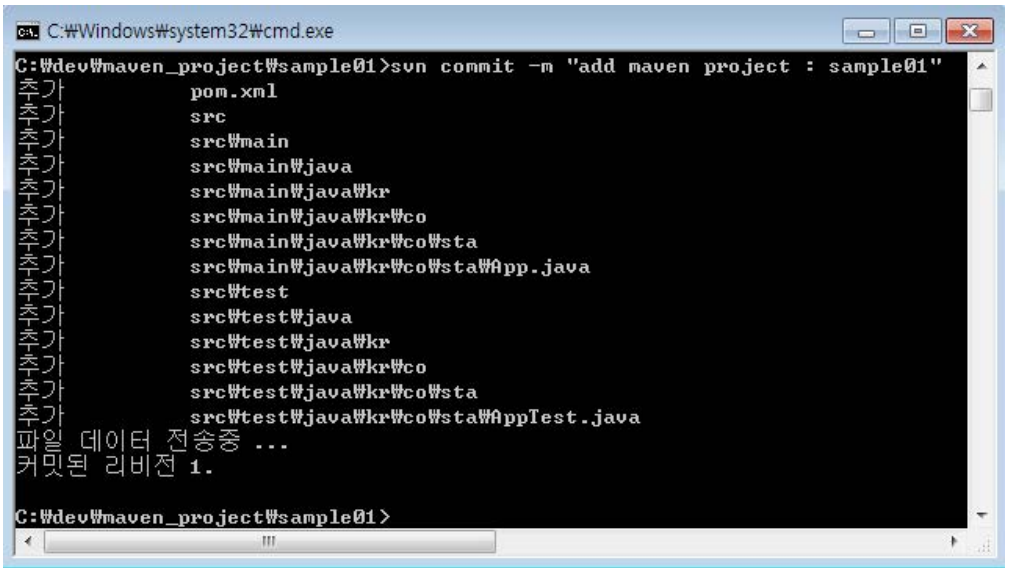
```
svn add *
```



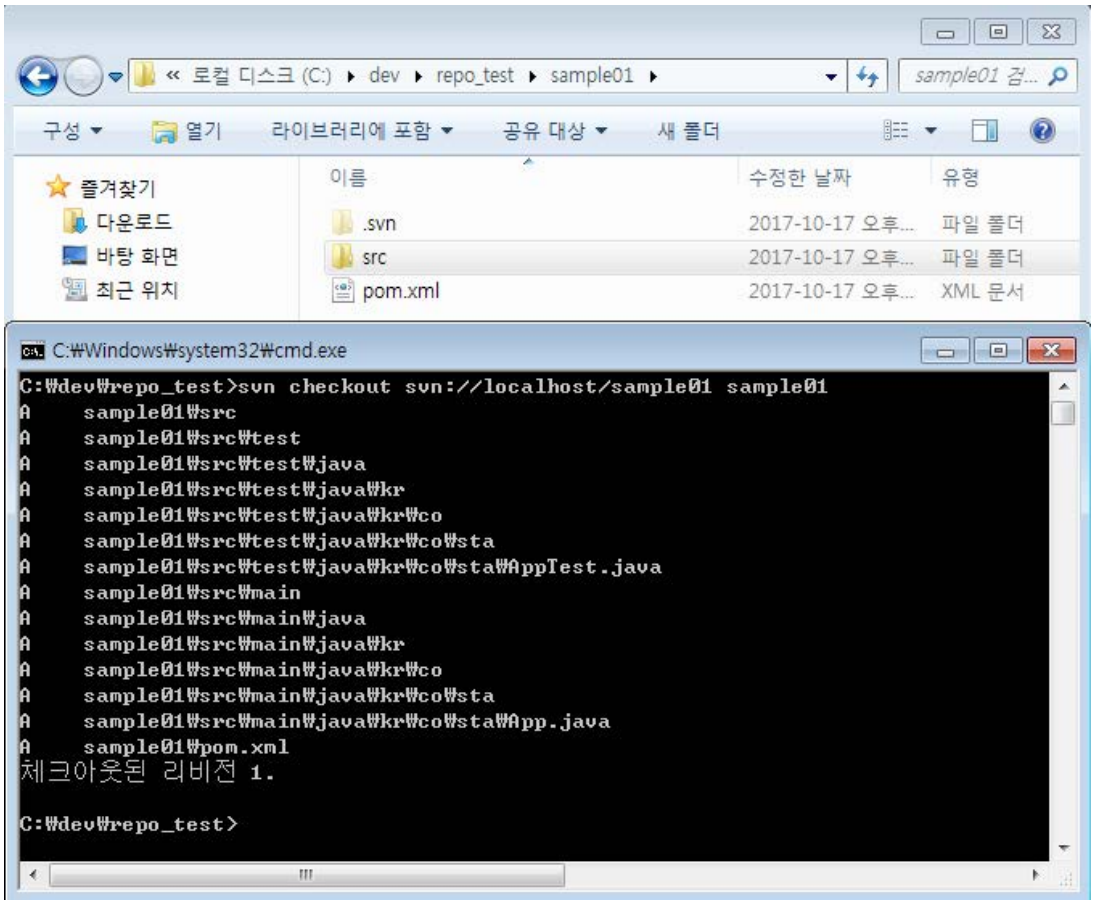
add한 파일을 SVN 저장소에 아래의 명령으로 올린다.

```
svn commit –m “메세지”
```

아래와 같이 파일들이 추가 되고 리비전 1이 되었다면 정상적으로 파일일 저장소에 올라간 것이다.



다른 원격저장소를 생성하여 확인을 해보면 아래와 같이 추가한 파일들이 저장소에 잘 올라간 것을 확인할 수 있다.



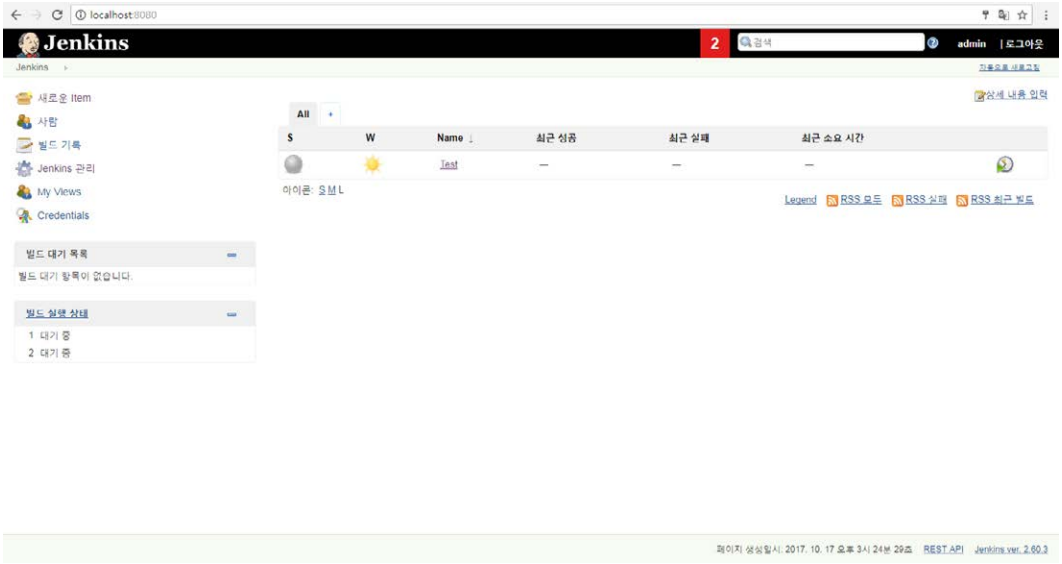
3) Jenkins 프로젝트 생성 및 SVN 연결

Jenkins에서 앞에서 생성한 SVN을 연결하여 Maven 프로젝트를 생성해 보겠다.

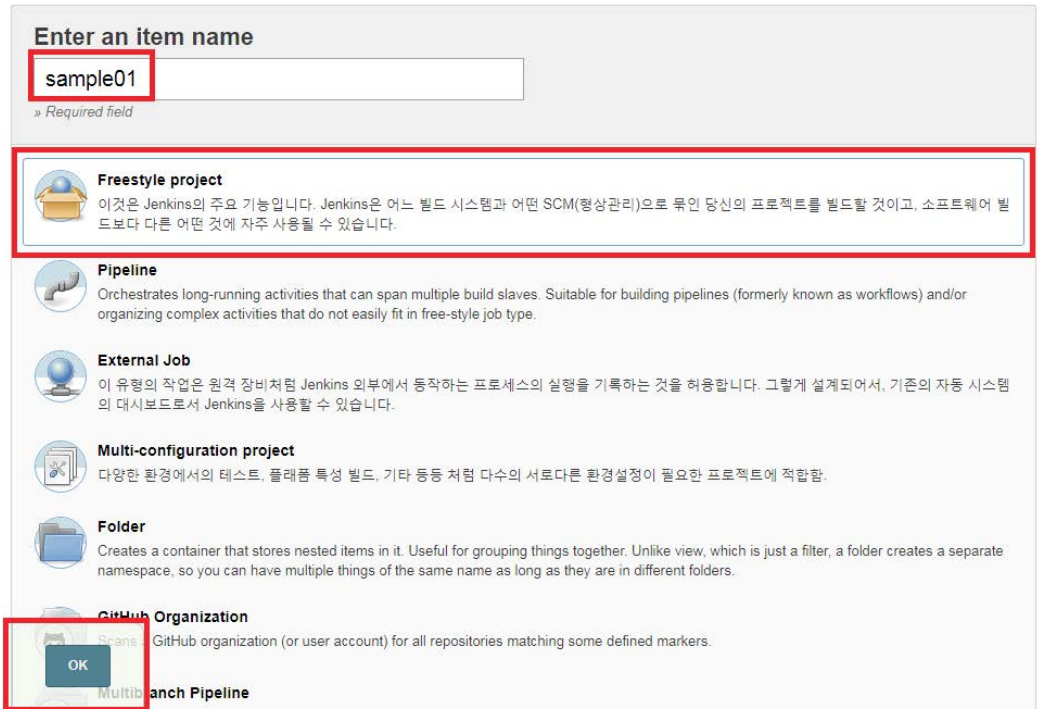
본 가이드에서 설치한 Jenkins로 접속하고 관리자 계정으로 로그인을 한다.

- URL : http://localhost:8080
- 계정 : 설치 시 생성한 관리자 ID와 Password 사용
  (본 가이드에서는 admin/admin 으로 생성 함)

Jenkins에 접속을 하고 로그인을 하면 아래와 같은 화면이 보인다.



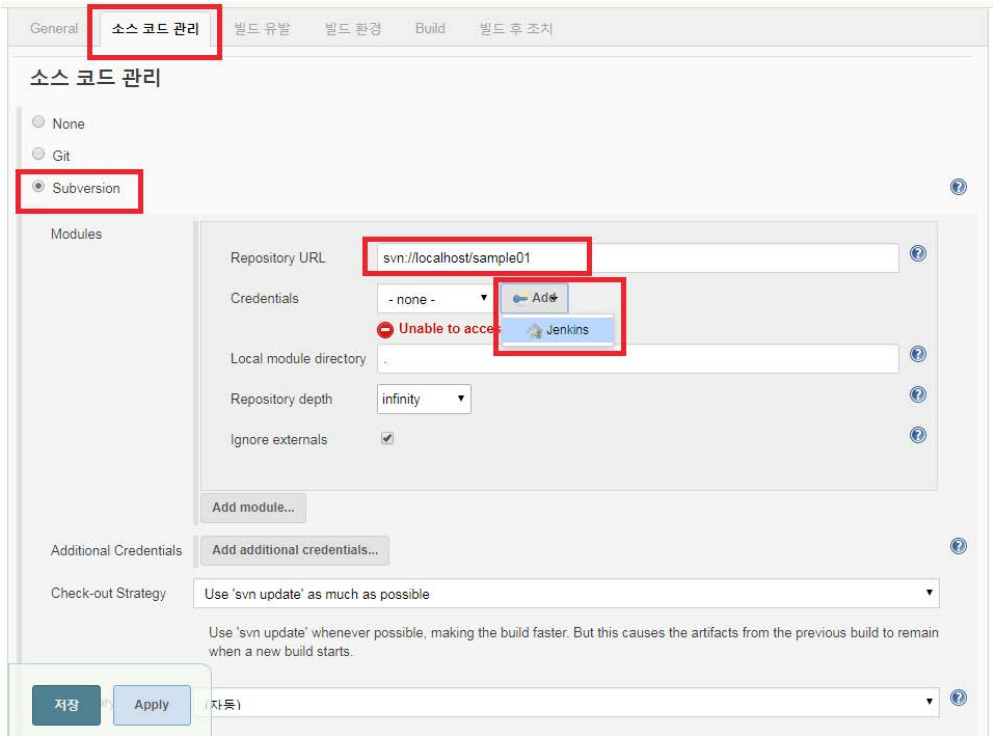
화면의 좌측 메뉴에서 '새로운 Item' 을 클릭하여 새로운 프로젝트를 생성한다.
'새로운 Item'을 클릭하면 아래와 같은 화면이 보이며, Enter an item name(프로젝트 명)은 'sample01' 으로 입력할 것이며, 'Freestyle project' 을 선택하고 하단의 'OK' 버튼을 클릭하면 생성이 된다.



'OK' 버튼을 클릭하면 상세한 설정을 할 수 있는 화면으로 이동을 한다.
앞에서 생성한 SVN 저장소 연결을 하기 위하여, 설정화면 상단의 '소스 코드 관리' 탭을 선택한다.
소스 코드 관리의 항목 중 Subversion 을 선택하면 세부 항목이 아래의 화면처럼 보이며, 각 항목을 입력하면 된다.

항목은 SVN 저장소 정보를 입력하면 되며, 입력은 아래와 같이 하면 된다.

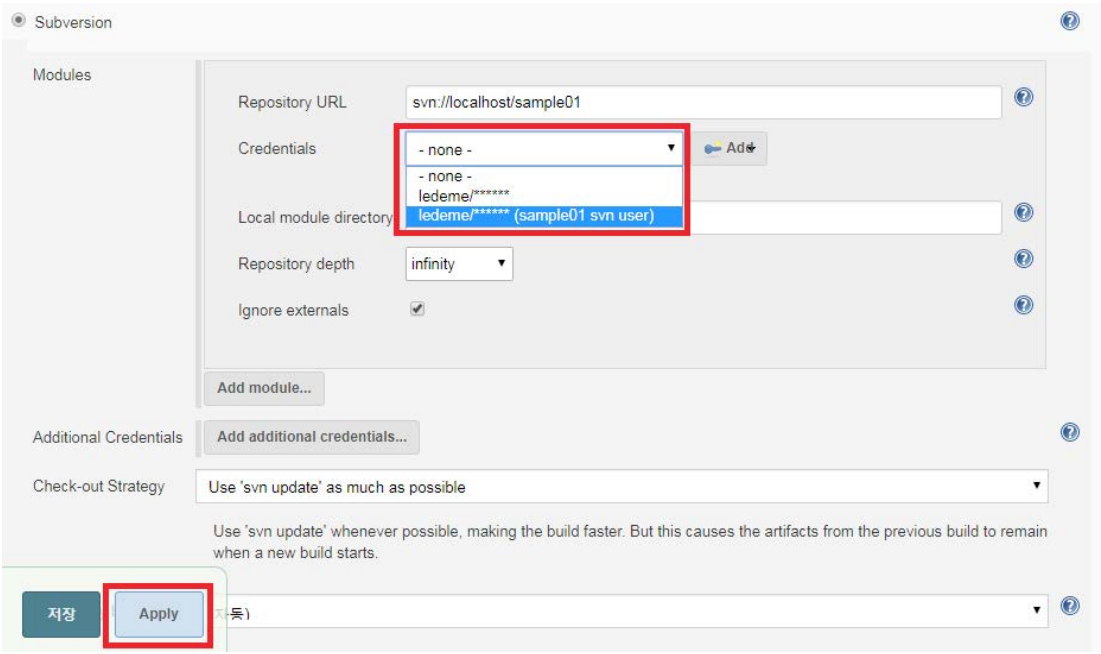
- Repository URL : svn://localhost/sample01
- Credentials : 저장소 계정 정보를 선택하면 되나, 해당 계정이 생성이 되어있지 않다면 'Add' 버튼을 클릭하여 추가한다. 추가하는 방법은 아래에 설명 할 것이다.



계정 추가는 'Add' 버튼을 클릭하면 아래에 'Jenkins' 라는 항목이 나오며 해당 항목을 클릭한다.
계정 추가 화면이 아래와 같이 나오면 연결할 SVN 계정의 정보를 입력하고, 하단의 'Add' 버튼을 클릭하면 계정이 추가 된다.



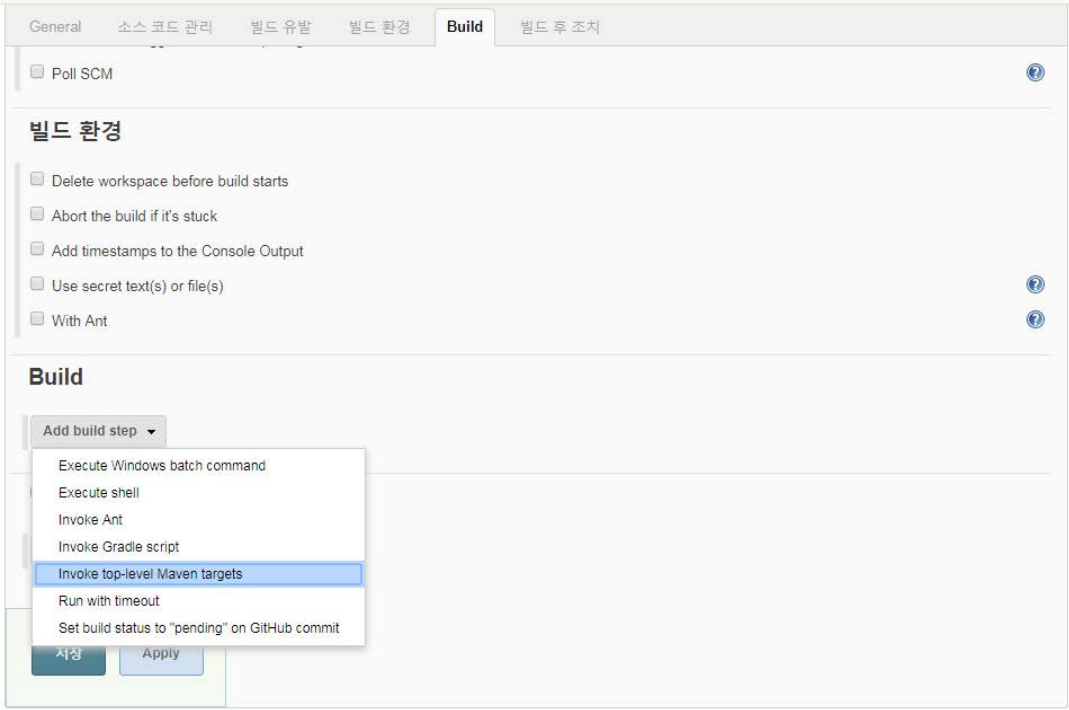
계정이 정상적으로 추가 되었다면 계정 목록에 추가한 계정이 보인다. 추가한 계정을 선택하고 하단의 'Apply' 버튼을 클릭하여 설정을 저장한다.



다음으로 Jenkins 프로젝트에 연결한 SVN이 Maven 프로젝트 이므로, Build 에서 Maven 관련 설정을 해주도록 하겠다.

상단의 Build 탭을 클릭하고, Build 항목에서 'Add build step' 을 클릭하면, 다양한 항목들이 보인다.

여러 항목 중 아래의 화면 처럼 'Invoke top-level Maven targets' 를 선택한다.



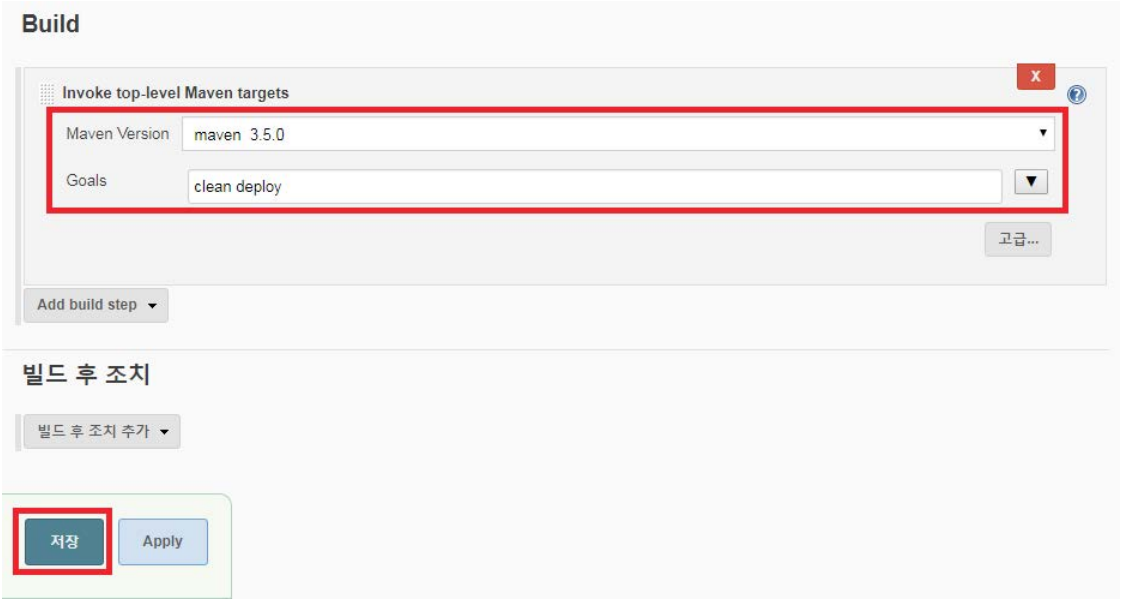
'Invoke top-level Maven targets' 를 선택하면 아래와 같이 Maven 정보를 입력하는 항목이 나온다.

Maven Version.은 기본값으로 하던지, jenkins에서 따로 등록한 버전이 있다면 목록에서 선택을 한다.

Goals는 아래와 같이 입력을 하고, 아래 고급 버튼을 클릭하여 추가 정보를 입력한다.

```
clean deploy
```

deploy는 Maven 빌드 시 배포와 관련된 옵션이다. 다음 단계에서 Nexus와 연결을 하여 배포하기 위하여 미리 입력하여 저장을 해두도록 하자.
입력을 하였다면 하단의 저장 버튼을 클릭한다.

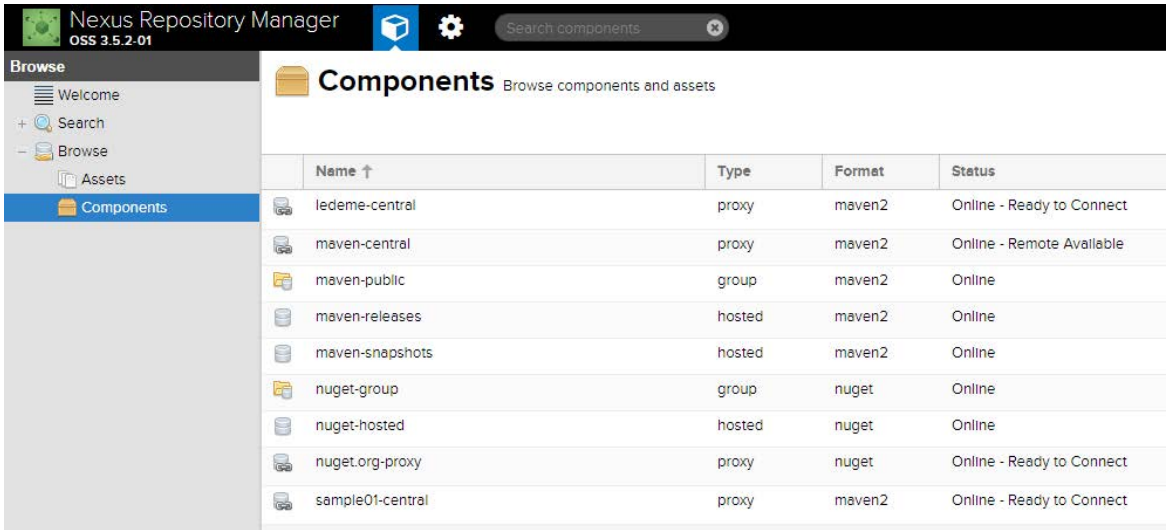


4) Nexus 배포를 위한 Maven 설정

Maven 빌드 후 Nexus 로 배포하기 위하여 설정하는 방법을 설명할 것이다.
우선 Nexus에 Maven Proxy 저장소를 생성할 것이다.
본 가이드에서 설치한 Nexus로 접속을 하고 로그인을 한다.

- URL : http://localhost:8081
- 계정 : admin / admin123

로그인을 하고 좌측의 Components를 클릭하면 우측에 설정되어있는 저장소 목록들이 보인다.



Nexus 저장소를 Maven 저장소로 사용할 수 있도록 Maven 설정을 변경해 주어야 한다.
Maven 설정 변경은 두 가지를 해주어야 한다. 하나는 Nexus 접근을 할 수 있도록 Maven에 Nexus 계정 정보를 설정하는 것과, Maven 프로젝트와 Nexus 저장소 연결 설정하는 것이다.
Nexus 계정 정보 설정은 계정의 추가 또는 변경 사항이 없다면 한번만 해주면 된다.
Maven 프로젝트와 Nexus 저장소 연결은 각 Maven 프로젝트 별로 설정을 해주어야 한다.

우리가 사용할 저장소는 maven-central, maven-snapshots, maven-releases 이며, 각 저장소는 아래와 같은 역할을 한다.

- maven-central : Maven 프로젝트의 공용 저장소라고 보면 되며, 프로젝트에 필요한 라이브러리 등이 관리되는 곳이다.
- maven-snapshots : pom.xml에서 〈version〉에 "-SNAPSHOT"이 있으면, 이곳에 파일이 저장된다.
  ex) 〈version〉1.0.0-SNAPSHOT〈₩version〉
- maven-releases : pom.xml에서 〈version〉에 "-SNAPSHOT"이 없으면, 이곳에 파일이 저장된다.
  ex) 〈version〉1.0.0〈/version〉

Nexus 계정 정보 설정은 setting.xml에서 할 수 있다.
본 가이드를 따라 Maven을 설치하였다면 아래의 경로에 해당 파일이 있을 것이다.

```
c:₩apache-maven-3.5.0₩conf₩
```

setting.xml 에 아래와 같이 Nexus 계정 정보를 추가 한다

```
〈settings〉
....
  〈servers〉
    〈server〉
      〈id〉nexus〈/id〉
      〈username〉admin〈/username〉 〈!--nexus 관리자 계정--〉
      〈password〉admin123〈/password〉 〈!--nexus 관리자 비밀번호--〉
    〈/server〉
  〈/servers〉
```

Nexus 저장소 정보를 추가 한다.

```
<settings>
  <mirrors>
    <mirror>
<id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
<repositories>
        <repository>
          <id>central</id>
          <url>http://localhost:8081/repository/maven-central/</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>

     <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>http://localhost:8081/repository/maven-central/</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
</profiles>

<!--위에 설정한 Profile을 사용하겠다고 선언 -->
  <activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
</settings>
```

Maven 프로젝트에 Nexus 저장소 설정은 pom.xml 에서 할 수 있다.
앞에서 생성하였던 sample01 Maven 프로젝트의 pom.xml 는 아래의 경로에 있을 것이다.

```
C:₩dev₩maven_project₩sample01₩
```

pom.xml 에 아래와 같이 Nexus 저장소 정보를 추가 한다.

```
<project>
...
<distributionManagement>
   <repository>
     <id>nexus</id>
     <name>Releases</name>
     <url>http://localhost:8081/repository/maven-central</url>
   </repository>
   <snapshotRepository>
     <id>nexus</id>
     <name>Snapshot</name>
     <url>http://localhost:8081/repository/maven-snapshots</url>
   </snapshotRepository>
 </distributionManagement>
...
```

여기까지 설정을 완료 하였다면 Maven 에서 아래의 명령을 실행하여 Nexus에 배포할 수 있다.

```
mvn clean deploy
```

우리는 앞서 Jenkins 프로젝트 설정을 하면서 Maven 설정에 위의 명령을 입력하였다.
여기까지 모두 따라 하였다면 모든 설정은 끝이다.

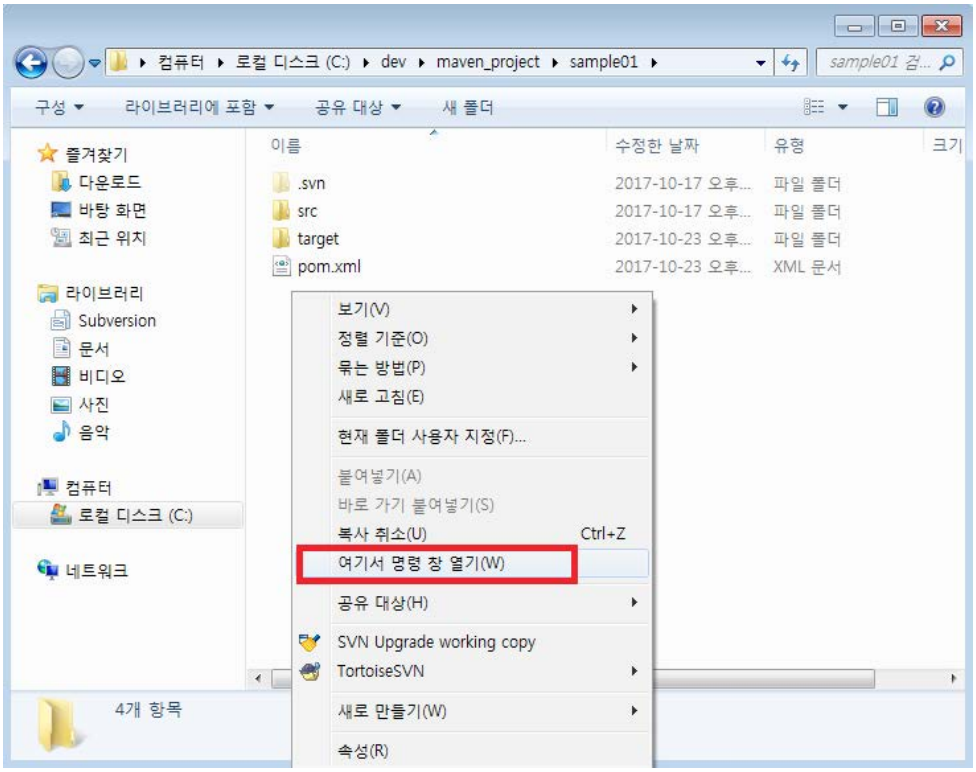
5) 변경 사항 SVN 저장소에 반영하기

앞서 Jenkins에서 svn과 연결을 하였기 때문에 Jenkins에서 빌드 시 svn에 접속하여 소스를 가져오고, 가져온 소스로 빌드를 진행하기 때문에 Maven 설정을 변경한 것을 svn에 반영하여야 한다.
svn에 반영하지 않고 Jenkins에서 빌드를 실행하면 Maven 변경 사항이 적용되지 않는다.
svn 에 변경 사항을 반영은 아래의 명령으로 할 수 있다.

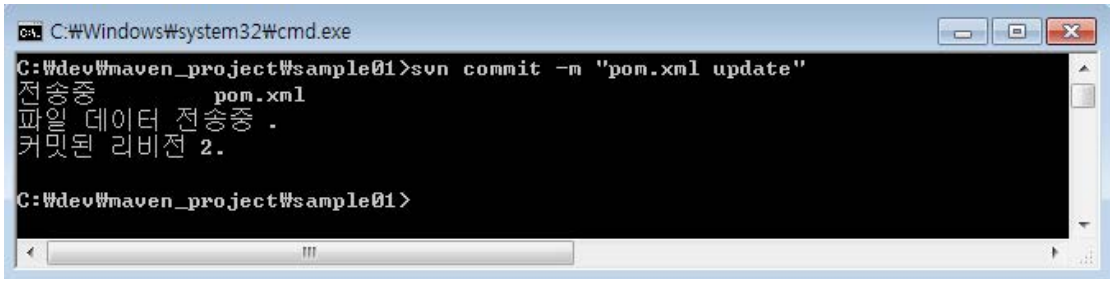
```
svn commit –m “메세지”
```

아래와 같이 위의 명령을 실행하면 local에서 변경한 내용이 svn 저장소에 반영이 되며, 다른 사용자 또는 Jenkins에서 svn 소스를 update하면 변경된 내용이 반영된 소스를 받을 수 있다.

Maven sample01 프로젝트 폴더로 이동을 하고, 'Shift key + 마우스 우클릭'을 하여 명령 창 열기를 클릭한다.



열린 명령 창에서 아래와 같이 명령을 실행하면 변경 사항이 저장소에 반영된다.



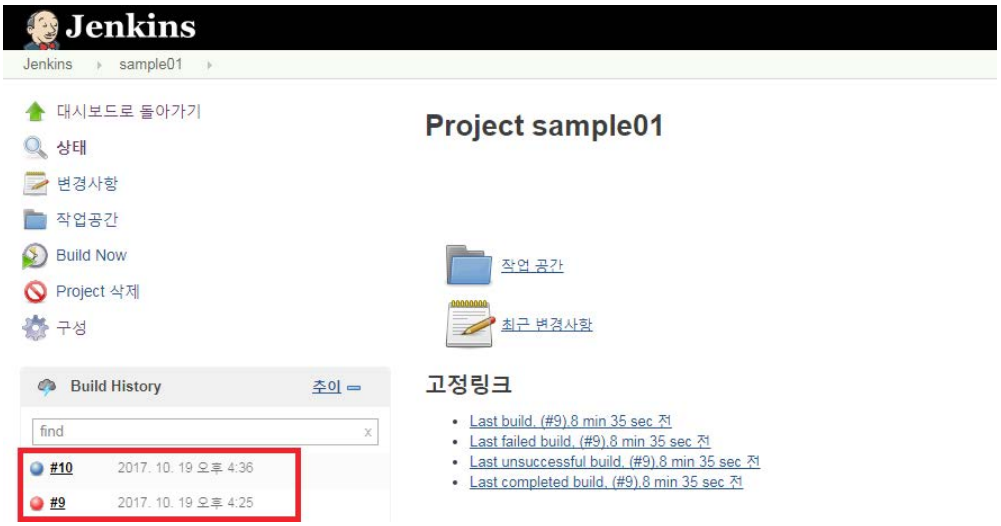
6) Toolchain 확인

이제 지금까지 Jenkins, SVN, Maven, Nexus 가 연동되어 잘 돌아가는지 확인해 보도록 하겠다.
Jenkins에 접속을 하고 프로젝트 목록에서 sample01 프로젝트를 클릭한다.

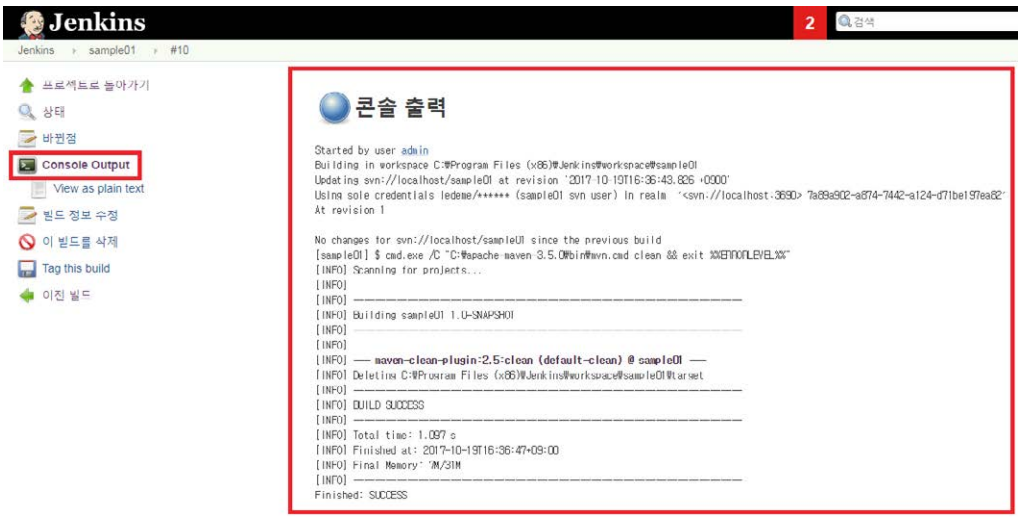
sample01 프로젝트 화면에서 좌측 메뉴에 'Build Now' 가 보일 것이다.
'Build Now'를 클릭하여 Build를 시작한다.



좌측 하단의 히스토리에 빌드가 진행되고 있는 것을 확인할 수 있을 것이며, 정상적으로 빌드가 완료 되면 녹색 블릿이, 빌드가 실패하면 적색 블릿으로 표시 된다.

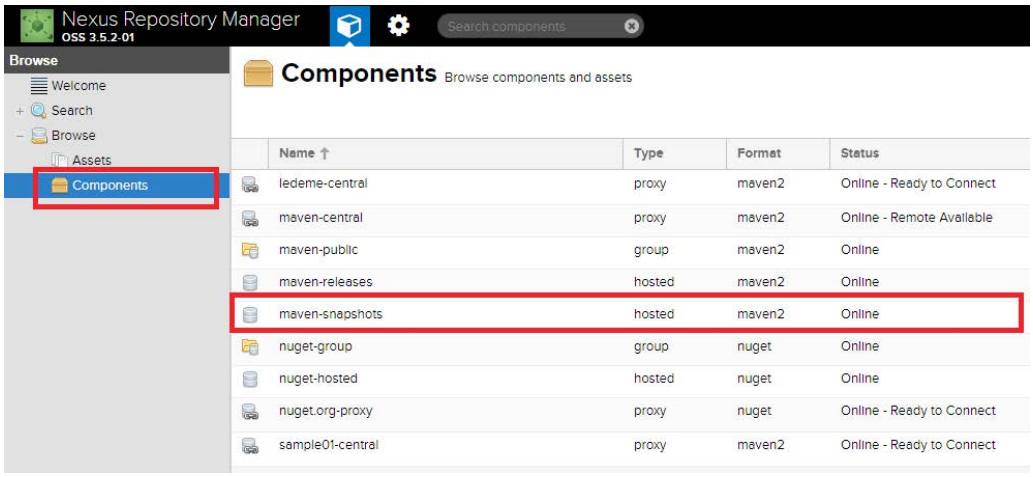


빌드가 완료되면 빌드를 클릭하여 세부 내용을 확인할 수 있다.
좌측의 'Console Output'을 클릭하면 빌드를 하면서 출력되었던 내용을 확인할 수 있다.
빌드가 실패하였다면 여기에서 실패한 내용을 확인할 수 있을 것이다.



빌드한 snapshot을 Nexus에서 확인해 보겠다.

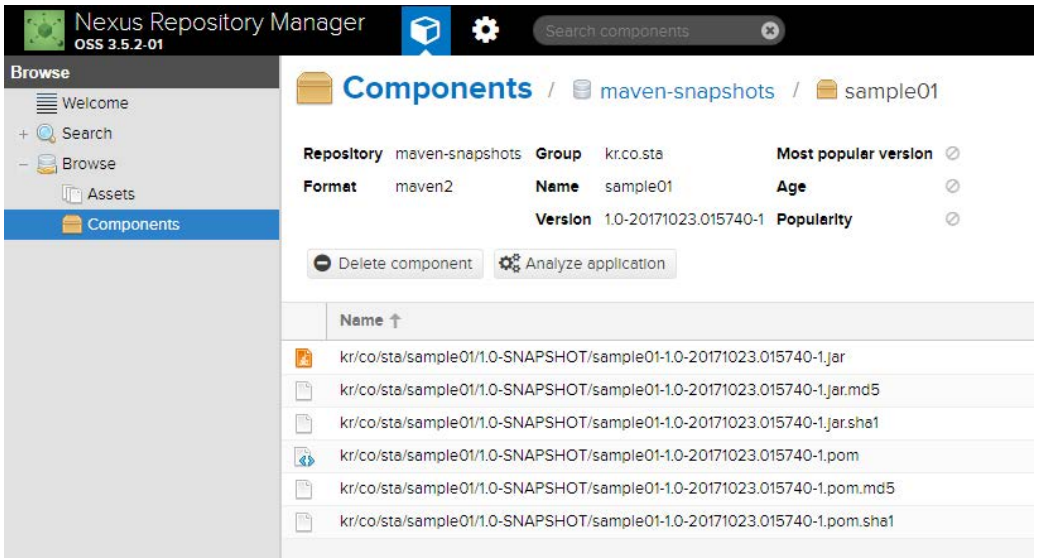
Nexus의 좌측 Components를 클릭하여 우측에 나타나는 저장소 목록에서 maven-snapshots를 클릭한다.



아래와 같이 snapshot이 올라가있는 것을 확인할 수 있다.

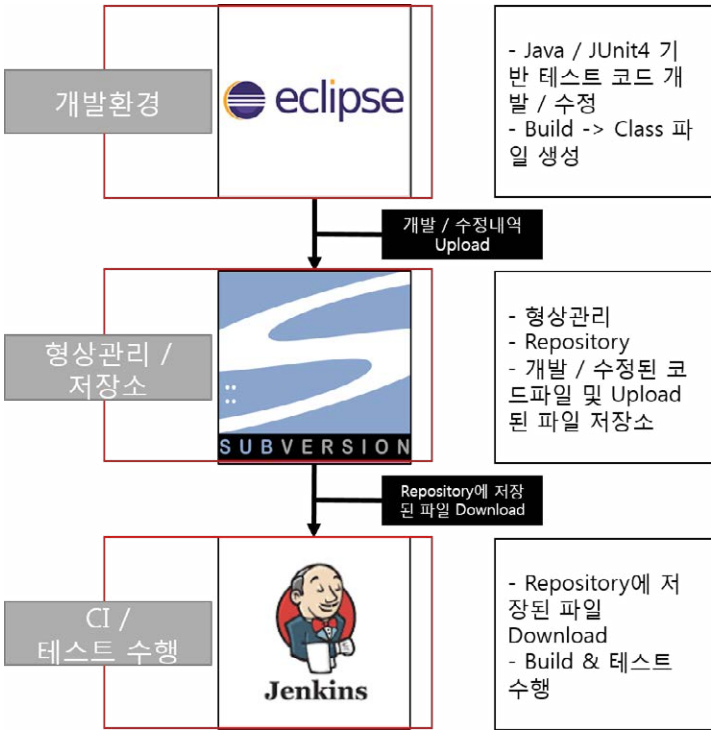


snapshot 을 클릭하면 해당 snapshot 정보와 파일들을 확인할 수 있다.



13.2 Tutorial 2 : Jenkins, SVN, Ant, Selenium, Junit

1) 프로세스



위 프로세스는 Java 기반의 Junit 형태로 가공되어 개발된 Selenium 기반의 테스트 스크립트를 개발하기 위하여 컴파일과 디버깅을 도와주는 통합 개발 환경 IDE 중 Java 개발자들이 가장 많이 사용하는 Eclipse를 이용하여 컴파일 및 빌드를 하고, 그 결과물을 SVN(Subversion)을 통한 형상관리 및 Repository 기능을 이용하여 그 결과물을 저장 / 관리 하여 이를 CI(Continuous Integration) 툴인 Jenkins를 활용한 통합 및 Build 과정을 보여주기 위한 프로세스 이다.

위 프로세스를 진행하기 위해서는 아래와 같은 사전 작업이 필요하다.

■ **개발환경**

1) Java JDK 설치 및 환경설정
2) Eclipse 설치 및 Project 생성
3) Selenium Library 다운로드 및 Referenced Libraries 추가
4) Web driver (Fire Fox, Chrome 등) 다운로드 및 Project 폴더로 이동
5) build.xml 추가 (Jenkins에서 Ant 빌드시 사용)

■ **형상관리 / 저장소**

1) SVN 설치 및 Repository 생성
2) Eclipse 에서 SVN plug in 설치
3) 개발환경에서 생성한 Project와 Repository 연결

■ **CI / 테스트 수행**

1) Jenkins 설치 및 Plug in 설치
2) Apache Ant 설치 및 환경설정
3) Jenkins Job 생성 및 Repository와 연결
4) Ant build 환경설정

위 사전조건이 완료된 이후에는 Jenkins를 통하여 Java로 개발된 결과를 형상관리, 저장소, 최종적으로 통합된 CI 환경에서 테스트 수행까지 진행 할 수 있다.

2) Eclipse

① 다운로드

Export 받은 소스를 실행하기 위해서 (GUI → API 변환) Eclipse 를 사용하여 프로젝트를 만들어야 한다. 많은 Java 편집도구가 존재하지만 보편적으로 Eclipse를 사용하므로 Eclipse를 이용하여 프로젝트 생성 및 실행하는 방법은 아래와 같다.

일단 PC에 자바 JDK가 설치가 되었다는 가정하에 아래 URL에 접속하여 Eclipse를 다운받아 설치한다.

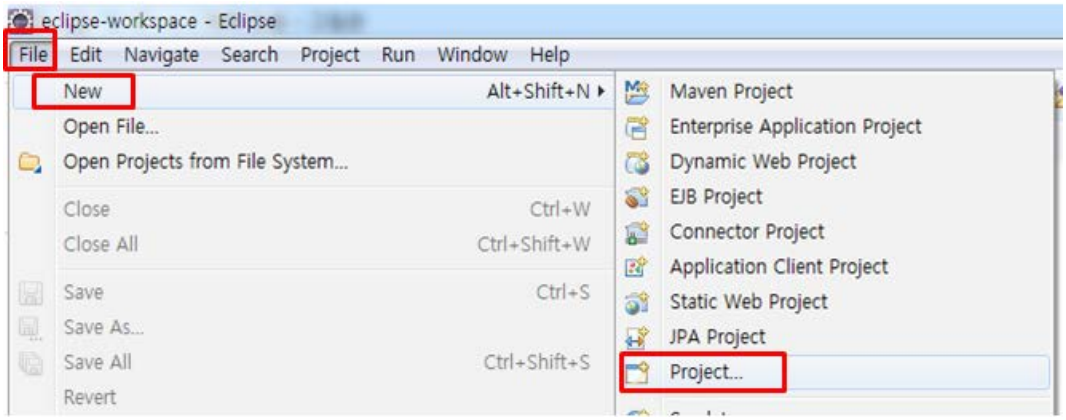
https://www.eclipse.org/

해당 페이지에서 Download 버튼을 이용하여 설치 프로그램을 다운 받은 후 실행하여 Eclipse를 설치하면 된다.

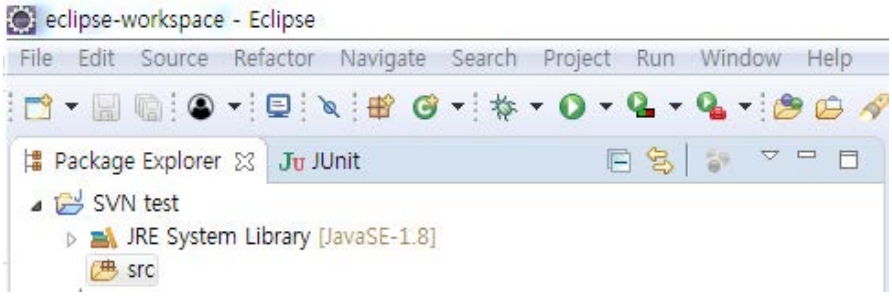
② 프로젝트 생성

Eclipse 가 설치가 완료되면 실행하여 아래와 같이 프로젝트를 실행한다.

File > Java Project > Project name 입력 (ex : SVN test) > Finish 버튼 클릭



아래 그림과 같이 프로젝트가 생성되었다.



프로젝트 생성이 완료 되었다면, Selenium 및 Junit 환경으로 테스트 하기 위한 Library를 추가 해 주어야 한다. 그 방법은 아래와 같다.

③ Selenium Library

Eclipse 프로젝트는 기본 JRE 에서 제공하는 Library만 가지고 있기 때문에, Selenium Java Libraries를 추가해 주어야 한다.

일단 아래 URL 접속 후 Java Client 를 다운로드 받아야 한다.

http://www.seleniumhq.org/download/

Language	Client Version	Release Date			
Java	3.5.3	2017-08-29	Download	Change log	Javadoc
C#	3.5.1	2017-08-15	Download	Change log	API docs
Ruby	3.5.1	2017-08-14	Download	Change log	API docs
Python	3.5.0	2017-08-10	Download	Change log	API docs
Javascript (Node)	3.5.0	2017-07-24	Download	Change log	API docs

위 URL 으로 다운로드가 안되면 아래 URL에서도 Libraries를 다운받을 수 있다.

https://npm.taobao.org/mirrors/selenium/3.5/

selenium-java-3.5.0.zip
selenium-java-3.5.1.zip
selenium-java-3.5.2.zip
selenium-java-3.5.3.zip

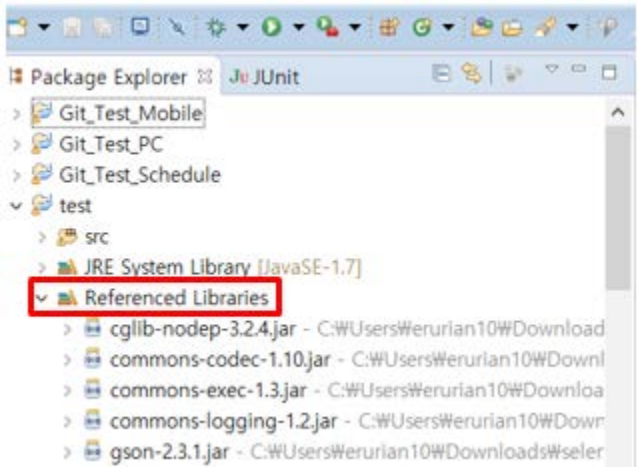
해당 다운로드 링크를 찾아 클릭하면 다운로드 된다

다운로드 받은 Libraries 파일의 압축을 푼 후 Eclipse 에서 라이브러리를 추가한다.

Eclipse 에서 순서는 아래와 같다.

File > Properties > Java Build Path > Libraries > Add External JARs > 다운받은 Libraries 추가 > Apply > OK

위 작업이 완료되면 아래와 같이 Referenced Libraries 가 추가된다.



④ Web Driver

Selenium은 Web Driver 라는 것을 통해 디바이스에 설치된 브라우저들을 제어할 수 있다.

코딩된 소스를 특정 웹 (Fire Fox, Chrome, Internet Explorer, Android 등)과 연결하여 사용하기 위해 사용되는 커넥터 프로그램으로 생각하면 된다.

해당 PC에는 Selenium IDE를 수행하기 위해. Fire Fox가 설치되어 있으므로, Fire Fox에 대해 수행하는 방법을 진행하자.

우선 아래 URL 에서 Fire Fox driver 를 다운 받아보자.

http://www.seleniumhq.org/download/

Third Party Drivers, Bindings, and Plugins

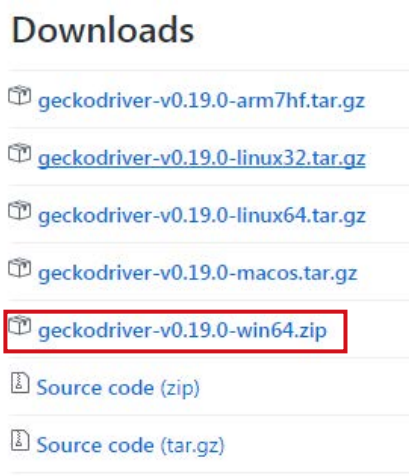
Selenium can be extended through the use of plugins. Here are a number of plugins created and maintained by third parties. For more information on how to create your own plugin or have it listed, consult the docs.

Please note that these plugins are not supported, maintained, hosted, or endorsed by the Selenium project. In addition, be advised that the plugins listed below are not necessarily licensed under the Apache License v.2.0. Some of the plugins are available under another free and open source software license; others are only available under a proprietary license. Any questions about plugins and their license of distribution need to be raised with their respective developer(s).

Third Party Browser Drivers NOT DEVELOPED by seleniumhq

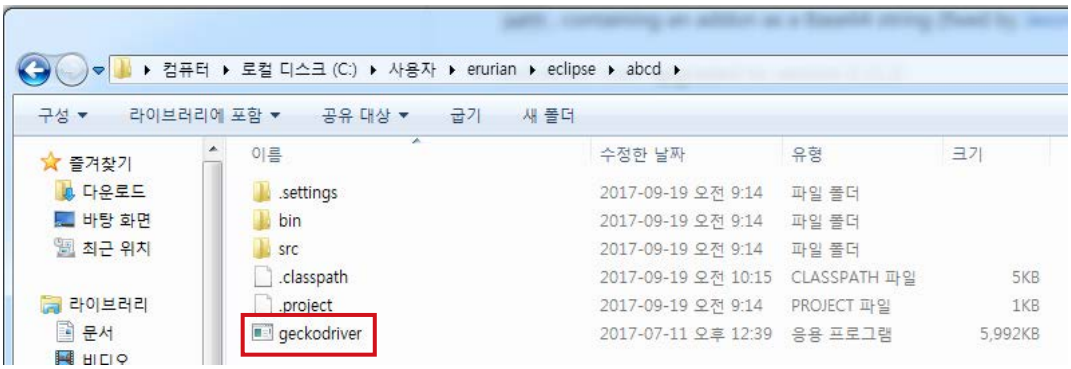
Browser					
Mozilla GeckoDriver	0.18.0	change log	issue tracker	Implementation Status	Released 2017-07-11
Google Chrome Driver	2.32	change log	issue tracker	selenium wiki page	Released 2017-08-31
Opera	2.29		issue tracker	selenium wiki page	Released 2017-06-27
Microsoft Edge Driver			issue tracker	Implementation Status	

위 다운로드링크를 클릭하게 되면 https://github.com/mozilla/geckodriver/releases 으로 이동 후 해당 PC의 운영체제에 맞는 버전을 다운 받으면 된다.



다운받은 압축파일의 압축 해제 후 해당 파일을 Eclipse 로 생성한 프로젝트의 Work space로 옮겨준다.

Ex) C:₩Users₩erurian₩eclipse₩abcd₩geckodriver.exe



⑤ 개발한 파일 실행하기 (Junit)

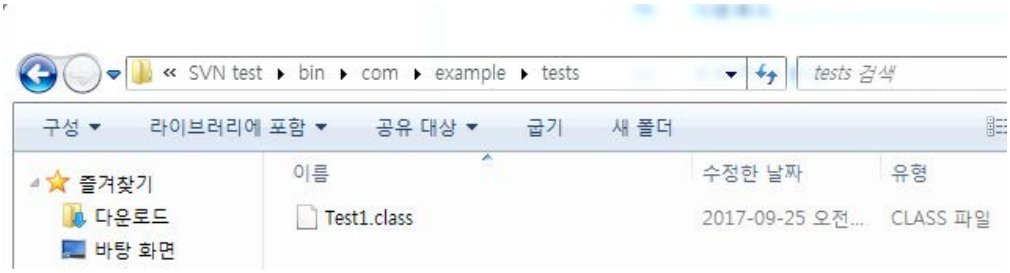
해당 예제는 Selenium의 Library를 활용하여 Junit4 를 이용하여 빌드를 진행하는 방법을 진행하다.

앞서 설명한 Selenium IDE에서 Export 받은 파일을 활용해도 좋고, 임의로 개발된 Junit 형태의 소스를 사용해도 무방하다.

빌드가 완료 된 이후에는 bin 폴더 안에 class 파일이 생성된다.

우리는 이 파일을 이용하여 최종적으로 Jenkins에서 자동화 테스트 빌드를 수행한다.

Eclipse 에서 빌드를 수행 한 후 workspace에서 class 파일이 생성된 것을 확인할 수 있다.



ex) C:₩Users₩erurian₩eclipse-workspace₩SVN test₩bin₩com₩example₩tests

⑥ build..xml

Apache Ant로 빌드할 때 필요한 필수파일로, 빌드에 대한 내용을 넣어주는 build.xml 파일을 추가하자.

임의로 만들어 추가할 수 있고, 프로젝트를 선택 후 New > Other 를 선택한 후 XML > XML File 을 선택하여 생성 할 수 있다. (메모장으로 파일을 생성해도 무관하다.)

〈build.xml 예제〉

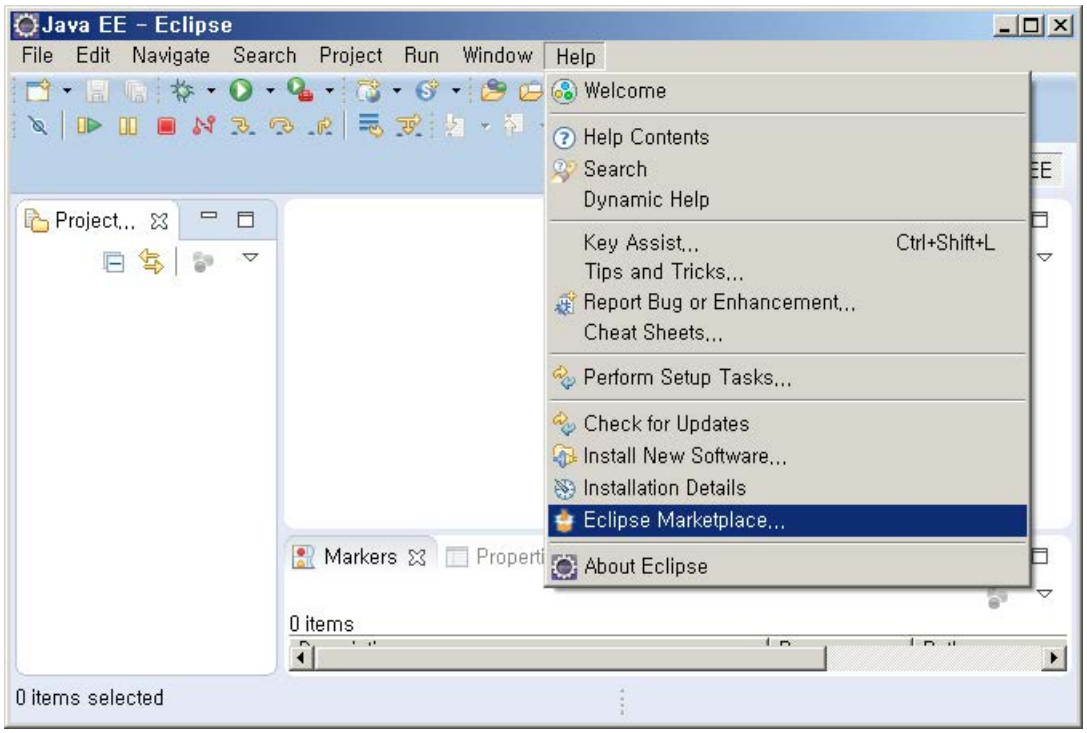
```
〈?xml version="1.0"?〉
〈project name="JUNIT" default="main" basedir="."〉
〈property name="test.class.dir" location="bin" /〉
〈property name="test.report.dir"
location="C:₩Users₩erurian₩.jenkins₩workspace₩svn_test₩testreport" /〉
〈path id="junit.class.path"〉
〈fileset dir="selenium-java-3.5.3"〉
      〈include name="**/*.jar"/〉
〈/fileset〉
〈/path〉
〈target name="makedir"〉
〈mkdir dir="${test.report.dir}" /〉
〈/target〉
〈target name="junit" depends="makedir"〉
〈junit printsummary="on" fork="true" haltonfailure="no" failureproperty="test.failed"〉
〈classpath refid="junit.class.path" /〉
〈classpath〉
  〈pathelement location="${test.class.dir}"/〉
 〈/classpath〉
〈formatter type="xml" /〉
〈batchtest todir="${test.report.dir}"〉
〈fileset dir="${test.class.dir}"〉
〈include name="**/**.class" /〉
〈/fileset〉
〈/batchtest〉
〈/junit〉
〈/target〉
〈target name="main" depends="junit"〉
〈description〉Main target〈/description〉
〈/target〉
〈/project〉
```

위 xml 파일을 이용하여 Apache Ant 빌드를 수행한다.
Eclpise 에서는 Ant 빌드를 수행하지 않고, Jenkns 에서 사용되지만, SVN과 연동 및 형상관리를 위해 추가하자.

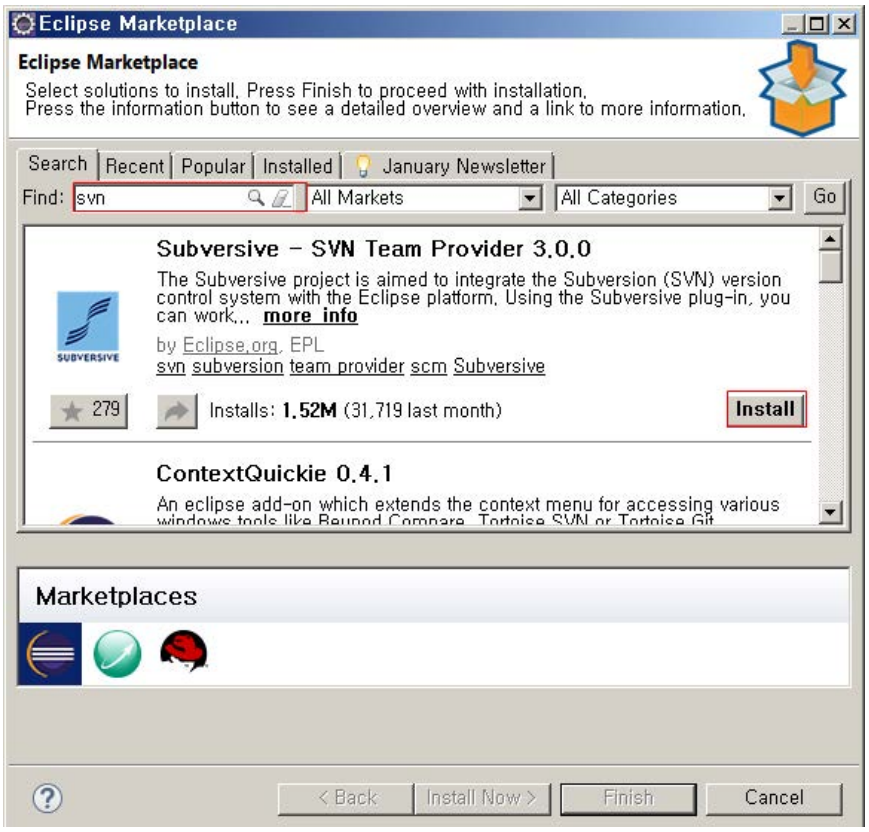
3) SVN

① SVN plug in 설치

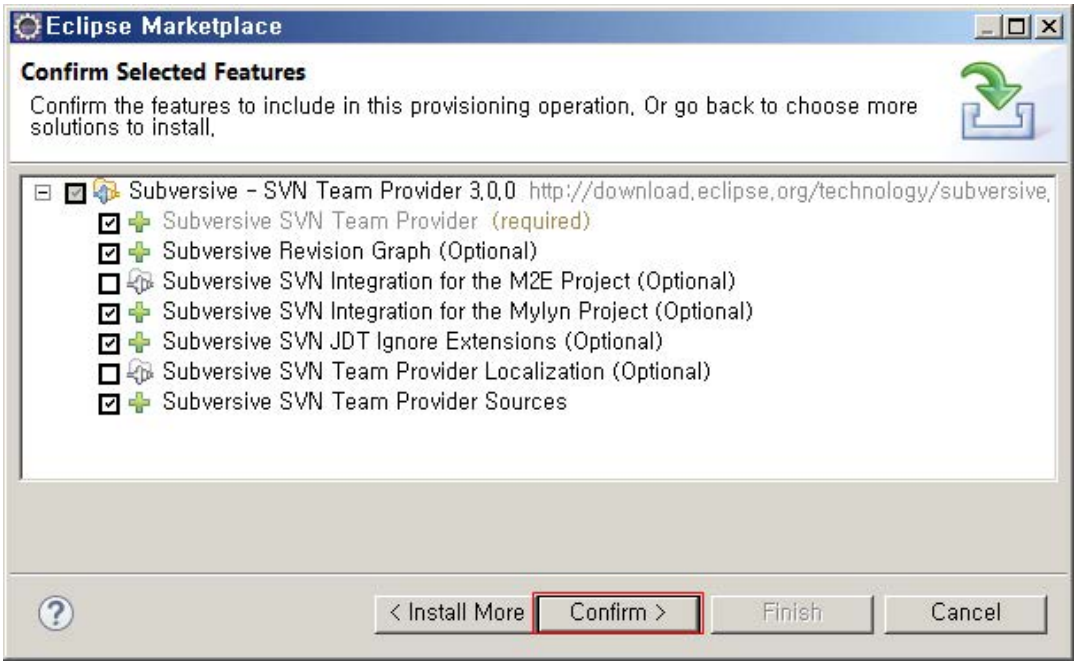
Eclipse 에서는 다양한 Plug in 을 지원한다. 그 중 SVN 과 연동하여 사용할 수 있도록 SVN Plug in 을 설치해 보자.



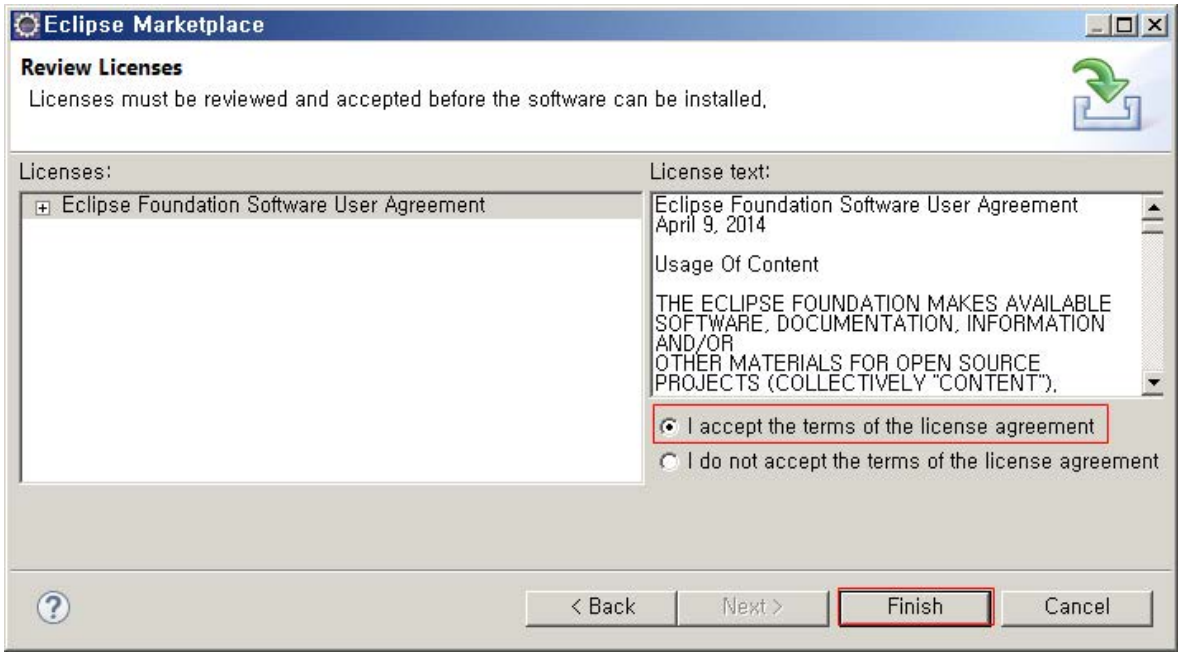
Eclipse를 실행한 후 상단의 메뉴 중Help > Eclipse Marketplace… 선택
Find 항목에 SVN 입력 후 검색 버튼(or Enter) 버튼 선택 > Subversive - SVN Team Provider *.*.* 의 Install 버튼 클릭 (해당 Plugin의 최신버전을 설치하면 된다.)



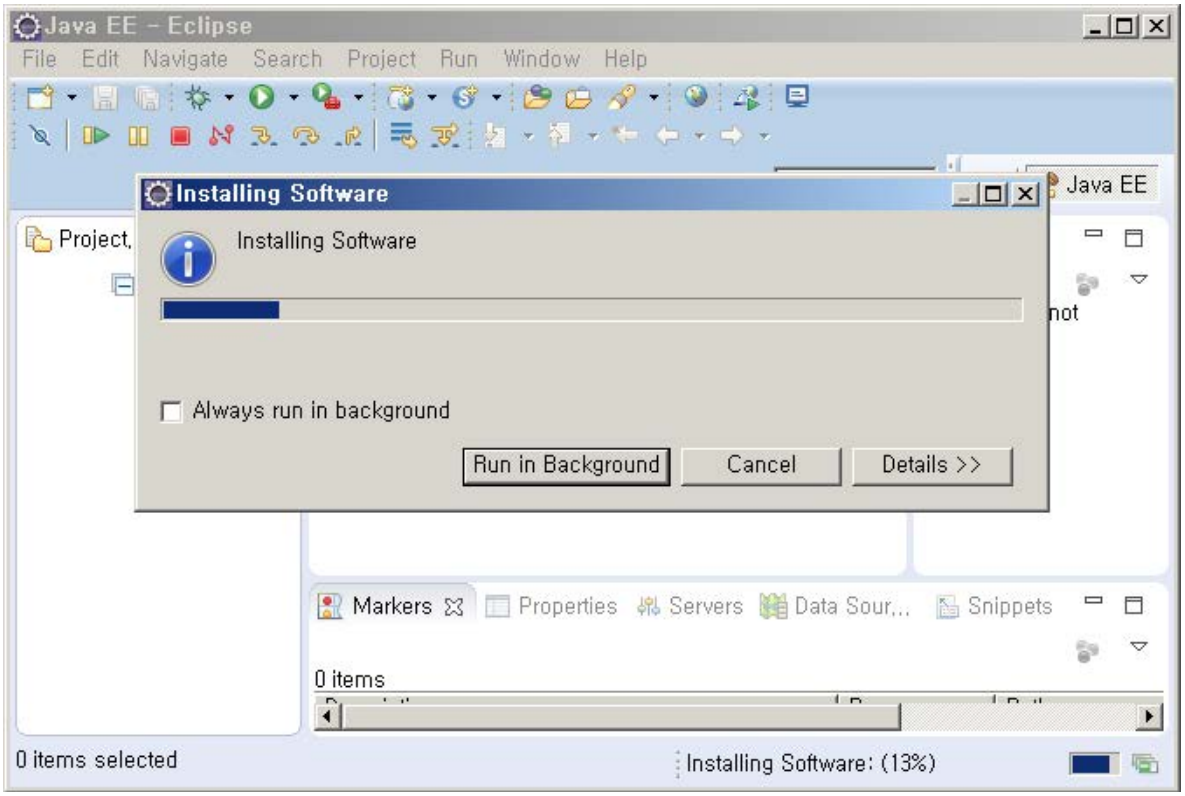
리스트업 된 항목을 확인하고 Confirm 버튼 선택



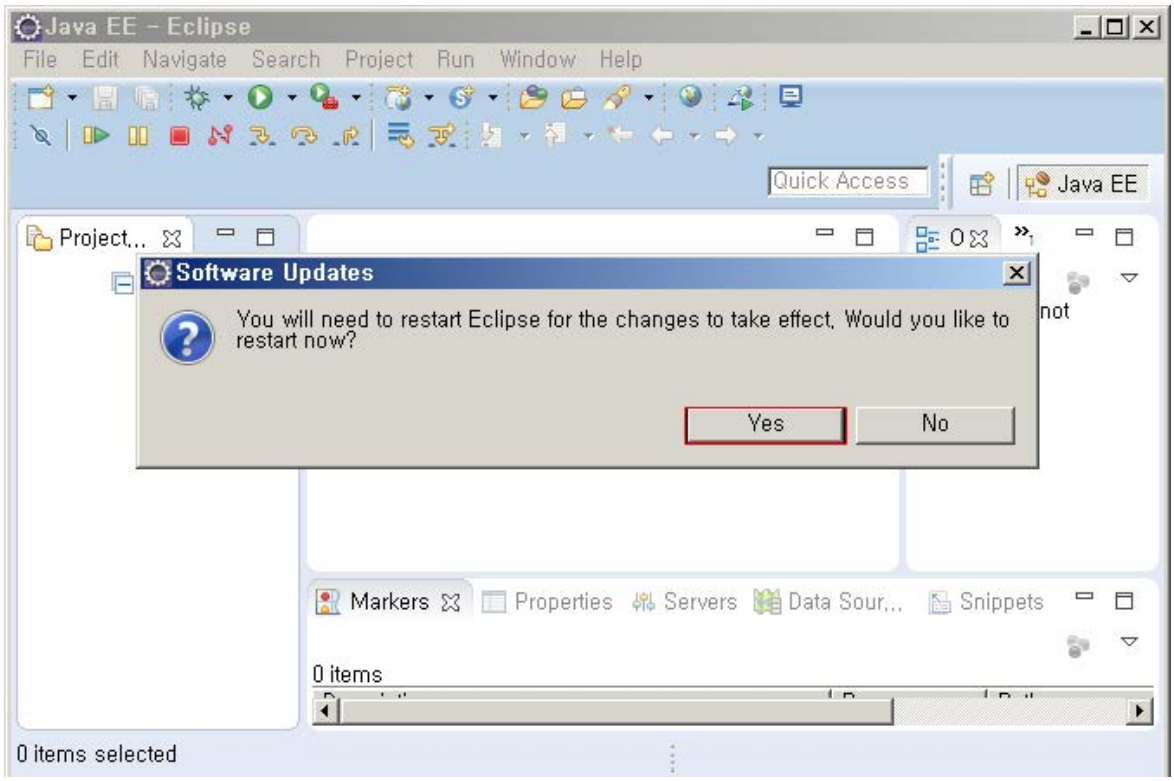
라이선스 동의 라디오 버튼을 선택하고 Finish 버튼 선택



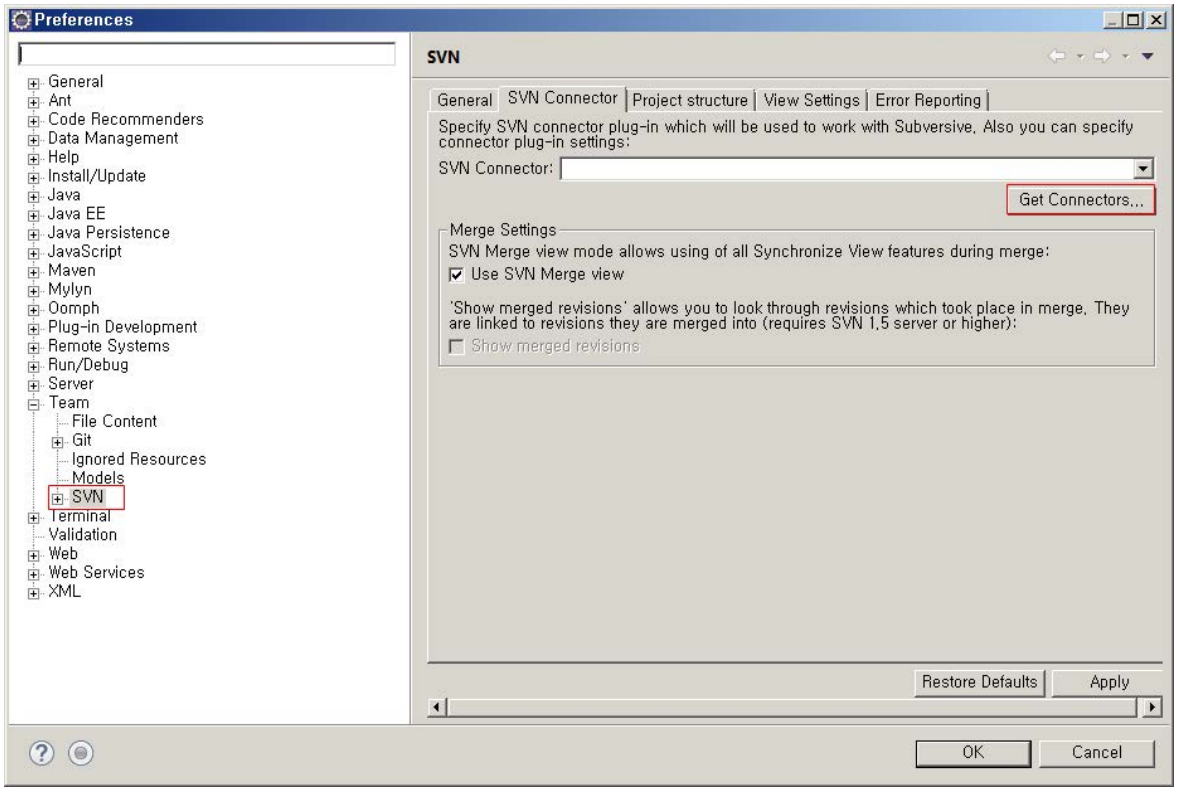
SVN plug in이 Installing 되고 있다.



설치된 SVN의 반영을 위해 Eclipse의 재시작이 필요하다. Yes 버튼 선택



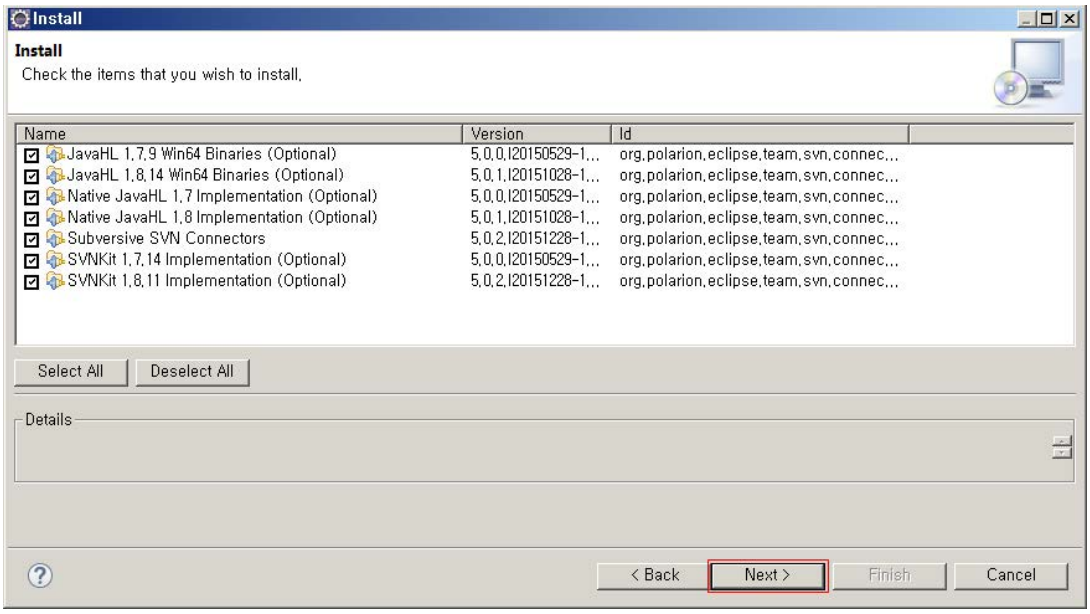
재시작된 Eclipse의 Menu > Window > Preferences 메뉴에서 아래와 같이 SVN 메뉴를 선택 > SVN Connector 탭을 선택 > Get Connectors… 를 선택



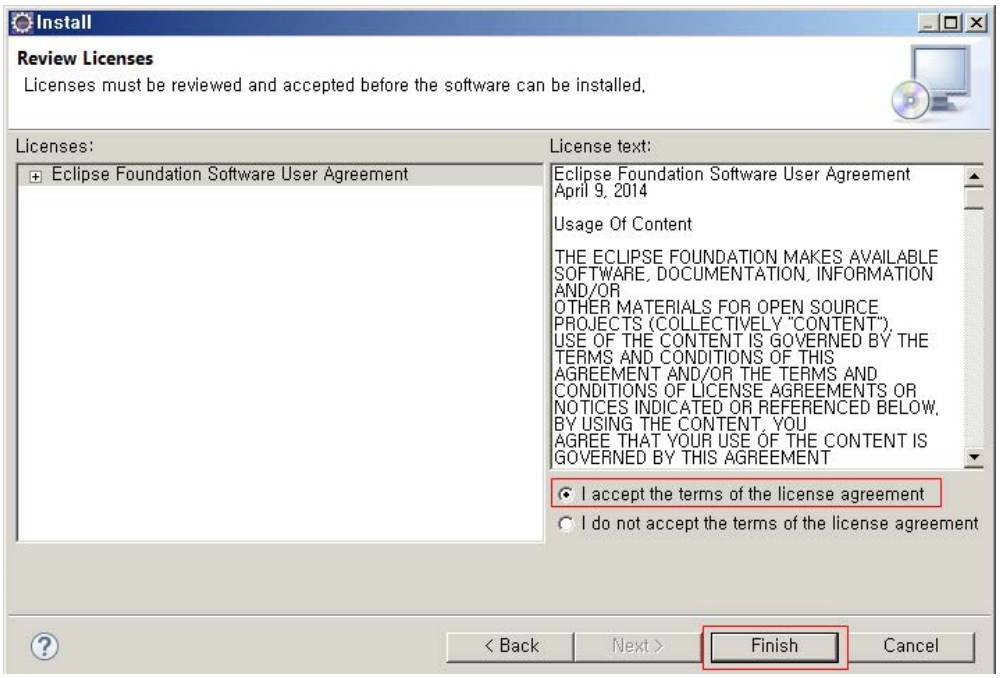
SVN Connectors를 선택할 수 있는 화면에서 Connector 전체를 선택하고 Finish 버튼 선택



Install 화면에서 리스트를 확인하고 Next 버튼을 선택



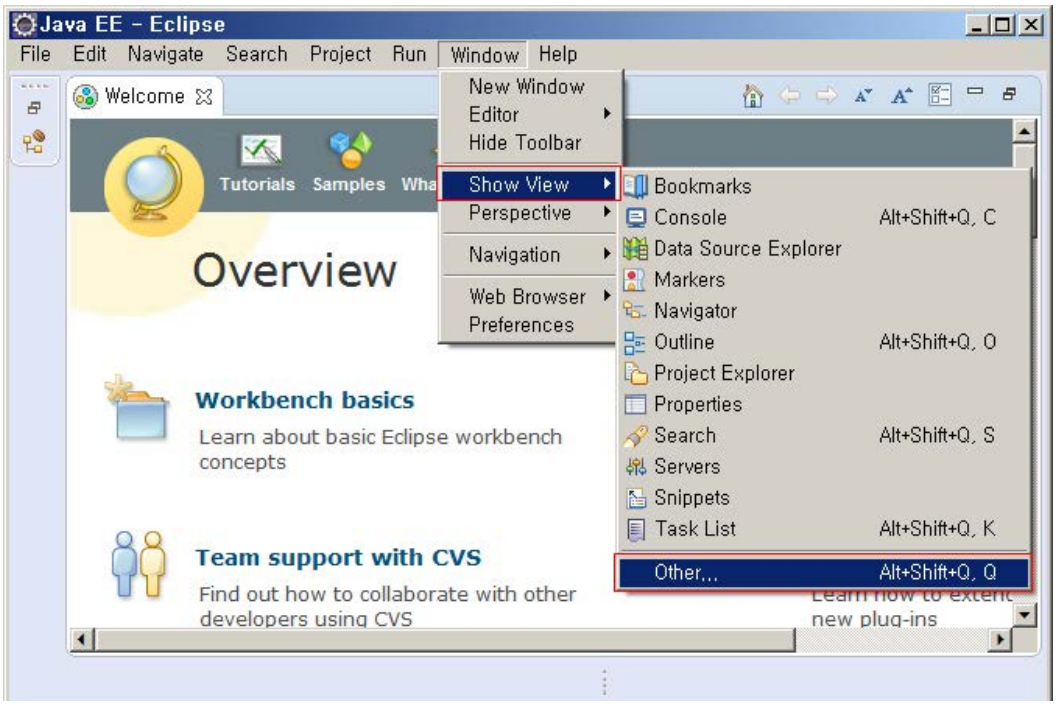
라이선스 동의 라디오 버튼을 선택하고 Finish 버튼 선택



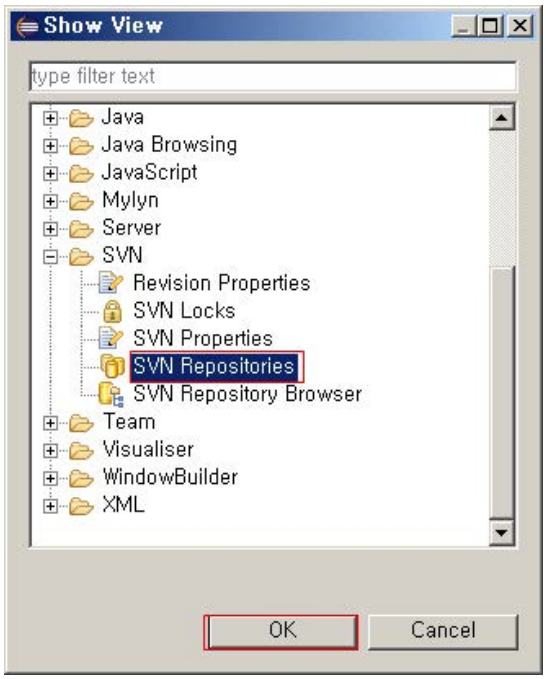
다시 SVN Connector의 반영을 위해 Eclipse의 재시작



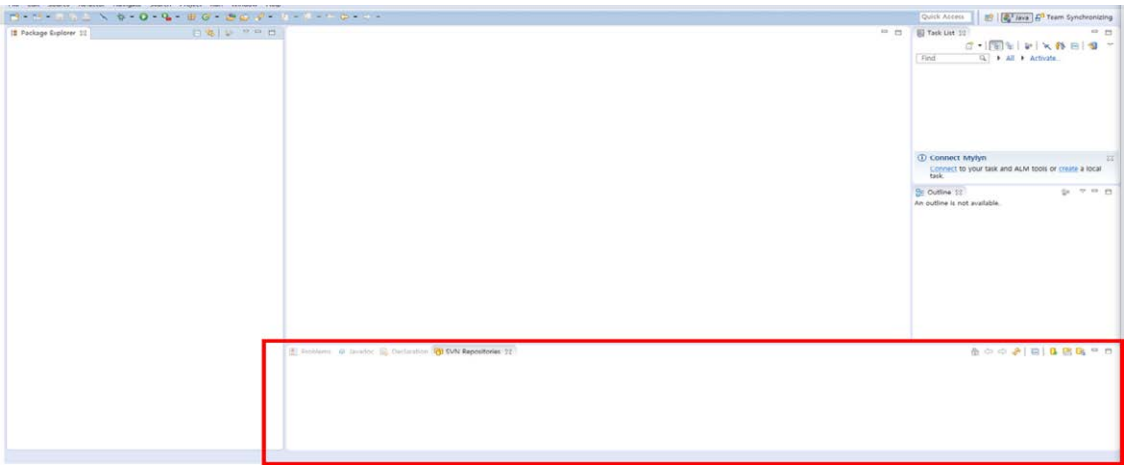
재시작된 Eclipse에서 SVN 저장소 등록을 위해 Menu > Window > Show View > Other… 선택
Show View 화면에서 SVN > SVN Repository를 선택 후 OK 버튼 선택



Eclipse 에서 SVN Repositories 창이 등록된 것을 확인

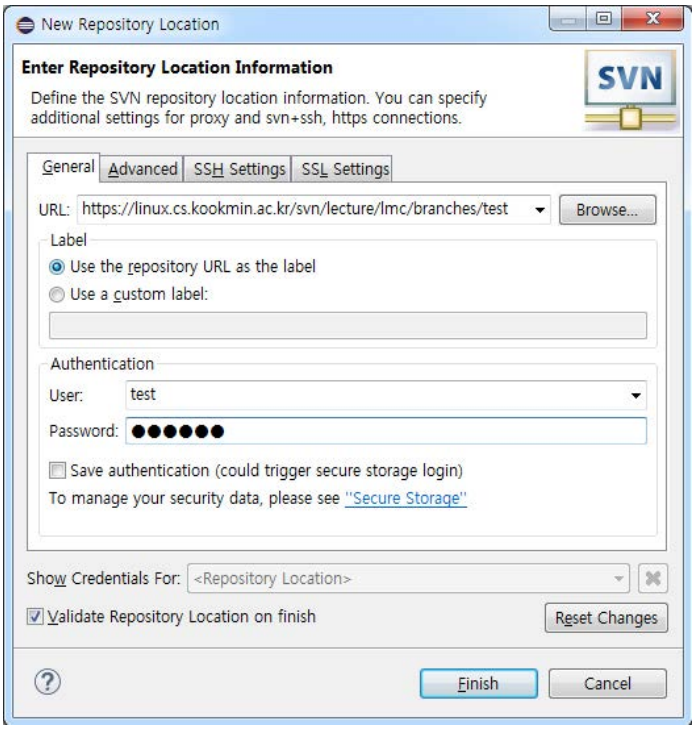


새로운 Repository를 등록하기 위해서 SVN Repository 창에서 마우스 오른쪽 버튼을 클릭하여 New >



Repository 를 통해 새로운 저장소를 등록하거나, 아래와 같이 버튼을 클릭하여 새로운 저장소를 등록하자.
새로운 저장소를 등록하기 위해 저장소의 URL과 본인의 계정 및 비밀번호를 입력한다.





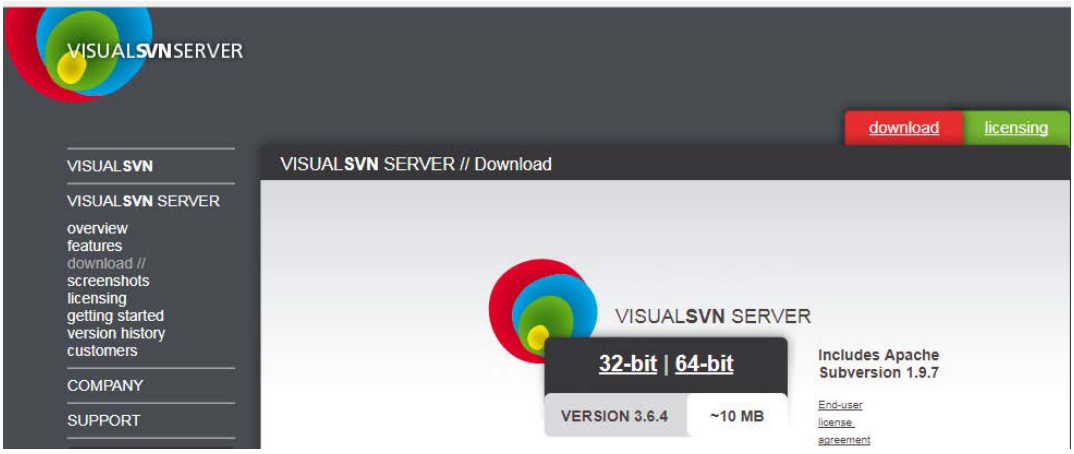
② Visual SVN 사용법

Visual SVN은 Windows에 Subversion Server를 설치, 설정, 관리할 때 필요한 모든 것을 포함한 것이다. Command 창에서 SVN Repository 관리 및 설치 보다는 간단한 UI 를 이용하여 Repository 생성 및 관리가 가능하고, 효과적으로 사용할 수 있다.

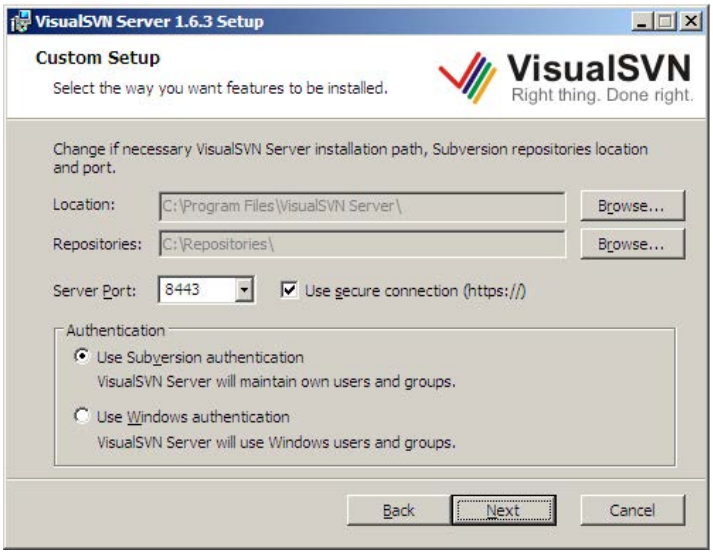
• 설치하기

아래 URL 에 접속하여 설치파일을 Download 받자.

https://www.visualsvn.com/server/download/

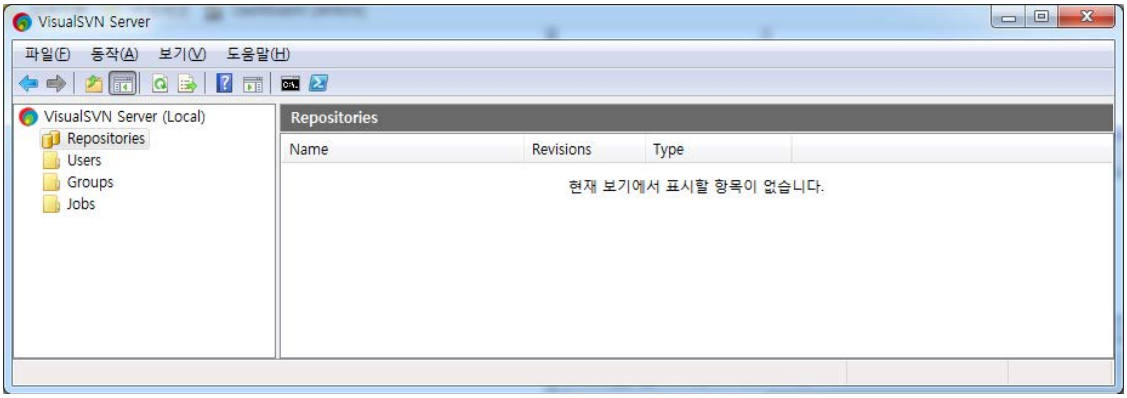


설치할 PC의 운영체제에 맞추어 다운 받으면 VisualSVN-Server-3.6.4-x64.,msi 파일이 다운 받아지고, 해당 파일을 설치하면 된다.



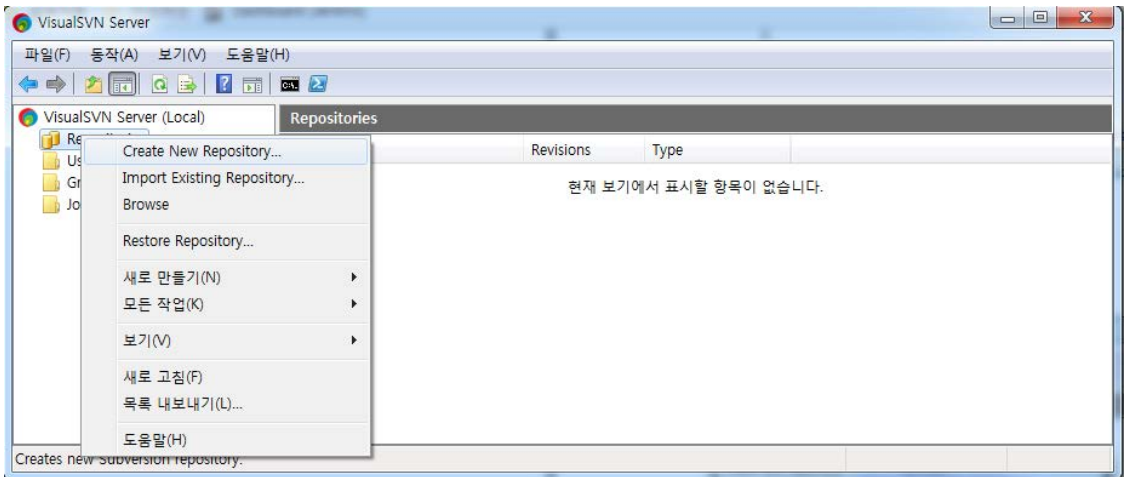
• VisualSVN 관리 콘솔(Management Console)

그림에서 볼 수 있듯이 저장소(repository)와 사용자, 사용자 그룹을 여기서 모두 제어할 수 있다. 이미 만들어진 저장소/유저/유저그룹의 정보를 볼 수 한눈에 볼 수 있는 것은 물론, 새로운 저장소를 만들거나 유저/유저 그룹을 추가/제거 하는 등의 일을 여기서 모두 할 수 있게 된다.

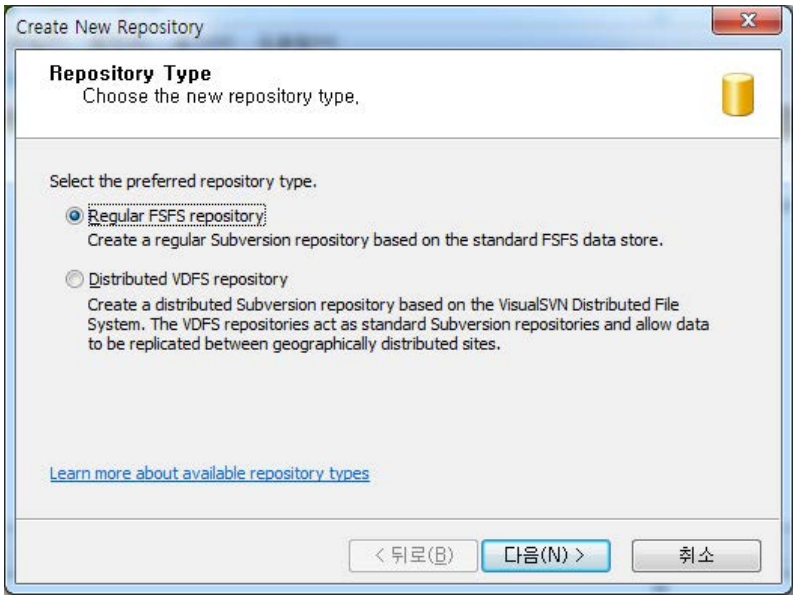


• 저장소 관리 (Repository Management)

Repositories > 마우스 우클릭 > Create New Repository 선택



Repository Type 선택 후 다음 버튼 클릭



Repository Name 입력 후 다음 버튼 클릭



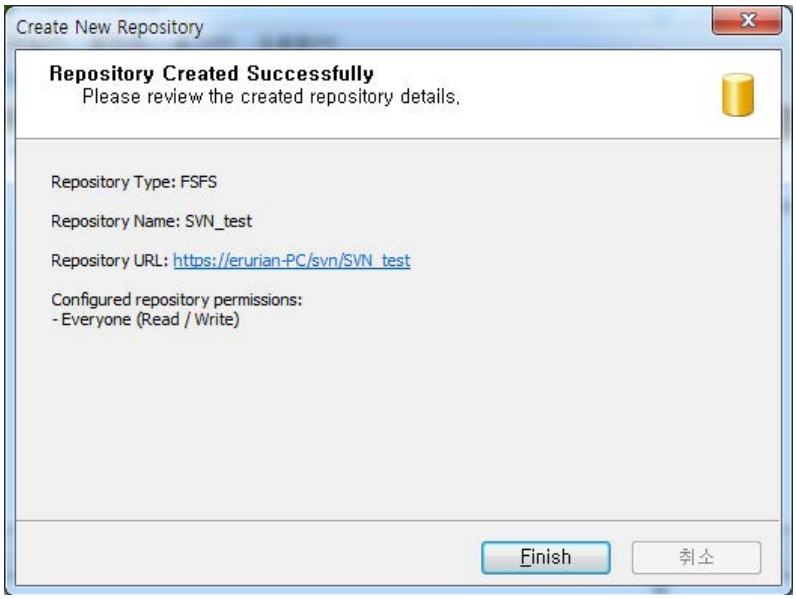
Repository Structure 선택 후 다음버튼 클릭



Repository Access Permission 선택 후 Create 버튼 클릭

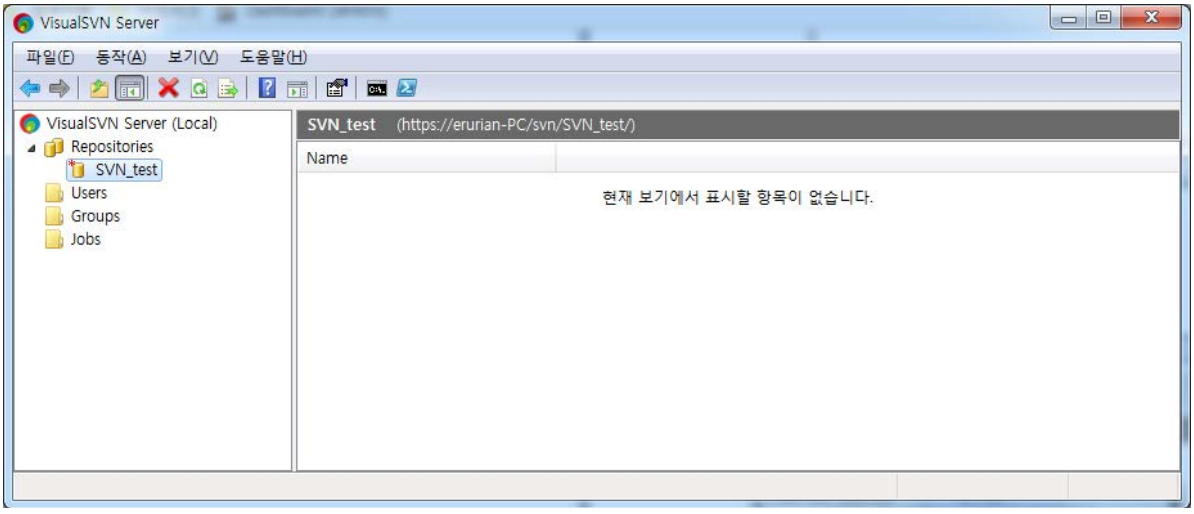


Repository 생성 확인



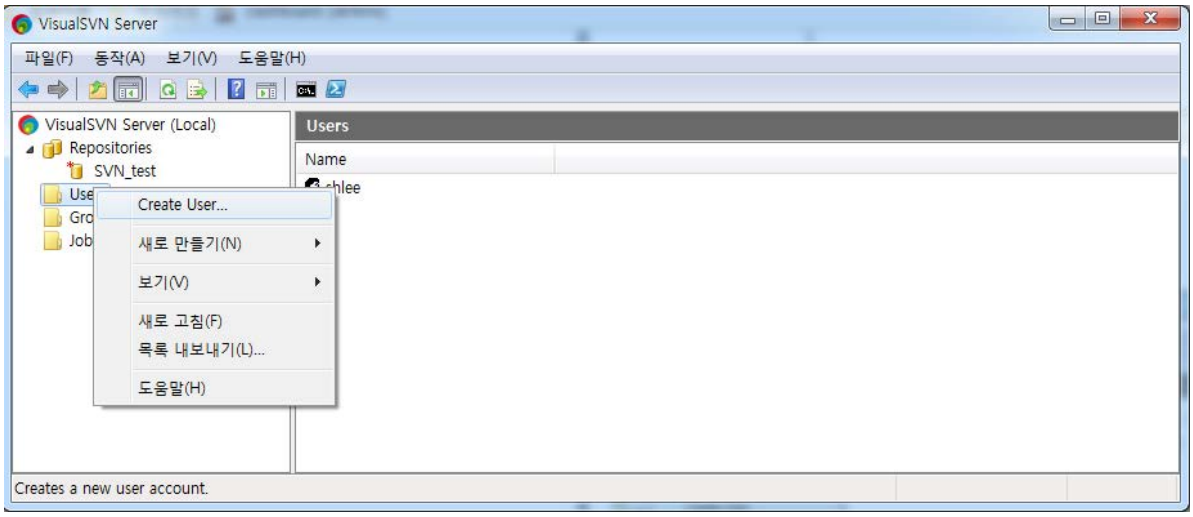
위와 같이 Repository 가 생성되었다.

생성된 Repository는 아래와 같이 표시된다.

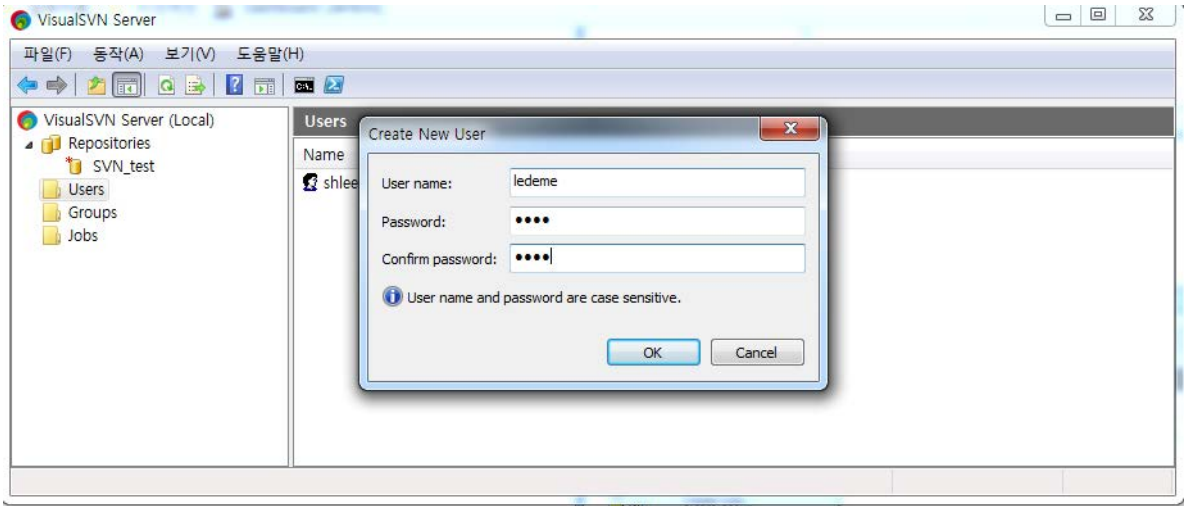


SVN 에서 사용할 계정 생성 방법은 아래와 같다.

User > 마우스 우클릭 > Create User 버튼 클릭



Username / Password / Confirm password 입력 후 OK 버튼 클릭



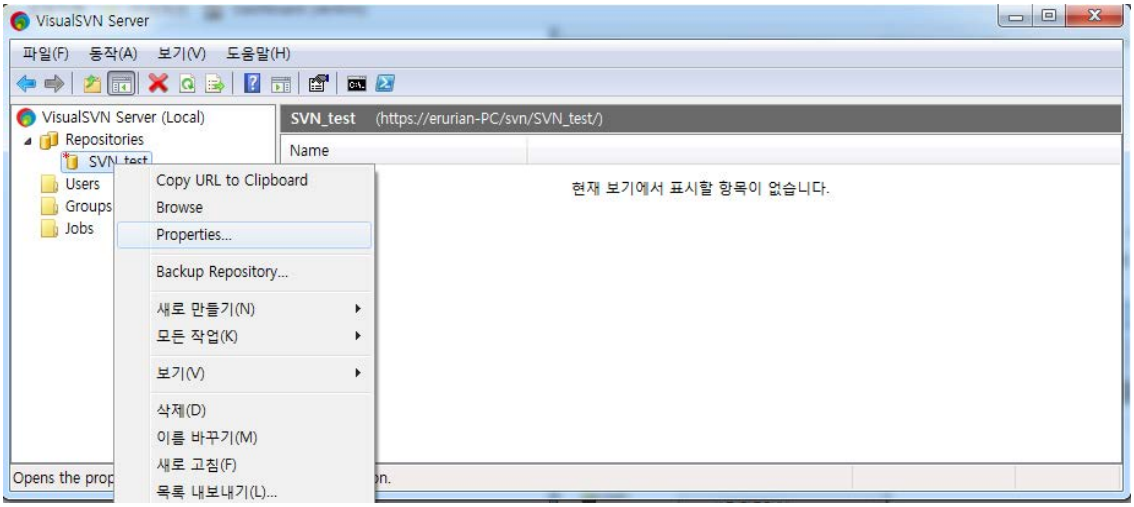
생성 후 아래와 같이 생성 확인



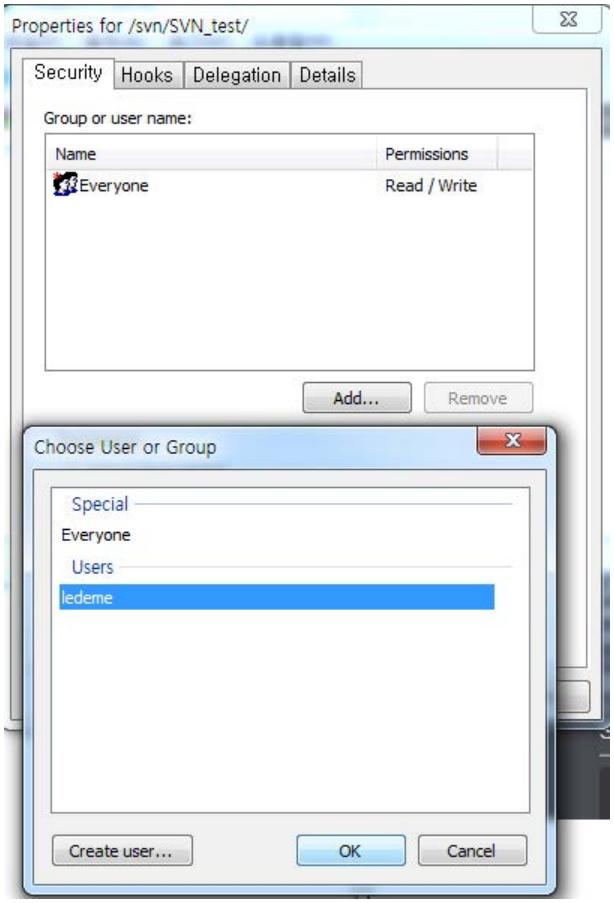
• 저장소에 대한 접근 권한 설정

각 저장소 별로 별도의 권한을 설정할 수 있다,

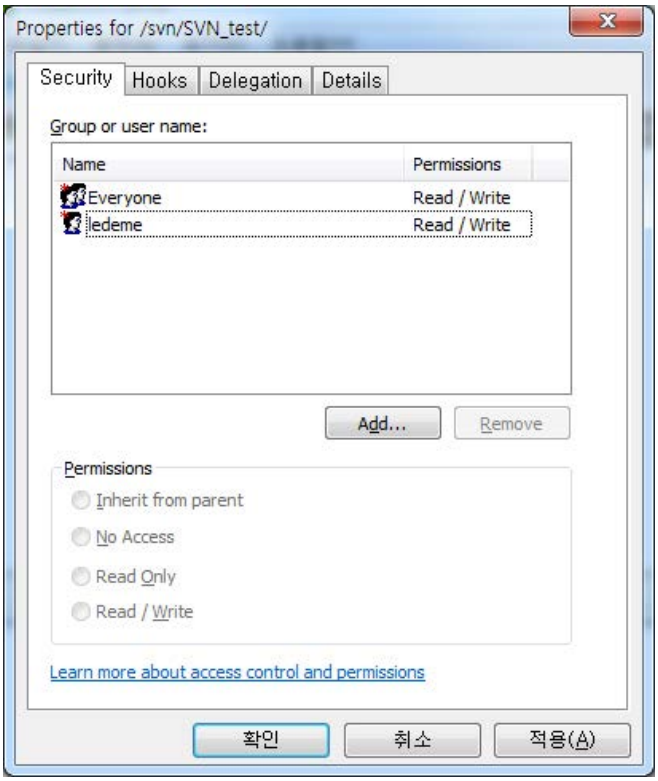
일단 생성된 Repository에서 마우스 우클릭 > Properties 버튼 클릭



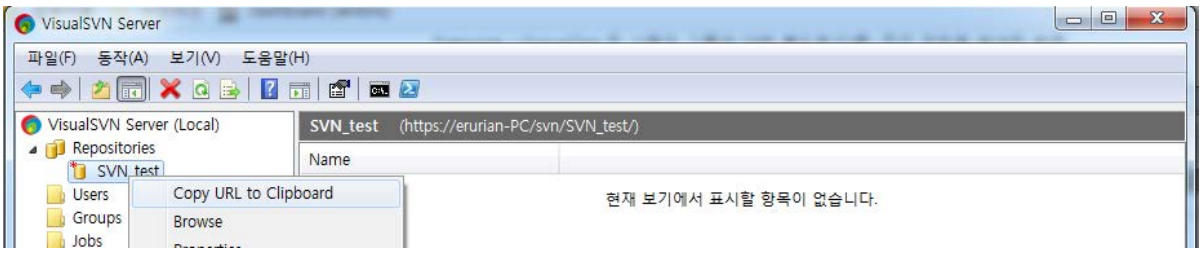
Add 버튼 클릭 후 권한이 필요한 User 선택 > OK 버튼 클릭



권한을 추가하면 아래와 같이 선택한 User가 해당 Repository의 권한이 추가된다.



해당 Repository URL 정보는 아래와 같이 Repository 선택 > 마우스 우클릭 > Copy URL to Clipboard 로 복사 할 수 있다.

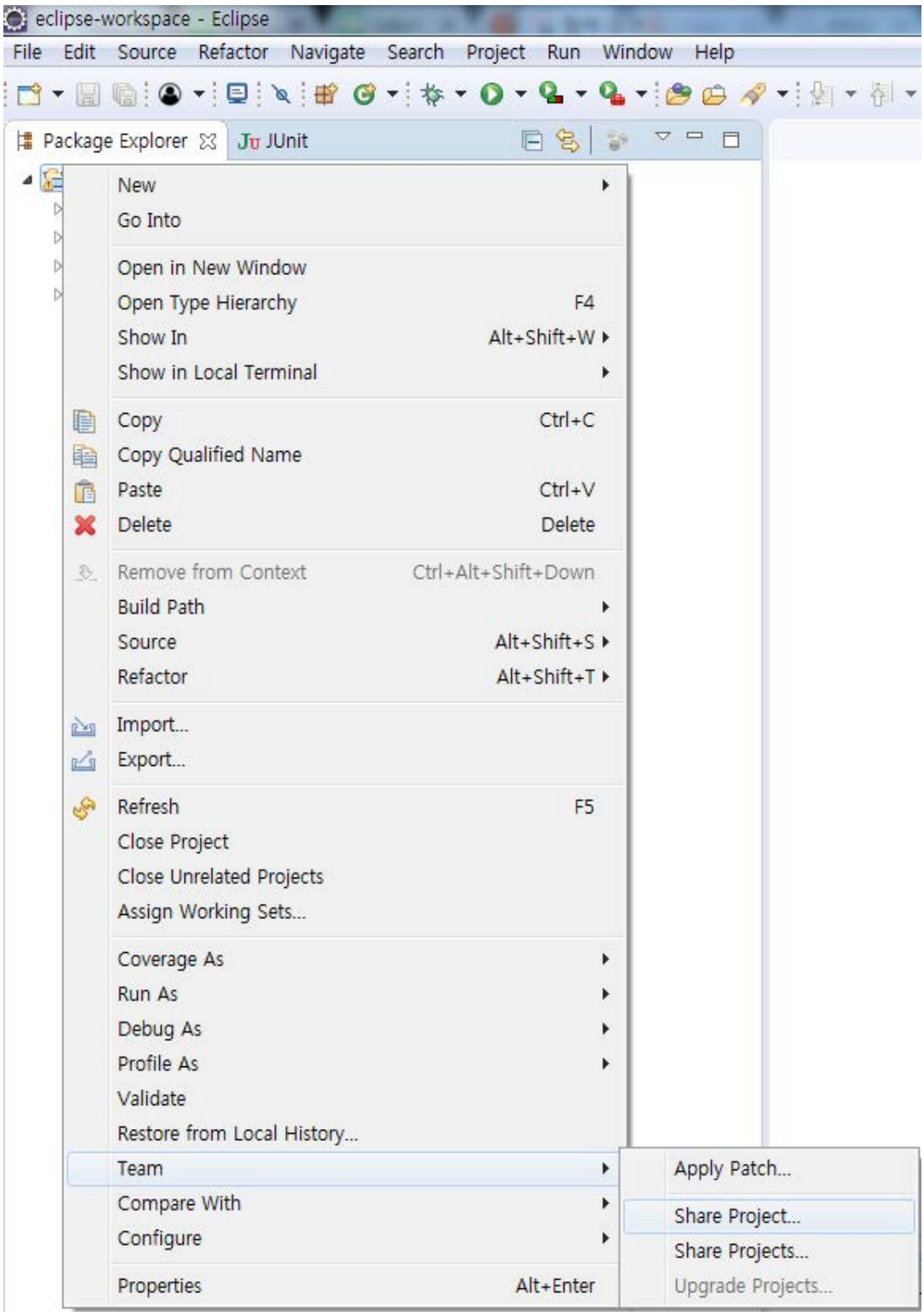


③ SVN Repository 연결

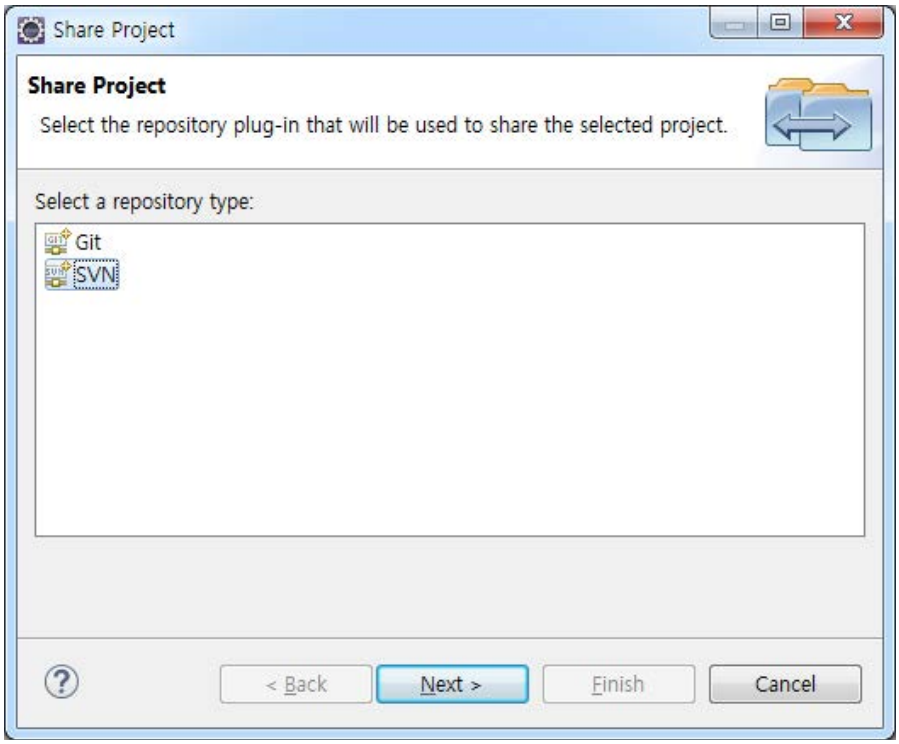
기존에 생성한 Project와 SVN 연결을 통하여 형상관리를 진행할 수 있다.

Visual SVN 에서 생성한 Repository와 Eclipse 로 생성한 Project를 연결해 보자.

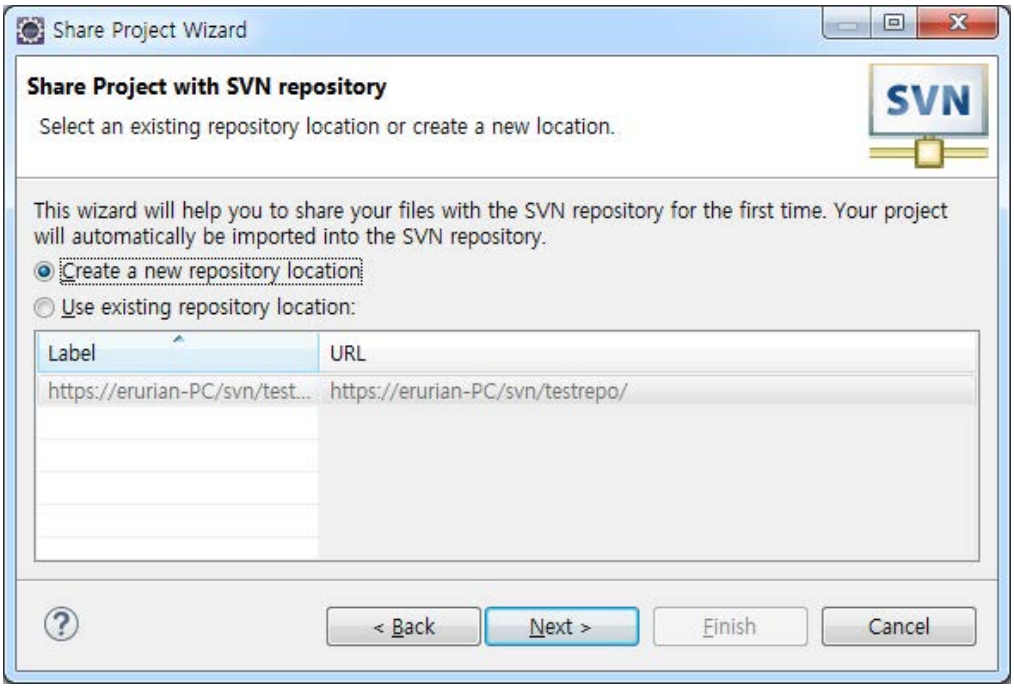
생성한 프로젝트 선택 > 마우스 우클릭 > Team > Share Project 선택



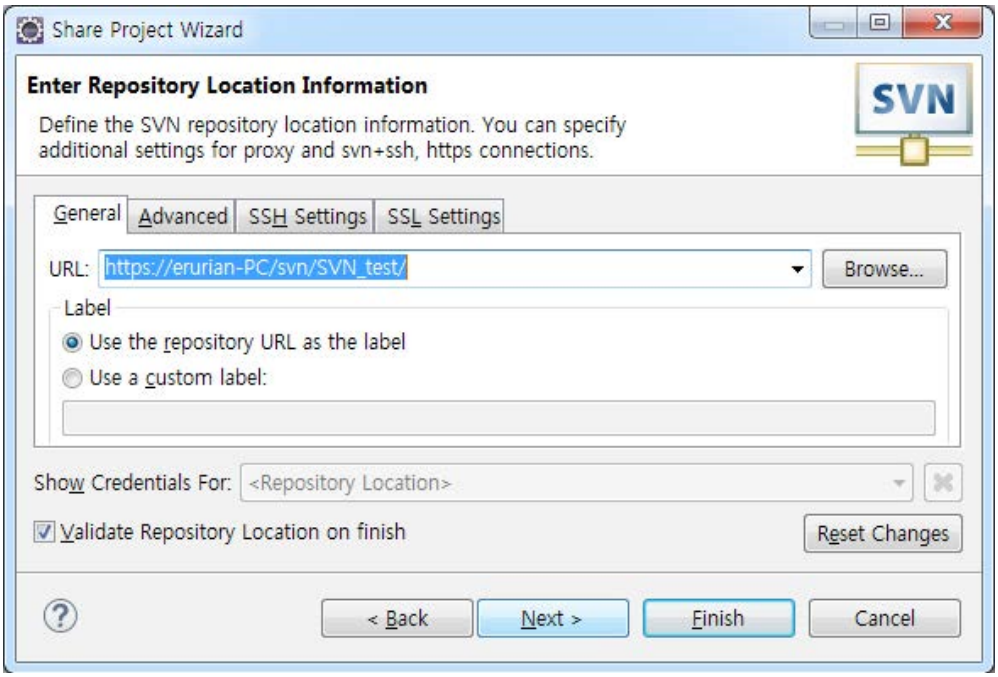
Share Project 팝업창에서 SVN 선택 > Next 버튼 클릭



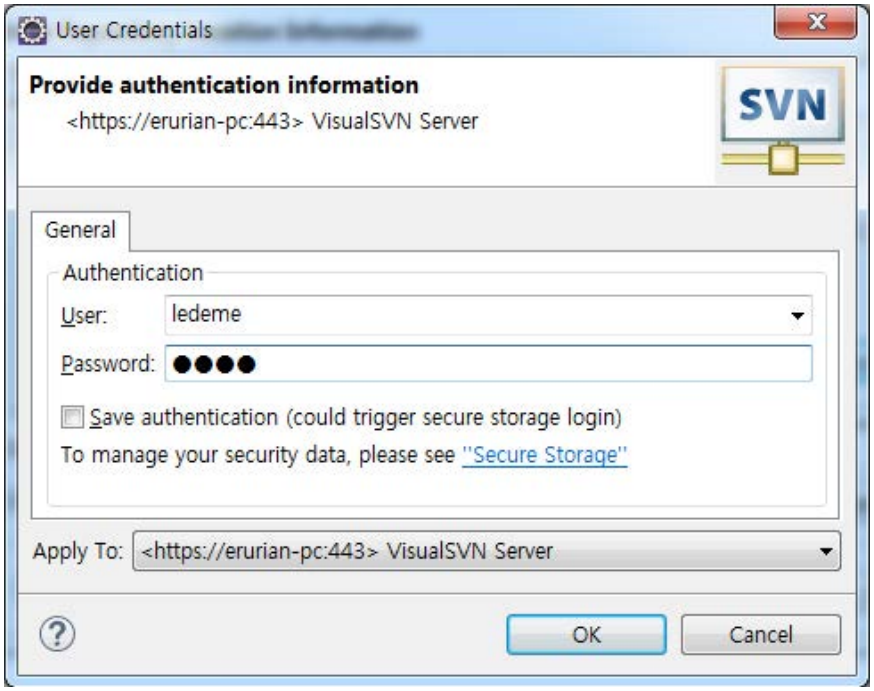
Create a new repository location 버튼 선택 > Next 버튼 클릭



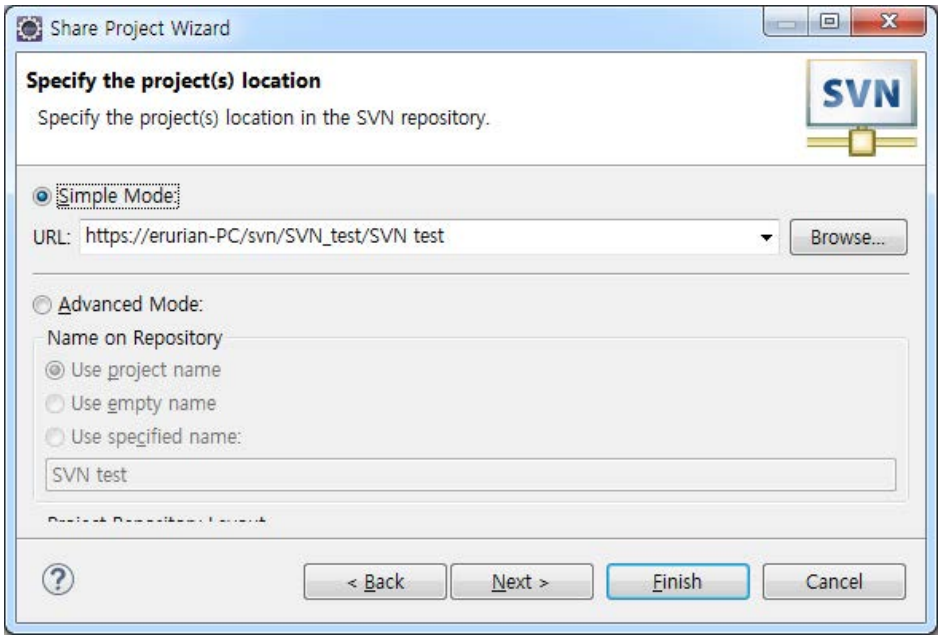
Visual SVN 에서 복사한 URL 주소 입력 후 Next 버튼 클릭



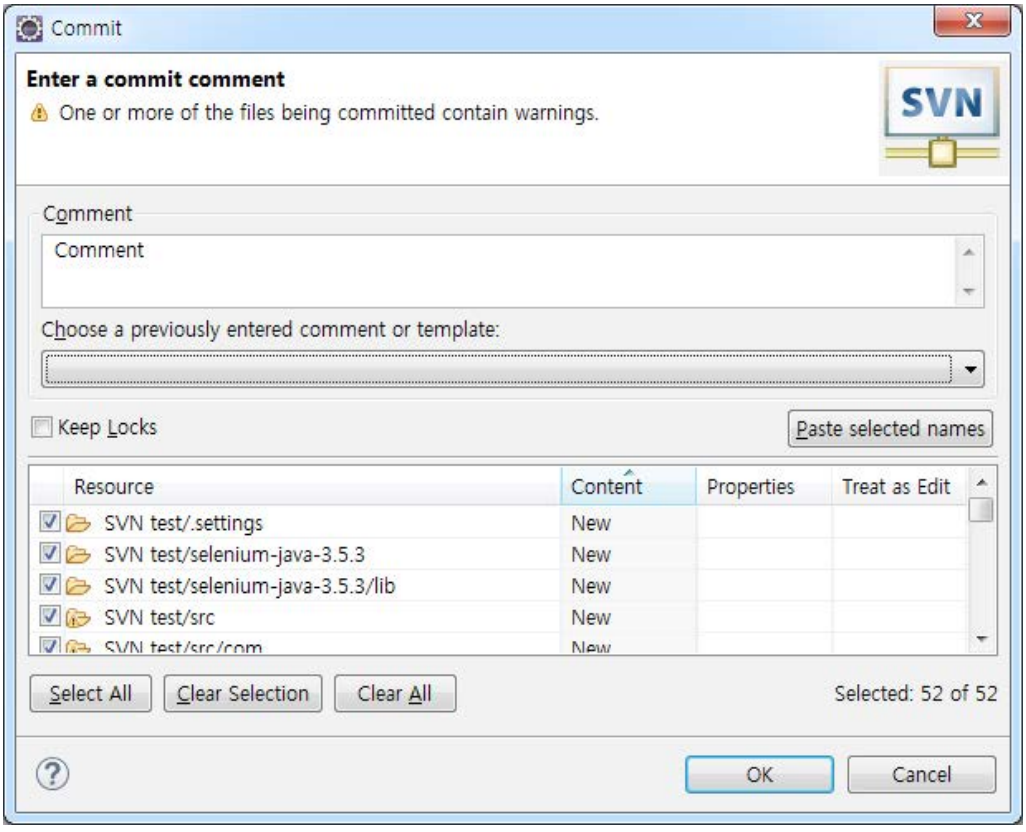
User / Password 입력 후 OK 버튼 클릭



Finish 버튼 클릭



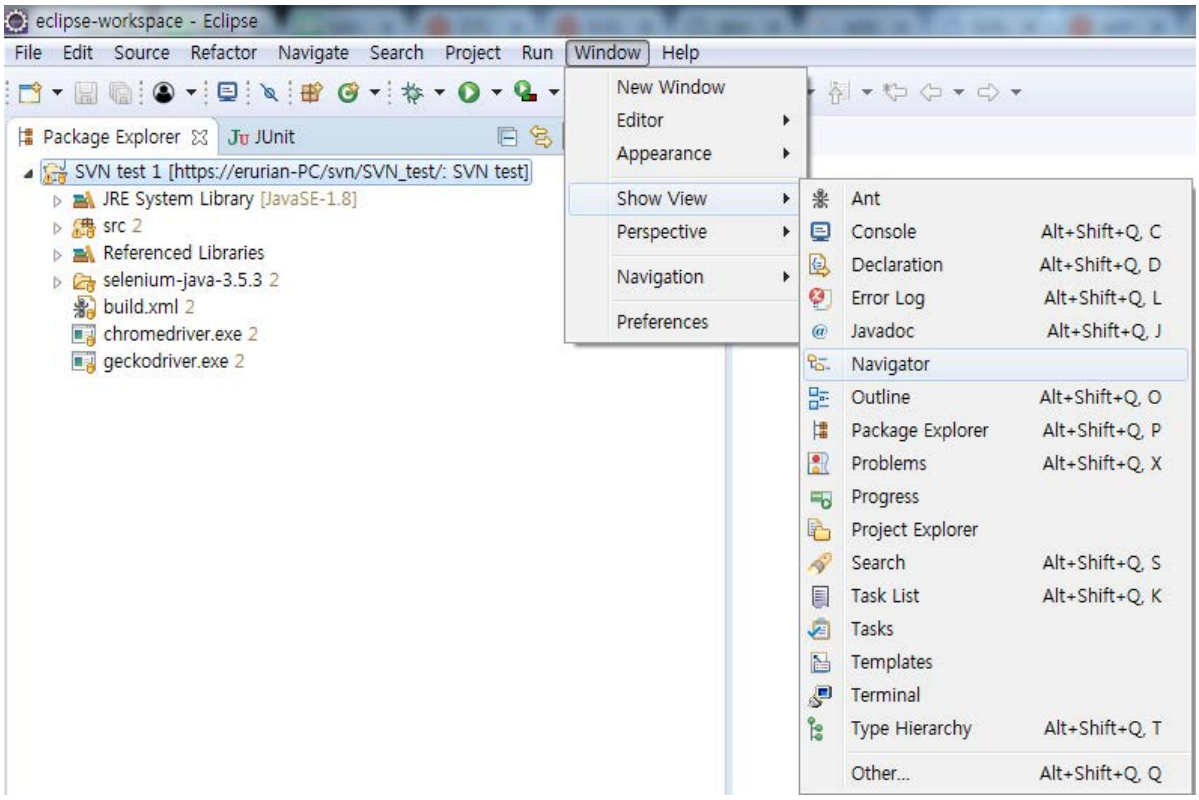
Repository와 연동이 끝나면 연결된 Project와 Repository의 동기화를 위해 Commit 한다.
아래 팝업창이 발생하면 Comment 입력 > OK 버튼 클릭 하면 Repository에 반영된다.



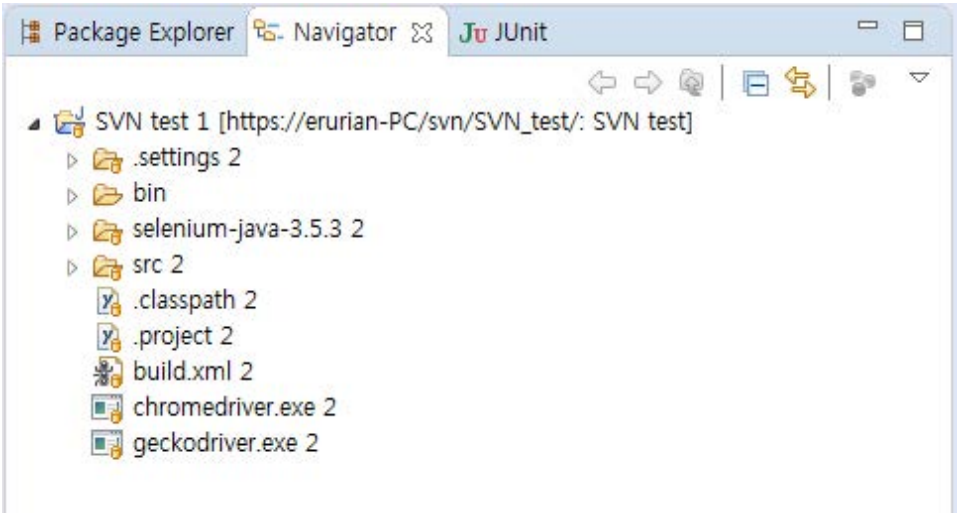
추가로 Eclipse 의 Package Explorer 에서는 bin 폴더가 보이지 않아 Repository에 업데이트 되지 않는다.

bin 폴더를 Repository에 적용하기 위해 아래와 같이 적용한다.

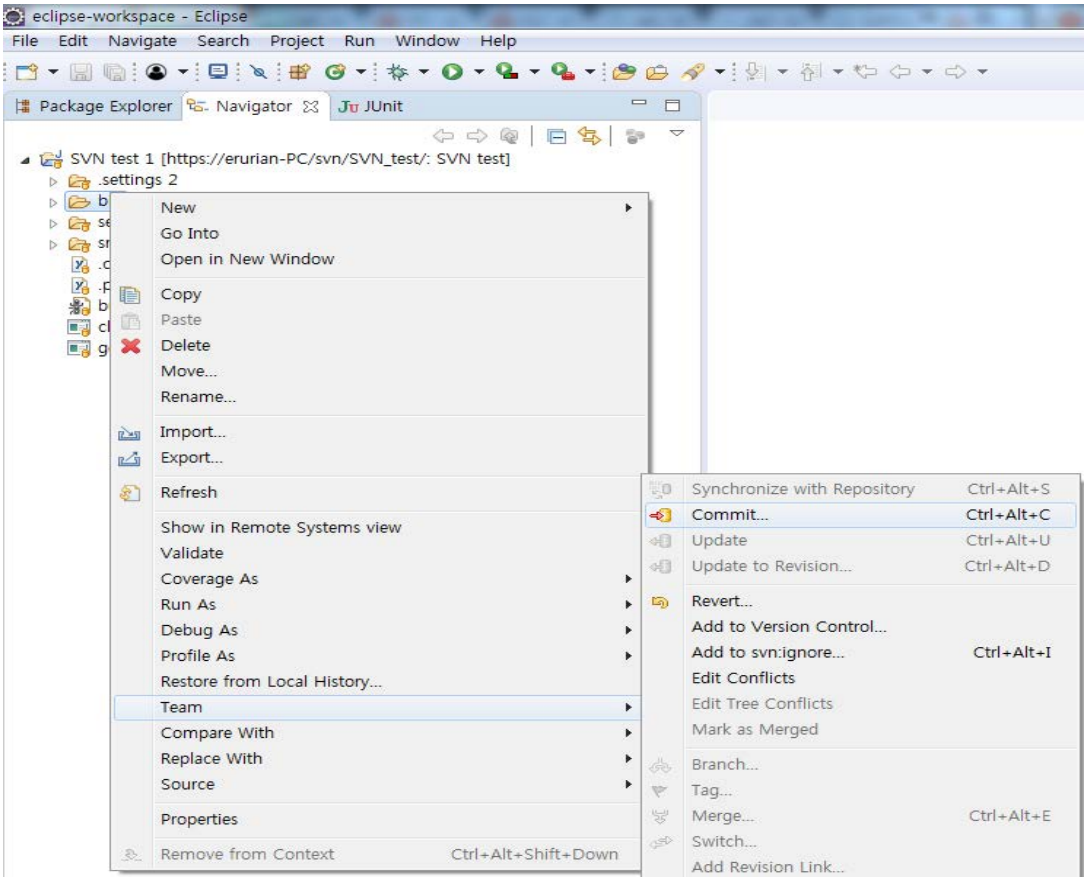
Eclipse Menu > Window > Show View > Navigator 클릭



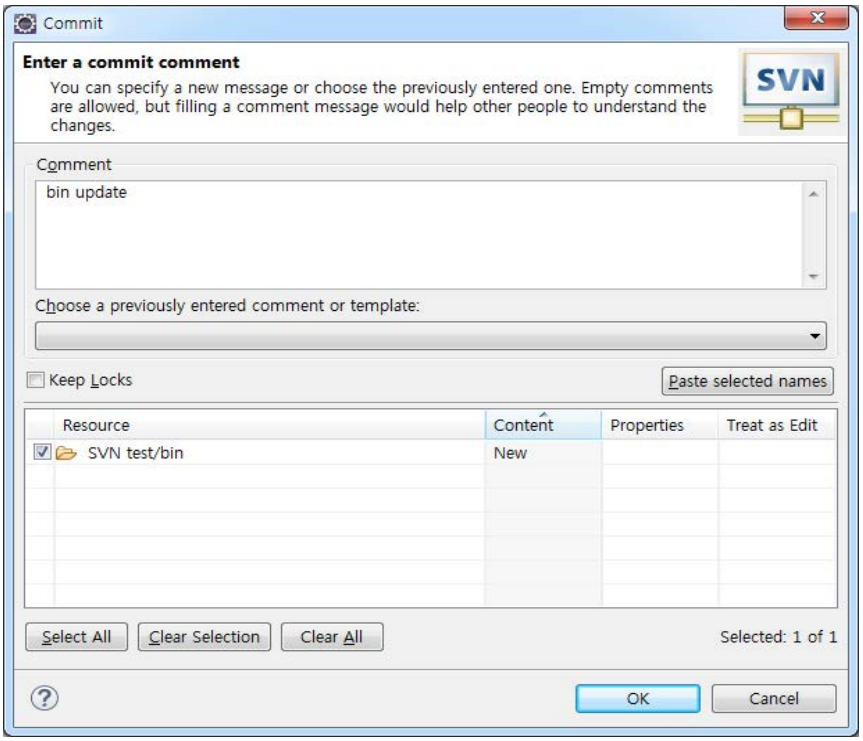
위 Navigator 가 활성화 되면 아래와 같이 bin 폴더가 보인다.



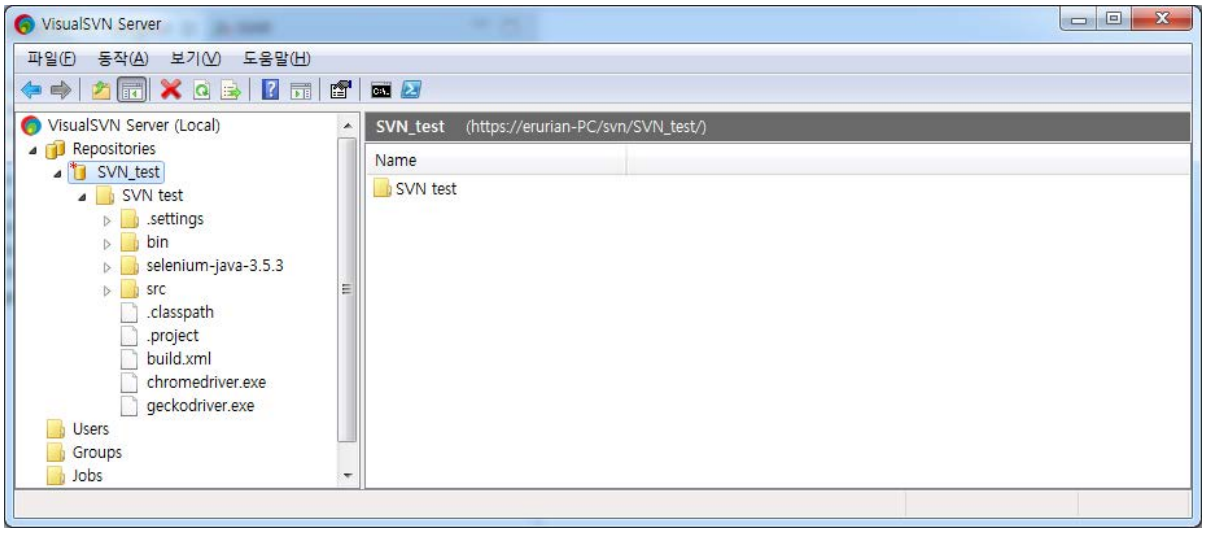
bin 폴더 마우스 우클릭 > Team > Commit 버튼 클릭 하여 bin 폴더의 class 파일도 Repository 에 업데이트 하자



Comment 입력 후 OK 버튼 클릭하여 Repository에 반영



반영이 완료되면 Visual SVN Server에 반영된 내용을 확인한다

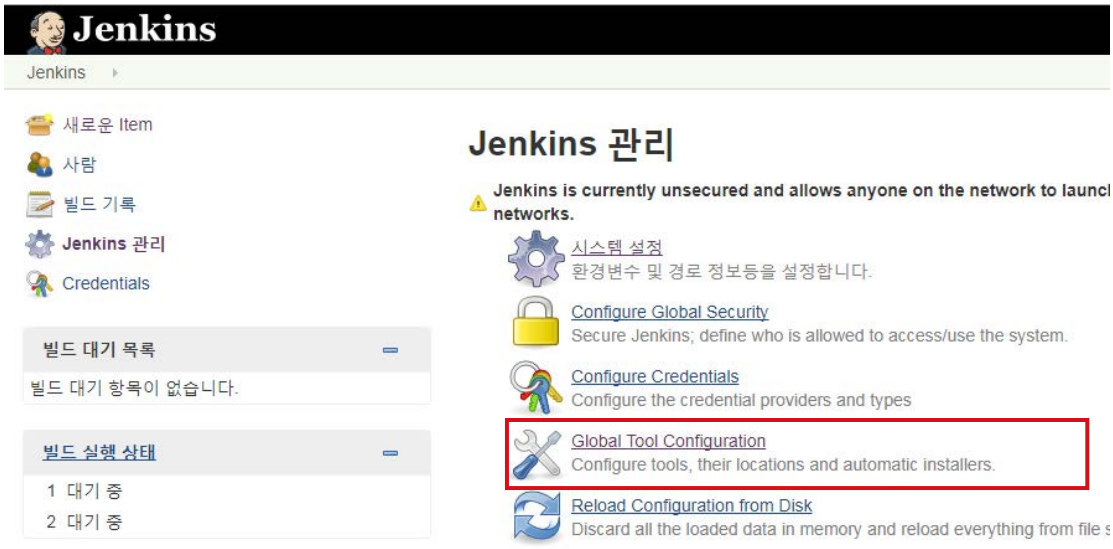


4) Jenkins

Jenkins 서비스 실행 후 http://localhost:8080 으로 접속한다.

접속 한 후에는 JDK 및 Ant 에 대한 기본 설정을 추가한다.

Jenkins 메인화면 에서 Jenkins 관리 > Global Tool Configuration 버튼을 클릭한다.



■ JDK

Name은 아무거나 사용해도 상관 없으나. 편의상 JDK 버전명 입력

JAVA_HOME 은 JDK가 설치된 실제경로로 환경변수 JAVA_HOME과 같음



■ Ant

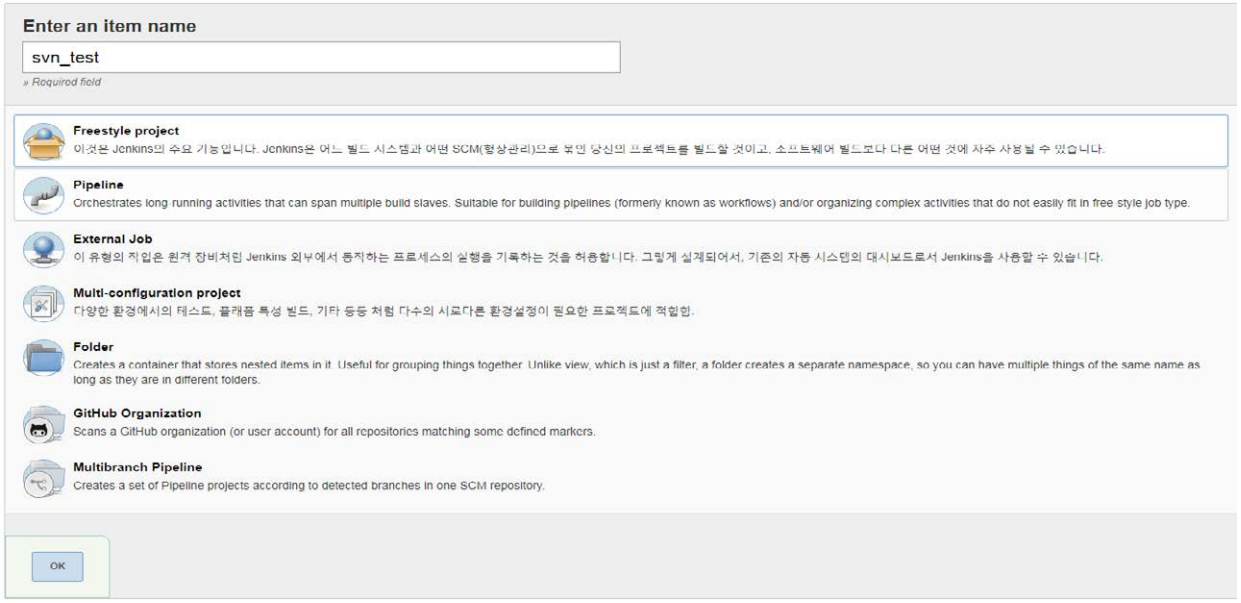
Name 역시 편의상 Apache Ant 의 버전명 입력



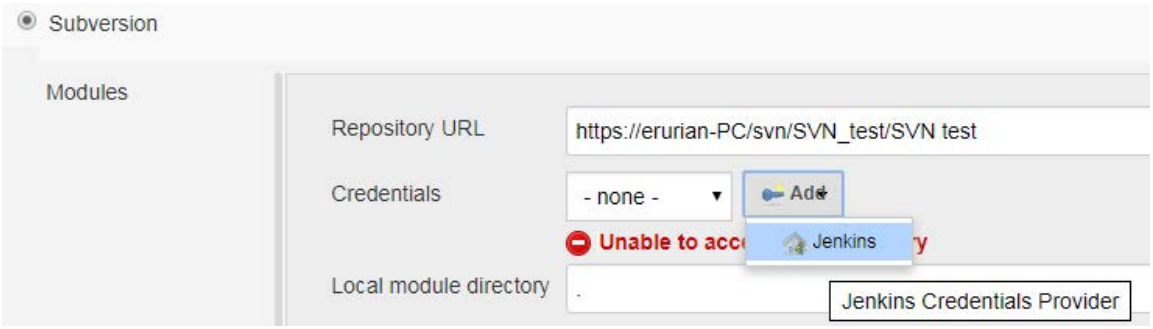
ANT_HOME 은 Ant가 설치된 실제경로로 환경변수 ANT_HOME과 같음

위 정보를 입력 후 하단의 SAVE 버튼을 클릭하여 저장 한다.

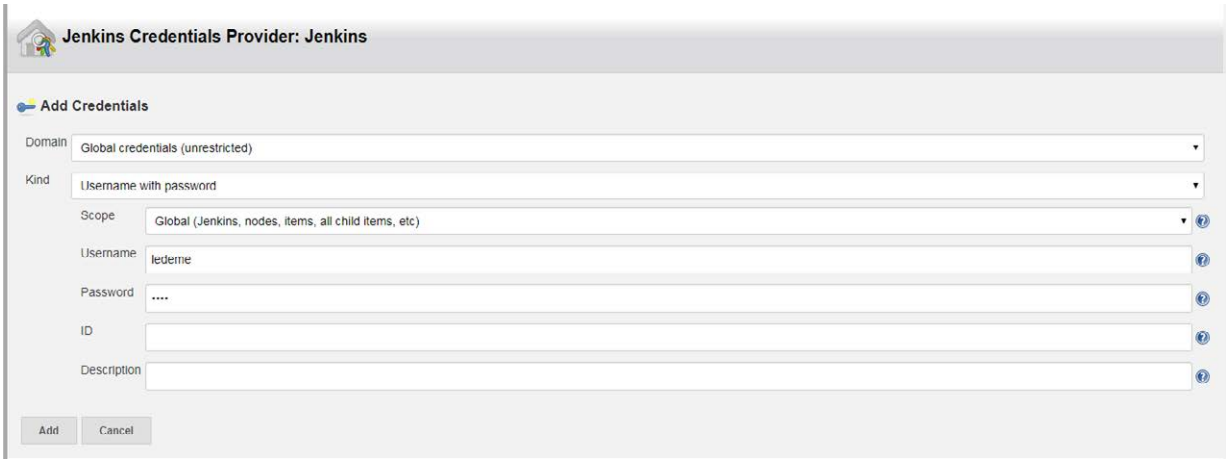
저장한 이후에는 새로운 Item 버튼 클릭 > Item name 입력 > Freestyle project 선택 > OK 버튼을 클릭하여 새로운 Job 생성



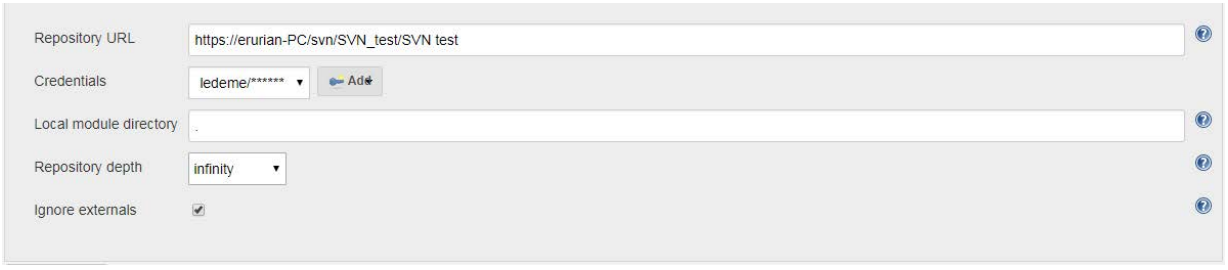
소스 코드 관리에서 Subversion 선택 > Repository URL 입력 (Visual SVN 에서 복사) > Add 버튼 클릭 > Jenkins 버튼 클릭



위 버튼을 클릭하면 Credentials를 추가할 수 있다.
생성된 Jenkins Credentials Provider:Jenkins 팝업에서 SVN의 Username / Password를 입력한 후 Add 버튼을 클릭하자

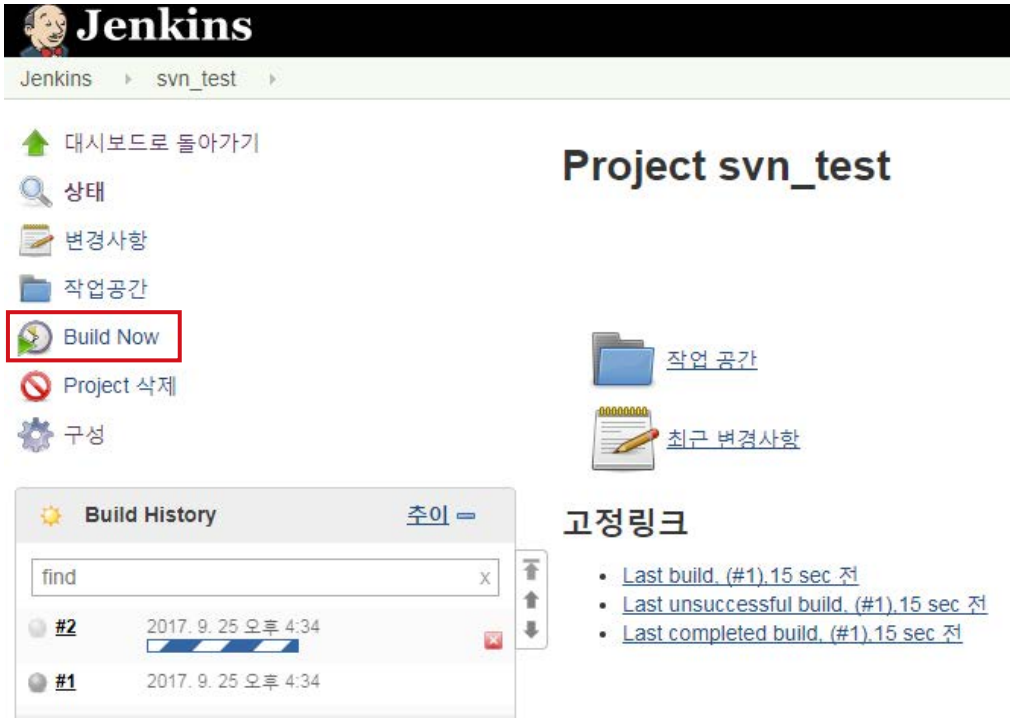


Credentials 에서 None 으로 표시된 Select Box 클릭 후 조금 전 추가해 보자

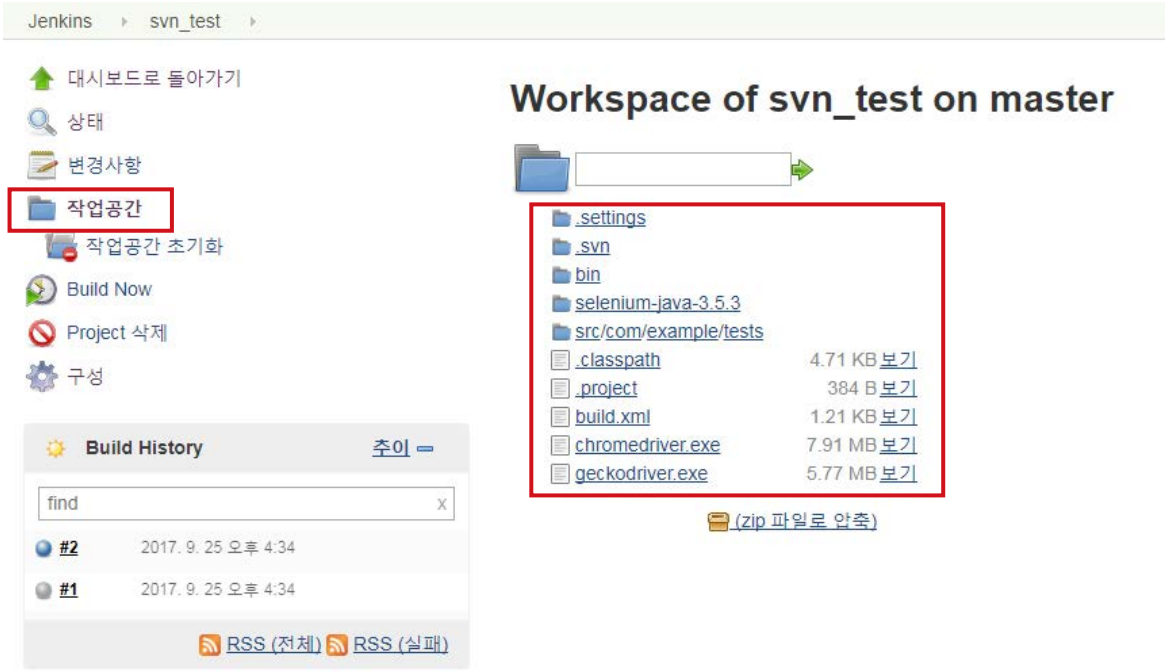


SVN Repository에서 연결된 Username / Password 가 만족할 경우 Credentials 하위에 표시되던 Unable to access the repository 가 해제된 것을 확인 할 수 있다.
위 설정까지 완료 후 하단의 저장 버튼을 클릭 하여 SVN Repository에서 정상적으로 저장된 내용을 받아오는지 확인해 보자

설정을 완료한 Job을 클릭한 뒤 Build Now 버튼을 클릭하여 빌드 진행해 보자

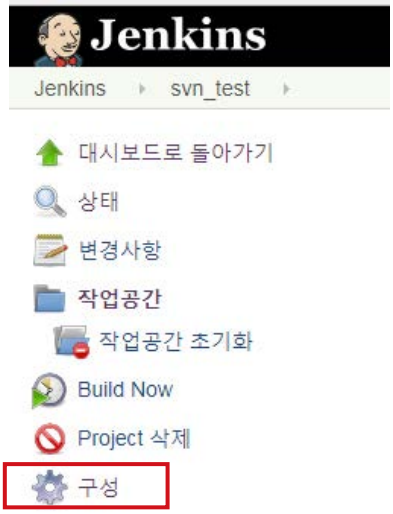


빌드가 완료되면 작업공간을 클릭하여 정상적으로 Repository 정보를 받아왔는지 확인해 보자

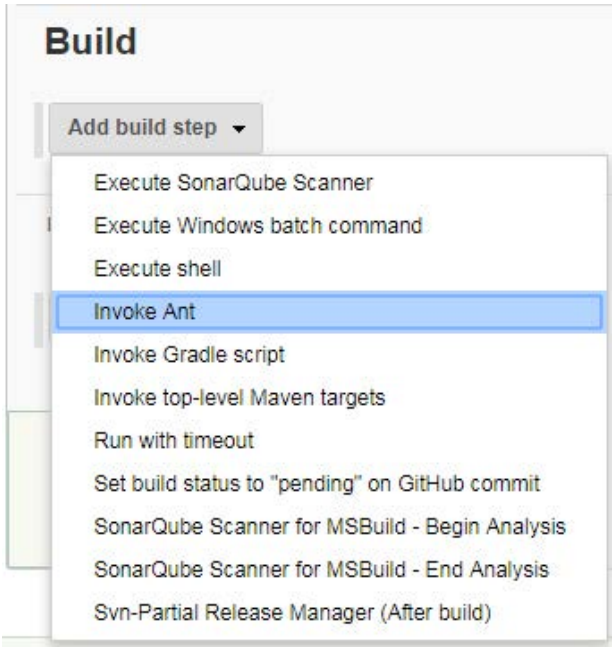


Repository의 모든 정보가 정상적으로 받아진 것을 확인하였다.

다음으로 Apache Ant로 Build를 진행해 보자. 성된 Job에서 구성 버튼을 클릭해 보자



하단의 Build 항목에서 Invoke Ant 를 선택한다.

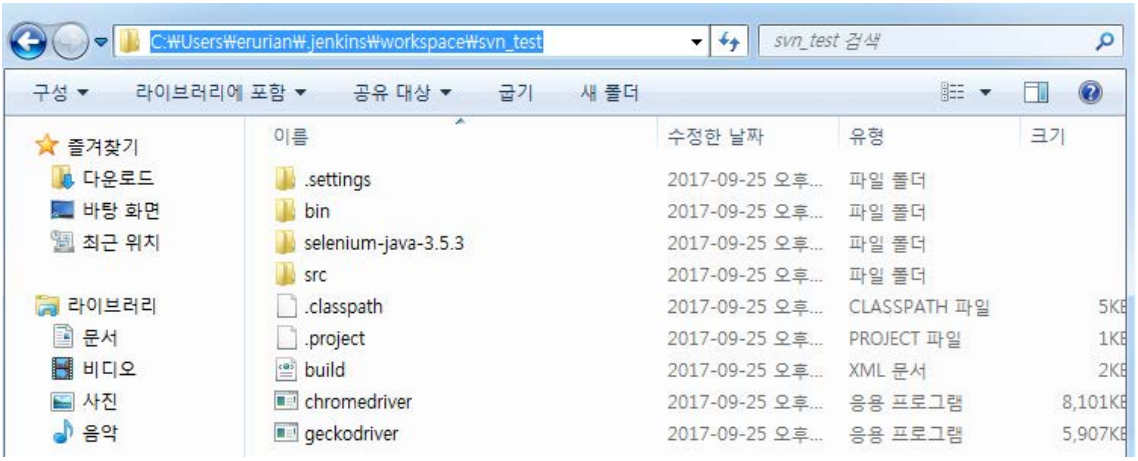


Invoke Ant 입력이 활성화 되면 하단의 고급 버튼 클릭 후 다음과 같이 입력한다.
Ant Version 클릭 > 기존에 입력한 ant-1.10.1 선택 > Build File 선택

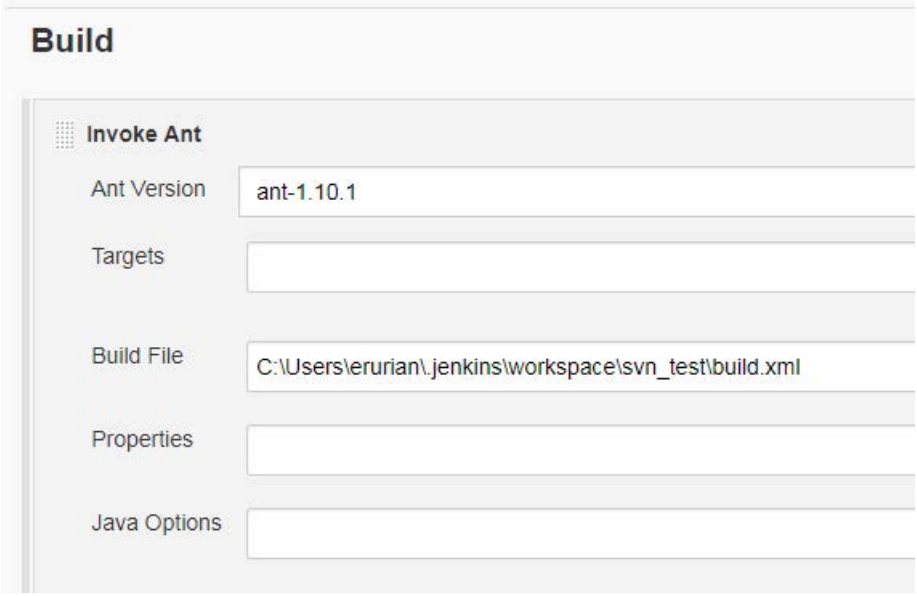


Build File은 조금전에 SVN Repository에서 받은 build.xml 파일을 확인하면 된다.

Jenkins가 설치된 폴더로 이동 후 해당 Build.xml 파일이 저장된 위치까지 입력해 주자.
특별히 Jenkins 설치경로를 지정해 주지 않았다면, C드라이브 > 사용자 > .jenkins > workspace > svn_test [Job Name] 에 Repository의 내용이 저장되어 있다.
해당 경로까지 모두 복사 한다.

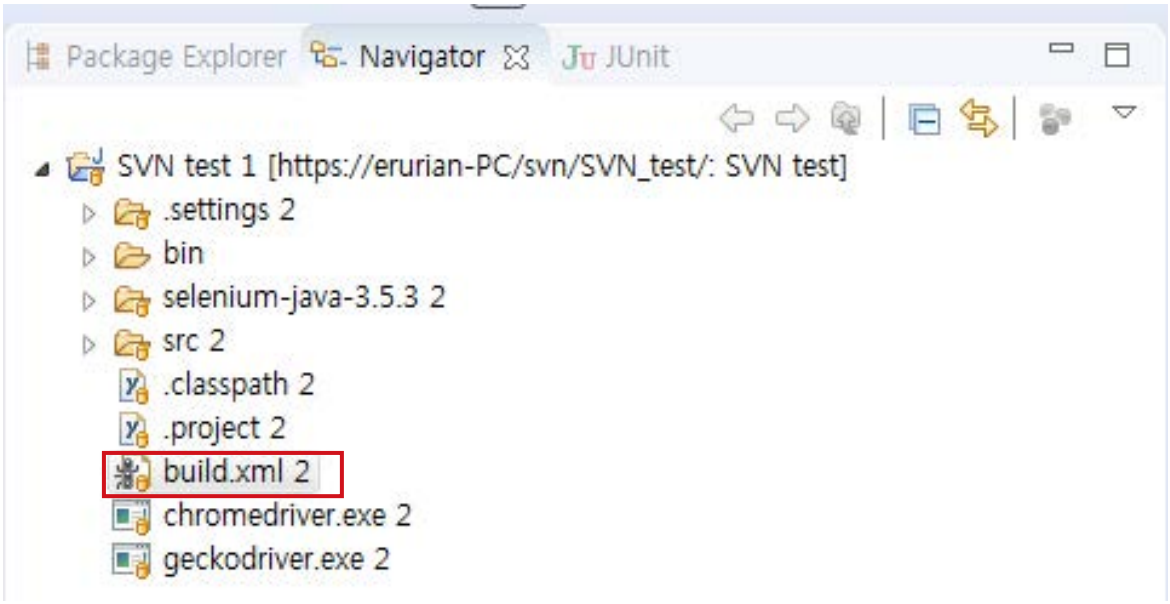


위 복사한 경로에 ₩build.xml 을 추가하여 Build File 항목에 입력한다.
Ex) C:₩Users₩erurian₩.jenkins₩workspace₩svn_test₩build.xml



위 정보 입력 후 저장 버튼을 클릭한다.

이제 build.xml 에 일부 경로를 변경 해 주어야 한다.
다시 Eclipse로 돌아와 추가해 줬단 build.xml 파일을 확인한다. (더블 클릭)



build.xml 파일에서 아래 정보를 실제 Jenkins Job 경로로 변경해 준다.

〈기존〉

```
...
〈property name="test.report.dir" location="" /〉
...
〈fileset dir=""〉
...
```

〈변경후〉

```
...
〈property name="test.report.dir"
location="C:₩Users₩erurian₩.jenkins₩workspace₩svn_test₩testrepot" /〉
...
〈fileset dir="selenium-java-3.5.3"〉
...
```

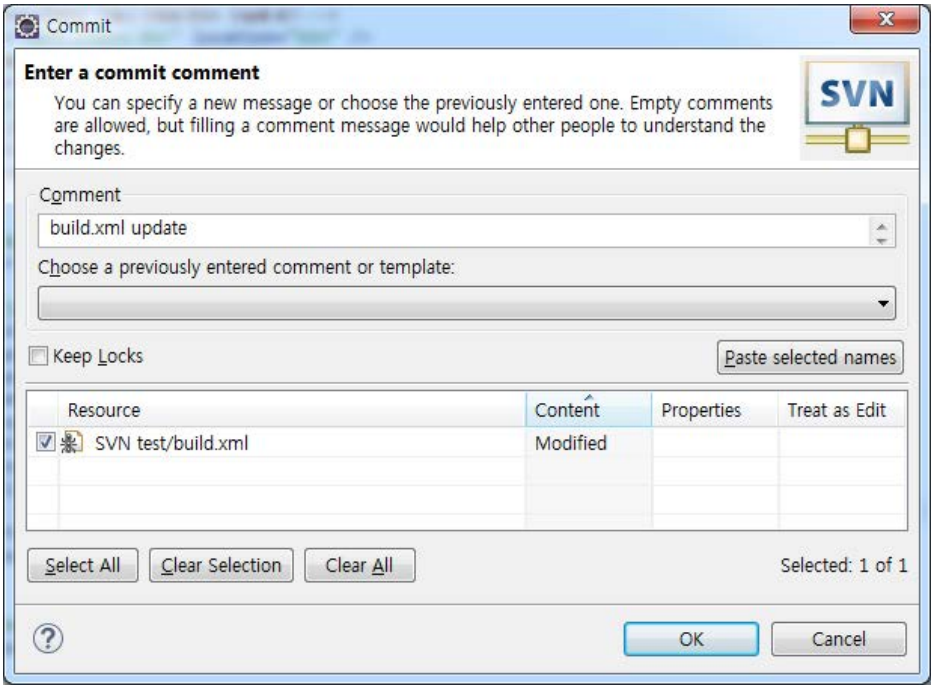
property name 의 location은 Junit 테스트 후 결과를 xml 파일로 저장되는데 그 파일이 저장되는 위치를 지정해 준다.
위치는 해당 Jenkins Job이 위치한 workspace 하위로 입력해 주면 된다.
workspace 마지막에 testreport 폴더를 추가하면 다음과 같다.
[Jenkins Job workspace 경로] + [testreport(xml 파일이 저장될 임의의 폴더)]
Ex) C:₩Users₩erurian₩.jenkins₩workspace₩svn_test₩testreport

fileset dir 에는 workspace에 위치한 Library 위치를 알려주는 것으로 Library 의 폴더명을 입력해 주면 된다.

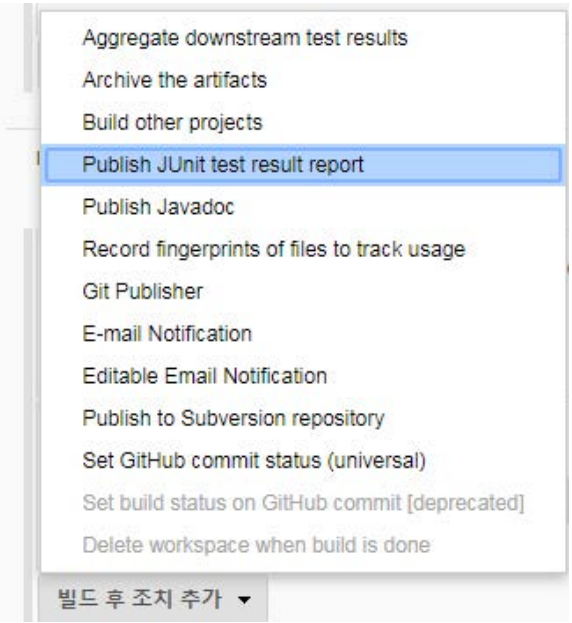
완성된 build.xml 은 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="JUNIT" default="main" basedir=".">
        <!-- workspace에서 class 파일이 저장된 bin 파일을 찾기 -->
        <property name="test.class.dir" location="bin" />
        <!-- Junit 으로 빌드된 결과를 xml 파일로 저장할 위치 -->
        <property name="test.report.dir"
location="C:₩Users₩erurian₩.jenkins₩workspace₩svn_test₩testreport" />
        <path id="junit.class.path">
        <!-- 빌드에 사용될 Library -->
                <fileset dir="selenium-java-3.5.3">
                        <include name="**/*.jar"/>
                </fileset>
        </path>
        <target name="makedir">
                <mkdir dir="${test.report.dir}" />
        </target>
        <target name="junit" depends="makedir">
                <junit printsummary="on" fork="true" haltonfailure="no"
failureproperty="test.failed">
                        <classpath refid="junit.class.path" />
                        <classpath>
                                <pathelement location="${test.class.dir}"/>
                        </classpath>
                        <formatter type="xml" />
                        <batchtest todir="${test.report.dir}">
                                <fileset dir="${test.class.dir}">
                                        <include name="**/**.class" />
                                </fileset>
                        </batchtest>
                </junit>
        </target>
        <target name="main" depends="junit">
                <description>Main target</description>
        </target>
</project>
```

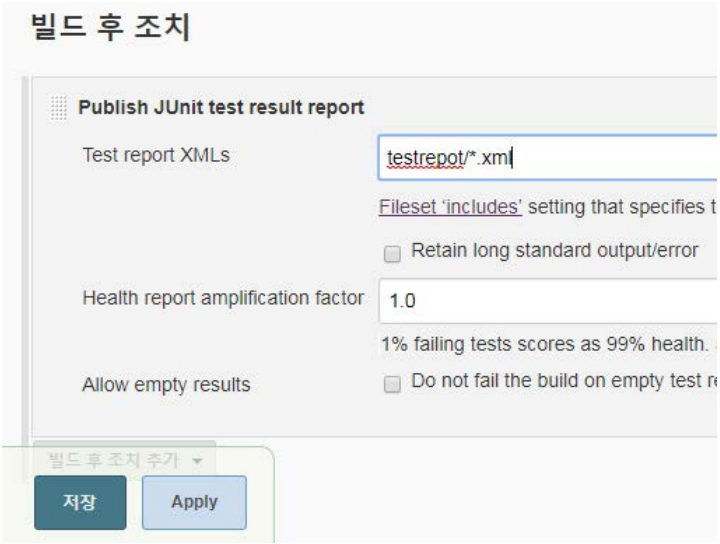
수정이 완료 되면 SVN Repository 에 Commit 하여 build.xml 파일의 내용을 갱신한다.



마지막으로 Jenkins > Job > 구성 버튼 클릭 > 빌드 후 조치 > 빌드 후 조치 추가 버튼 클릭 > Publish Junit test result report 클릭



Test report XMLs 에 testreport/*.xml 입력 후 저장 버튼 클릭한다.



위 환경이 완료 된 이후 해당 Jenkins 의 Job에서 Build Now 를 클릭하면 Eclipse 에서 Junit으로 테스트와 동일하게 Test 후 Test Result 에 결과를 표시해 준다.



5) Jenkins 문제해결

작업 수행시 빌드가 정상적으로 이뤄지지 않는 경우가 있다.

- Java 파일 Build 후 생성되는 class 파일이 Repository에 반영되지 않은경우
  ⇨ 에서 Repository로 다시 Commit 해주면 해결된다.
- build.xml 의 경로가 잘못 지정된 경우
  ⇨ testreport 쪽 경로를 Jenkins Job 의 workspace 경로 안에 포함시키면 해결된다.

6) SVN 대신 Git 으로 연결하는 경우

SVN 으로 구성 할 때와 큰 차이점은 없다. 형상관리 프로그램을 SVN을 쓰느냐, Git 을 쓰느냐의 차이다.
Git 으로 구성할 경우 로컬 PC의 Repository를 구성할 수도 있고, Git hub, Bitbucket 등 Repository와 연동하여 사용할 수도 있다.
그 구성된 환경을 Jenkins와 연동하여 사용하는 방법의 차이만 있을 뿐 SVN 과 큰 차이없이 환경구성이 가능하다.

방법은 아래와 같다.

1) Eclipse 에서 코드 생성 환경구성 (Library 추가, build.xml 추가, webdriver 추가)
2) Eclpise 에서 Git Plug in 설치 (최신버전의 Eclipse 에는 기본으로 설치되어 있음)
3) Repository 생성 (로컬 PC or Git hub, bitbucket등)
4) Eclipse 와 Repository 연결
5) Jenkins Plugin 설치
6) Jenkins와 Repository 연결
7) build.xml 수정
8) Jenkins에서 Build Now 버튼으로 정상 동작 확인

13.3 Tutorial 3 : Jenkins(Tutorial 1), Findbugs

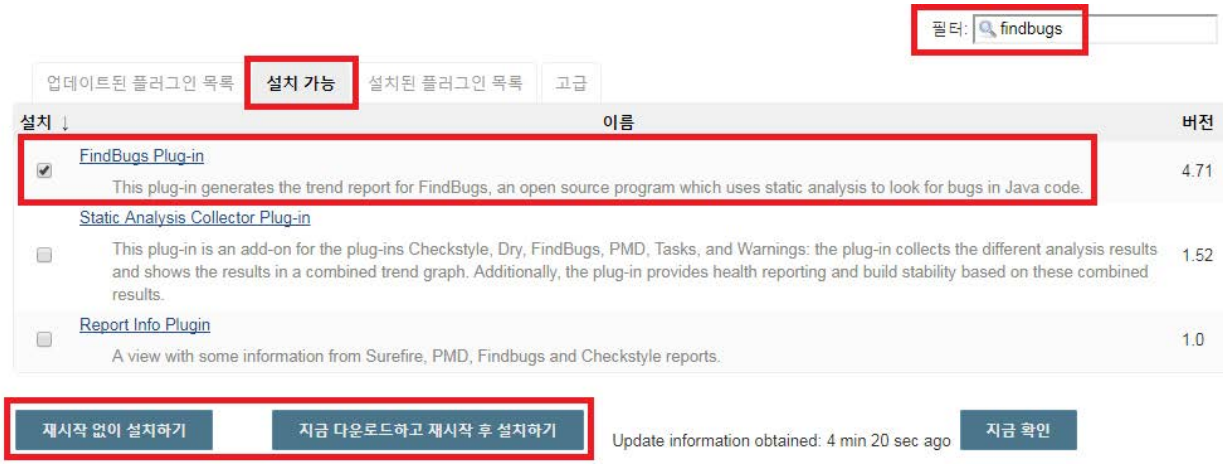
Tutorial 1에서 Maven 프로젝트를 구성하고 해당 소스를 SVN으로 형상관리를 할 수 있도록 하며, Nexus로 배포할 수 있도록 Jenkins로 CI를 구성하였다.
Tutorial 3은 Tutorial 1에서 구성한 환경을 기준으로 설명을 진행할 것이므로 원활한 진행을 위해서는 Tutorial 1이 선행되어야 한다.
본 가이드에서 제안하는 Toolchain의 기본적인 구성(Tutorial 1)에서 Findbugs를 Jenkins에서 어떻게 구성하고 사용 하는지를 알아 볼 것이다.

1) Jenkins에서 정적분석 도구 사용하기(Findbugs)

Findbugs는 Java 정적분석 도구이다. Jenkins에서 Findbugs를 직접적으로 사용하는 것은 아니고 Maven에서 사용한다고 볼 수 있다. Jenkins에서는 플러그인을 통하여 Findbugs에서 실행한 결과를 확인할 수 있는 리포트를 제공한다.
이를 사용하기 위해서 Jenkins에서 Findbugs 플러그인을 설치한다.
플러그인 설치를 위해서 Jenkins 메인 화면에서 Jenkins 관리 > 플러그인관리 메뉴로 이동한다.



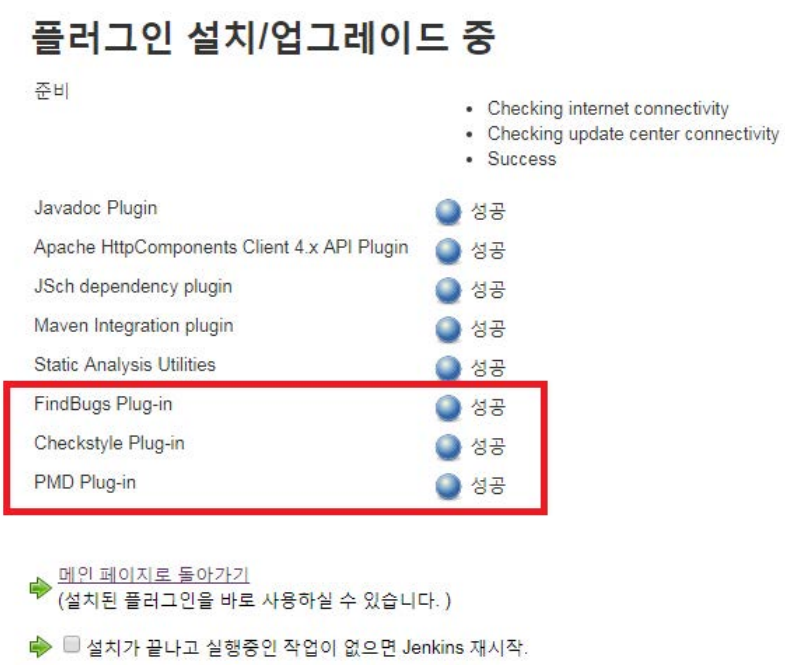
플러그인 관리 화면에서 설치가능 탭을 선택하고 상단의 필터 창에 findbugs 를 입력하여 검색한다. 검색 결과 목록에 'Findbugs Plug-in'을 체크하고 하단의 '재시작 없이 설치하기'를 클릭한다.



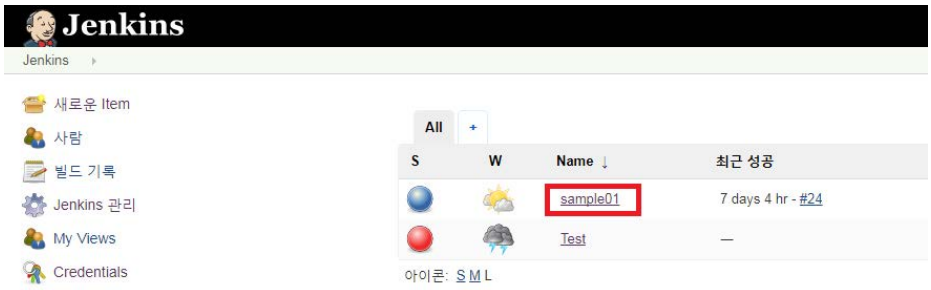
Findbugs외에 여러 언어를 지원하는 PMD, CheckStyle 등의 정적분석 도구를 사용할 수도 있다. Findbugs 플러그인을 설치하는 것 처험 해당 도구의 플럭그인을 아래와 같이 검색하여 설치하여 사용할 수 있다.



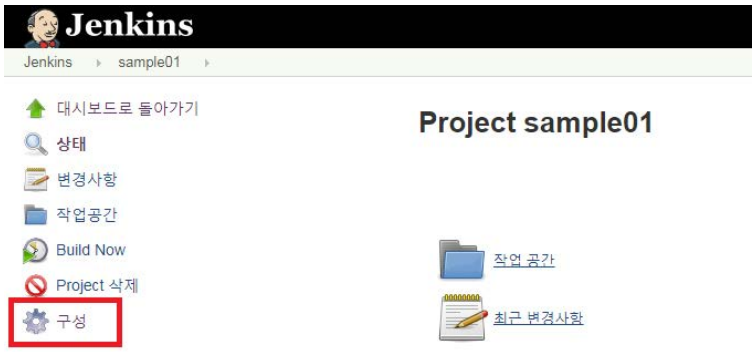
플러그인을 선택하여 설치를 진행하면 아래와 같이 플러그인이 성공적으로 설치되는 것을 확인할 수 있다.



이제 프로젝트에서 사용하기 위하여 프로젝트 설정을 해주어야 한다.
Jenkins 메인화면에서 프로젝트를 선택한다.



프로젝트 메인 화면 좌측에 구성을 클릭한다.

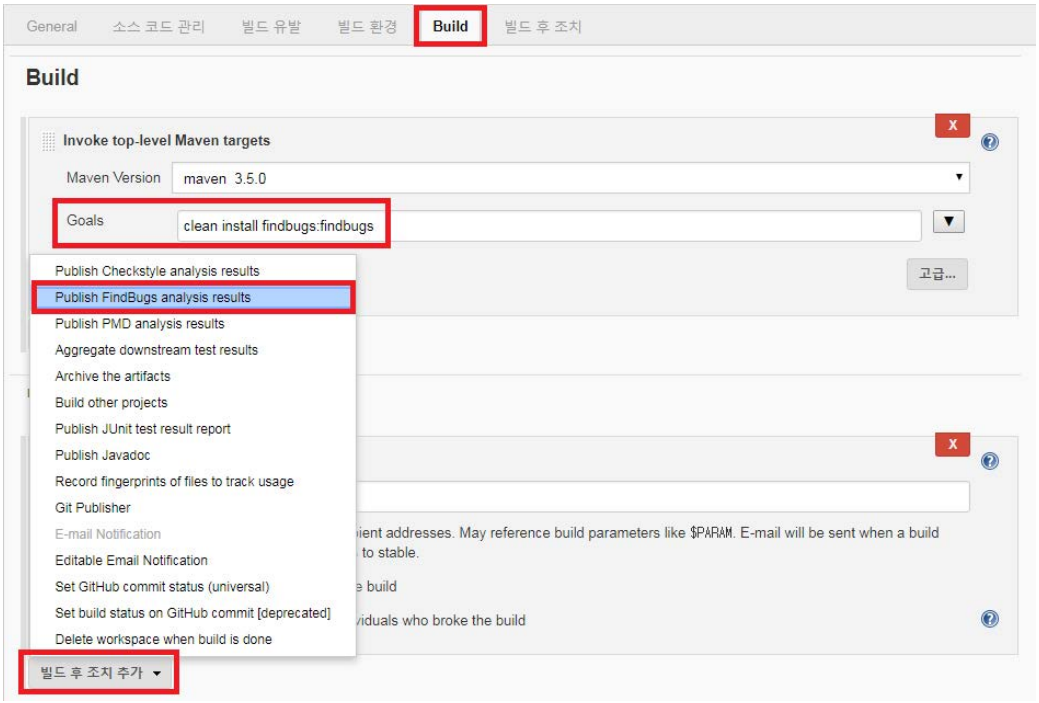


프로젝트 구성(설정)화면에서 Build를 선택한다.
해당 프로젝트는 Tutorial 1에서 구성하였던 것이라 Maven 설정이 되어있을 것이다.

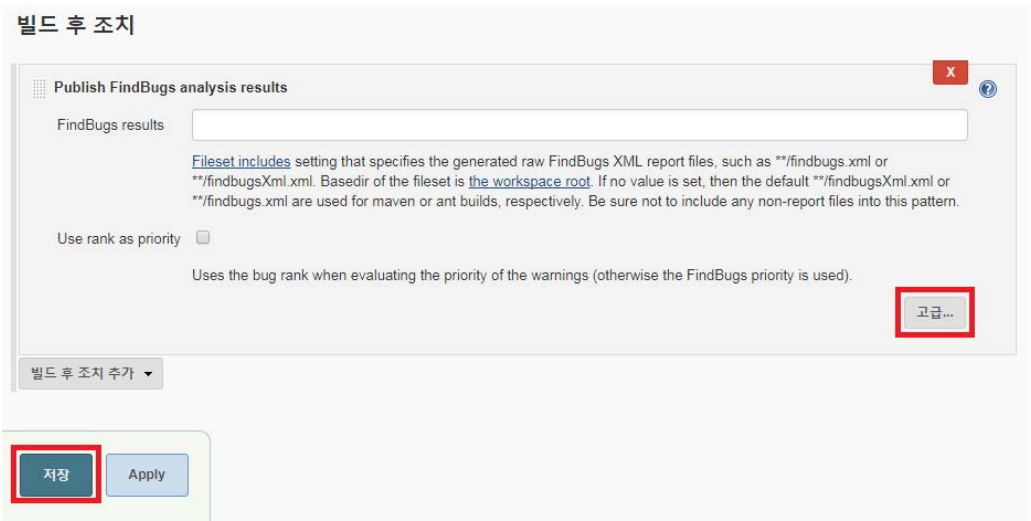
Maven 설정의 Goals 입력항목에 아래와 같이 입력을 한다.

```
clean install findbugs:findbugs
```

다음으로 '빌드 후 조치 추가'를 선택하면 'Publish Findbugs analysis results' 항목이 보이며 해당 항목을 선택한다. Findbugs 플러그인이 정상적으로 설치되지 않았다면 해당 항목이 보이지 않을 것이다.



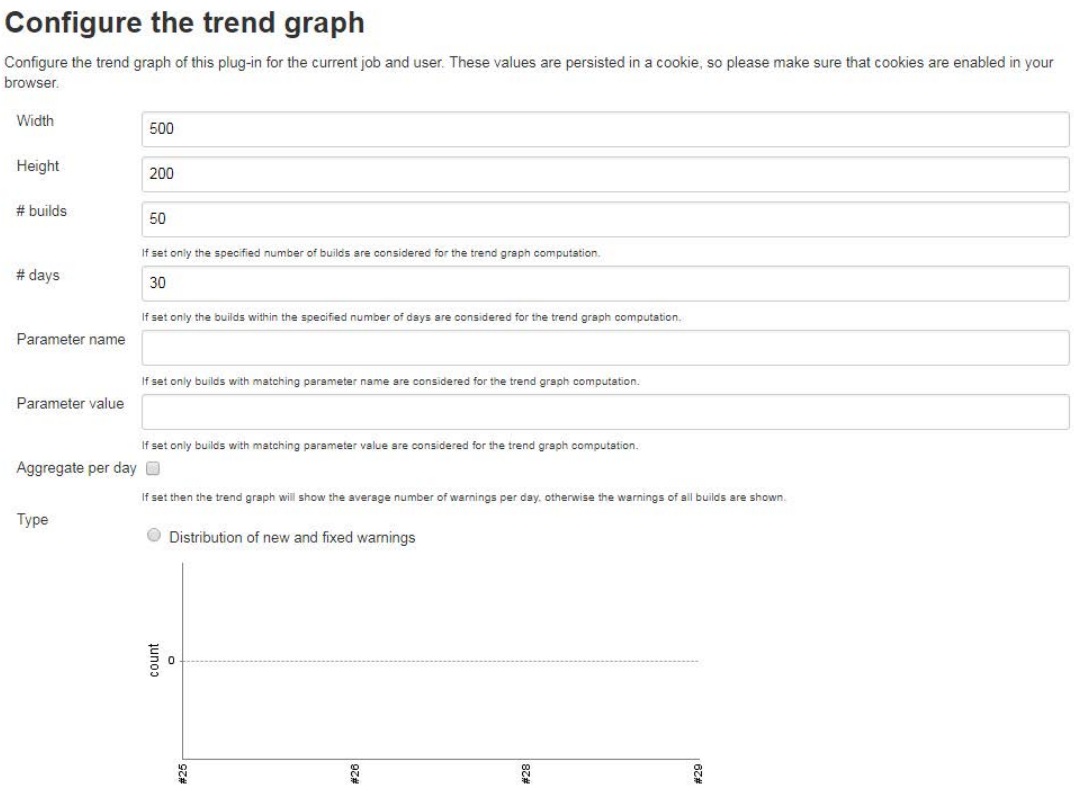
빌드 후 조치로 추가가 되면 아래와 같이 추가된 것을 확인할 수 있다. 고급 버튼을 선택하면 추가적으로 설정할 수 있는 항목들이 보인다. 필요한 설정이 있다면 설정을 하면 된다.
설정을 하였으면 아래의 저장 버튼을 클릭한다.



저장 버튼을 클릭하면 프로젝트 메인화면이 보이며 좌측의 'Build Now' 메뉴를 클릭하면 빌드가 시작되고 끝나면 프로젝트 메인화면에서 Findbugs Trend라는 리포트를 확인할 수 있다. 빌드가 끝나고 리포트가 자동으로 갱신이 되지 않으므로 새로 고침을 해주어야 한다.



해당 리포트 아래에 Configure를 클릭하여 그래프의 크기, 종류 등 다양한 리포트에 대한 설정을 할 수 있다.



#참고 : PMD, CheckStyle 사용하기

앞서 같이 설치한 PMD, CheckStyle 플러그인을 사용하는 방법은 Findbugs와 동일하게 사용할 수 있다. Maven 설정의 Goals 입력항목에 아래와 같이 입력하였던 것을 기억할 것이다.

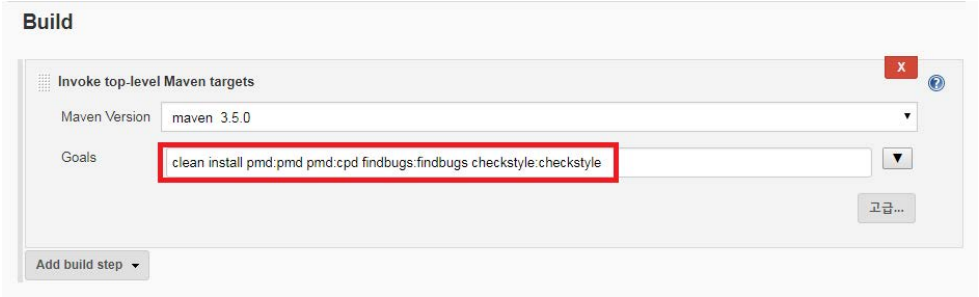
```
clean install findbugs:findbugs
```

여기에 Findbugs와 마찬가지로 PMD와 CheckStyle 을 추가해 주고, 빌두 후 조치도 추가를 해주면 된다.
PMD 와 CheckStyle 의 Goals는 아래와 같이 추가하면 된다.

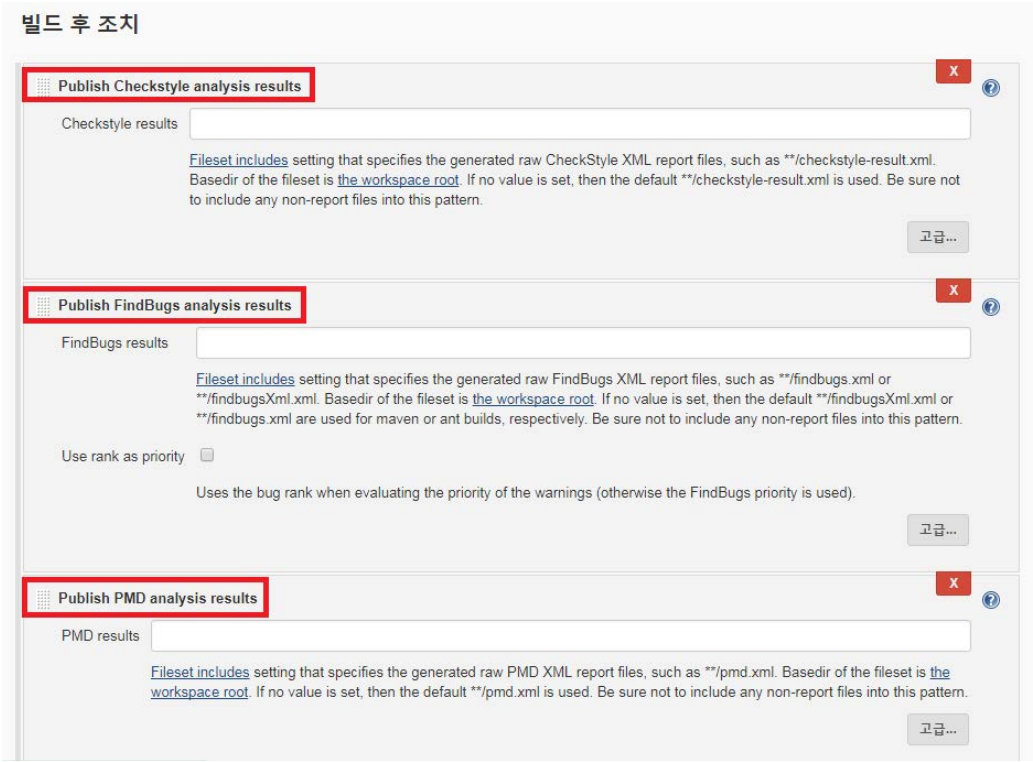
```
clean install pmd:pmd pmd:cpd checkstyle:checkstyle
```

Findbugs를 포함하여 설정하려면 아래와 같이 입력하면 된다.

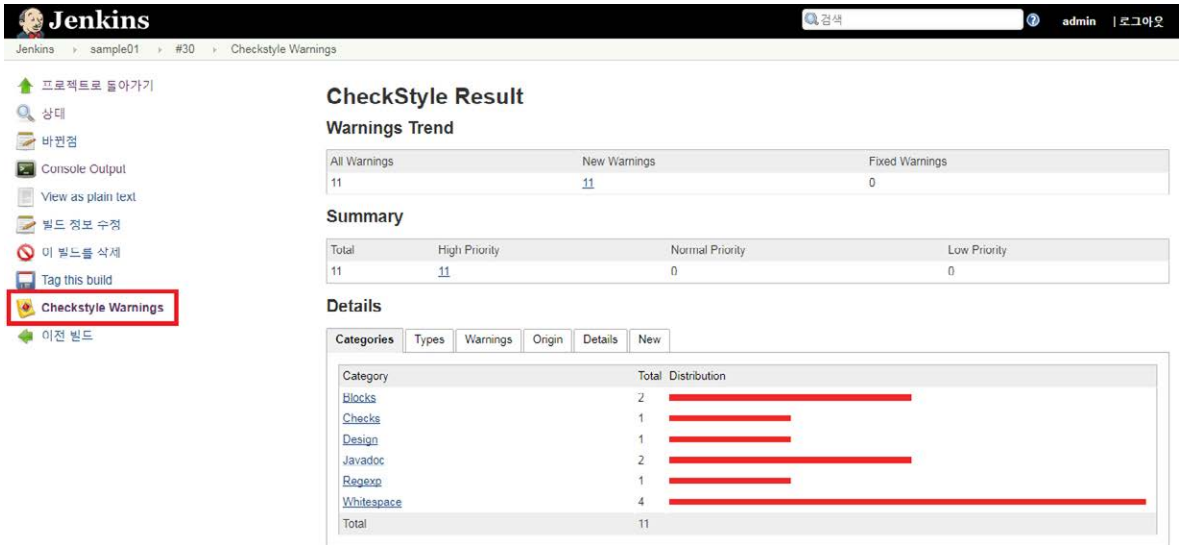
```
clean install pmd:pmd pmd:cpd findbugs:findbugs checkstyle:checkstyle
```



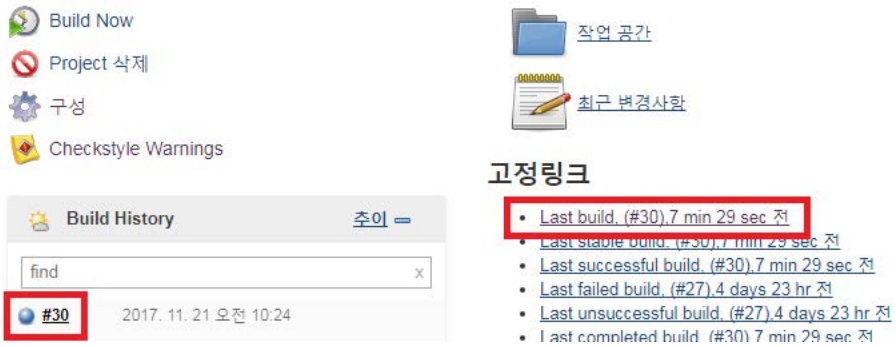
빌드 후 조치에서 추가하면 Jenkins에서 결과를 확인할 수 있다.
각 도구의 결과 파일 저장 정보를 입력하는 항목이 있으며 입력하지 않으면 기본 경로의 결과 파일을 사용하게 된다. 필요한 경우 결과 파일 저장 정보를 입력한다.



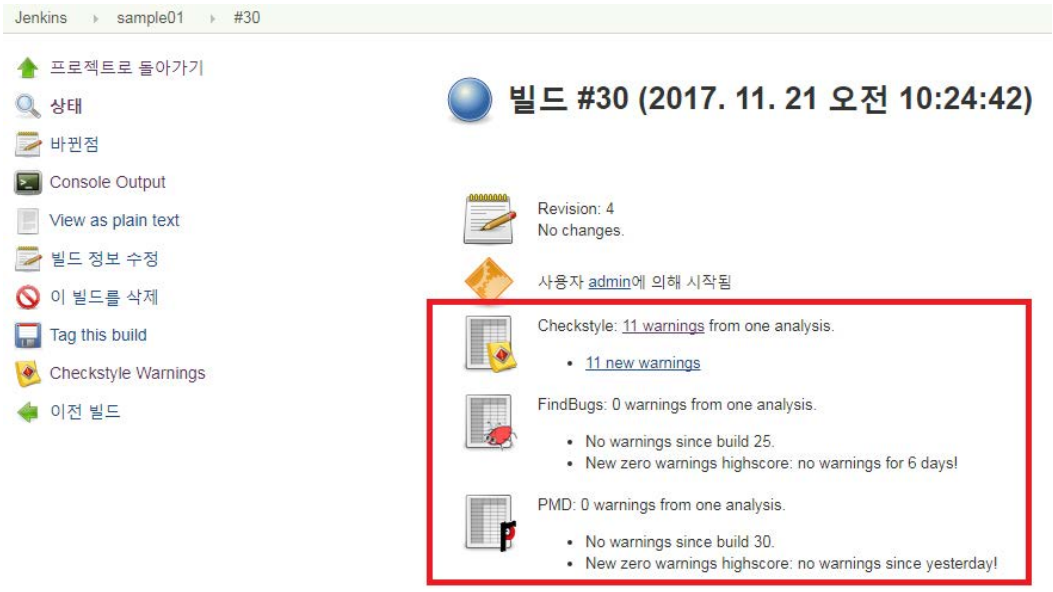
빌드를 하고 프로젝트 메인화면 좌측에 CheckStyle Warnings 메뉴를 클릭하면 아래와 같이 결과를 확인할 수 있다.



프로젝트 메인 화면에서 Build History에서 빌드 링크를 클릭하면 요약 정보를 확인할 수 있다.



아래와 같이 요약 정보를 확인 할 수 있다.

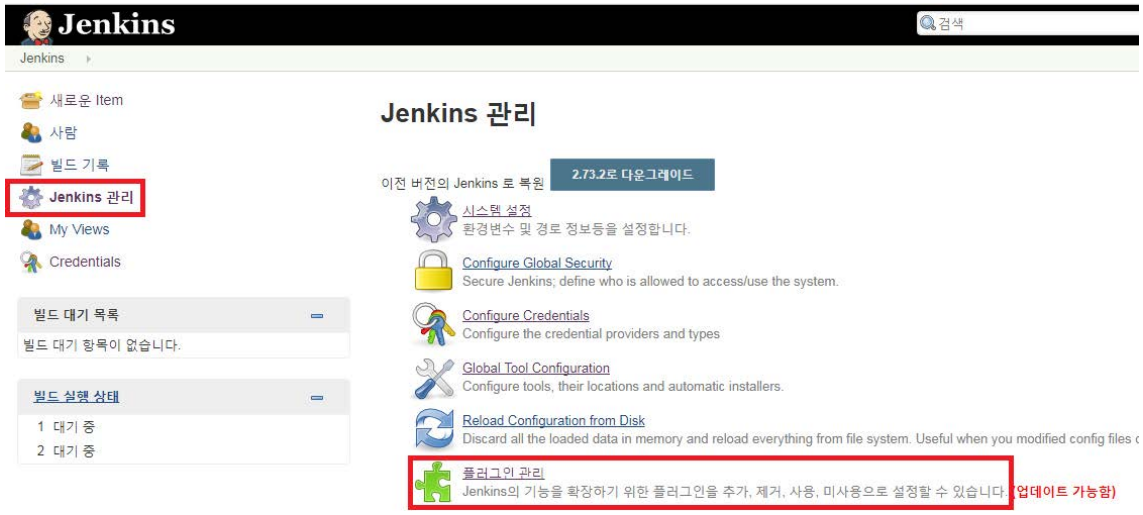


13.4 Tutorial 4 : Jenkins, Fitnesse

Tutorial 1에서 Maven 프로젝트를 구성하고 해당 소스를 SVN으로 형상관리를 할 수 있도록 하며, Nexus로 배포할 수 있도록 Jenkins로 CI를 구성하였다.
Tutorial 4은 Tutorial 1에서 구성한 도구 및 설정들이 모두 필요한 것은 아니지만 Tutorial 1에서 구성한 환경을 기준으로 설명을 진행할 것이므로 원활한 진행을 위해서는 Tutorial 1이 선행 하길 권장 한다.

또한 Tutorial 4에서는 Jenkins와 FitNesse와 어떻게 구성하고 사용하는지에 대한 내용만 다루어지므로 FitNesse를 구성하는 것은 본 가이드 10. FitNesse를 참고하거나 먼저 수행을 하고 진행하여야 한다.

Jenkins에서는 플러그인을 통하여 FitNesse를 실행하고 실행한 결과를 확인할 수 있는 리포트를 제공한다.
이를 사용하기 위해서 Jenkins에서 FitNesse 플러그인을 설치해야 한다.
플러그인 설치를 위해서 Jenkins 메인 화면에서 Jenkins 관리 > 플러그인관리 메뉴로 이동한다.



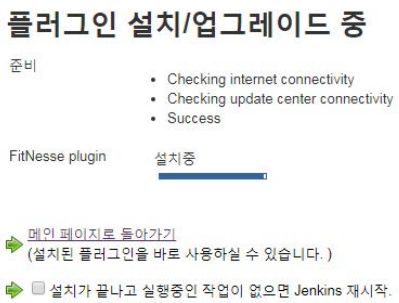
플러그인 관리 화면에서 설치가능 탭을 선택하고 상단의 필터 창에 fitnesse를 입력하여 검색한다. 검색 결과 목록에 'FitNesse Plug-in'을 체크하고 하단의 '재시작 없이 설치하기'를 클릭한다.



FitNesse 플러그인 이름의 링크를 클릭하면 설정 등의 정보를 확인 할 수 있으며, 링크는 아래와 같다.

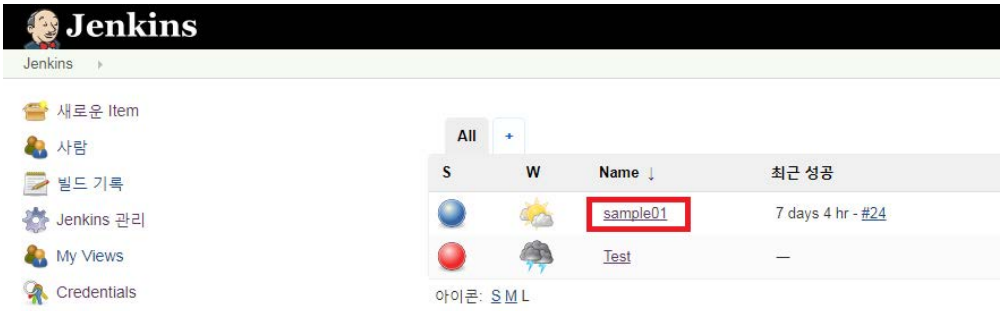
https://plugins.jenkins.io/fitnesse

설치하기를 클릭하면 아래와 같이 플러그인 설치가 진행되며 설치가 되면 설치 중이라는 메시지가 완료라고 바뀌어 알려준다.

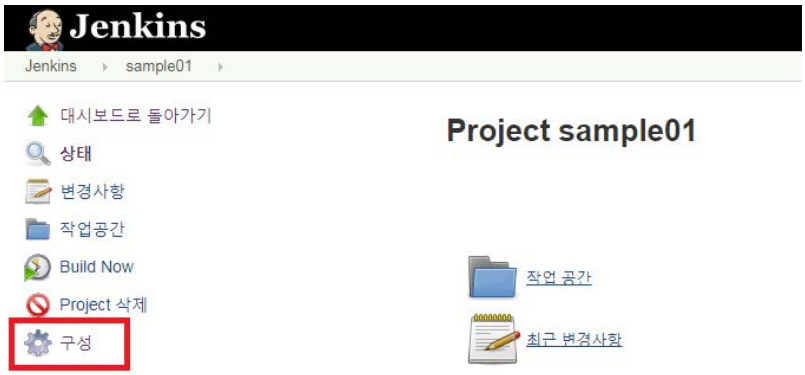


FitNesse는 프로젝트 마다 구성을 하여 사용한다. FitNesse 플러그인을 설치하면 프로젝트 구성에 FitNesse를 구성할 수 있는 항목이 추가되어있을 것이다. 이제 프로젝트에서 어떻게 FitNesse를 사용하는지 알아보겠다.

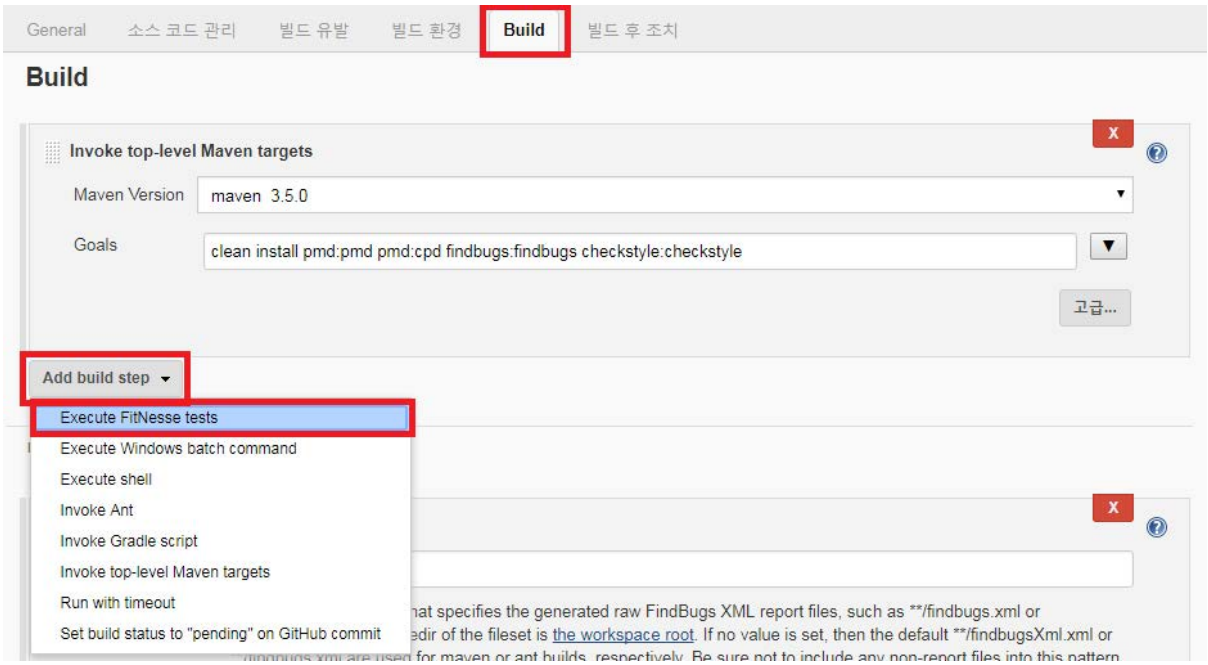
FitNesse를 사용하고자 하는 프로젝트를 선택한다.



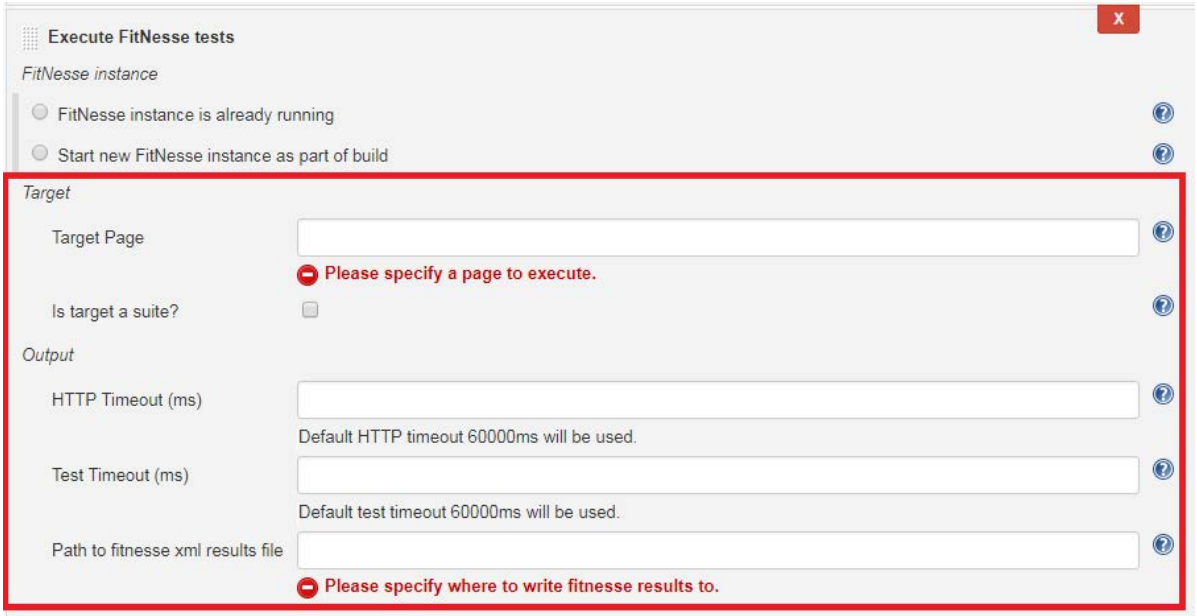
프로젝트 메인 화면 좌측에 구성을 클릭한다.



상단의 Build 메뉴를 클릭하여 Build 설정 화면으로 이동한다.
아래에 Add build step을 클릭하고 Execute FitNesse tests라는 항목을 클릭하여 추가한다.



아래와 같이 FitNesse을 실행하기 위한 설정 항목이 보인다.
기본적으로 테스트 실행 관련된 Target과 Output 설정을 한다.



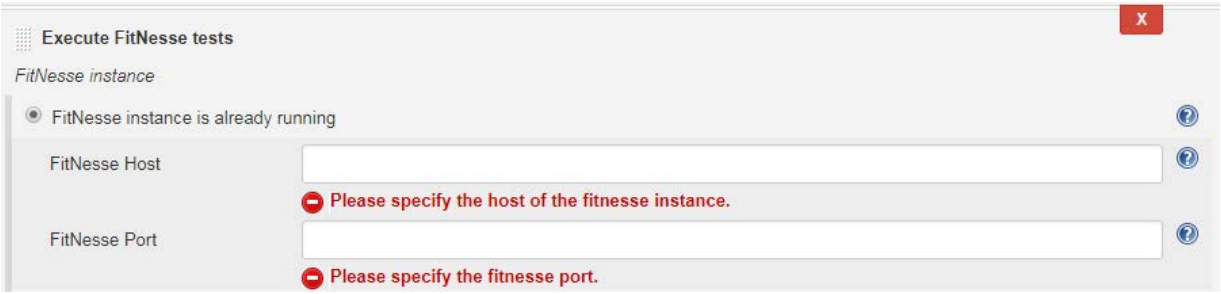
Target Page 는 본 가이드 FitNesse에서 만들었던 패이지를 사용하도록 하겠다. 앞서 만들었던 페이지는 아래와 같으며 Target Page에 입력하면 된다.

FrontPage.Sample01

Output에서 Timeout 설정은 기본값으로 사용하면 되며 실행 결과 파일은 원하는 임의의 경로에임의의 파일을 설정하면 된다.
여기서는 앞서 설치한 FitNesse 경로로 입력할 것이며 아래와 같이 입력할 것이다.

```
C:₩ledeme₩fitnesse-results.xml
```

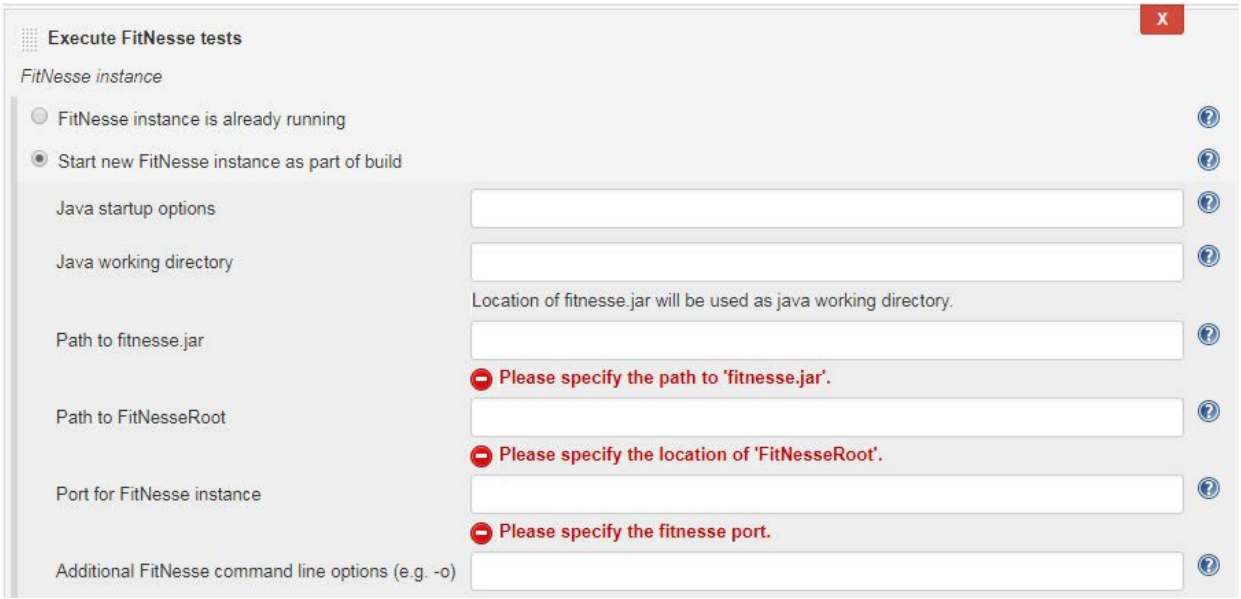
다음으로 FitNesse 서버 정보를 설정하는 것을 알아보겠다.
FitNesse instance에 선택할 수 있는 두 가지가 있다.
FitNesse가 항상 실행 중이라면 'FitNesse instance is already running' 을 선택하고 Host와 Port를 입력한다.



본 가이드에서 설치한 FitNesse 정보를 아래와 같으므로 Host와 Port를 각각 입력한다.

- Host : localhost
- Port : 80

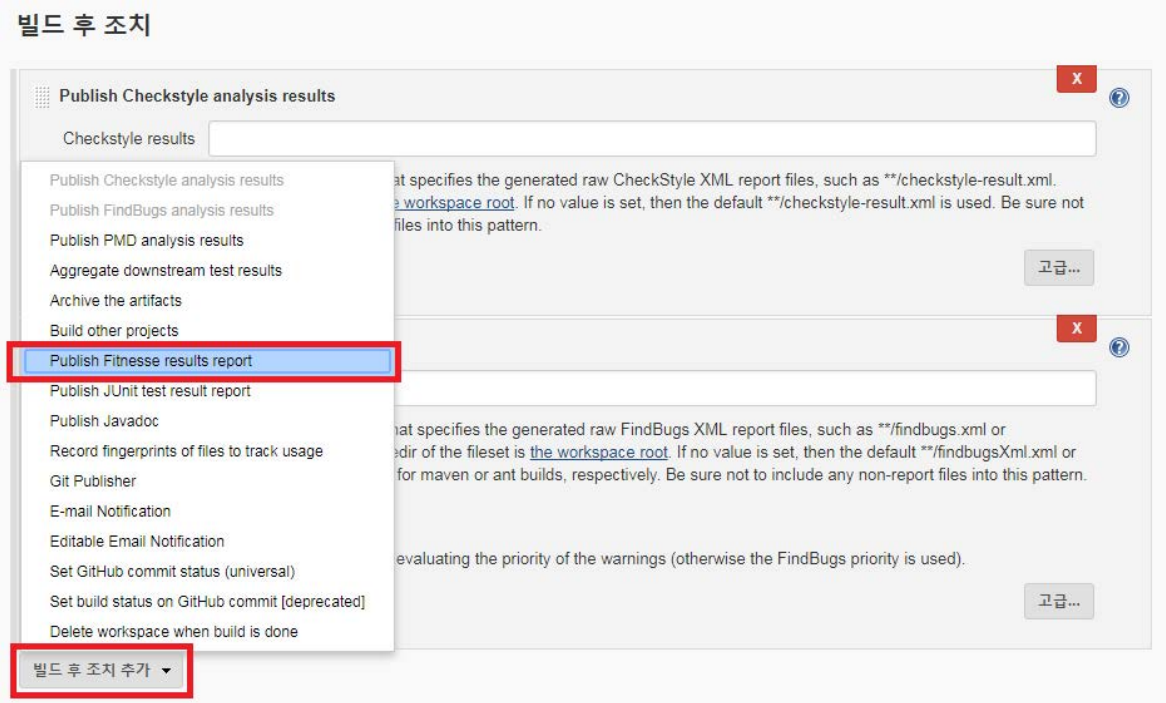
빌드를 할 때 FitNesse를 시작해야 한다면 'Start new FitNesse instance as part of build'를 선택하고 FitNesse 설치 정보를 입력해 주어야 한다.
평소에는 FitNesse를 실행하지 않고 빌드 시에 실행하여 사용하고자 할 경우 사용하면 된다.



본 가이드 FitNesse 설치 정보는 아래와 같다.

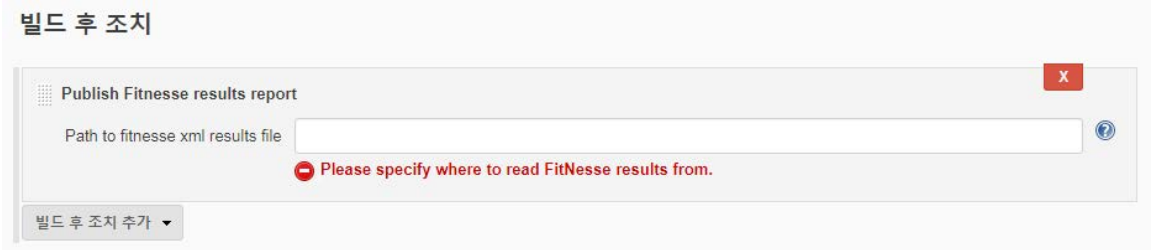
- Java startup options : (입력하지 않는다. 환경에 따라 필요하면 입력을 한다.)
- Java working directory : C:₩ledeme
- Path to fitnesse.jar : C:₩ledeme₩fitnesse-standalone.jar
- Path to FitNesseRoot : C:₩ledeme₩FitNesseRoot
- Port for FitNesse instance : 80
- Additional FitNesse command line options : (입력하지 않는다. 환경에 따라 필요하면 입력을 한다.)

다음으로는 실행 결과를 Jenkins에서 볼 수 있도록 설정하는 방법을 알아볼 것이다.
빌드 후 조치에서 빌드 후 조치 추가를 클릭하여 Publish Fitnesse results report를 선택하여 추가한다.

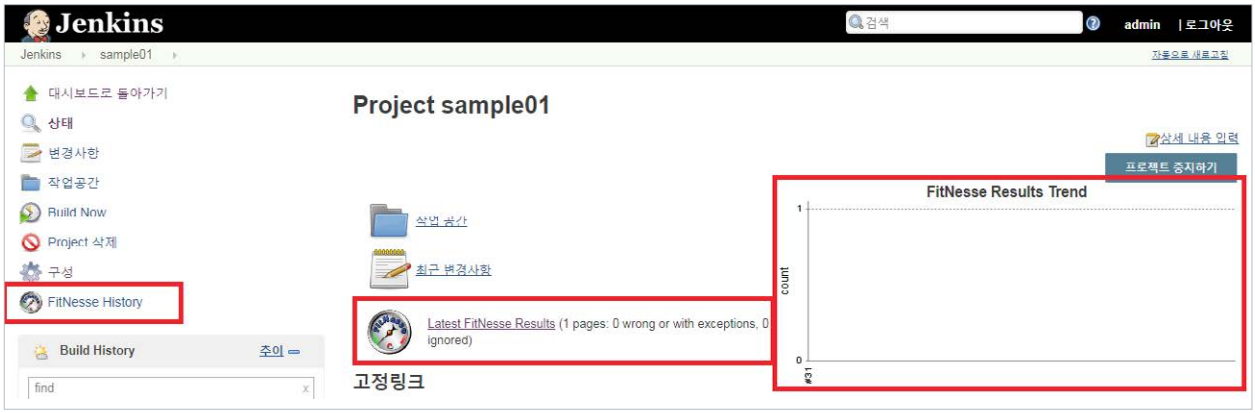


결과 파일은 앞서 입력하였던 것과 동일하게 아래와 같이 입력하면 된다.

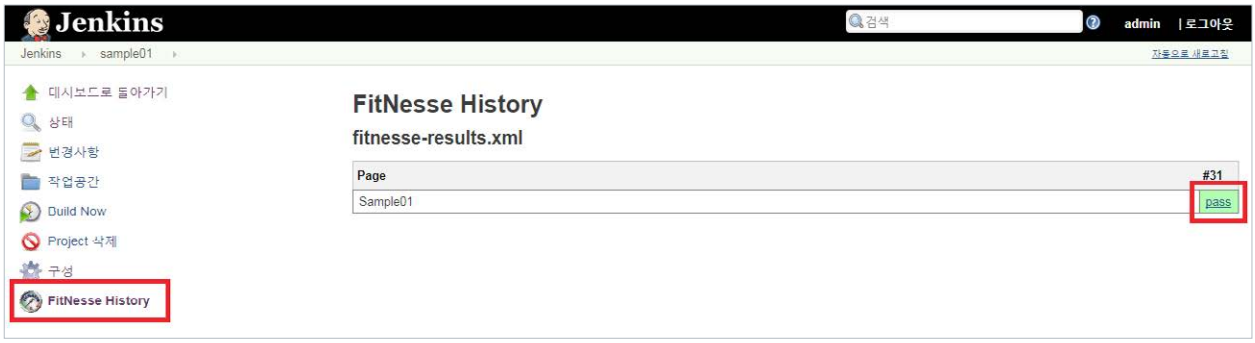
C:₩ledeme₩fitnesse-results.xml

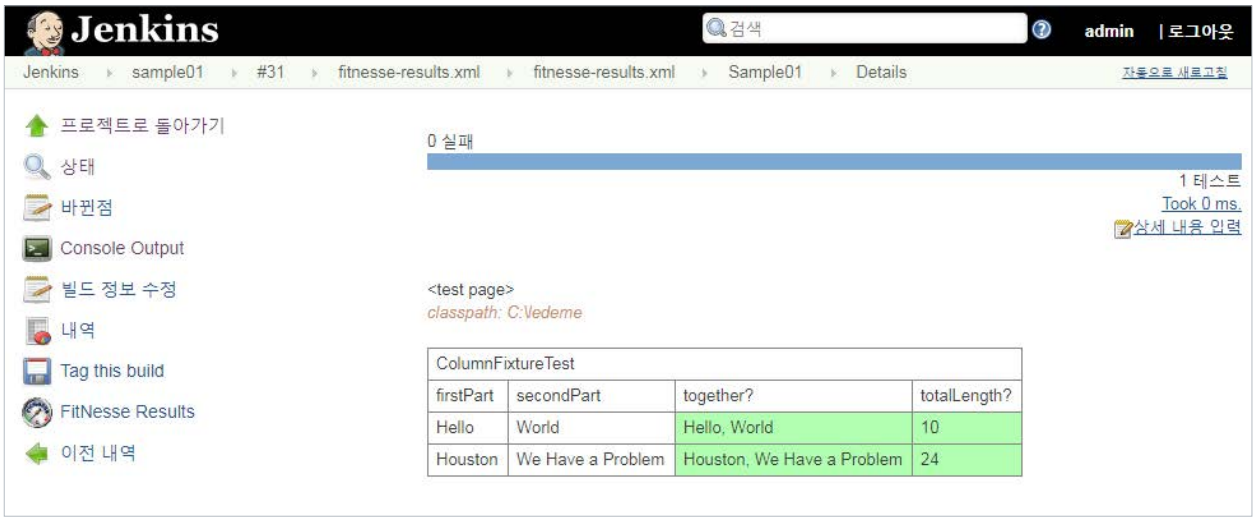


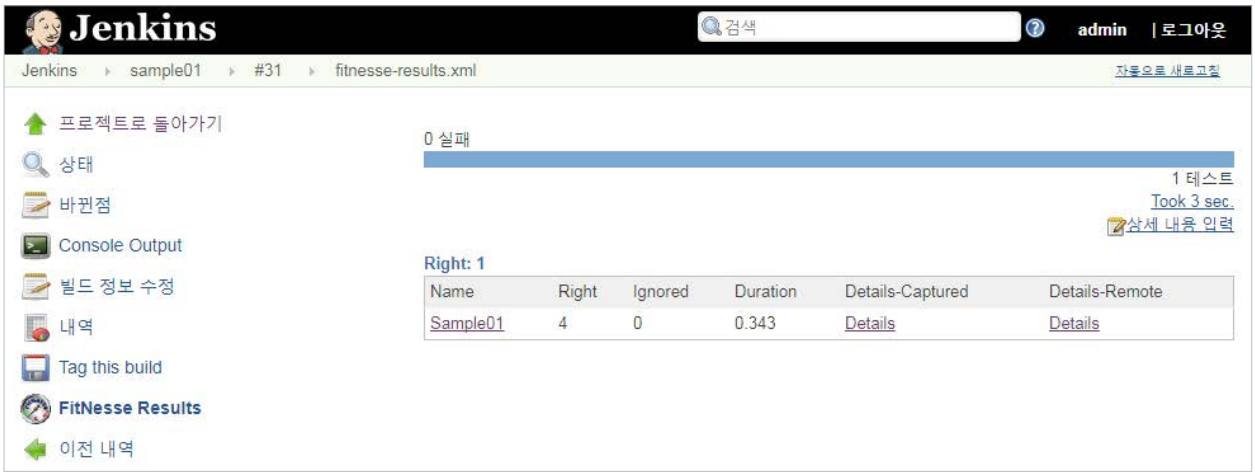
설정을 모두 마쳤다면 저장을 하고 프로젝트 메인 화면 좌측에서 Build Now 메뉴를 클릭하여 빌드를 실행 한다. 빌드가 완료되고 프로젝트 메인 화면을 갱신하면 FitNesse History 와 FitNesse Results 라는 항목이 보이는 것을 확인할 수 있다.
좌측의 메뉴에 FitNesse History 를 클릭하면 실행 결과 이력을 확인할 수 있고, Latest FitNesse Results 링크를 클릭하면 마지막 실행한 결과 정보를 확인할 수 있다.



FitNesse History 를 클릭하면 실행 이력을 확인할 수 있다. 각 이력 우측에 실행 결과가 보이며 클릭을 하면 상세한 정보를 확인할 수 있다.






Jenkins
검색
admin | 로그아웃
Jenkins ▸ sample01 ▸ #31 ▸ fitnesse-results.xml
자동으로 새로고침
프로젝트로 돌아가기
상태
바뀐점
Console Output
빌드 정보 수정
내역
Tag this build
FitNesse Results
이전 내역
0 실패
1 테스트
Took 3 sec.
상세 내용 입력
Right: 1
Name Right Ignored Duration Details-Captured Details-Remote
Sample01 4 0 0.343 Details Details