허드슨(Hudson)을 이용한

지속적인 통합

(Continuous integration with Hudson)

Version 1.0

자바월드(http://www.javaworld.com/)에 기고된 CI서버 허드슨(Hudson)관련 글을 번역해서 올려 봅니다. 매우 상세하게 설명되어 있어서 CI 서버가 생소한 사람들도 어렵지 않게 따라갈 수 있도록 되어있습니다

작년에 사이트(doortts.tistory.com)에 올린 글 들을 오타 조금 잡고, 문장 조금 고치고 하는 식으로 약간이나마 정리해서 출력하기 쉽게 모아서 올려봅니다.

문서 관련 문의 및 의견은 (doortts+hudson@gmail.com)으로 보내주세요. 감사합니다.

채수원

doortts.tistory.com

doortts.com

목차

번역시에 큰 고민하지 않고 진행한 관계(=발로 번역한 관계)로 다소 어색할 수 있습니다만, 이해가 곤란한 수준이 되진 않도록 노력하였습니다. CI 가 다양한 프로젝트에서 여유롭게 쓰여서 개발자의 삶에 도움이 조금이나마 되길 바라며 올려봅니다.

생산성 향상의 길은 먼 곳에 있는 것만은 아닙니다. **Let's start today!!!**

원문은 http://www.javaworld.com/javaworld/jw-12-2008/jw-12-hudson-ci.html 에서 살펴보실 수 있습니다.

---------- Started Here ----------

허드슨을 이용한 지속적인 통합
(Continuous integration with Hudson)

쉬운 설치와 설정을 제공하는 오픈소스 CI 서버
By Nicholas Whitehead, JavaWorld.com, 12/18/08

지속적인 통합(CI)은 소프트웨어 개발 생명주기에 걸쳐서 코드 품질 보증에 대해 초점을 맞추고 있는 팀들에겐 흔한 실천방법이 되었다. 이 글에서, 니콜라스 화이트헤드(Nicholas Whitehead) 는 인기 있는 오픈소스 CI 서버인 **허드슨(Hudson)**을 소개한다. 당신의 어플리케이션 개발 환경에 허드슨 서버를 구축하는 방법을 배워보고, 허드슨이 제공하는 많은 설정 옵션들의 전체적인 개념을 얻어보자. (예제는 윈도우 XP 에 톰캣6 혹은 우분투 리눅스에 JBoss AS 로 되어있다.) 그리고 자동화된 빌드, 테스트, 그리고 예제 프로젝트용 레포팅 프로세스를 구현하는 방법을 보기로 한다. **레벨은: 초급!**

지속적인 통합(CI)은 소프트웨어 빌드를 만들어내는 절차를 쉽게 하고 안정화 하기 위한 실천방법들의 세트이다. CI는 아래와 같은 난관이 있는 개발팀을 도와준다

> **소프트웨어 빌드 자동화**: 당신은 버튼을 하나 누른다던가, 혹은 미리 일정을 정해놓고, 그것도 아니면 특정한 이벤트에 대한 반응 같은 식으로 CI를 이용해서 어떤 소프트웨어 구조물(이하 아티팩트, Artifact)의 빌드 프로세스를 실행시킬 수 있다. 만일 당신이 소스로부터 아티팩트를 빌드하고자 할 때, 당신의 빌드 프로세스는 특정 IDE나 컴퓨터, 혹은 특정인에 속박되지 않게 된다. (당연히 속박되어서는 안 된다.)
>
> **지속적이고 자동화된 빌드 검증**: CI 시스템은 새로운 소스코드나 수정된 소스코드가 체크인 될 때마다 끊임없이 빌드가 실행되도록 설정될 수 있다. 이 말은, 어떤 소프트웨어 개발팀이 주기적으로 신규 코드나 수정된 코드를 체크인 하는 동안, CI 시스템이 새로운 코드가 빌드를 깨뜨리는지의 여부를 지속적으로 검증할 수 있다는 뜻이다. 이로인해 개발자들이 상호의존적인 컴포넌트들의 변경에 대해 각각 점검해야만 하는 필요성을 줄여준다.

지속적이고 자동화된 빌드 테스트: 빌드 검증의 확장에 해당하는 이 프로세스는, 신규 코드나 수정된 코드가 미리 정의된 빌드 아티팩트의 **테스트 스위트** 실패를 일으키지 않는다는 것을 보증해 준다. **실패(Failures)**는 빌드 검증과 테스트 둘 다에 있어 해당 작업이 실패했다는 **'통지(notification)'**를 관련 부서들에게 보내게 되는 방아쇠(...trigger)가 된다.

빌드 후속 절차 자동화: 한 소프트웨어 아티팩트의 빌드 생명주기에서는 빌드 검증과 테스트가 완료된 다음에도 문서 생성, 소프트웨어 패키징, 그리고 해당 아티팩트들을 실행환경이나 소프트웨어 저장소로 전개(deploy)하는 것 같은 자동화될 수 있는 추가적인 작업이 필요할 수 도 있다. 이런 형태로 아티팩트들이 사용자들이 사용할 수 있도록 신속히 만들어 질 수 있다.

CI 서버를 구현하기 위해, 당신은 최소한 하나의 접근 가능한 소스코드 저장소(그리고 그 안에 들어있는 소스코드), 빌드 스크립트 한 세트와 절차들, 그리고 빌드된 아티팩트용 테스트 스위트 하나가 필요하다. 그림1은 CI시스템의 기본 구조의 개략도(outline)이다.

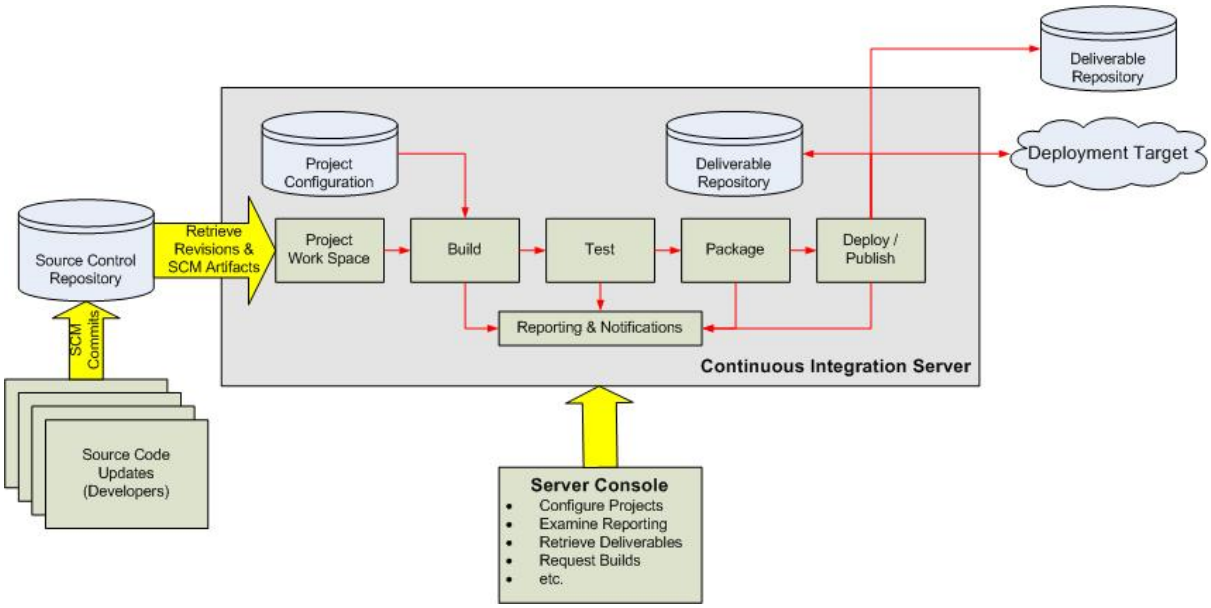


그림 1. CI 시스템의 기본 구조

시스템 컴포넌트들은 아래와 같은 순서로 동작한다.

개발자들은 소스코드 저장소 안으로 새로운 코드나 수정된 코드를 체크인 한다.

CI 서버는 각각의 프로젝트를 위한 전용 작업공간을 만든다. 하나의 새로운 빌드가 요청되거나 예정되면, 소스는 저장소로부터 빌드가 수행될 이 작업공간으로 불려온다.

CI서버는 새로이 만들어진, 혹은 새롭게 갱신된 작업공간에서 빌드 프로세스를 수행한다.

일단 빌드가 완료되면, CI서버는 추가적으로 새로운 아티팩트에 대해 정의된 테스트 스위트를 수

행할 수 있다. 만일 빌드가 실패하면, 등록된 개개인들은 이메일, 인스턴스 메시징, 혹은 어떤 다른 방식으로, 통지 받게 된다.

만일 빌드가 성공적이면, 해당 아티팩트들은 패키징 되어서 (이를테면 어플리케이션 서버 같은) 배포처로 전송되거나 아니면 소프트웨어 저장소에 새로 버전이 붙여진 아티팩트로 저장되기도 한다. 이 저장소는 CI서버의 일부가 될 수도 있고, 혹은 파일서버나 Java.net, SourceForge같은 소프트웨어 배포 사이트 식의 외부 저장소가 될 수도 있다. 소스코드 저장소와 아티팩트 저장소는 분리될 수 있고, 몇몇 CI서버들을 사용하면 공식적인 소스관리 시스템을 전혀 갖고 있지 않는 것이 가능하다.

CI서버는 보통 어떤 형태든 프로젝트를 설정하고 디버그 할 수 있는, 그리고 경우에 따라 특별하게 즉시 빌드를 만들어 내게 하거나 보고서 생성, 혹은 빌드된 아티팩트들을 검색하는 식의 동작들을 요청할 수 있는 콘솔을 갖는다.

허드슨: 지속적인 통합 서버

지속적인 통합은 지난 몇 년에 걸쳐 인기리에 성장해 왔고, 현재 상업용/공개용 둘 다 포함해서 당신이 선택할 수 있는 꽤 많은 수의 CI 서버들이 존재한다. 나는 동료가 허드슨(Hudson)을 추천하기 이전에 개인적으로 4개의 CI 서버들을 사용하고 있었는데, 쓰게 되자마자 곧바로 감명받게 되었다. 처음에 난 허드슨이 잘 알려져 있지 않다고 생각했었지만, 자바 파워 툴즈 사이트의 설문은 응답자들 사이에서 허드슨이 가장 폭넓게 사용되는 CI서버라는 걸 보여줬다. (이 글이 쓰여지는 시점에서) 투표 중 37.8퍼센트를 차지하고 있다.

지원하는 SCM 들

서브버전은 허드슨에서 곧 바로 사용할 수 있는 수준으로 내장 되어 지원되고 있고, CVS는 통합하기 위해서 약간의 설정만이 필요하며, CVS클라이언트가 허드슨 호스트에 인스톨 되어 있다. 몇몇 다른 소스 코드 관리(SCM) 솔루션들은 허드슨 플러그인 형태로 지원되고 있다. 이 글을 쓰는 시점에서 아래와 같은 SCM들이 지원된다.

- Accurev
- BitKeeper
- ClearCase
- Git
- Mercurial
- Perforce
- StartTeam
- Team Foundation Server
- Visual SourceSafe

- URL SCM (SCM에 URL 을 사용할 수 있게 해주는 특별한 SCM 플러그인)

이 글에서, 나는 서브버전과 Java.net의 소스 저장소를 사용할 것이다. 따라서 당신은 위에 있는 어떠한 플러그인들도 설치할 필요가 없다. (여담으로, MKS 소스통합 허드슨 플러그인을 만들고 있는 사람을 알고 있는데, 혹시 관심 있으면 메일(email) 주길 바란다)

허드슨은 공짜이고 오픈소스 제품으로 Java.net에 자리잡고 있다. 허드슨은 썬 마이크로 시스템의 엔지니어인 코수케 카와구치(Kohsuke Kawaguchi)에 의해 최초로 작성되었으며, 2005년 2월에 그의 블로그에서 릴리즈가 발표되었다. 허드슨은 그 이후 거의 154번의 릴리즈가 있었다.

왜 내가 허드슨을 좋아하는 지, 그리고 왜 내가 (몇몇 특별한 요구사항이 필요한 경우를 제외하고) 허드슨을 당신에게 추천하는 지에 대한 이유는 아래와 같다.

> 내가 사용했던 모든 CI 제품 중에서, 인스톨하고 설정하기가 가장 쉬웠다. 웹 기반 유저 인터페이스는 매우 친숙하고, 직관적이며, 빠르게 반응했고, 많은 경우에 있어 개별적인 설정 필드에 대해 Ajax를 이용한 즉각적인 피드백을 제공하였다.
>
> 허드슨은 (만약 당신이 자바 개발자라면 유용할 수 있는)자바 기반이지만 자바 기반의 소프트웨어를 빌드하는 데에 한정되어 있지는 않다.
>
> 허드슨은 깔끔하게 컴포넌트화 되어 있고 잘 정의되고 문서화되어 있는 확장 가능한 API를 허드슨 플러그인 형태로 제공한다. 그로 인해 서버의 기능을 확장하는 커다란 허드슨 플러그인 라이브러리를 갖게 하였다. 이 라이브러리들은 허드슨 콘솔에서 자유롭게 사용가능하고 설치 가능하다.

허드슨 인스톨 하기: Windows XP 혹은 우분투 리눅스

허드슨을 사용하기 위해서, 당신은 접속가능하고 허드슨을 지원하는 소스 관리 시스템과 아티팩트로 빌드될 수 있는 소스, 그리고 동작하는 빌드 스크립트가 필요하다. 그 외에 당신이 정말 필요한 것은, 동작하는 허드슨 서버를 인스톨하고 설정하기 위해 자바 1.5 이상과 허드슨 인스톨 파일을 설치하는 것이다.

```
C:₩hudson> java -jar hudson.war
```

위와 같이 할 수도 있지만, 서블릿2.4와 JSP 2.0스펙에 기반한, GlassFish, Tomcat, JBoss, 혹은 Jetty 같은 자바 서블릿 컨테이너에 허드슨을 디플로이 하는 것이 좀더 일반적일 수 있다. 다음 섹션에서, 두 개의 허드슨 인스톨 시나리오로 설치를 진행하려고 한다. 하나는 윈도우즈XP에 Tomcat 6을 이용하는 것이고 다른 하나는 우분투(Ubuntu) 리눅스에 JBoss 4.2.3 를 이용하는 것이다. (JBoss AS 5.0 는 이 글의 제출일 이후에 릴리즈 되었다)

허드슨 인스톨 하기: 톰캣 6 와 윈도우즈 XP

나는 당신의 윈도우즈XP 머신 위에 JAVA 버전 1.5 이상이 설치되어 있다고 가정할 것이다. 아래에 이어지는 단계는 윈도우즈 서비스 인스톨러를 이용해서 톰캣 6.0.18을 설치할 것이다. 그래서 윈도우즈 XP 가 부팅되면 곧바로 허드슨이 시작되게 할 것이며, 로그인 하지 않더라도 백그라운드로 동작하도록 할 것이다. 다운로드 해서 설치할 톰캣 파일은 apache-tomcat-6.0.18.exe이다.

톰캣 설치는 인스톨 옵션을 선택하게 할 것이다. **Custom** 옵션을 선택해서 그림 2에 보이는 것처럼 **Service**를 반드시 선택하자. 그래서 톰캣이 서비스로 동작한다.

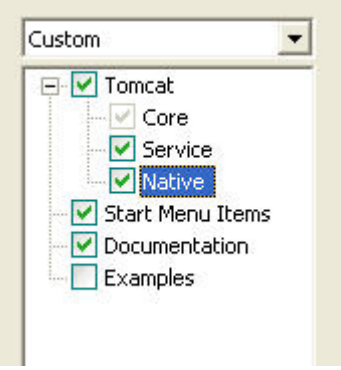


그림 2. 톰캣 설치 옵션

다음으로, 톰캣을 설치할 디렉터리를 그림 3에 보이는 것처럼 선택하자. 디렉터리를 정할 때 빈 공백이 없는 디렉터리를 고르는 걸 추천한다. 이에 대해 나중에 나한테 감사 하게 될 것이다.

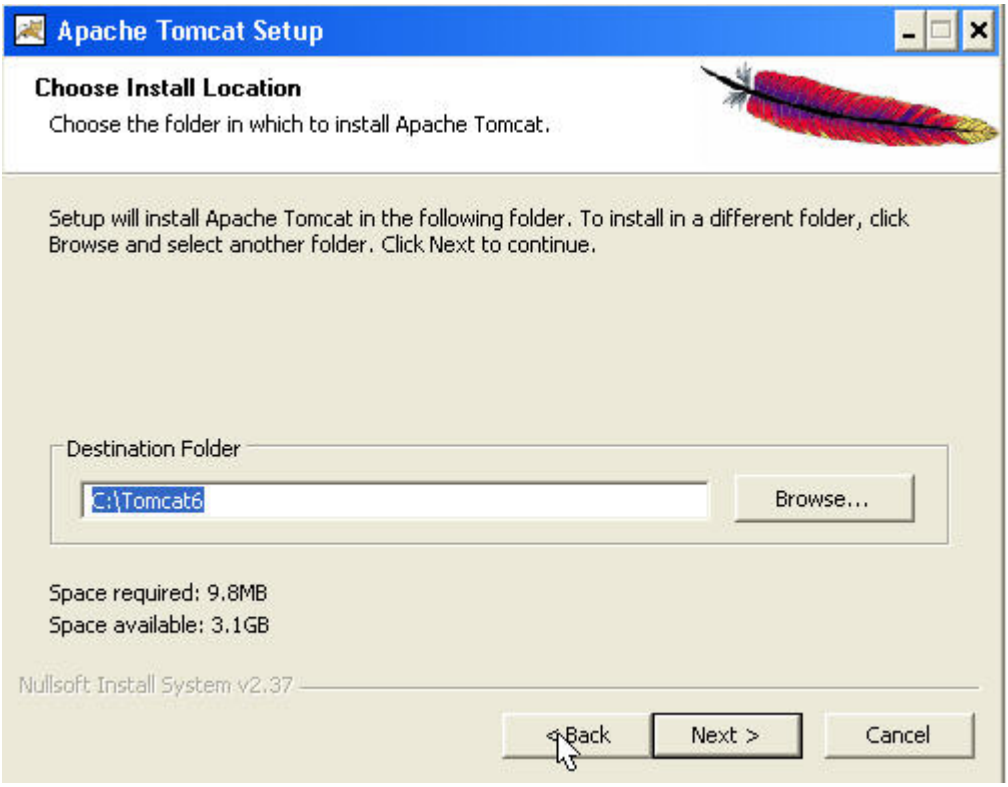


그림 3. 인스톨 디렉터리 고르기

이제 인스톨러는 응답대기 할 포트를 선택하라고 물을 것이다. 기본값은 포트 8080인데, 적당하다. 해당 포트를 사용하는 다른 어플리케이션이 없는 지만 유의하자. 만일, 해당 포트를 사용하는 어플리케이션이 있으면, 톰캣은 적절히 뜨지 않을 것이다. (뭐. 당연하잖아!!) 톰캣 관리자 유저이름과 패스워드도 물어볼 거다. 그림 4에 모두 나와 있다.

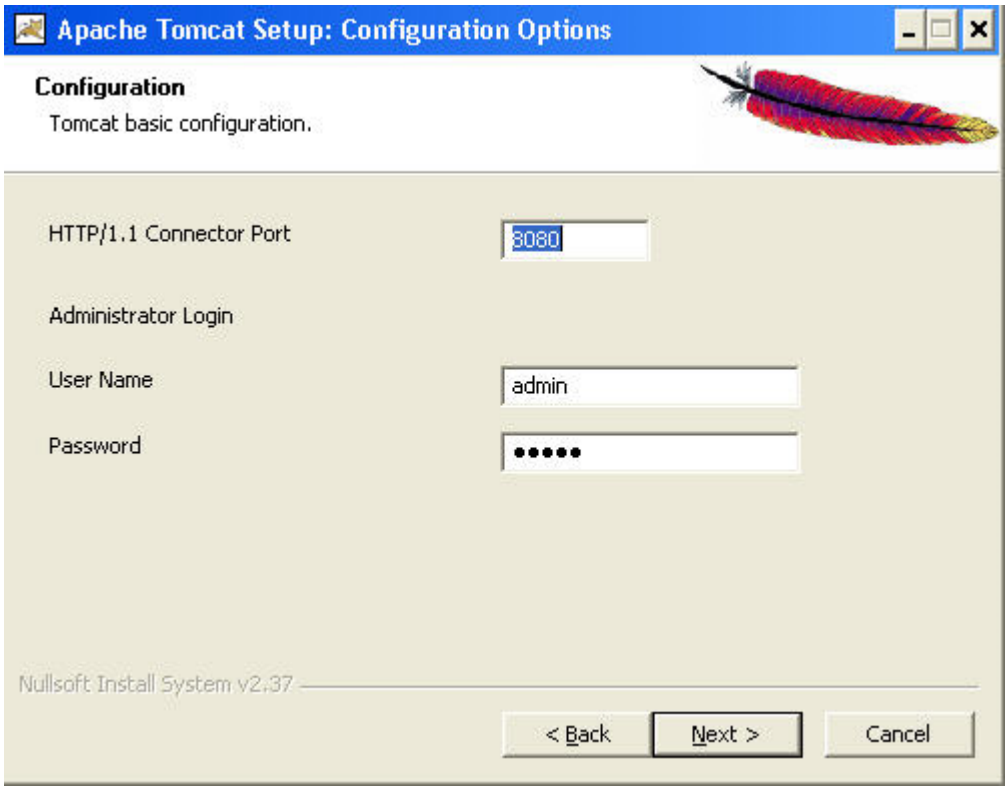


그림 4. 리스닝포트와 관리자 유저 이름과 패스워드 고르기

그리고 나면 인스톨러는 자바 JRE가 설치된 위치를 물어볼 것이다. 그림 5에 보이는 것처럼, 나는 Sun Java 1.6.0_07을 사용했다.

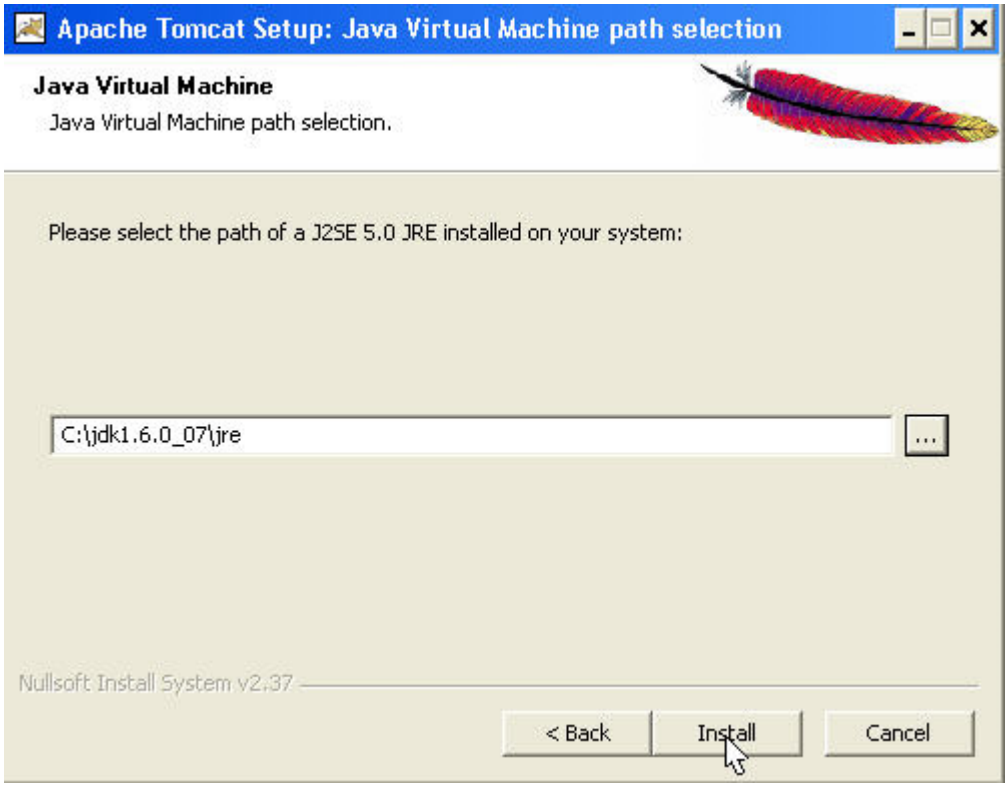


그림 5. 톰캣을 띄울 Java JRE 고르기

Install 을 클릭하고 나면, 인스톨이 실행&완료 되고 서비스가 시작될 것이다. 웹 브라우저에 http://localhost:8080 을 입력해서 톰캣이 제대로 동작하는지 확인할 수 있다. (톰캣이 인스톨 된 컴퓨터에서 웹 브라우저를 띄우는 것이 아니라면 localhost 대신에 IP 주소나 적절한 이름으로 치환한다) 그림 6에 보이는 스크린샷 처럼 웹 페이지가 표시될 것이다.

그림 6. 톰캣 인스톨과 동작 검증

이제, 허드슨을 인스톨하기 위해, hudson.war 파일을 톰캣이 설치된 디렉터리의 webapps 하위 디렉터리로 복사한다. 만일 당신이 그림3에 보이는 것과 동일한 인스톨 디렉터리를 사용했다면 webapps 는 C:₩Tomcat6₩webapps가 될 것이다. 톰캣은 핫-디플로이(뜨거운 배포...-_- sorry for my garbage joke)를 할 텐데, 지금 할 수 있는 쉬운 방법은 톰캣을 재 시작하는 것이다. 재 시작 하는 데는 두 가지 방법이 있는데, 첫 번째는 DOS 쉘을 열여서 아래와 같은 명령어를 입력하는 것이다.

```
C:₩Tomcat6>net stop Tomcat6
C:₩Tomcat6>net start Tomcat6
```

두 번째 옵션은 서비스 애플릿을 여는 것이다. 제어판(Control Panel)의 관리자 툴에서 해당 애플릿을 찾을 수 있다. 제어판은 시작버튼에 있는 윈도우 툴바에서 설정을 선택해서 제어판을 고르면 된다. 서비스 애플릿에서, **Apache Tomcat** 이라고 이름 붙은 서비스를 찾아서 재시자 버튼을 누른다. 그림 7에 설명되어 있다.

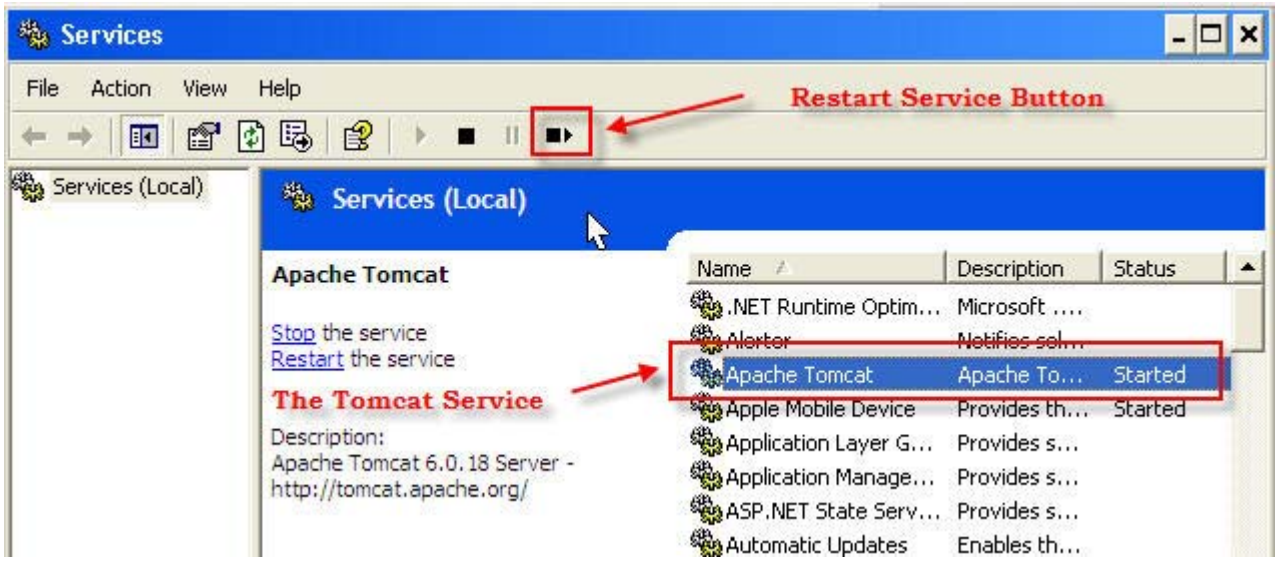


그림 7. Services 애플릿 Figure 7. The Services applet

이제 허드슨이 설치되었다. 웹 브라우저에 http://localhost:8080/hudson 이라고 지정해서 확인할 수 있다. 메인 허드슨 스크린은 그림 8과 같다.

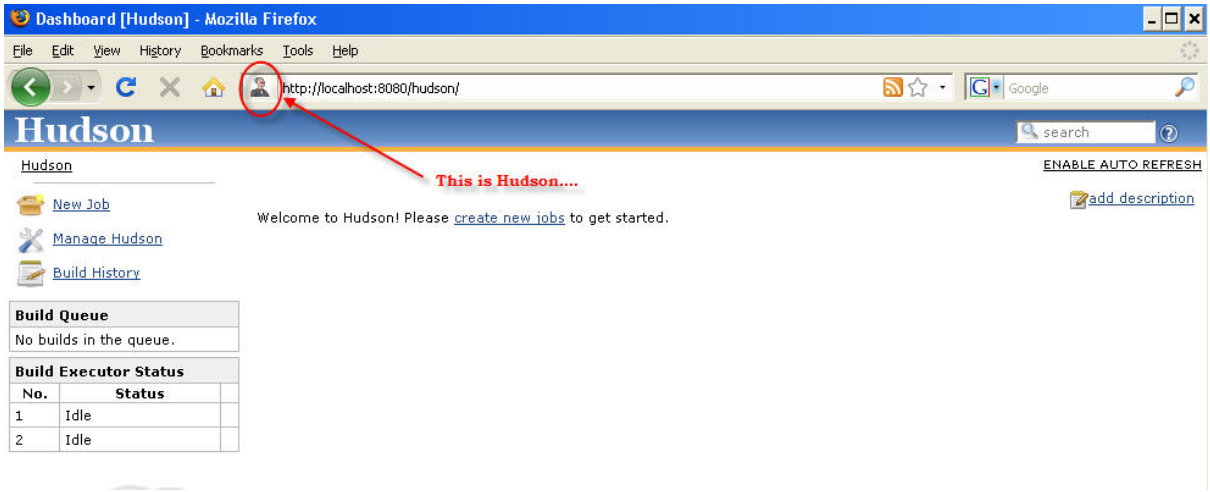


그림 8. 허드슨 시작 페이지

당장 필요한 것은 이게 전부다! 만일 당신이 Windows XP 와 Tomcat 기반의 어플리케이션 개발 환경에 친숙하다면, 세팅은 다 끝났다. 만약 당신이 우분투 리눅스에 JBoss 를 실행하는 시스템을 선호한다면, 계속 읽기 바란다.

허드슨 인스톨 하기: JBoss 4.2.3 on Ubuntu Linux 8.04 (Hardy Heron)

우분투에 썬 자바 1.6을 설치하려면, 쉘을 열어서 아래와 같은 커맨드를 실행하여라.

```
sudo apt-get install sun-java6-jdk
```

sudo 커맨드를 실행하면, 패스워드를 물어볼 것이다.

JBoss를 인스톨 하는 데는 몇 가지 방법이 있다. 여기에선 jboss 전용 유저를 만들 것이다. 이 방식이 최고의 선택이고, 유저를 만들어서 당신의 home 디렉터리에 JBoss를 인스톨하는 것이 나은 방법으로 여겨지고 있다. 여기에 요약된 절차는 우분투 포럼에 있는 a useful description 을 응축한 것이다.

첫째로, the JBoss 4.2.3.GA package 를 다운로드 한다. jboss-4.2.3.GA.zip 이라고 이름 붙은 파일을 찾아라.

다음으로, jboss로 이름 지어진 group 과 home 디렉터리, 유저를 만들어야 한다. group 은 편리하지만, 이 글에서 자세히 다루진 않는다. 우분투 서버에 있는 다른 유저들에게 JBoss권한을 확장할 수 있게 해준다.

리스트1 은 jboss home 디렉터리, 유저, 그룹을 생성하는 명령어들과 JBoss서버를 설치하는 것을 코멘트로 보여 주고 있다.

리스트 1. jboss 계정 생성과 서버 인스톨

```
echo Create the jboss group
sudo groupadd jboss
echo Create the jboss user, define bash as the user's default shell and
/home/jboss as the home directory
echo and make the user jboss part of the group jboss
sudo useradd -s /bin/bash -d /home/jboss -m -g jboss jboss
echo Copy the jboss-4.2.3.GA file to /home/jboss or download directly into that
directory
sudo mv jboss-4.2.3.GA /home/jboss
echo Change the owner of the file to jboss
sudo chown jboss:jboss /home/jboss/jboss-4.2.3.GA
echo Log into the jboss account
sudo su jboss
echo Go to the jboss home directory
cd
echo Unzip the file jboss-4.2.3.GA
unzip jboss-4.2.3.GA
echo Create a symbolic link "jboss" for "jboss-4.2.3.GA".
echo This allows you to change JBoss versions with minimal changes
ln -s jboss-4.2.3.GA jboss
```

unzip 이 설치 되어 있지 않으면, (sudo 명령이 실행 가능한 유저로 로그인해서) 아래와 같은 명령어를 입력해서 인스톨하여라.

```
Sudo apt-get install unzip
```

JBoss 서버는 이제 기본적으로 인스톨 되었다. 다음과 같은 명령어를 사용해서 띄울 수 있다.

```
/home/jboss/jboss/bin/run.sh
```

그러나 명령어로 띄우는 것이 아니라, 이 예제에서는 장비가 시작될 때 서비스가 자동적으로 시작되도록 자동 시작 스크립트를 설치할 것이다. JBoss 를 내려 받을 때 세 개의 서로 다른 init.d 스크립트도 함께 다운로드 되었지만, 손 본 필요가 있다. 자동으로 기동과 정지를 가능하게 해줄 jboss-init.sh 스크립트를 다운로드 할 수도 있다. 그런 다음, 리스트2에 있는 명령어들을 실행하여라.

리스트2. JBoss를 위한 자동 시작 스크립트 인스톨 하기

```
echo Move the jboss-init.sh file to /etc/init.d/ and rename it to jboss
sudo mv jboss-init.sh /etc/init.d/jboss
echo Change the owner of the /etc/init.d/jboss file to root
sudo chown root:root /etc/init.d/jboss
echo Make the /etc/init.d/jboss file executable
sudo chmod ug+x /etc/init.d/jboss
echo Activate the /etc/init.d/jboss file in the rc.d lifecycle process.
sudo update-rc.d jboss defaults
```

자 이제 아래와 같은 명령어를 이용해서 백그라운드 프로세스로 JBoss 서버를 시작할 수 있다. (로그아웃 해도 프로세스는 종료되지 않을 것이다)

```
sudo /etc/init.d/jboss start
```

JBoss가 동작하는 것을 확인하기 위해 브라우저에 http://localhost:8080/jmx-console 라고 지정해보자. (뜨는데 몇 분쯤 걸릴 수 있다) 그림9는 나와야 하는 JBoss JMX 콘솔을 보여준다.

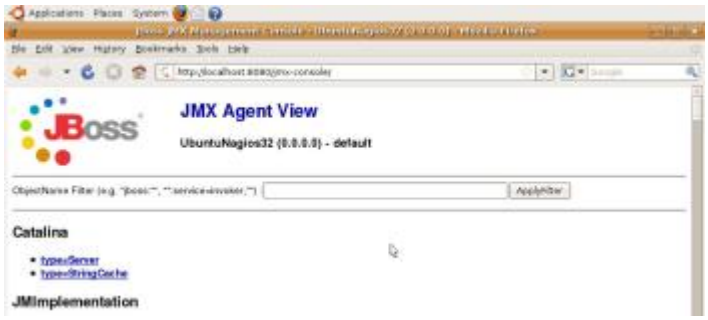


그림 9. The JBoss JMX console

허드슨을 디플로이 하는 것은 간단하다. hudson.war 파일을 /home/jboss/jboss/server/default/deploy 에

복사하면 끝난다. jboss 유저를 이용해서 JBoss서버가 해당 파일을 읽을 수 있는 권한을 갖도록 하는 것이 최선이다. 허드슨이 몇 초 이내에 디플로이 될 것이다. 그러면 웹 브라우저에 http://localhost:8080/hudson 이라고 지정해서 디플로이가 잘 되었는지를 확인할 수 있다. 그림 8에 보이는 것과 같은 페이지가 보여질 것이다.

허드슨에서 빌드 세팅하기(Setting up a build in Hudson)

당신은 윈도우즈XP에서 동작하는 톰캣, 혹은 우분투 리눅스에서 동작하는 JBoss 에 허드슨을 인스톨하는 절차를 지금까지 거쳐왔다. 대부분의 단계들은 운영체제에서 어플리케이션 서버를 설치하고 관리자 작업을 수행하는 것에 관련되어 있었다.; 나는 당신이 허드슨을 인스톨 하는 것 그 자체는 사소한 부분이라는 것에 동의하리라 생각한다. 다음으로는, 빌드 업무를 구축할 수 있도록 허드슨에서 필요한 몇 가지 기본 설정 방법을 보게 될 것이다.

전제조건

빌드 업무(build job)를 셋업하기 전에, 아래와 같은 조건들이 반드시 만족되어야 한다.

- 접근 가능한 소스 코드 저장소(repository)가 있어야 한다.
- 저장소는 빌드를 필요로 하는 소스 코드를 담고 있어야 한다.
- 저장소는 소스를 빌드하는 빌드 스크립트를 반드시 포함하고 있어야 한다. 허드슨이 간단한 쉘을 지원하지만, 여기서 말하는 스크립트는 보통 Ant 나 Maven 스크립트, NAnt 나 MSBuild 같은 빌드 스크립트를 말한다.

본 글에서는, 예제 소스를 빌드하는데 Ant version 1.7 을 사용할 것이다.

허드슨 설정하기

소프트웨어 빌드작업을 시작시키기에 앞서 허드슨에 있는 몇 가지 마이너한 설정에 주의를 기울일 필요가 있다. 우선, 허드슨에게 JAVA JDK 와 Ant 의 인스톨위치가 어디인지 알려주어야 한다. http://localhost:8080/Hudson의 허드슨 메인 페이지에서, **Manage Hudson** 링크를 클릭하여라. 이어지는 페이지에서, **Configure System** 을 클릭한다.

시스템 설정 페이지에서, 목록의 첫 번째 항목인 **Home directory** 를 주의 깊게 보기 바란다. 이곳이 바로 허드슨의 모든 설정이 저장되고 모든 작업이 이루어지는 곳이다. 본 글의 뒤 부분에서 이 디렉터리를 다시 살펴볼 것이다.

JDK와 Ant 인스턴스 둘 다 설정하기 위해서, 각각의 섹션 하단에 있는 **Add** 버튼을 누르자. 한 세트의 필드들이 설정 섹션 아래로 확장될 것이다. 이들 두 개의 섹션은 그림 10에서 보여주고 있다.

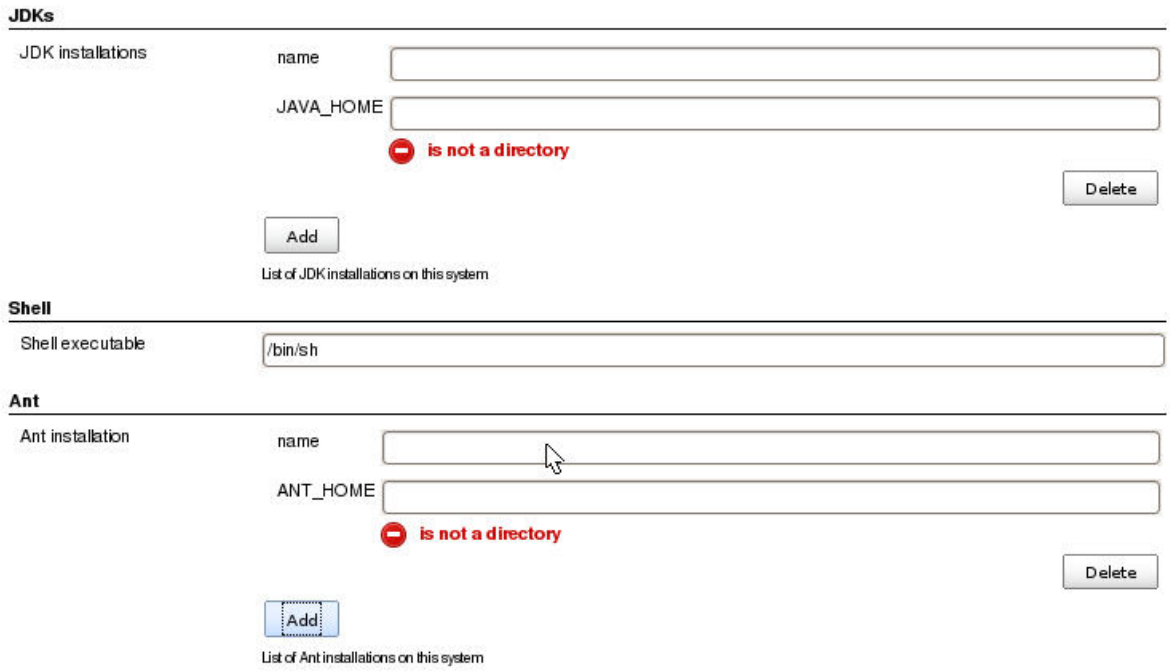


그림 10. Ant 와 JDK 설정하기

각각의 경우에 있어, JDK와 Ant 인스톨을 식별해 주는 임의의 이름을 할당한다. 당신의 프로젝트 요구사항에 따라서 복수개의 인스턴스가 설정될 수 있기 때문에 겹치지 않는 임의의 이름을 할당할 필요가 있다. 예를 들면, 당신은 Java JDK 1.4, 1.6, 그리고 1.7용 인스턴스와 Ant 1.6, 1.7 용 인스턴스를 갖고 있을 수도 있다. 당신이 당신 프로젝트 빌드를 설정할 때에는, 사용하기 원하는 인스턴스를 선택하는 것이 가능하다.

현재 빈 공백 필드로 보여지고 있는 JAVA_HOME과 ANT_HOME이 유효한 디렉터리가 아니라는 걸 허드슨이 즉각적으로 당신에게 알려준다는 점에 주목하길 바란다. 유효한 디렉터리가 입력되면, 경고가 사라지게 될 것이다. 이게 바로 허드슨이 유저 친화적이라고 말할 수 있는 간단한 예의 하나이다. 허드슨은 디렉터리를 찾게 될 때야 비로소 에러가 발생하게 되는 방식이 아닌 즉각적인 피드백을 준다.

나는 이미 Java 1.6 JDK를 설치했기 때문에, 다음과 같이 JDK 인스톨 정보를 설정할 수 있다.

- **name**: JDK 1.6.0_07
- **JAVA_HOME**:
  - **Windows**: C:₩jdk1.6.0_07
  - **Linux**: /usr/lib/jvm/java-6-sun

만일 당신인 윈도우즈를 사용하고 있다면, 간단하게 Ant 설치파일 을 다운로드 받아 대상 디렉터리에 압축을 풀어버리면 된다. 우분투라면, Ant 를 설치하기 위해 아래와 같은 명령어를 사용할 것이다.

```
sudo apt-get install ant

sudo apt-get install ant-optional
```

```
ant -version

Apache Ant version 1.7.0 compiled on August 29 2007
```

우분투의 인스톨 명령은 Ant를 /usr/share/ant 에 설치할 것이다.

그림11은 우분트에서의 JDK와 Ant 설정을 보여주고 있다. 인스톨된 위치를 지정하는 필드들을 입력하면, 경고표시가 사라진다. 나는 올바르지 않게 설정된 Ant의 두 번째 인스턴스를 또 하나 추가했다. 만일 당신이 Ant 인스턴스가 설치된 올바른 디렉터리는 아니지만, 유효한 디렉터리 경로를 입력했다면, 그림에서처럼 당신이 입력한 항목이 올바르지 않은 것 같다는 허드슨의 경고를 보게 될 것이다. JDK나 Ant 둘 다 마찬가지다. 나는 그 후에 이 잘못된 Ant 항목을 지웠고, 유효한 것 하나만 남겼다.

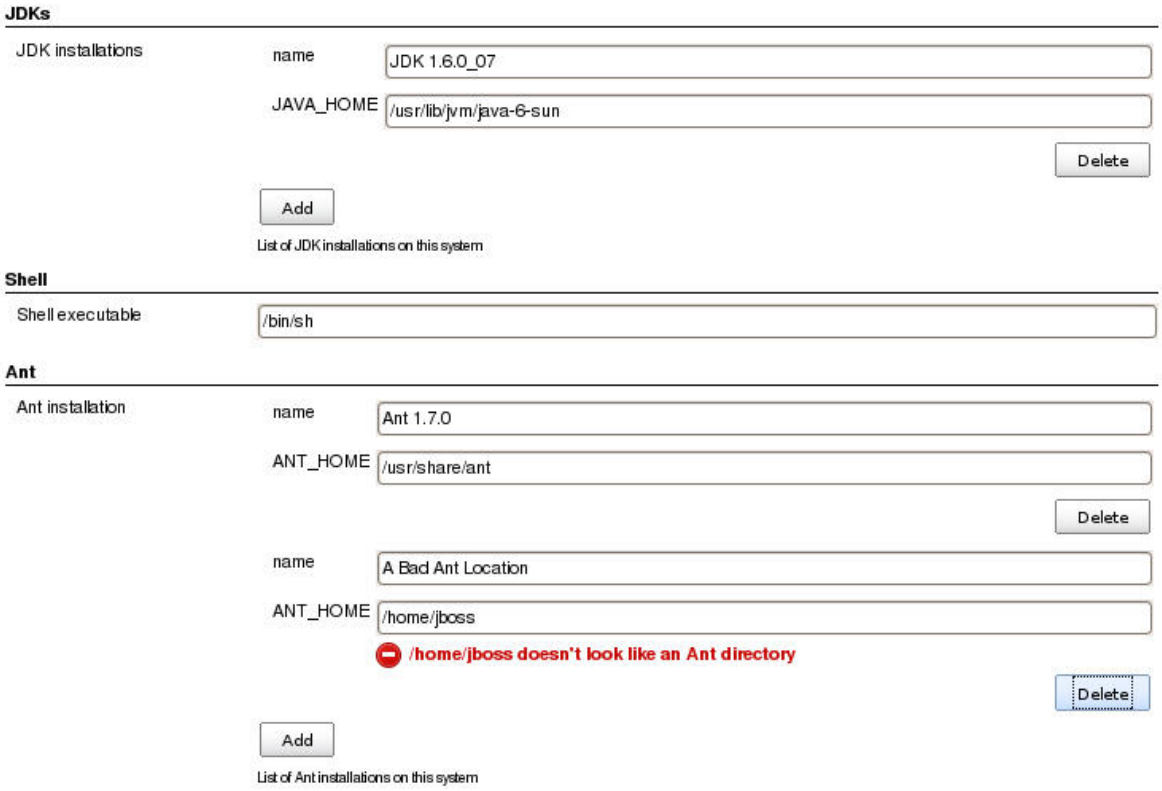


그림 11. 우분투에서의 JDK 와 Ant 설정

같은 폼의 약간 아랫부분에서는, 실행 가능한 CVS 의 장소를 지정할 수 없다는 경고가 보일 것이다. 만일 CVS를 사용하지 않는다면, 가볍게 무시해도 된다.

이 페이지에서 중요한 마지막 항목은, 허드슨이 빌드 실패 같은 중요한 이벤트들을 이메일로 당신에게 알려줄 수 있게 하는 SMTP 설정이다. 만일 당신의 SMTP 서버가 인증을 필요로 한다면, 고급 옵션(advanced)을 사용할 필요가 있다. 나는 내 Google Apps 호스트 도메인을 사용한다. 하지만, 만일 당신이 Gmail 계정을 가지고 있다면, 당신은Google SMTP 서버와 Gmail 어드레스를 사용할 수 있다. 그림12에서 보이는 것이 본인의 허드슨 서버 이메일 설정이다.

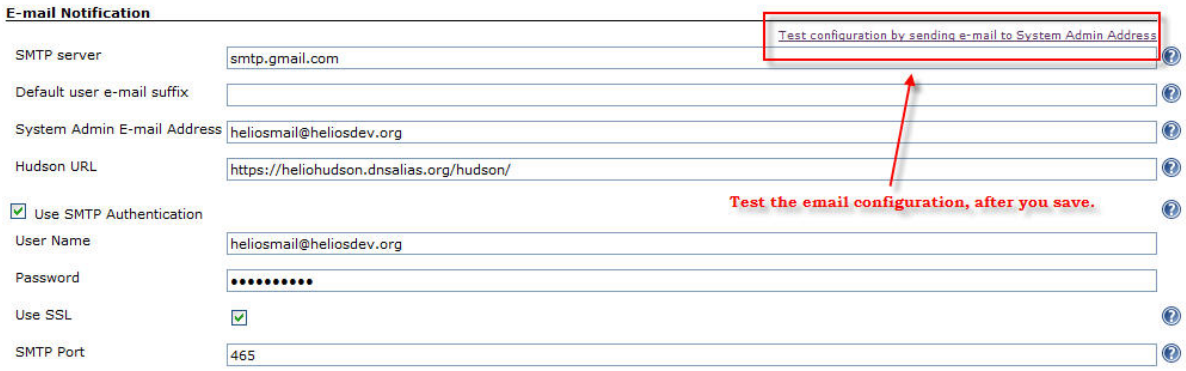


그림 12. 구글 메일 서버를 이용한 허드슨 SMTP 발송 설정 (클릭하면 확대됨)

기본적인 시스템 레벨의 설정은 이제 끝났다. 당신은 이제 특정 빌드 작업(build job)을 설정할 준비가 되었다.

허드슨에서 빌드 업무 설정하기 (Configuring a build job in Hudson)

http://localhost:8080/Hudson 에 있는 허드슨 메인 페이지에서, **New Job** 링크를 클릭하여라. 그림 13은 그 다음에 나타날 화면을 보여주고 있다. 여기에서, 당신은 새로운 빌드 업무(build job)에 이름을 할당한다. 새로 빌드 업무 만드는 데에는 몇 가지 옵션이 있다. 하지만, 이 글의 범위에서는, **Build a free-style software project** 라고 이름 붙여진 타입에 집중할 생각이다. 내가 종종 사용하는 다른 옵션 타입 하나는 **Copy existing job** 이다. 이 타입은 이미 존재하는 빌드 업무를 복사해서 새 빌드 업무를 만들고자 할 때 손 쉽게 사용 가능한 타입이다.

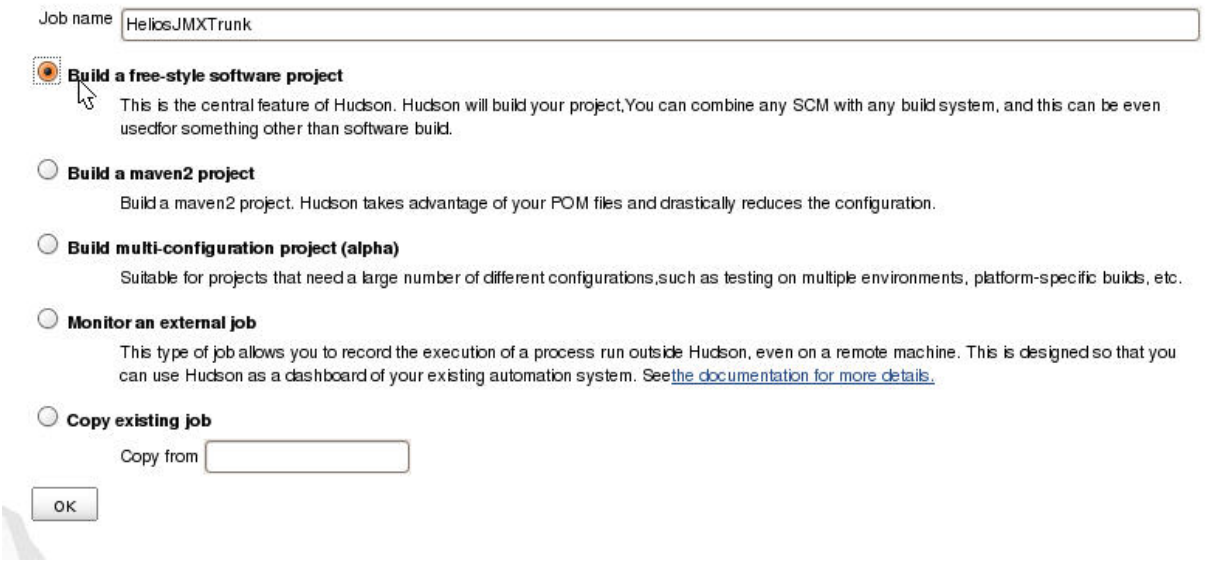


그림 13. 새로운 업무(Job) 이름 (마찬가지로 클릭하면 확대 됨)

본 예제에서는, java.net의 Subversion에 저장되어 있는 HeliosJMX 프로젝트 용 빌드 업무를 세팅할 예정이다. 나는 trunk에서 바로 꺼낸 신선하고 따끈따끈한 소스를 빌드할 예정이기 때문에, 해당 빌드 업무를 **HeliosJMXTrunk** 라고 이름 붙였다. 그렇게 이름을 입력한 다음, **OK** 를 클릭했다.

다음으로는 새로운 빌드업무에 대한 다량의 설정 항목이 꽤 길게 이어진다. 이 설정의 상세한 내용은 아래 목록과 같다. 이 폼에 있는 각각의 옵션 오른쪽에는 작은 물음표(?) 아이콘이 있다. 어떤 설정 옵션이던지 좀 더 상세화된 설명을 보려면 망설이지 말고 클릭해라. 클릭한다고 해서 다른 페이지로 이동되어 버리는 식으로 진행중인 작업내용이 소실되거나 하진 않는다. 대신에, 허드슨은 도움말을 바로 페이지 안으로 간단히 나타나게 한다. 그래서 도움말 기능(Help)을 사용하는 것은 무시해도 좋을 정도로만 간섭한다. 사실상 이 모든 설정들은 윈도우즈나 리눅스 둘 다 동일하다.

- **프로젝트 이름(Project name)**: 난 이미 HeliosJMXTrunk 라고 프로젝트 이름을 지었지만, 원할 경우 여기에서 바꿀 수 있다.
- **설명(Description)**: 빌드 업무에 대해 기술할 수 있는 자유 형식 폼 필드
- **오래된 빌드 버리기(Discard old builds)**: 허드슨은 이 체크박스를 선택하지 않는다면, 이전 기록을 보관할 것이다.
- **이 빌드작업은 파라미터화 됨(This build is parameterized)**: 만일 이 옵션을 선택하면, 허드슨은 빌드 프로세스에 전달 가능한, 이름-값 쌍으로 이루어진 임의의 파라미터 세트 제공을 허락 할 것이다. 설정 파라미터들은 러닝 빌드 환경에서 환경 변수로 지정될 것이다.
- **프로젝트 기반 보안 활성화(Enable project-based security)**: 허드슨은 유저들이 허드슨 웹 페이지를 접근할 때 인증작업을 하기 위한 전체 보안 스키마를 지원한다. 또한 이 보안 스키마는 어떤 유저들이 어떤 빌드 업무를 건드리는 것이 가능한지에 대한 제어를 제공할 수 있다. 나는 이 허드슨 인스턴스에서 보안은 설정하지 않았다.
- **빌드 불가(Disable build)**: 이 항목이 체크되면, 옵션이 비 활성화 될 때까지 이 빌드 업무는 실행되지 않을 것이다.
- **고급 옵션(Advanced options)**: 이 옵션 중에 내가 사용하는 것은 없다. 하지만 이 체크 박스를 선택하면, 다음과 같은 옵션들이 인터페이스에 나타나게 된다.
    - 정숙 기간(Quiet Period): 빌드 수행이 예정되었을 때 실제 수행 이전에 발생하는 정숙 기간(=휴지기간)을 설정할 수 있다. (즉, Delay time 과 동일의미, 0 이면 build now! 의 의미)
    - 커스텀 워크스페이스를 사용(*Use custom workspace)*: 기본적으로, 허드슨은 ${jboss-home}/.hudson/jobs/[project name] 에 빌드 업무용 워크스페이스를 만든다. 이 옵션은 당신 다른 장소를 지정하는 것을 가능케 해준다.
- **소스 코드 관리(Source code management)**: 기본적으로 사용 가능한 세 가지 옵션은 다음과 같다.
    - Subversion
    - CVS
    - None

None 옵션은 소스 코드 저장소가 선행조건이라고 앞서 말한 나의 단정사항에 위배된다. 그러나

대부분의 경우에 있어, 허드슨과 함께 어떤 유용한 일을 하려면 종류는 어찌되었든 저장소가 필요하다는 걸 여전히 지지한다. 이 글 뒷부분에서, 허드슨 플러그인들이 어떻게 설치되는지 소개할 것이다. 만약 당신이 SCM 관련 플러그인을 설치한 다음 허드슨을 재 시작하면, 이 목록에 새로운 옵션이 보이게 될 것이다. 이 글에서, 나는 Subversion 을 사용하고 있다. 당신이 Subversion을 선택한다면, 설정 섹션이 폼에 확장되어 나타날 것이다. 다음 섹션에서 이 섹션에 대한 설정을 따로 소개할 것이다. ("서브버전 업무 설정"을 볼 것)

- **다른 프로젝트들이 빌드 된 다음에 빌드(Build after other projects are built)**: 이 옵션은 (한 업무가 다른 업무의 결과물에 의존을 갖는 업무 의존성) **조립 라인**이나 몇 개의 관련 프로젝트를 그룹으로 묶어 간단히 함께 빌드하길 원하는 것 같은 **시나리오**를 지원한다. 이것을 선택하면, 이 빌드가 실행하기 전에 먼저 실행할 콤마(,)로 분리된 다른 프로젝트 이름들을 입력하기 위한 필드가 제공될 것이다.
- **SCM 폴링(Poll SCM)**: 이것은 CI 시스템들을 위한 고전적인 옵션이다. 이 옵션을 선택하면, 얼마나 자주 허드슨이 당신의 저장소 변경을 체크할 것인지를 정의하는 크론 표현식(cron expression)을 지정할 수 있다. 만일 변경이 발견되면, 빌드가 수행될 것이다. 예를 들어, 표현식이 0,15,30,45 * * * * 이라면 허드슨은 매 15 분 마다 변경을 위해 당신의 저장소를 체크할 것이다. 크론 문법에 대한 좀더 상세한 내용은 the Quartz CronTrigger javadoc 을 살펴보길 바란다.
- **주기적인 빌드(Build periodically)**: (마찬가지로 크론 표현식으로 정의된) 이 옵션은 허드슨에게 SCM 의 변경과 관계없이 특정 주기로 프로젝트를 빌드할 것을 간단하게 지시한다. 이 옵션은 SCM 은 비교적 고정적인데 반해, 테스트 환경은 어떤 식으로든 주기적으로 생기는 경우, 그런 테스트의 실행을 하고자 할 때에 도움이 될 수 있다.
- **빌드 단계 추가하기(Add build step)**: 빌드 스크립트를 실행하기 위한 지시사항을 추가하려면 이 버튼을 클릭하여라. 지시사항은 다음 중 하나가 될 수 있다.
    - 쉘 실행하기
    - 윈도우 배치 명령어 실행
    - Ant 호출 (이게 내가 사용할 옵션이다. 자세한 내용은 아래에서 설명할 것이다)
    - 탑 레벨 Maven 타켓 호출
- **아티팩트 아카이브 하기(Archive the artifacts)**: 이 옵션을 선택해서 파일이나 디렉터리 마스크(mask = include 와 exclude 를 지정할 수 있는 Ant 스타일의 마스크)를 지정할 때, 마스크와 일치하는 아티팩트들은 빌드가 완료되면, 허드슨 아티팩트 저장소에 추가되고 빌드 업무와 빌드 시퀀스 넘버를 사용해서 키 값(key)으로 만들어지게 될 것이다. 옵션에 따라, 허드슨 서버의 디스크 공간을 절약하기 위해서 성공적으로 빌드된 이전의 모든 아티팩트들을 버릴 수 있다.
- **사용 추적을 위해 파일들의 사용 흔적을 기록 (Record fingerprints of files to track usage)**: 이 옵션을 선택하면, 비슷한 형태의 Ant 스타일 마스크(mask)를 사용해서, 허드슨으로 하여금 생성된 아티팩트들의 사용흔적(fingerprints)을 유지하도록 지시할 수 있다. 그렇게 해서, 시스템 내의 다른 어느 곳에서 이들 아티팩트들이 사용되는지를 당신이 좀 더 쉽게 추적할 수 있게 해준다.
- **Javadoc 만들어내기(Publish javadoc)**: 만일 당신의 빌드 스크립트가 javadoc 콘텐트를 생성해낸다면, 이 옵션은 허드슨으로 하여금 해당 콘텐트를 만들어 낸 다음 곧바로 업무 홈

페이지에 게시할 것을 지시할 것이다. 빌드가 성공한 모든 경우에 있어 javadoc 콘텐트는 계속 유지될 것이다. 그러나 기본적으로 마지막에 해당하는 것만 유지된다.

- **JUnit 테스트 결과 보고서 만들어내기(Publish JUnit test result report)**: 만일 당신의 빌드 스크립트가 JUnit 테스트를 수행한다면, 이 옵션은 허드슨으로 하여금 XML 테스트 결과 문서를 처리해서 각각의 성공적인 빌드의 진행 보고서를 생성해 내고, 지속적으로 결과들을 모으도록 지시한다. 모인 결과가 바로 업무 홈 페이지에 있는 보고서에 해당하며, 오랜 시간에 걸쳐 수행된 빌드 업무용 단위 테스트의 기록이 될 수 있는 트랜드(historical trends)로 보여주는 보고서이다.
- **다운스트림 테스트 결과 모으기(Aggregate downstream test results)**: 어떤 경우에는, 한 업무의 단위 테스트 스위트의 수행 시간이 실제 빌드 시간 보다 엄청 더 길다. 이런 경우에는, 빌드와 테스트를 각각 다른 업무로 분리하는 것을 선택할 수 있다. 그래서 상대적으로 빌드가 빠르게 종료될 수 있도록 말이다. 하지만, 구현되어 있는 하나 이상의 테스트 업무들은 빌드가 성공적으로 완료되면 그 즉시 수행되게 된다. 만일 당신이 이 옵션을 선택하면, 허드슨은 모든 빌드 후속 업무들의 테스트 결과들을 모으고, 소급 적용하는 식으로 해당 내용들을 주요 빌드 업무에 대한 테스트 결과로 보고할 것이다.
- **다른 프로젝트 빌드하기(Build other projects)**: 앞선 항목들과 관련되어서, 이 옵션은 하나의 논리적인 빌드와 테스트 프로세스를 연속적으로 수행되어야 하는 두 개 이상의 물리적인 업무로 구현하기 위해 사용된다. 이 옵션을 선택되면, 필드가 보여지게 될 텐데, 이 필드에는 현재 업무 이후에 수행되길 원하는 업무 이름들을 콤마(,)로 분리해서 기입하면 된다. 이 작업은 비록 현재 업무 수행이 빌드가 불안정하다고 판단할 지라도 선택한 내용에 따라 수행된다.
- **이메일 공지(E-mail notification)**: 이 옵션을 선택하면, 하나 혹은 그 이상의 공백으로 분리된 이메일 주소를 입력할 수 있고, 해당 이메일로 허드슨의 업무 수행 완료 공지가 보내지게 된다. 빌드 실패, 불안정 빌드 등의 이벤트들은 이메일 발송을 일으키게 된다. 여기에 있는 추가적인 옵션은 허드슨이 판단하기에 빌드를 깨뜨린 코드를 SCM 에 체크인 한 작업자에게 "특별한" 이메일을 보내도록 만드는 것이다.

이것들은 숙지해야 하는 적지 않은 정보들이다. 다음으로, 이 모든 것이 어떻게 작동하는지 좀 더 잘 이해하기 위해서, 빌드 업무의 예제를 살펴 볼 것이다. 이를 통해, 설정 옵션이 실제로 의미하는 것이 무엇인지를 알 수 있을 것이다.

허드슨에서 빌드 업무 설정하기: 예제

이어지는 내용은 내 허드슨 서버에 설정되어 있는 실제 빌드 업무의 예제이다. 이 빌드 업무는 HeliosJMX라고 불리는 내가 작성한 유틸리티 라이브러리를 빌드한다. 이 섹션에 있는 모든 이미지들은 이전 섹션에서 살펴본 New Job 화면에서 따왔다.

그림 14에서, 프로젝트 이름, 설명, 그리고 허드슨이 마지막 5개의 빌드만 유지하도록 하는 버림 정책(discard policy)을 볼 수 있다.

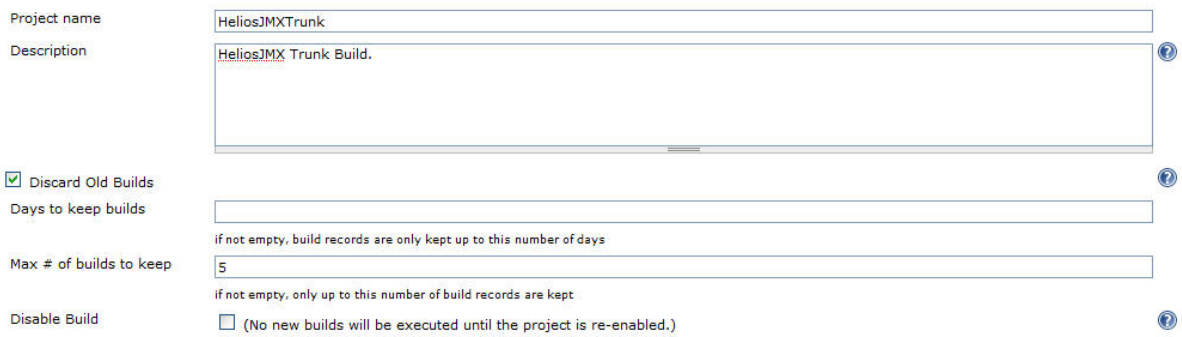


그림 14. 업무 셋업 예제, 파트 1

그림 15는 해당 업무(job)의 서브버전 셋업을 보여준다. java.net 에 있는 서브버전 저장소용 URL은 https://helios.dev.java.net/svn/helios/helios-jmx/trunk 이다. 로컬 모듈 디렉터리(**Local module directory**) 프로퍼티는 업무 워크스페이스에 만들어지게 되는 선택적이고 추가적인 하위 디렉터리 이다.

Use update 체크박스는 매우 중요하다. 허드슨이 빌드 수행용 워크스페이스를 준비하기 위한 가장 빠른 방식은 간단하게 서브버전 저장소로부터 해당 디렉터리를 refresh 하는 것이다. 이 작업은 대부분의 경우에 적절하며 매우 빠르다. 그러나 어떤 인스턴스들에서는, 워크스페이스가 업데이트 될 때에 저장소로부터는 이미 제거된 소스 아티팩트들이 남아있을 수도 있다. 차선책은 이 옵션을 불가로 만드는 것이다. 해당 경우에 있어 허드슨은 워크스페이스를 깨끗이 비우고 저장소로부터 다시 불러올 것이다.

마지막 옵션은 FishEye 나 VisualSVN 같은 소스 저장소 브라우저를 할당한다. 만약 당신 이들 제품 중 사용 가능한 한 것이 있다면, 적용 가능한 해당 브라우저를 선택해서, 당신의 소스 저장소를 지정해 놓아라.

그림 15. 업무 셋업 예제, 파트 2

그림 16에서 내가 선택한 빌드 트리거를 볼 수 있다. 나는 매 5분마다 서브버전을 조사해서 변경사항이 있으면 빌드작업을 수행하길 원한다.

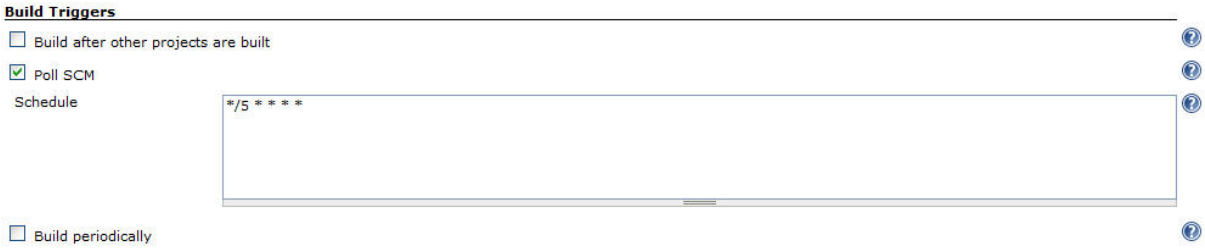


그림 16. 업무 셋업 예제, 파트 3

그림17은 빌드 프로세스를 수행하기 위해 내가 정의해 놓은 Ant 태스크의 셋업을 보여준다. 설정 옵션들은 아래와 같다.

- **Ant 버전(Ant version)**: 빌드를 수행 시에 사용하려고 미리 정의해 놓은 Ant 인스턴스
- **타겟들(Targets)**: 지정한 Ant 스크립트 중에서 수행되어야 하는 타겟들의 목록. 스크립트의 디폴트 태스크가 실행될 경우에는, 빈 칸으로 남겨 놓을 수 있다.
- **빌드파일(Build file)**: 유효 워크스페이스를 기준으로 놓고 봤을 때, 수행되어야 하는 Ant 스크립트의 상대적인 위치.
- **프로퍼티들(Properties)**: Ant 스크립트로 전달될 추가 시스템 프로퍼티 정의값. 나는 서브버전 인증정보를 스크립트로 전달하기 위해 이들 프로퍼티들을 사용했다. 왜냐하면 내 작업 절차에는 몇 가진 변경사항들을 저장소로 반영하는 단계가 포함되어 있기 때문이다. 추가적으로, 나는 내 단위 테스트들 위한 설정 파라미터들에 해당하는 몇몇 파라미터들을 정의했다.
- **자바 옵션들**: 자바 명령행 옵션들을 여기에서 전달할 수 있다. 여기에서 나는 Ant 의 -debug 옵션을 할당했다. 왜냐하면 나는 Ant 스크립트에 있는 디버깅 문제에 대해 작업하고 있었기 때문에, 해당 옵션을 사용해서 Ant 가 추가적인 분석정보들을 로그에 생성하게 하였다. 다른 흔한 옵션은 기동되는 JVM 의 초기 힙 사이즈(-Xms)와 최대 힙사이즈(-Xmx)를 특정하는 지시문 들이다. 이 옵션은 빌드 스크립트를 수행하기 위해 허드슨에 의해 실행되는 모든 새로운 JVM 인스턴스가 따르게 된다.

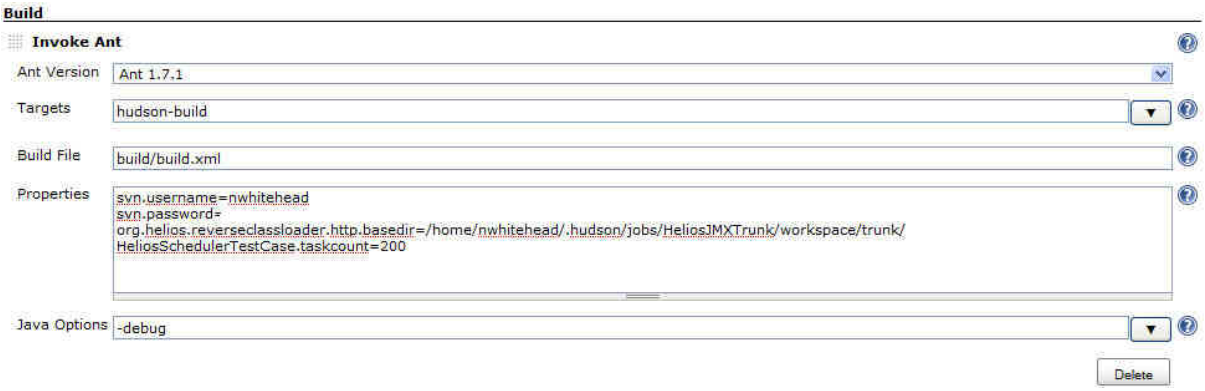


그림 17. 업무 셋업 예제, 파트 4

그림18은 내가 정의한 빌드 후속 액션들을 보여준다.

- **아티팩트 아카이브하기(Archive artifacts)**: 허드슨에게 빌드가 완료되면 해당 아티팩트들을 아카이브 할 것을 지시한다. 이것은 Ant 스타일의 파일 마스크로, 유효한 워크스페이스에 상대적인 디렉터리와, 파일이름들, 그리고 아카이브 될 확장자들을 지정한다. 아카이브된 아이템들은 업무 빌드 인스턴스 홈 페이지를 통해 접근 가능하게 될 것이다. 고급 옵션(advanced options)은 마스크에서 제외될 것들을 지정 가능하게 해주며, 마지막으로 성공한 빌드에서 생성된 것들을 제외한 모든 아카이브 된 아티팩트들을 지우는 옵션을 갖는다.
- **javadoc 만들어내기**: 이전 옵션과 비슷하다. 하지만 빌드 프로세스에 의해 생성된 임의의 javadoc 콘텐트에 대해 적용 가능하다.
- **JUnit 테스트 결과 보고서 만들어내기**: 허드슨에게 미리 정의된 위치에 JUnit XML 결과 문서를 얻어내고 기록이 될 수 있는 트랜드 보고서로 취합할 것을 지시한다.

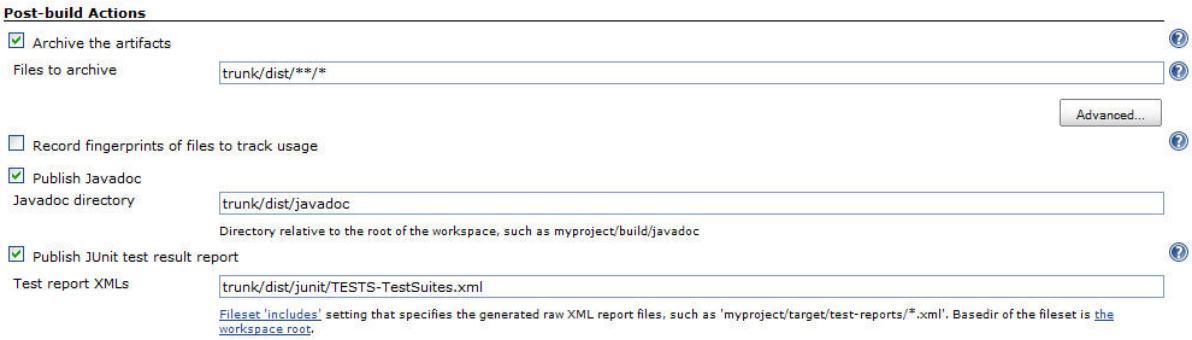


그림 18. 업무 셋업 예제, 파트 5

그림19는 내가 정의해 놓은 빌드 후속 액션들을 좀 더 많이 보여준다.

- **FindBugs 분석 결과 만들어내기**: 나의 빌드 스크립트는 해당 빌드 업무와 연관되어 있는 소스코드에 대한 FindBugs 정적 코드 분석을 수행해서 findbugs 보고서를 생성한다. 이 옵션은 허드슨 FindBugs 플러그인이 설치된 경우에 사용 가능하다. 이 옵션은 허드슨으로 하여금 정의된 FindBugs XML 결과 보고서를 불러와서 업무 홈 페이지에 게재되는 히스토리컬 FindBugs 경향 보고서로 취합할 것을 지시한다. FindBugs 플러그인용 고급 옵션은 보고되는 FindBugs 단정 항목들과 허드슨이 판단하는 해당 빌드 업무의 상태에 관한 최종 결정에 그 단정 항목들이 어떻게 영향을 끼치게 할 것인지에 대한 카테고리를 정할 수 있게 해준다.
- **이메일 공지**: 빌드 실패 시 공지를 보낼 공백으로 분리된 이메일 주소를 지정한다. 빌드 업무가 영구적으로 불안정한 상태이거나 깨진 상태일 때, **Send email for every unstable build** 옵션을 언 체크 하는 것은 이미 알고 있는 상태에 대해서 계속적으로 공지가 되는 것을 막아준다.
- **Cobertura 커버리지 보고서 만들어내기**: 나의 빌드 스크립트는 생성된 클래스를 조율하기 위해 코드 커버리지 지시자로 Cobertura 를 사용한다. JUnit 테스트가 실행할 때, Cobertura 는 코드 커버리지를 추적하고 테스트가 완료되면 커버리지 보고서를 생성한다. 이 옵션은 허드슨 Cobertura 플러그인이 설치되면 사용 가능하다. 이 옵션은 허드슨으로 하여금 정의된 Cobertura XML 커버리지 리포트를 불러와서 마찬가지로 업무 홈페이지에 기록으로 남을 수 있는 Cobertura 트랜드 보고서로 게재하기 위해 취합된다. **Coverage Metric Targets** 이라고 이름 붙은 섹션은

빌드 업무의 상태에 대한 허드슨 최종 결정에 영향을 미치는 **코드 커버리지 맵**의 레벨을 정하는 방법을 지정할 수 있게 해준다. (이것에 관해서는 '빌드 업무의 상태' 섹션을 보아라)

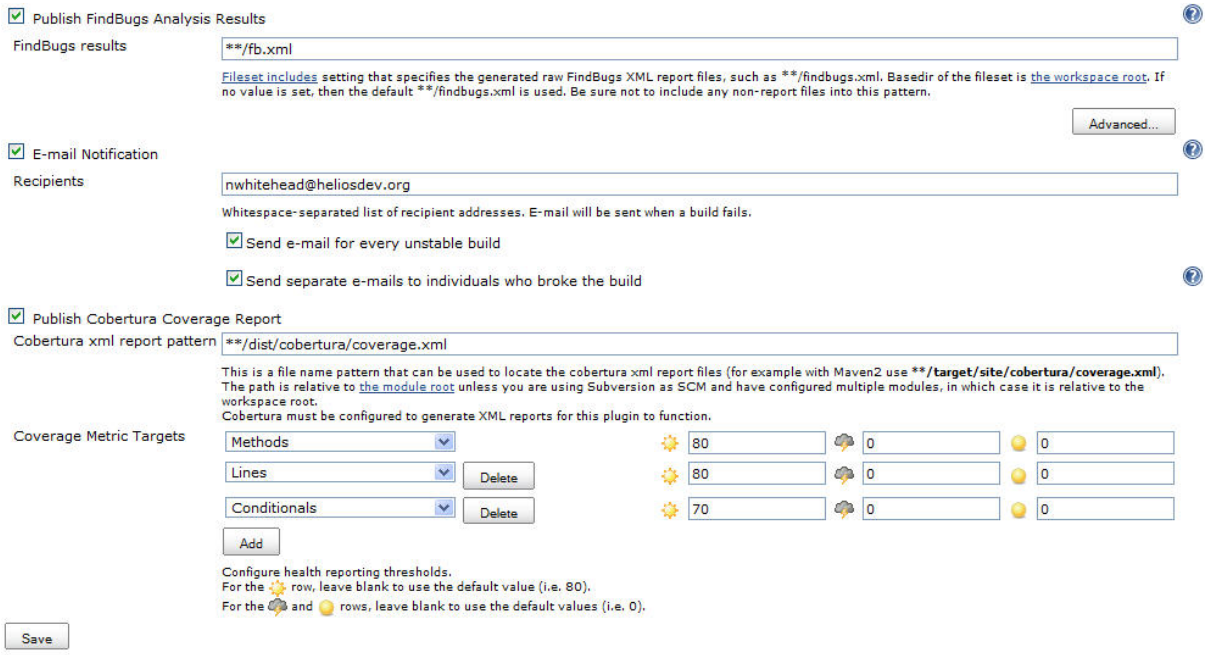


그림 19. 빌드 업무 셋업 예제, 파트 6

이 시점에서, 만일 당신이 새 업무를 정의를 마쳤다면, 실행해서 어떻게 동작하는지 볼 시간이다. 당신의 빌드 트리거들이 어떻게 설정되었든 상관없이, **Disable Build** 를 체크하지 않는 한 언제든 애드 혹(ad hoc, 특별한)빌드를 요청할 수 있다. 당신이 새로운 빌드 업무를 구축하거나 위에 있는 스크릿샷에서 보이는 것처럼 내가 했던 디버깅을 구축할 때, 빌드 트리거를 기다리는 것은 사리에 맞지 않는다. 다음 섹션에서, 나는 빌드를 실행하도록 애드 혹을 요청하는 방법과 수행되고 있는 빌드 업무를 볼 수 있는 방법을 보여줄 것이다.

빌드 업무 실행과 지켜보기 Running and watching a job

최근에 만든 빌드 업무의 첫 번째 빌드를 실행하기 위해서, http://localhost:8080/hudson 에 있는 허드슨 대시보드로 이동한다. 그림 20은 내 우분투 서버에 본인이 구축한 새로운 허드슨 인스턴스의 브랜드를 보여주고 있다.

그림 20. 허드슨 대시보드(Hudson dashboard)

내가 정의해 놓은 HeliosJMXTrunk 빌드 업무가 화면 중앙에 보인다. 이 화면에서 주목할 만한 몇 가지 항목은 다음과 같다.

- 회색의 원형 아이콘은 빌드 업무의 상태를 나타낸다. 본 예제에서는, 한 번도 빌드가 일어난 적이 없기 때문에 회색이다.
- 왼쪽에 **빌드 큐(Build Queue)**라고 라벨이 붙여져 있는 테이블은, 현재 실행 중이거나 실행하기 위해서 대기 열에 들어있는 빌드 업무를 보여준다. **빌드 수행 상태(Build Executor Status)**라고 라벨이 붙은 테이블은 할당된 빌드 쓰레드의 상태를 보고한다. 기본값으로, 허드슨은 두 개의 빌드 쓰레드를 할당하며, 이것은 허드슨이 동시에 두 개까지 빌드 업무를 수행할 수 있음을 의미한다. 이 숫자를 수정하려면, **Manage Hudson** 을 클릭한 다음 **Configure Executors** 를 클릭한다. 소프트웨어 빌드는 꽤나 자원 집약적인 경향이 있는 관계로, 쓰레드 숫자는 조심스럽게 설정해야 한다. 동시 빌드를 너무 많이 지정하면 허드슨 서버에 적당하지 않은 수준의 부하를 주게 될 것이고, 결과적으로 당신의 빌드 업무를 느리게 만들 것이다.
- RSS 피드(feed) 아이콘을 주목하여라. 허드슨은 이벤트 공지 방법 중 하나로 RSS 피드를 제공한다. 그리고 이 피드에는 전반적인 시스템용 피드와 각각의 개별적인 빌드 업무 보고용 피드가 있다.
    - 성공적인 빌드
    - 불안정한 빌드
    - 깨진 빌드
    - SCM 변경

빌드를 요청하기 위해서는, 빌드 업무 리스트 테이블의 오른쪽 끝에 있는 빌드 아이콘을 클릭하면 된다. 혹은, 빌드 업무 이름을 클릭해서 빌드 업무 홈페이지로 이동한 다음, **Build Now** 라고 이름 붙은 아이콘 링크를 클릭할 수도 있다.

한번 빌드 업무가 실행되면, 실행중인 빌드 업무 리스트를 대시보드나 빌드 업무 홈 페이지에 있는 대기열(Queue)에서 볼 수 있다. 이들 두 개의 뷰는 그림 21에서 볼 수 있다.

그림 21. 실행중인 빌드 업무에 대한 두 개의 서로 다른 뷰

전형적인 형태로, 어떤 시점에서 당신은 현재 실행중인 빌드 업무의 결과물을 봄으로써 빌드 업무의 진척

상황을 보길 원할 것이다. 그렇게 하려면, 빌드 업무 홈페이지로 이동해서 **Console Output** 을 클릭하여라. 만일 빌드 업무가 끝나면, 빌드 스크립트에 의해 생성된 정적인 결과물을 보여줄 것이다. 하지만 만약 빌드 업무가 여전히 실행 중이라면, 허드슨은 계속해서 페이지의 내용을 갱신해서 현재 발생하는 결과물들을 볼 수 있게 해줄 것이다. 그림 22는 매우 유용한 해당 기능에 대해 보여주고 있다.

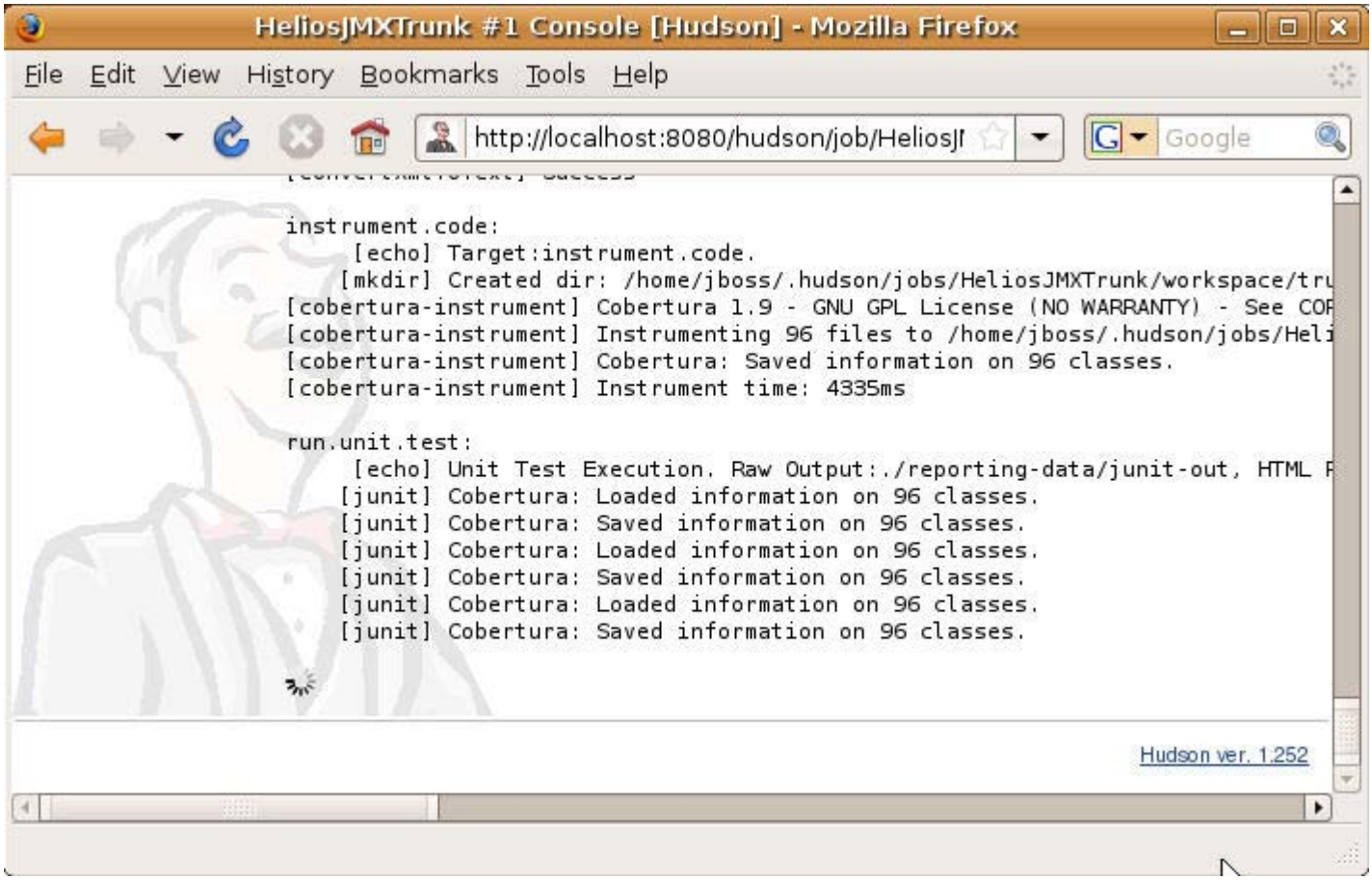


그림 22. 실행중인 빌드 업무의 결과물을 보여주는 라이브 콘솔

빌드가 한 번 완료되고 나면, 세 곳에서 빌드 업무 완료가 보여질 것이다.

- 그림 23 처럼, 허드슨 대시보드에서 해당 내용을 볼 수 있다.
- 그림 24 처럼, 빌드 업무 홈 페이지에서 해당 내용을 볼 수 있다.
- 빌드 이력에 있는 특정 빌드 링크를 클릭해서, 허드슨이 빌드 인스턴스용으로 특별해 생성한 해당 빌드의 홈 페이지로 이동할 수 있다. 이 내용은 그림 25 에서 보여주고 있다.

그림 23. 완료된 빌드 업무를 보여주고 있는 허드슨 대시보드

대시보드에 있는 심볼들의 의미는 "빌드 업무 상태(Job state)"라고 이름 붙은 아래의 섹션에 요약되어 있다. 간략하게 말하자면, 노란 원은 빌드가 성공은 했지만 불안정하다는 것을 나타낸다. W 컬럼에 있는 작은 태양 아이콘은 "날씨"를 나타내는데, 빌드가 성공했고 후속 작업 플러그인들이 허드슨에게 문제사항이 없다고 보고했기 때문에 해당 날씨는 "햇볕 쨍쨍"이다. 여기에서 또 하나 주목할 내용은 마지막 빌드의 지속 시간이 15분이었다는 점이다. 그렇게 된 이유는 의도적으로 오래 실행되게 스케줄 잡혀있는 컴포넌트용 단위 테스트가 내 테스트 스위트 내에 들어 있기 때문이다. 이 업무는 빌드와 테스트를 나누어 놓은 좋은 예가 될 수 있을 것이다.

그림 24. 완료된 빌드 업무를 보여주는 빌드 업무 홈 페이지

빌드 업무 홈 페이지는 몇몇 흥미로운 항목을 포함한다. 왼편에 있는 링크들은 (**New Job** 에서 정의한 것을 수정하기 위해서) 해당 빌드 업무 설정용 명령어와 빌드 업무를 삭제하고, 해당 빌드 업무를 실행하기 위한 명령어들이다. 오른편에 있는 것들은 가장 최근의 프로젝트와 아티팩트들에 대한 링크들이다.

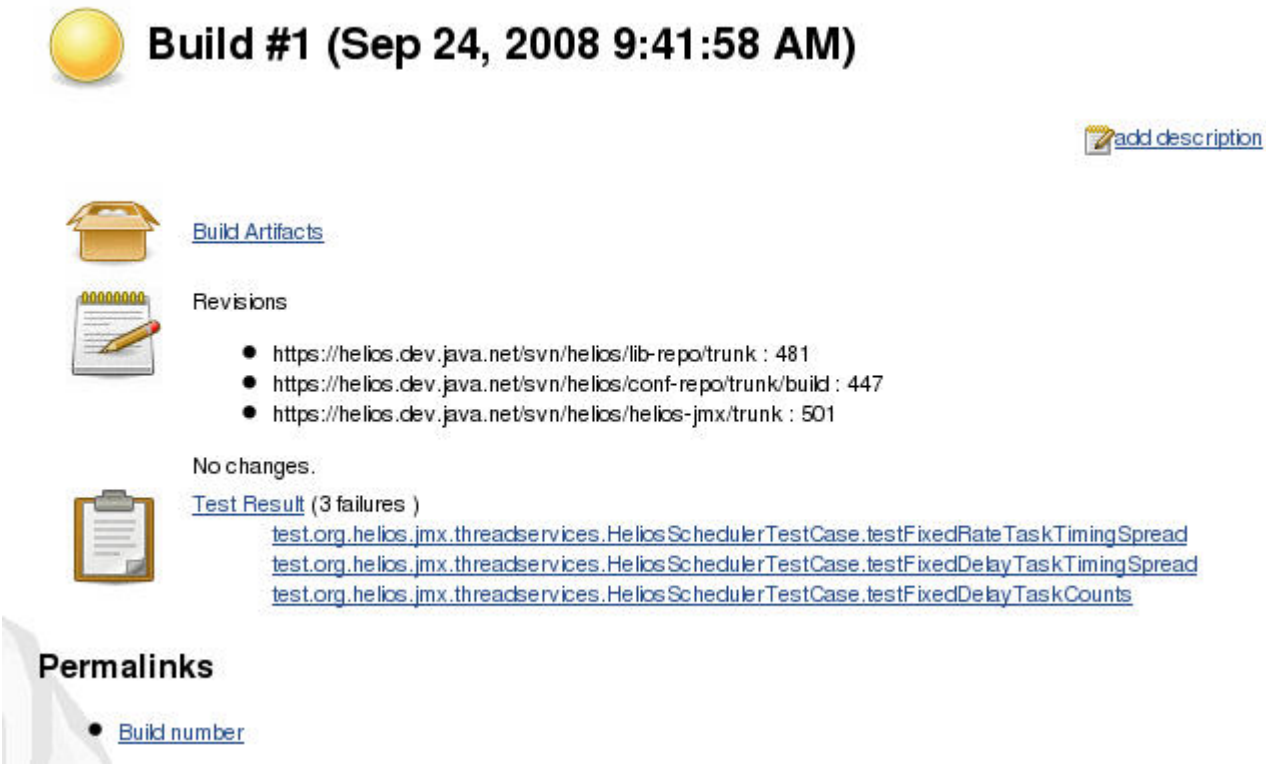


그림 25. 완료된 빌드를 보여주는 빌드 홈 페이지 Figure 25. The build home page for the completed build

빌드 홈 페이지는 해당 빌드 인스턴스를 지정한다. 허드슨이 내부적인 빌드 넘버를 할당한다는 점에 주목하여라. 이 빌드 넘버는 분산된 빌드들을 추척하는 데에 유용하다. 이 페이지는 실패한 세 개의 JUnit 단위 테스트 케이스뿐 아니라 서브버전에서 발견된 리비전들(revisions)도 목록으로 보여준다. (이번 빌드에서 리비전 변경은 없었다). 여기에서 이 들 세 개의 실패가 허드슨이 해당 빌드를 '불안정상태(unstable)'로 표시하게 된 원인이다. 불안정 상태란 해당 소프트웨어가 에러 없이 빌드는 되었으나, 단위 테스트는 에러가 있음을 의미한다. 그림26은 단위 테스트에 에러가 없고, 그래서 안정된 빌드(stable build)로 지정된, 새로운 빌드가 끝난 후의 대시보드와 프로젝트 페이지의 섹션들을 보여준다.

그림 26. #2 빌드 후의 대시보드와 빌드 업무 홈 페이지

나는 이제 빌드 업무를 설정하고 빌드 트리거를 지정하고 애드 혹(ad hoc) 빌드를 요청하기 위해 필요한 모든 단계들을 아우를 수 있게 되었다. 남은 것은 허드슨이 빌드 업무의 상태를 표시하기 위해 사용하는 스키마에 대해 짧게 논의해 보는 것이 남았다.

빌드 업무 상태(Job states)

위에 보이는 스크린샷에서, 빌드 업무의 현재 상태를 나타내는 두 개의 아이콘을 보았을 것이다. 허드슨은 빌드 업무의 전체적인 현황을 나타내기 위해서 두 개의 개념을 사용한다.

- **빌드 업무 상태(Job state)**: 그림 27 은 가장 최근에 수행된 빌드 업무가 가질 수 있는 네 개의 상태 심볼을 요약해 보여준다.
    - 성공(*Successful*): 빌드가 완료되었고 안정적으로 간주됨
    - 불안정(*Unstable*): 빌드가 완료되었고 불안정한 것으로 간주됨
    - 실패(*Failed*): 빌드가 실패함
    - 사용불가(*Disabled*)빌드 업무가 사용불가임

- **빌드 업무 안정도(Job stability)**: 빌드 업무가 완료되고 이슈 없이 대상 아티팩트를 생성할지라도, 허드슨은 플러그인 형태로 구현된, 그러니까 암묵적으로 안정도를 평가하기 위해 만들어 놓은 후속 작업들에 기반하여 해당 빌드에 (0~100 사이의) 안정도 점수를 부여할 것이다. 이 암묵적인 작업들이란 단위 테스트들(JUnit), 커버리지(Cobertura), 정적 코드 분석(FindBugs)을 포함한다. 해당 점수가 높으면 높을 수록 빌드는 더 안정적이다. 그림 28 은 안정도 점수 범위에 따른 심볼들을 요약해서 보여준다.

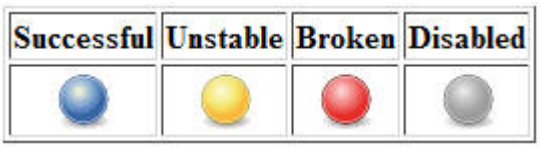


그림 27. 빌드 업무 상태(Job states)

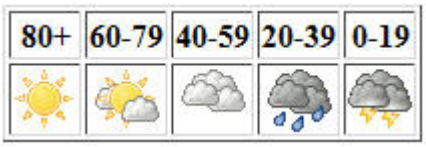


그림 28. 빌드 업무 안정도(Job stability)

빌드 트랙들 Build tracks

허드슨은 복수개의 빌드 트랙을 만들 수 있게 해준다. 당신의 소프트웨어 개발 절차에 따라, 당신은 소프트웨어 프로젝트 당 하나 이상의 빌드 트랙을 만들어 내기를 원할 수 있다. 빌드 트랙은 고유한 설정을 갖고 있는 하나의 특정한 프로젝트나 프로덕트를 위한 빌드 업무이다. 동일한 프로젝트에 대한 빌드 트랙들을 서로 구분 짓는 요소들은 해당 빌드 업무에서 가져온 소스 코드의 SCM 브랜치(branch)가 될 수도 있다. (즉 한 프로젝트 내에서 브랜치가 여러 개 발생하면, 그에 따라 빌드들이 개별적인 '트랙'이라는 형태로 존재하게 될 수 있다는 이야기이다) 다른 식으로는, 서로 다른 트랙들이란 같은 소스에 대해 수행하게 되는 서로 다른 작업들의 세트가 될 수도 있다. 논리적으로 동일한 소프트웨어 프로젝트에 대해 어떻게 서로 다른 빌드 트랙들을 만들지를 지시하는 것에 따른 영향은 다음과 같다.

- **소스 컨트롤 브랜치들(Source control branches)**: 많은 개발 팀들은 그들의 소스 컨트롤 시스템에서 소스 코드의 여러 버전에 대한 개발을 동시에 깔끔히 지원하기 위해서 일련의 브랜치들을 유지해 나간다. 이와 같은 경우에, 오래되어 사용하지 않는 브랜치들에 대해서 뿐 아니라, 개발에 사용되는 각각의 브랜치에 대해서 분리된 빌드 트랙을 만드는 것은 사리에 맞는다. (여기에서 구 버전들을 관리하는 이유는 소프트웨어의 구 버전들이 카탈로그화 되어서 경우에 따라서는 쉽사리 다시 만들어 내기 위해서이다)
    - 트렁크 빌드: 가장 최신의 활성화된, 그리고 종종 변경이 일어나는 기반 소스
    - 릴리즈 빌드: 지속적인 릴리즈에 한정되어 있는 개별 분리된 브랜치. 소스 코드는 대부분 안정화 되어 있다. 하지만 트렁크 개발은 향후 릴리즈를 위해서 지속적으로 변경된다.
    - 버그 수정 빌드: 메인라인(=trunk) 활동으로부터 격리된 상태에서 버그를 수정하기 위해 생성된 개별 브랜치
    - 실험적인 빌드: 기반 소스에 실험적인 아이디어를 테스트 하기 위해서 생성된 개별 브랜치. 결국엔 메인라인 소스가 되지 않을 수 있음.
- **빌드 업무 작업 카테고리들**: 다른 소스 기반에 대한 빌드는 어떠한 특정 브랜치들의 현재 활동이나 상태에 따라 다른 빌드 세트나 연속적인 빌드 작업이 필요할 수도 있다. 예를 들면, 다음과 같은 경우에 개별 트랙들을 구축하길 원할 수도 있다.

  - 안정성에는 관심이 없지만 빌드 여부와 빌드 아티팩트들이 사용가능 여부를 정기적으로 심플하게 확인하길 원하는 경우 같은 **간단한 빌드 확인 업무(Simple build verification jobs)**
  - 다소 띄엄띄엄 수행할 수도 있는, 그리고 아티팩트들을 빌드하고 소스 코드와 빌드된 아티팩트들에 대해서 풀 테스팅 스위트를 수행하는, **빌드와 안정성 확인 빌드(Build and stability verification builds)**
  - 전체 빌드와 안정성 테스트 스위트들을 수행하고 배포를 위해 전달하게 되는 것들을 패키징하기 위한 일련의 작업들을 완료하는 **릴리즈 후보 업무(Release candidate jobs)**
  - 릴리즈 후보가 배포를 위해 준비되고 패키지 된 아티팩트들이 공개 저장소에 자동적으로 업로드 되었을 때, 전체 릴리즈를 완료하기 위해 필요한 다른 작업들과 함께 수행되는 **전체 릴리즈 업무(Full release jobs)**
- 전달 가능한 고려사항들: 정확하게 동일한 코드 브랜치와 빌드 작업은 다소 다른 툴을 사용해서 빌드될 필요가 있을 수 있다. 예를 들면, 당신은 당신 클래스들의 버전들을 자바 1.5 용과 자바 1.4 용으로 분리해서 릴리즈 하도록 원할 수 도 있다.

그림29는 소스 컨트롤 브랜치들과 그에 관련된 허드슨 빌드 트랙들 사이에서 생길 수 있는 논리적인 구조를 보여준다.

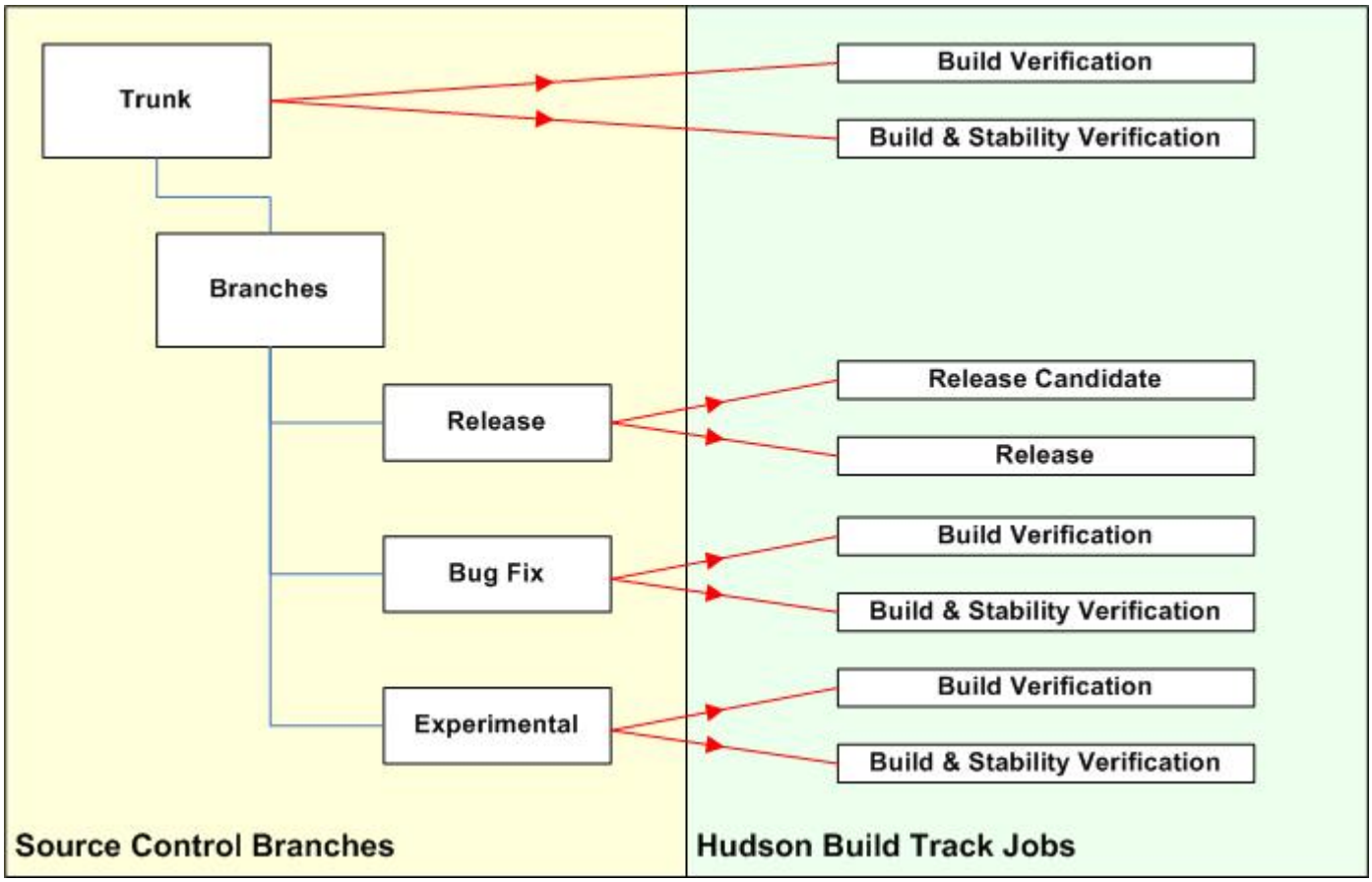


그림 29. 소스 컨트롤 브랜치들과 허드슨 빌드 트랙들

허드슨은 기존 빌드 업무를 복사하는 식으로 새로운 빌드 업무를 쉽게 만들 수 있다. 그렇게 하기 위해서는, 허드슨 대시보드로 가서, New Job 링크를 클릭한다. 새로운 빌드 업무의 이름을 입력한 다음 **Copy existing job** 을 선택한다. 당신이 타이핑을 시작할 때에, 허드슨은 당신이 타이핑하는 것과 일치하는 복사 가능한 기존 빌드 업무 목록을 보여준다. 그리고 나서 OK를 누르면, 새로운 빌드 업무가 만들어질 것이다. 이 절차는 그림 30에서 보여주고 있다.

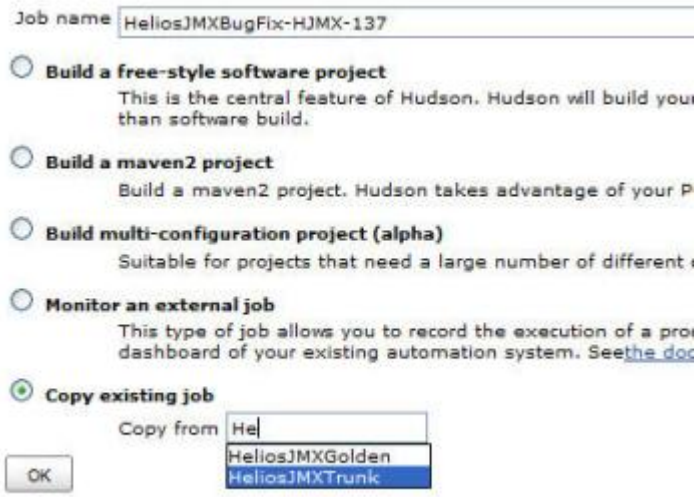


그림 30. 기존 빌드 업무를 복사해서 새 빌드 업무 생성하기

당신은 이미 이런 식으로 새 빌드 업무를 한번 만들었을 텐데, 빌드 업무의 이름을 제외하고는 기존 빌드 업무를 복사한 것과 실제로 동일하다. 그래서 항목을 변경하는 식으로 새로운 복사본을 변경하고 싶을 것이다. 예를 들면 다음과 같은 작업을 원할 수 있다.

- 워크스페이스가 가져온 소스 컨트롤 대상 브랜치를 수정하기
- 허드슨이 실행할 빌드 스크립트나 빌드 스크립트 대상을 수정하기
- 허드슨에서 Ant 스크립트로 넘길 시스템 파라미터 수정하기

뷰 사용하기 Using views

이미 당신은 몇 개의 빌드 트랙을 만들었기에, 당신의 대시보드가 긴 빌드 업무 목록으로 다소 어지럽게 되었다고 느낄 수 있다. 대시보드를 정리하는 쉬운 방법은 뷰(Views)를 만드는 거다. 대시보드 뷰는 대시보드에 분리된 탭 안에서 보여지는 연관 있는 빌드 업무들의 그룹이다. 그리고 그 그룹은 당신이 임의로 정의할 수 있다. 관련 있는 빌드 업무들을 그룹핑하는 뷰를 만들 때, 일관성 있는 빌드 업무 네이밍 컨벤션을 구현하는 것이 유익하다는 것을 알게 될 것이다.

뷰를 만들려면, 대시보드에 + 로 라벨이 붙은 작은 탭을 클릭하여라. 새로운 뷰 페이지에서, 새로운 그룹의 이름과 선택적인 설명을 입력하여라. 당신이 해당 뷰에 포함하길 원하는 특정 빌드 업무를 선택할 수 있도록 허드슨은 현재 설정된 모든 빌드 업무에 대해 라벨 붙은 체크박스를 제공할 것이다.

그러나, 내 생각하기에 빌드 업무의 그룹을 만드는 더 나은 방법은 **Use a regular expression to include jobs in the view(뷰에 빌드 업무를 포함하기 위해서 정규 표현식을 사용하기)**라 고 이름 붙은 체크박스를 클릭해서 당신이 그룹에 포함시키기를 원하는 빌드 업무의 이름을 매치시킬 수 있는 정규 표현식을 적는 것이다. 이게 바로 손쉬운 일관성 있는 네이밍 컨벤션이다. 당신은 일반적인 소프트웨어 프로젝트나 혹은 가능한 경우, 빌드 타입으로 빌드 업무를 하나로 그룹핑하는 뷰를 설정할 수 있다. 나는 후자의 접근법을 선택해, 그림31에서처럼 빌드 업무에 release 라는 단어가 들어간 모든 빌드 업무를 포함하는 릴리즈 빌드(Release Build)라고는 이름의 뷰를 만들었다.

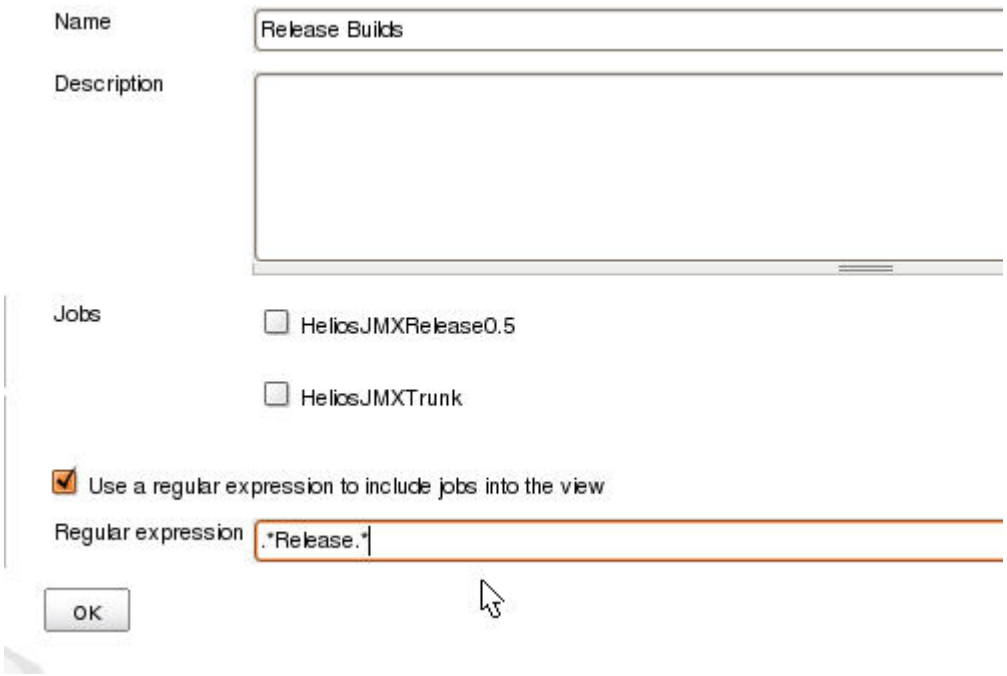


그림 31. 대시보드 뷰 만들기

정규 표현식으로 빌드 업무를 포함시키는 방법을 사용했을 때의 추가적인 이점은 새로운 빌드 업무가 생길 때, 빌드 업무 이름과 매치되는 어떤 뷰에든 자동적으로 추가될 것이라는 점이다. 반면에 특정 빌드 업무를 선택해서 그룹핑하는 뷰는 손수 업데이트를 해주어야 한다.

그림32는 빌드 타입 기반으로 새롭게 정리된 대시보드를 보여주고 있다.

그림 32. 빌드 타입으로 나눈 대시보드 뷰

허드슨 플러그인들 Hudson plugins

*허드슨 플러그인*이라는 말은 허드슨의 기존 기능을 확장하는 라이브러리들과 허드슨에 새로운 기능을 추가하기를 원하는 개발자들용 확장성을 제공하는 방법, 이 두 가지를 다 의미할 수 있다. 어떤 플러그인들은 단순히 당신의 빌드 프로세스에 유용한 추가기능이 될 수도 있고, 혹은, CVS나 Subversion 이외의 다른 소스 컨트롤 시스템을 지원하기 위해 구현한 SCM 플러그인, 환경구축 시에 허드슨을 사용하기 위해 필요한 무언가가 될 수도 있다.

현재 수 많은 플러그인들이 사용 가능한 관계로, 여기에서 모든 리스트를 다루진 않을 것이다. 하지만, 일반적인 플러그인 카테고리는 다음과 같다.

- **SCM**: 허드슨이 CVS 나 서브버전 이외의 소스 컨트롤 시스템을 지원하기 위해 구현한 플러그인들
- **트리거들**: 이벤트를 감시하고 빌드를 촉발하는 플러그인들. 예를 들자면, URL 변경 트리거는 URL 을 감시할 것이다. 해당 주소에 있는 콘텐트가 변경되면, 트리거는 빌드 업무를 수행할 것이다.
- **빌드 도구들**: MSBuild 나 Rake 같은 추가적인 빌드 도구를 구현한 플러그인들. 이 플러그인들은 허드슨에서 자바가 아닌 소프트웨어를 빌드하고자 할 때 매우 유용하다.
- **빌드 랩퍼들(Build wrappers)**: 자신의 빌드 프로세스 전 후에 제어된 이벤트들을 수행하는 것에 특별히 관련된 플러그인들. 예를 들면, VMware 플러그인은 빌드 전에 게스트 VM 을 시작시키고, 빌드가 끝난 다음에 중지시킬 것이다. 단위 테스트들을 수행하기 위해서 게스트 VM 이 필요한 상황에서 유용하다.
- **빌드 공지 관련(Build notifiers)**: 이 플러그인들은 빌드 업무 이벤트 관련 공지를 발행하는 대안 방법들을 지원해 준다. 대안 방법에는 Twitter 나 IRC, 구글 캘린더 이벤트 같은 것들이 있다.
- **슬레이브 런처와 컨트롤러들(Slave launchers and controllers)**: 본 글에서 소개되지 않은 허드슨의 매우 강력한 기능 중 하나는, 마스터 허드슨 인스턴스를 대신해 작업을 수행하는 슬레이브 허드슨을 가지는 능력이다. 현재 이 카테고리에는 단 하나의 플러그인 만이 존재한다. SSH 링크를 통해 슬레이브를 관리하는 SSH 슬레이브 플러그인이 바로 그것이다.
- **빌드 보고서들(Build reports)**: 몇 가지 양식에 기반하여 당신의 소스코드나 생성된 아티팩트를 분석한 유용한 보고서들을 만드는 일련의 플러그인 들이다. 예를 들면, Cobertura 플러그인은 당신의 빌드 스크립트에 의해 생성된 지속적 커버리지 보고서들을 모은다.
- **외부 사이트 통합(External site integrations)**: 허드슨과 Jira 나 버그질라 같은 다른 어플리케이션들을 통합하는 걸 도와주는 플러그인들
- **아티팩트 업로더(Artifact uploaders)**: java.net 파일 저장소나, FTP 서버 같은 어떤 네트워크로 연결된 종단점에 빌드된 아티팩트 배포를 돕는 플러그인들
- **페이지 데코레이터(Page decorators)**: 허드슨 웹 페이지의 미관을 꾸미거나 치장하는데 유용한, 몇몇 플러그인들. 허드슨에서 제공하는 모든 페이지들에 구글 트래킹을 추가하는 구글 애널리틱스(Google Analytics) 같은 플러그인.

허드슨 플러그인 매니저는 당신의 허드슨 서버에 새로운 플러그인들을 인스톨하고 기존 것들을 업데이트 할 수 있게 해준다. 이 매니저는 사용 가능한 플러그인들과 플러그인 업데이트 목록을 가져오기 위해 온

라인 저장소에 접속할 것이다. 만일 당신의 허드슨 서버가 외부 리소스에 접속할 수 없다면, 허드슨 웹사이트에서 당신이 원하는 플러그인들을 다운로드 할 수도 있다. 사이트에서 plugins 폴더를 클릭하면 사용 가능한 플러그인 목록을 보게 될 것이다. 개별 플러그인 파일들은 .hpi 확장자를 갖는다. 당신이 플러그인을 다운받은 다음에는, 허드슨 홈 디렉터리에 있는 플러그인 하위디렉터리에 해당 플러그인을 복사한다. (허드슨 홈 디렉터리는 .hudson 이라고 불리고, 허드슨 서버를 운영하는 사용자의 홈 디렉터리에 위치할 것이다.) 일단 해당 플러그인 파일이 복사되고 나면, 플러그인이 동작하도록 허드슨을 재시자 할 필요가 있다.

허드슨 플러그인 매니저를 사용하기 위해서는, 대시보드에서 **Manage Hudson** 링크를 클릭한 다음, **Manage Plugins** 를 클릭한다. 플러그인 매니저는 그림33에 보여지고, 해당 매니저는 4개의 탭을 갖고 있다.

- **업데이트들(Updates)**: 허드슨이 업데이트 가능하다고 탐지한 플러그인 목록. 리스트에 올라있는 각각의 플러그인은 업데이트를 적용하기 위해서 선택될 수 있다.
- **사용 가능한 것들(Available)**: (현재 설치되어 있지 않은) 설치 가능한 플러그인 목록. 리스트에 올라있는 각각의 플러그인은 인스톨하기 위해서 선택 가능하다.
- **인스톨(Install)**: 이미 설치된 플러그인 목록
- **고급(Advanced)**: 허드슨이 온라인 플러그인 저장소와 통신할 수 있도록 거쳐야 하는 HTTP 프록시를 설정할 수 있게 해준다. 추가적으로, 허드슨 외부에서 다운로드 한 플러그인들을 설치하기 위해 사용할 수 있는 업로드 기능이 있다.

그림 33. 허드슨 플러그인 매니저

일단 당신이 원한 플러그인이나 업데이트가 설치되면, 허드슨은 그것들을 적용하기 위해 재기동 되어야 한다. 만일 당신이 JBoss 를 사용한다면, 최초에 당신이 디플로이한 JBoss 디플로이 디렉터리에 있는 hudson.war 파일을 터치(unix의 touch)하는 걸로 효과를 발휘시킬 수 있다는 걸 주목해라. (touch는 특정 파일의 타임스탬프를 업데이트 하는 유틸리티이다)

어떤 종류들의 빌드 보고서 플러그인들을 이해하기 위한 한 가지 중요한 컨셉은, 그것들이 당신을 위한 핵심 보고서를 반드시 생성하는 건 아니라는 점이다. 그보다는, 보고서들이 생성될 때마다 지속적으로 보고서들을 모으는 추가적인 작업에 관심을 갖는다. 어떤 경우에는, 생성된 보고를 통합된 허드슨 네이티브 보고서로 다시 포맷팅해서 생성해 낼 것이다. 예를 들면, Cobertura 플러그인은 당신의 대상 테스트 클래스들의 절차를 구현하고 단위테스트를 수행하고, 해당 빌드 한정의 커버리지 보고서를 생성할 빌드 스크

립트가 필요하다. 그러면 해당 플러그인은 동작중인 히스토리컬 커버리지 경향 보고서를 업데이트 한다. 그렇게 해서 당신은 당신의 커버리지 퍼센트 율이 시간에 따라 어떻게 변해가는지 비주얼화 할 수 있다. 이들 경향 보고서의 예제는 그림 34에서 볼 수 있다.

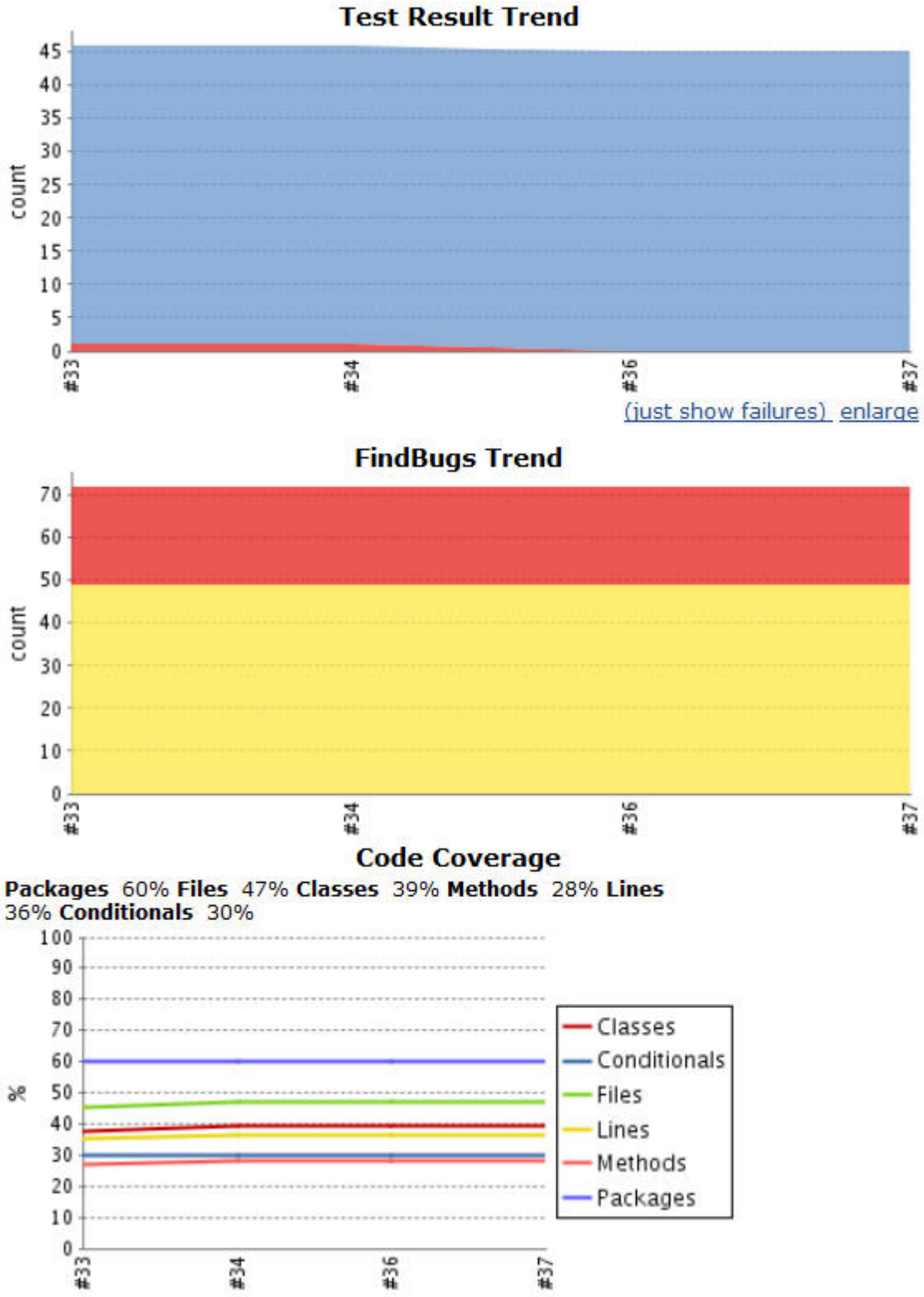
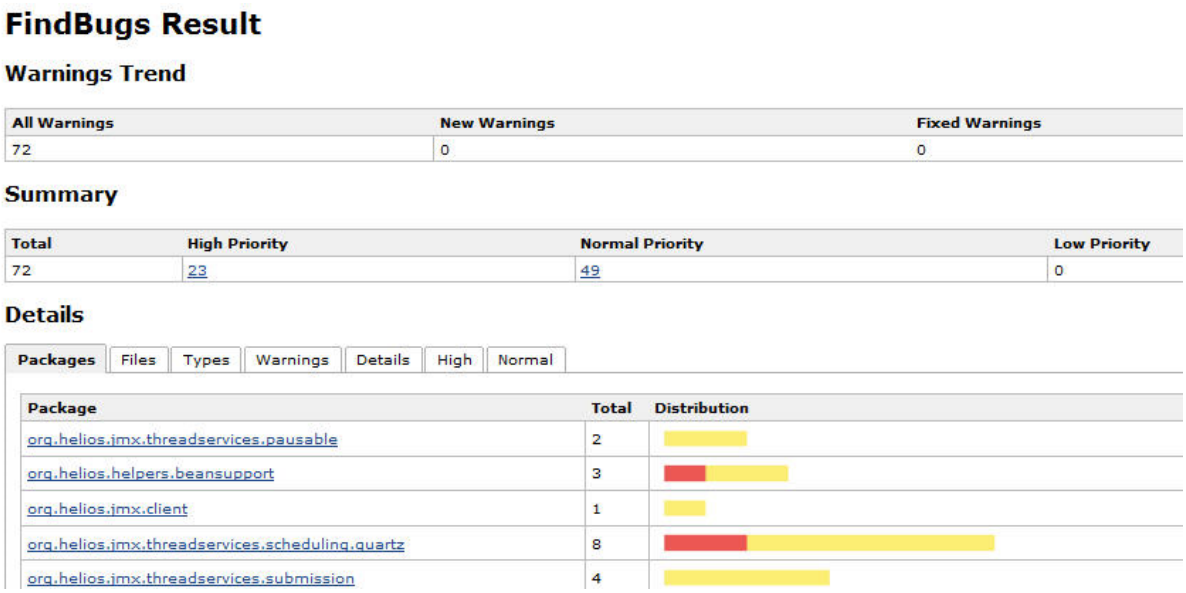


그림 34. 단위 테스트들, 정적 코드 분석, 코드 커버리지에 대해 이력으로 남은 경향들

추가로, JUnit 테스트 결과들과 FindBugs 들은 빌드 업무 홈페이지나 인스턴스 홈 페이지에 표시되는 허드슨 네이티브 보고서를 만들어 내는 플러그인의 예시들이다. (JUnit 플러그인은 사실 빌트인 되어있어서 설치할 필요가 없다) 그림35는 빌드 인스턴스 페이지에 볼 수 있는 FindBugs 플러그인에 의해 생성된 빌트인 보고서의 예시이다. .



FindBugs Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
72	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
72	23	49	0

Details

Packages | Files | Types | Warnings | Details | High | Normal

Package	Total	Distribution
org.helios.jmx.threadservices.pausable	2	
org.helios.helpers.beansupport	3	
org.helios.jmx.client	1	
org.helios.jmx.threadservices.scheduling.quartz	8	
org.helios.jmx.threadservices.submission	4	

그림 35. 빌드에 대한 FindBugs 요약 보고서

당신은 당신 생각 가능한 어떤 종류의 허드슨 확장이든 실제로 제공하기 위해서 당신 스스로 플러그인을 구현할 수 있다. 만일 당신이 플러그인 확장을 구현하는데 관심이 있다면, 허드슨 위키에서 레퍼런스와 문서, 그리고 따라 하기식 도움말을 찾을 수 있다. (아래 부분에 있는 리소스 섹션을 보길)

결론

이로서 허드슨 지속적인 통합 서버에 대한 소개는 끝났다. 나는 허드슨이 끝내주는 소프트웨어라는걸 당신이 알게 될 거라 생각한다. 쉬운 설치와 설정 덕분에, 당신은 매우 빠른 시간 안에 구축해서 띄우고 동작시켜 볼 수 있다. 허드슨 개발 프로젝트가 자리잡고 있는 java.net 사이트에서의 활동 규모를 근거로 놓고 봤을 때, 허드슨은 확실히 강한 추진력을 갖고 있다. 내가 띄엄띄엄 보는 메일링 리스트에서도 사람들이 일관되고 빠르게 그들의 질문에 대한 응답을 받는 것을 알 수 있다. 언젠가는 내가 해당 내용을 필요할 날이 올 수도 있겠지만, 어쨌든 현재까지는 지원을 찾기 위해 날 멈추게 하는 이슈에 맞닥뜨려본 적은 없다. 나는 당신이 허드슨에 대해 함께 탐험해 본 것을 즐겼기 바란다. -- 더 많은 글과 다운로드 할 꺼리, 관련 링크는 아래의 리소스 섹션을 체크해 보아라.

저자에 대해

Nicholas Whitehead 은 시니어 테크놀로지 아키텍트이다. (a senior technology architect at the Small Business Services division of ADP in Florham Park, N.J.)

리소스들(Resources)

Downloads

- Get the latest Hudson WAR file here or here.
- Download apache-tomcat-6.0.18.exe to install Tomcat on your Windows machine. Download JBoss 4.2.3.GA for a Linux environment (look for the file named jboss-4.2.3.GA.zip).
- Ant is the build tool used for examples in this article.
- jboss-init.sh enables the automatic start and stop of the JBoss server.
- If your Hudson server cannot connect to outside resources, you can download the plugins you need from the Hudson Website.

Learn more

- "Introducing continuous integration" (Paul Duvall, Steve Matyas, Andrew Glover; JavaWorld, 2007): Get an overview of CI in this chapter excerpt from *Continuous Integration: Improving Software Quality and Reducing Risk* (Addison-Wesley Professional, June 2007).
- "Which open source CI server is right for you?" (John Smart, JavaWorld, 2006): Introduces and compares four top CI servers -- CruiseControl, Continuum, Luntbuild, and Hudson.
- *Java Power Tools* author John Smart talks shop about tools he's most likely to use for his own projects, including Hudson, in this podcast episode of JavaWorld's Java Technology Insider.
- Check out the Hudson project on Java.net.
- If you're ready to start using Hudson seriously, you should study up on best practices (from the Hudson Wiki). You might also want to be aware of CI anti-patterns (Paul Duvall, IBM developerWorks, December 2007).
- Hudson was the most used CI server in an informal developer survey in 2008. Learn more about the results here.
- The procedure for installing JBoss outlined in this article has been condensed from this useful thread in the Ubuntu forums.
- Check out the complete list of SCM plugins for Hudson.
- Find out more about the Hudson FindBugs plugin.
- If you are interested in implementing your own plugin, check out these references, documentation, and tutorials.
- Learn more about Cobertura.
- Read the Quartz CronTrigger javadoc for information on the cron syntax Hudson uses