

Software Verification

#2. Build Environmnet



T3. SangYoon Kim
Green Kim

INDEX



• **CI**

• **Ant**

• **Jenkins**



CI

CI

Ant

Jenkins

CI



CI

CI

Ant

Jenkins

- 1. What is CI?**
- 2. Why we need CI?**
- 3. What can we do with CI?**
- 4. Which Consist CI?**



CI

CI

Ant

Jenkins

1. What is CI?

Continus Integration



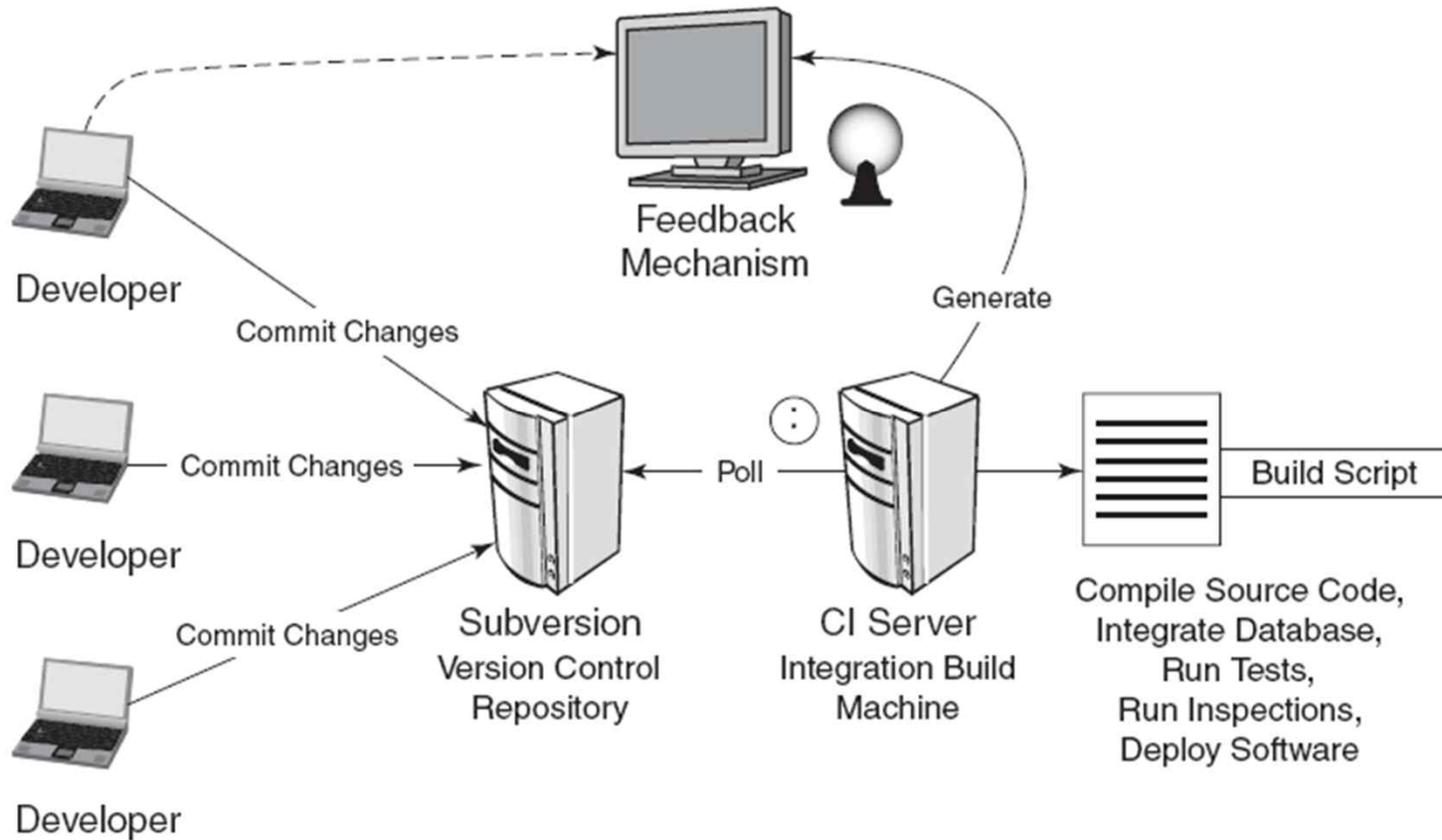
CI

CI

Ant

Jenkins

1. What is CI?





CI

CI

Ant

Jenkins

2. Why we need CI?

- 자동화의 필요성
- 프로젝트의 생산성과 안전성



CI

CI

Ant

Jenkins

3. What can we do with CI?

- **Keep the build fast**
- **Test in a clone of the production environment**
- **Make it easy to get the latest deliverables**
- **Everyone can see the results of the latest build**
- **Automate deployment**



CI

CI

Ant

Jenkins

- **Maintain a code repository**
- **Automate the build**
- **Make the build Self-Testing**
- **Everyone commits to the baseline every day**
- **Everyone commit to baseline should be built**



CI

CI

Ant

Jenkins

Vanilla vs CI tool

1. 프로그램 설치 한 번으로 모든 것을 해결 가능하다.
2. 개발환경과 구동환경의 격차를 최소화 할 수 있다.
3. 가볍다.



CI

CI

Ant

Jenkins

4. Which consist CI?

Build Automation

+

Build Self-Testing



Ant

CI

Ant

Jenkins



Ant



Ant

CI

Ant

Jenkins

Another Neat Tool



Ant

CI

Ant

Jenkins

- 1. Why we need Build Automation?**
- 2. Why we choose Ant?**
- 3. What is Ant?**
- 4. How to Use Ant?**
- 5. Ant is the Best Tool?**



Ant

CI

Ant

Jenkins

1. Why we need Build Automation?

- 라이브러리의 부족
- 파일이 없을 경우
- 파일에 오류가 있을 경우
- 서로 다른 개발환경에서 차이가 있을 경우

서버에서 자동으로 최신파일을 가져와 빌드 하고 통보 - 오류를 신속히 확인가능, 자동화 처리



Ant

CI

Ant

Jenkins

2. Why we choose Ant?

1. 우리나라에서 정보를 구하기 가장 쉬운 툴
- 업계 표준으로 정착
2. JUnit과 밀접하게 통합
3. JAVA를 사용하여 손쉽게 확장 가능
4. 문법이 간단하여 쉽게 배울 수 있다.
5. 쉽게 사용할 수 있다.



Ant

CI

Ant

Jenkins

Is different from Makefile?

- Makefile이 가지고 있는 문제점을 해결한 Makefile?
- 셸 기반이 아닌 XML기반
- 자바클래스들의 사용성을 확장



Ant

CI

Ant

Jenkins

3. What is Ant?

JAVA기반으로 크로스 플랫폼과 사용의 용이성, 확장성, 범위성을 고려하여 설계된 빌드 도구



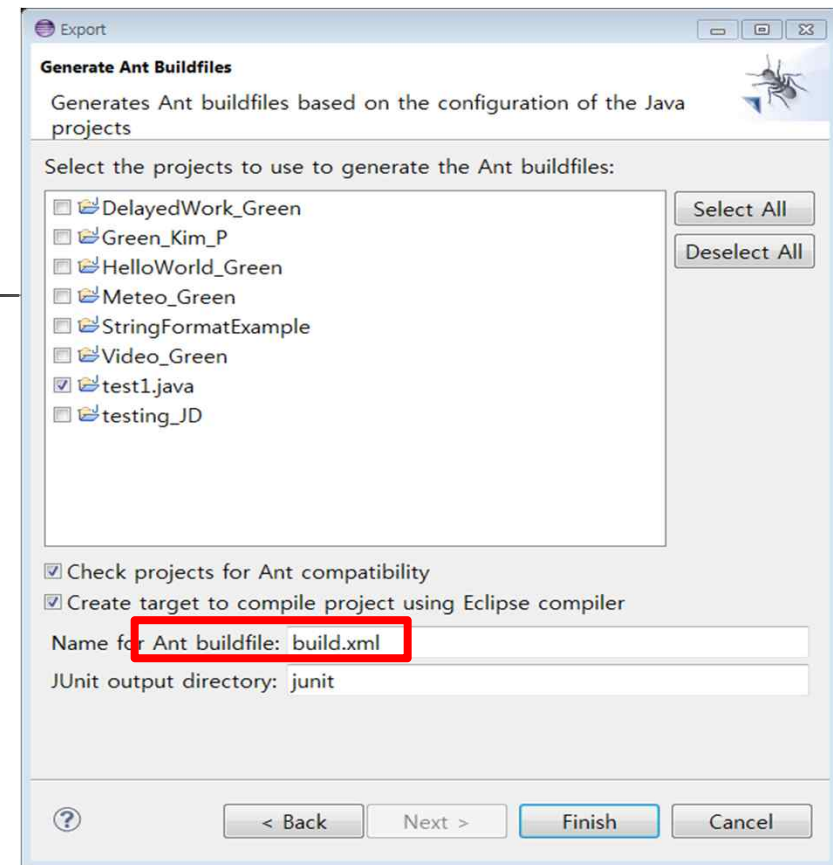
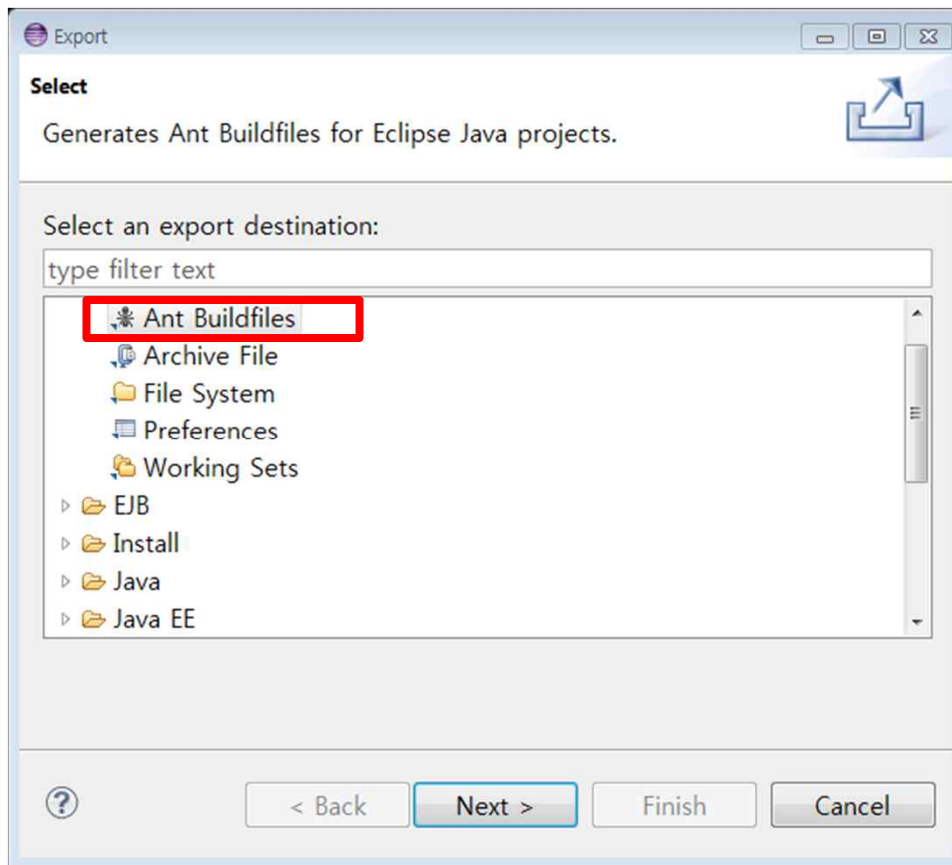
Ant

CI

Ant

Jenkins

4. How to use Ant?





Ant

CI

Ant

Jenkins

The screenshot shows the Eclipse IDE interface with the following components:

- Left Panel (Type Hierarchy):** Lists project elements including `ork_Green`, `l_P`, `d_Green`, `ien`, `natExample`, `stem Library [JavaSE-1.7]`, `ml`, and `en`.
- Editor (build.xml):** Displays the following XML content:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- WARNING: Eclipse auto-generated file.
Any modifications will be overwritten.
To include a user specific buildfile here, simply create one in the
directory with the processing instruction <?eclipse.ant.import?>
as the first entry and export the buildfile again. --><project based:
<property environment="env"/>
<property name="ECLIPSE_HOME" value="D:/######/Tool_coding/eclipse-SDK-4.2.1-
<property name="debuglevel" value="source,lines,vars"/>
<property name="target" value="1.7"/>
<property name="source" value="1.7"/>
<path id="test1.java.classpath">
  <pathelement location="bin"/>
</path>
<target name="init">
  <mkdir dir="bin"/>
  <copy includeemptydirs="false" todir="bin">
    <fileset dir="src">
      <exclude name="**/*.java"/>
    </fileset>
  </copy>
</target>
<target name="clean">
  <delete dir="bin"/>
</target>
<target depends="clean" name="cleanall"/>
<target depends="build-subprojects,build-project" name="build"/>
<target name="build-subprojects"/>
<target depends="init" name="build-project">
  <echo message="${ant.project.name}: ${ant.file}"/>
  <javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantrun
    <src path="src"/>
    <classpath refid="test1.java.classpath"/>
  </javac>
</target>
<target description="Build all projects which reference this project. Useful to
<target description="copy Eclipse compiler jars to ant lib directory" name="in
  <copy todir="${ant.library.dir}"/>
```
- Outline (test1.java):** Shows a tree view of the project structure:
 - test1.java
 - environment=env
 - ECLIPSE_HOME
 - debuglevel
 - target
 - source
 - test1.java.classpath
 - init
 - clean
 - cleanall
 - build [default]
 - build-subprojects
 - build-project
 - build-refprojects
 - init-eclipse-compiler
 - build-eclipse-compiler
- Right Panel (Index):** Contains search and navigation options like `Contents`, `Search`, `Related Topic`, `Bookmarks`, and `Index`.
- Bottom Right Panel (About Ant Editor):** Provides information about the Ant Editor, stating that each workbench window contains one or more perspectives, which are made up of various views and editors. It also lists "See also:" `Workbench`, `Perspectives`, and `Customizing the Workbench`, and "More results:" `Search for Ant Editor`.



Ant

CI

Ant

Jenkins

The screenshot shows the Eclipse IDE interface with the 'build.xml' file open in the editor. A context menu is displayed over the file, with 'Run As' and '2 Ant Build...' highlighted by red boxes. The 'build.xml' content is as follows:

```
<property environment="env"/>
<property name="ECLIPSE_HOME" value="D:/#####/Tool_coding/eclipse-SDK-4.2.1-
<property name="debuglevel" value="source,lines,vars"/>
<property name="target" value="1.7"/>
<property name="source" value="1.7"/>
<path id="test1.java.classpath">
<pathelement location="bin"/>
</path>
<target name="init">
<mkdir dir="bin"/>
<copy includeemptydirs="false" todir="bin">
<fileset dir="src">
<exclude name="**/*.java"/>
</fileset>
</copy>
</target>
<target name="clean">
<delete dir="bin"/>
</target>
<target depends="clean" name="cleanall"/>
<target depends="build-subprojects,build-project" name="build"/>
<target name="build-subprojects"/>
<target depends="init" name="build-project">
<echo message="${ant.project.name}: ${ant.file}"/>
<javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantrun
<src path="src"/>
<classpath refid="test1.java.classpath"/>
</javac>
</target>
<target description="Build all projects which reference this project. Useful to
<target description="copy Eclipse compiler jars to ant lib directory" name="in
<copy todir="${ant.lib.dir}"
```

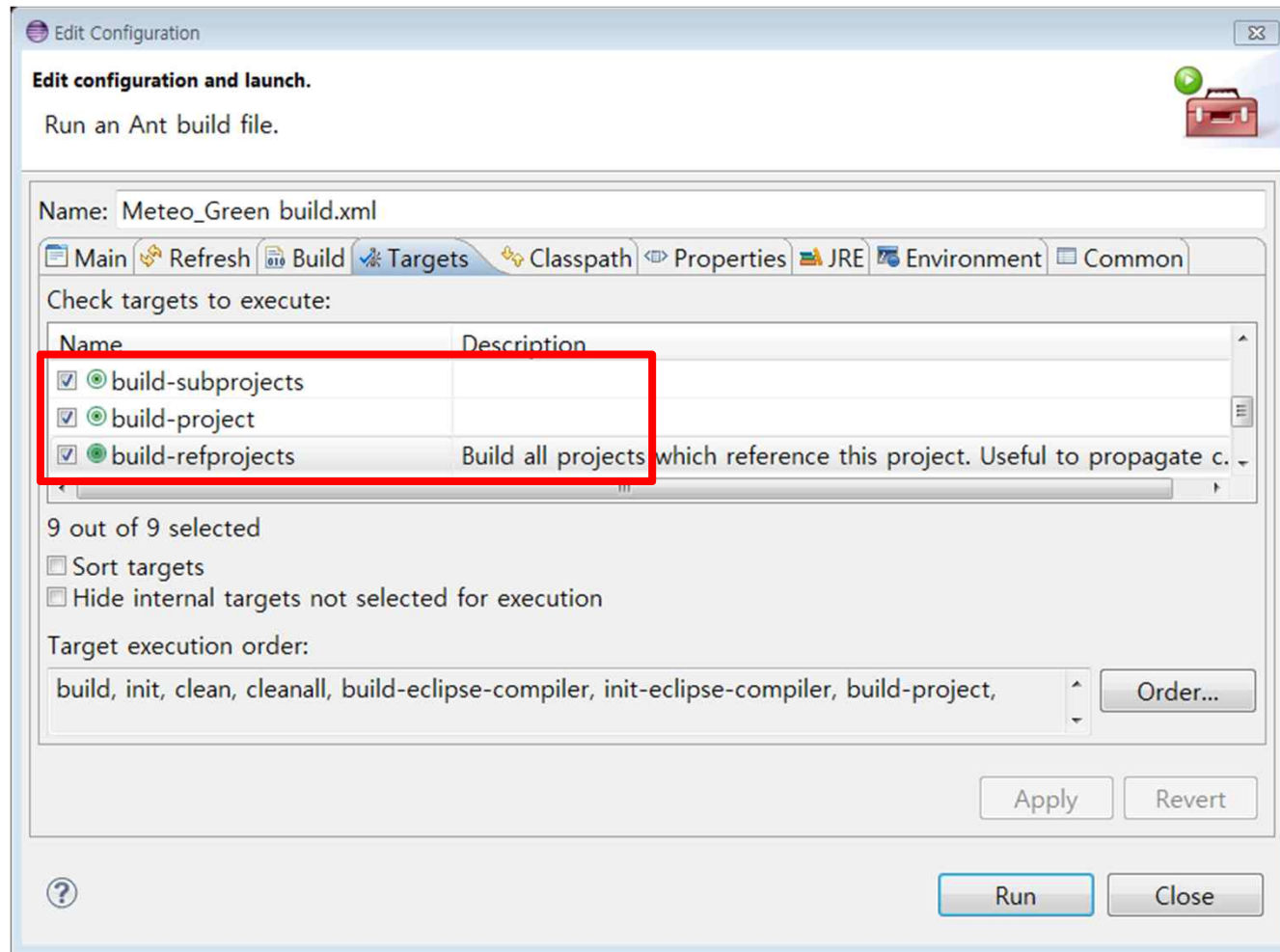


Ant

CI

Ant

Jenkins





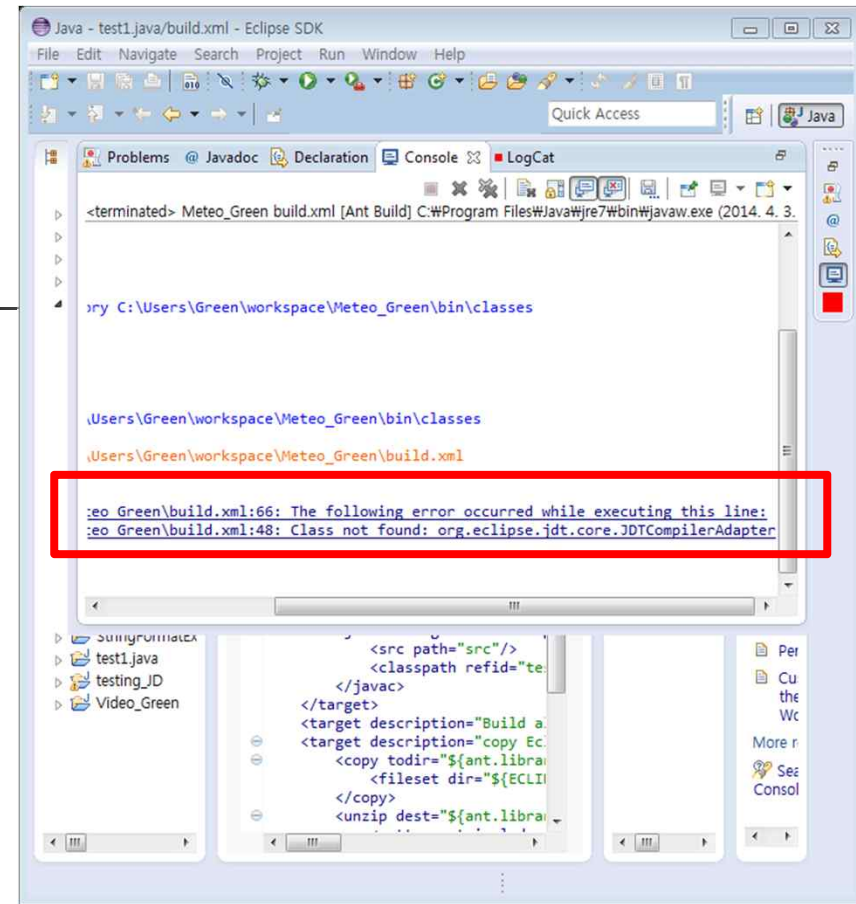
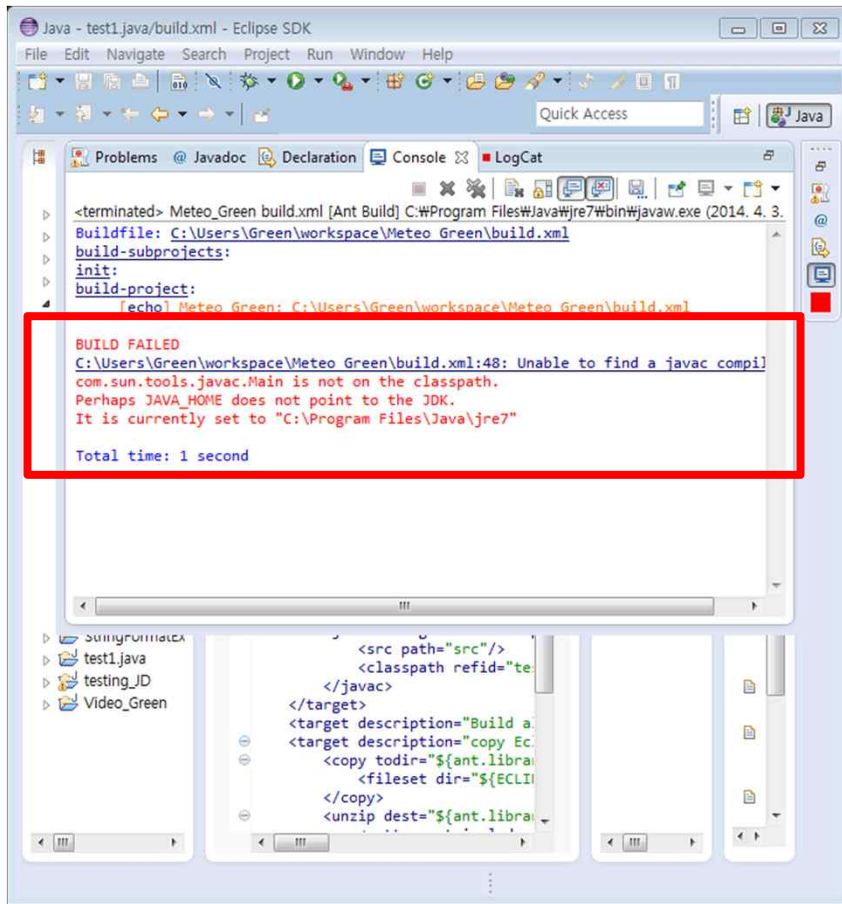
Ant

CI

Ant

Jenkins

Problems





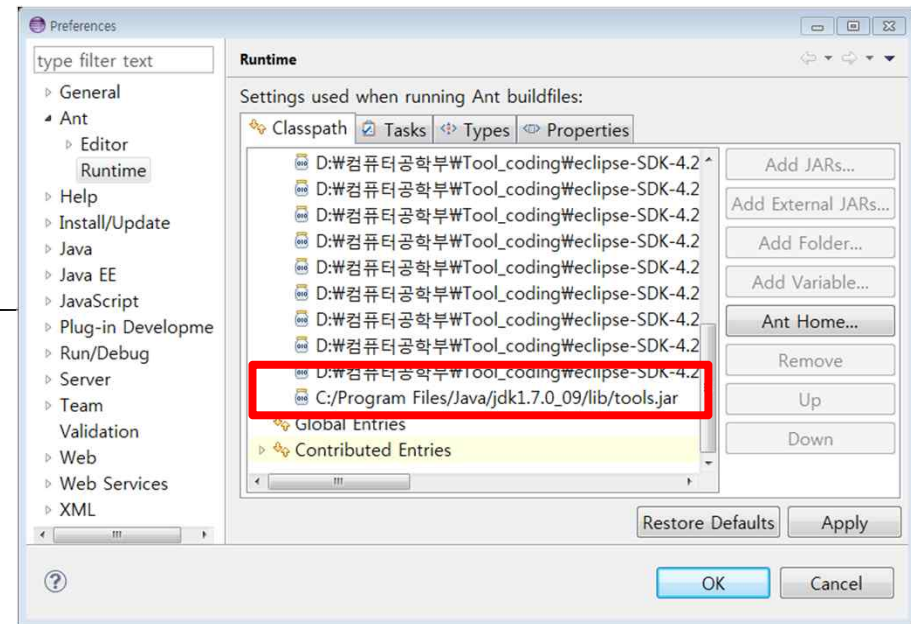
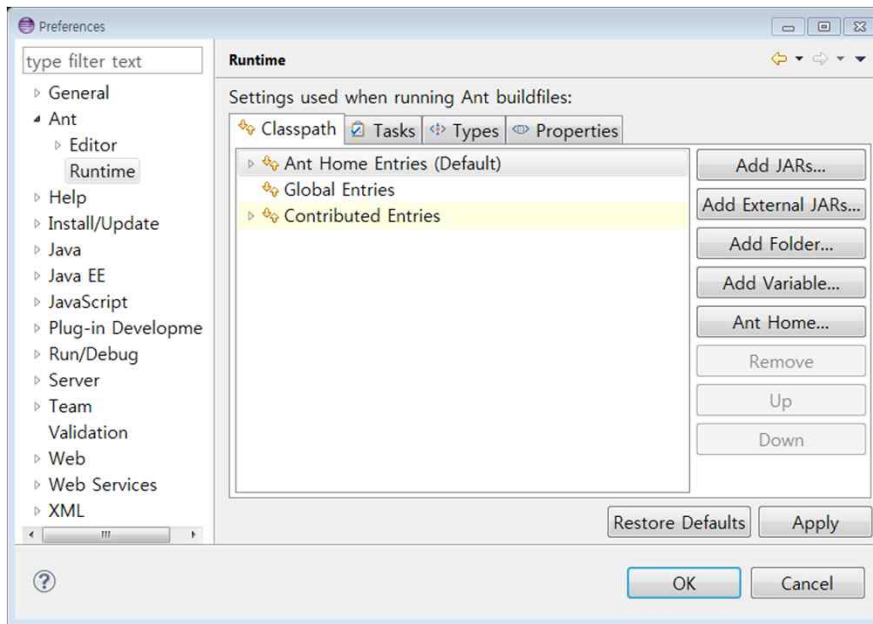
Ant

CI

Ant

Jenkins

Solutions





Ant

CI

Ant

Jenkins

5. Ant is The Best Tool?

- 아주 작은 규모에서부터 대규모에 이르기까지 빌드, 테스트, 배치를 수행해줄 수 있는 기반의 도구
- But, 그 자체로는 JAVA 프로젝트를 성공시킬 수 없다.



Jenkins

CI

Ant

Jenkins

Jenkins



Jenkins

CI

Ant

Jenkins



Jenkins



Jenkins

CI

Ant

Jenkins

- 1. Why we choose Jenkins?**
- 2. What is Jenkins?**
- 3. How to install Jenkins?**
- 4. How to use Jenkins?**



Jenkins

CI

Ant

Jenkins

1. Why we choose Jenkins?

1. Easy installation
2. Easy configuration
3. Change set support
4. Permanent links
5. RSS/E-mail/IM integration



Jenkins

CI

Ant

Jenkins

1. Why we choose Jenkins?

6. After-the-fact tagging

7. Junit/TestNG test reporting

8. Distributed builds

9. File fingerprinting

10. Plugin Support



Jenkins

CI

Ant

Jenkins

Jenkins vs Cruise Control

1. Easy GUI vs XML
2. Fast
3. Easy



Jenkins

CI

Ant

Jenkins

2. What is Jenkins?

JAVA로 만들어졌으며 400여개 이상의 플러그인을 제공하는 CI를 지원하는 가장 유명한 도구

Hudson이라는 이름에서 Oracle에 흡수되면서 별도 프로젝트인 Jenkins로 분리



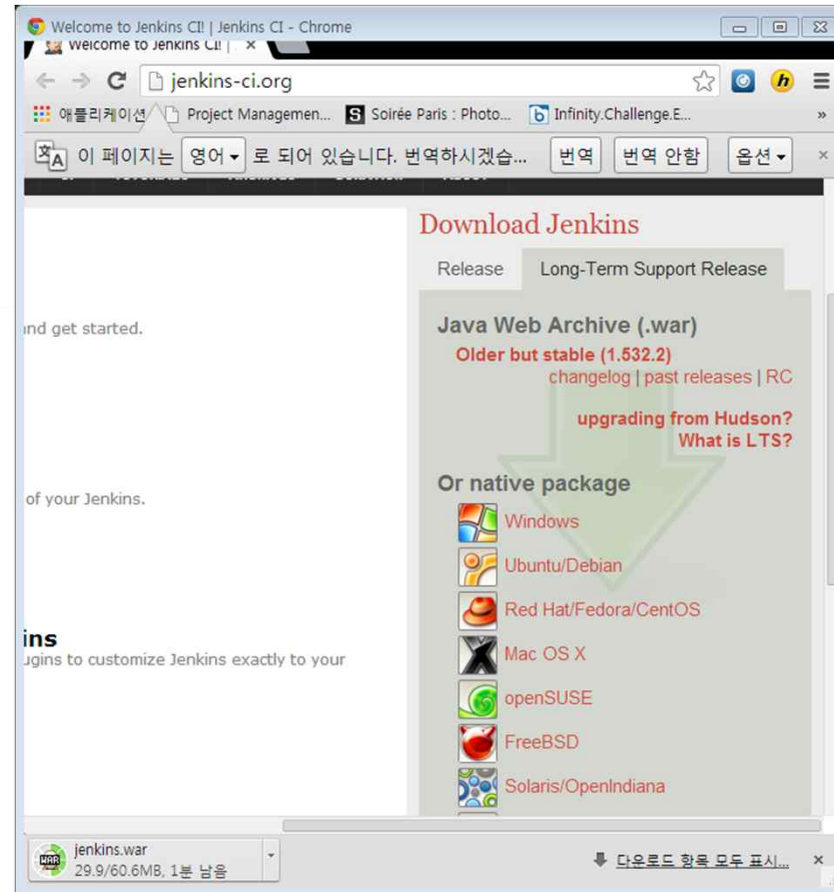
Jenkins

CI

Ant

Jenkins

3. How to install Jenkins?





Jenkins

CI

Ant

Jenkins

4. How to use Jenkins?

The screenshot shows the Jenkins 'New Job' configuration page. The browser address bar shows 'http://localhost:8080/jenkins/newJob'. The page title is 'New Job [Jenkins]'. The main content area is titled 'Job 이름 [MyProject]'. There are four radio button options for job types: 'Build a free-style software project', 'Build a maven2/3 project', 'Build multi-configuration project', and 'Monitor an external job'. The 'Build a free-style software project' option is selected. Below the options is a table for '빌드 대기 목록' (Build Queue) with columns for '#', '상태', and '대기 중'. The table shows two entries, both in '대기 중' (Waiting) status. At the bottom, there are links for '이 페이지 한글화 도와주기' and '작성된 페이지: 2014. 4. 3 오후 9:25:54 REST API Jenkins ver. 1.532.2'.

The screenshot shows the Jenkins 'MyProject Config' configuration page. The browser address bar shows 'http://localhost:8080/jenkins/configure?job=MyProject'. The page title is 'MyProject Config [Jenkins]'. The main content area is titled 'configuration'. There are several sections: 'Project' with checkboxes for 'This build is parameterized', '빌드 안함 (프로젝트가 다시 빌드를 할 때까지 새로운 빌드가 실행되지 않습니다.)', and 'Execute concurrent builds if necessary'; '고급 프로젝트 옵션' (Advanced Project Options) with a '그림...' button; '소스 코드 관리' (Source Code Management) with radio buttons for 'CVS', 'CVS Projectset', 'None', and 'Subversion'; 'Repository URL' with a text input field and a red error message 'Repository URL is required.'; 'Local module directory (optional)' with a text input field; 'Repository depth option' with a dropdown menu set to 'infinity'; 'Ignore externals option' with a checkbox; and 'Check-out Strategy' with a dropdown menu set to 'Use 'svn update' as much as possible'. At the bottom, there are '저장' (Save) and 'Apply' buttons.



Conclusion



References

http://en.wikipedia.org/wiki/Continuous_integration

http://en.wikipedia.org/wiki/Apache_Ant

[http://en.wikipedia.org/wiki/Jenkins_\(software\)](http://en.wikipedia.org/wiki/Jenkins_(software))

<http://jhoony.tistory.com/368>

<http://whatwant.tistory.com/583>

<http://rimmomo.tistory.com/13>

<http://blog.naver.com/iamfreeman?Redirect=Log&logNo=50180429646>

<http://blog.naver.com/2000yujin?Redirect=Log&logNo=130161252153>

<http://blog.naver.com/rosekingdom?Redirect=Log&logNo=60001529696>

<http://coronasdk.tistory.com/750>



Q&A





Thank You

