

## Ant

1	도구 개요
2	도구 설치 방법
3	도구 기능 소개
4	도구 활용 예제
5	FAQ
6	도구 평가
7	용어집

# 목차

- 1. 도구 개요
- 2. 도구 설치 방법
  - 2.1 Ant 다운받기
  - 2.2 Ant 설치하기
- 3. 도구 기능 소개
  - 3.1 Ant 실행 주요 옵션 목록
  - 3.2 Ant의 사용
  - 3.3 Project
  - 3.4 target
  - 3.5 태스크
  - 3.6 Property
  - 3.7 예제 소개
  - 3.8 Ant 빌드 파일 작성하기
- 4. 도구 활용
  - 4.1 예제 소개
  - 4.2 프로젝트 생성
  - 4.3 Ant 빌드 스크립트의 제작
  - 4.4 Ant의 동작 확인
- 5. FAQ
- 6. 도구 평가
- 7. 용어집

『Java 세상을 뒤흔친 Eclipse ver3.1 을 참고하였음』

# 1. 도구개요

# 1. 도구개요

Ant

소개	Java 기반의 빌드 프로세스 자동화 도구로서 크로스 플랫폼과 사용의 용이성, 확장성, 범위를 고려하여 설계 되었습니다. Ant는 소규모의 개인용 프로젝트에서부터, 여러 팀으로 구성된 대규모 소프트웨어 프로젝트에서 사용 될 수 있습니다.		
주요 기능	CVS 체크아웃, 컴파일, 테스트, 배포 관리		
카테고리(용도)	Implementation	세부 카테고리	빌드관리
커버리지	Implementation	도구 난이도	중급
License 형태 / 비용	Apache Software License / 무료	사전 설치 도구	JDK
사용 환경 / 개발 상태	운영체제	Windows XP / Linux	
	Eclipse 환경	-	
특징	<ul style="list-style-type: none"><li>• Eclipse는 Ant 플러그인을 기본으로 내장합니다.</li><li>• Ant를 좀더 편리하게 사용할 수 있는 다양한 사용자 인터페이스를 제공합니다.</li><li>• 플랫폼 독립적인 Java 클래스를 사용함으로써 OS가 바뀌어도 빌드 파일을 수정할 필요가 없습니다.</li><li>• Ant 테스크들은 필요한 일 이외의 다른 일을 수행하지 않기 위하여 종속성 검사를 하도록 설계 되었습니다.</li></ul>		
적용 회사 / 프로젝트	-		
관련 도구	GNU Make		
제작사	Apache		
공식 홈페이지	http://Ant.apache.org		
개발자	-		

## 2. 도구 설치 방법

## 2. 도구 설치 방법

### 세부 목차

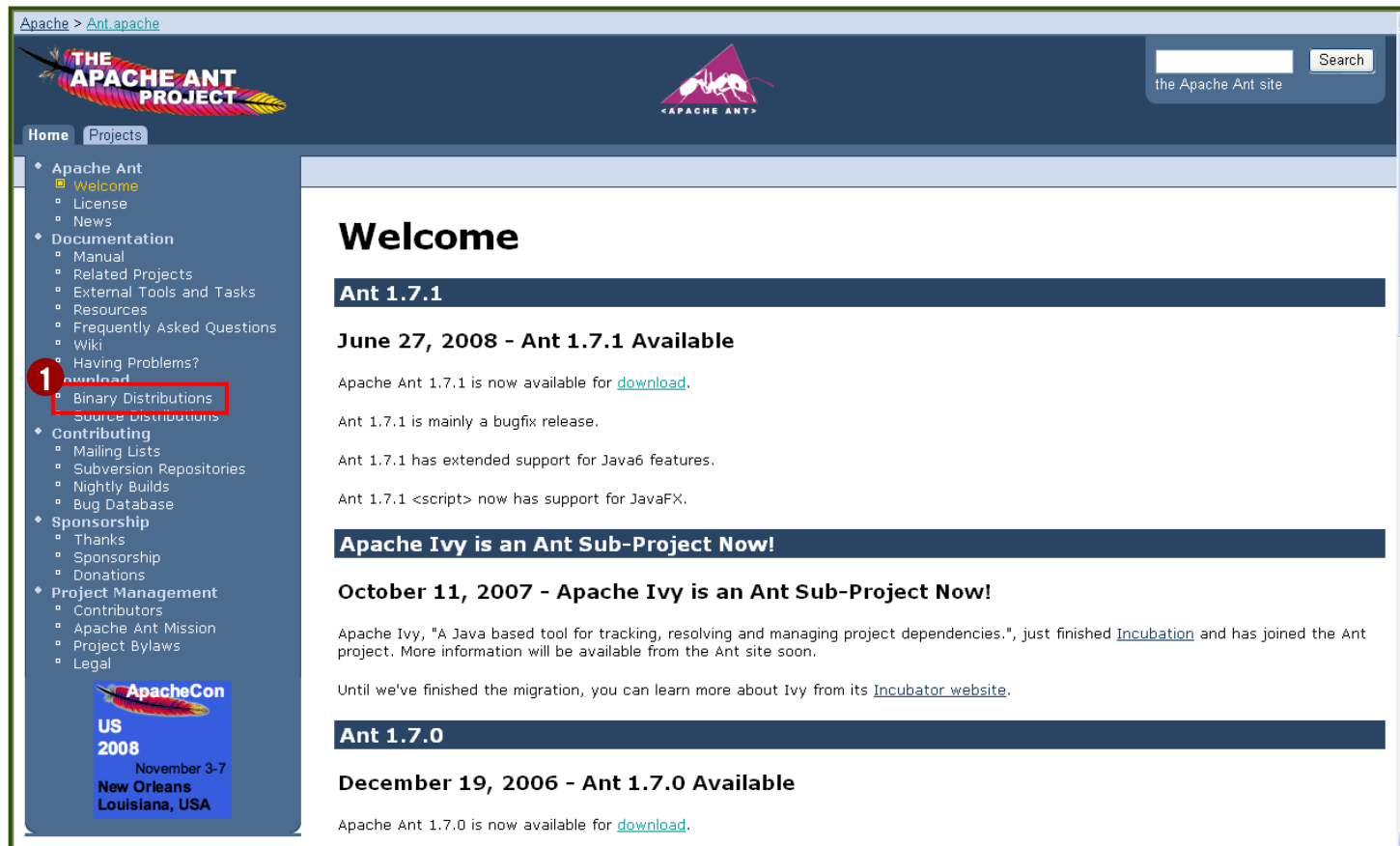
Ant

- 2.1 Ant 다운받기
- 2.2 Ant 설치하기

## 2. 도구 설치 방법

### 2.1 Ant 다운받기 (1/4)

- Eclipse를 설치하면 Ant를 자동으로 설치하지만, 여기서는 수동으로 설치하는 방법을 설명합니다.
- 다음의 사이트에서 Ant를 다운 받습니다.
  - <http://Ant.apache.org>
  - Download 카테고리의 Binary Distributions를 클릭합니다.



## 2. 도구 설치 방법

### 2.1 Ant 다운받기 (2/4)

Ant

- 자신에게 적합한 Ant 설치 파일을 선택합니다.
  - Mirror 항목에서 다운로드 받을 사이트를 설정합니다.
  - Current Release of Ant 항목 아래 나와있는 세 개의 파일 중 자신에게 맞는 파일을 클릭합니다.
    - 본 매뉴얼에서는 Other mirrors : <http://mirror.apache-kr.org> , .zip archive: [apache-Ant-1.7.1-bin.zip](#)을 다운받도록 하겠습니다.

#### Mirror

You are currently using <http://mirror.apache-kr.org>. If you encounter a problem with this mirror, please select another mirror. If the mirrors you select are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

2

Other mirrors:

#### Current Release of Ant

Currently, Apache Ant 1.7.1 is the best available version, see the [release notes](#).

#### Note

Ant 1.7.1 was released on 27-Jun-2008 and may not be available on all mirrors for a few days.

#### Tar files may require gnu tar to extract

Tar files in the distribution contain long file names, and may require gnu tar to do the extraction.

3

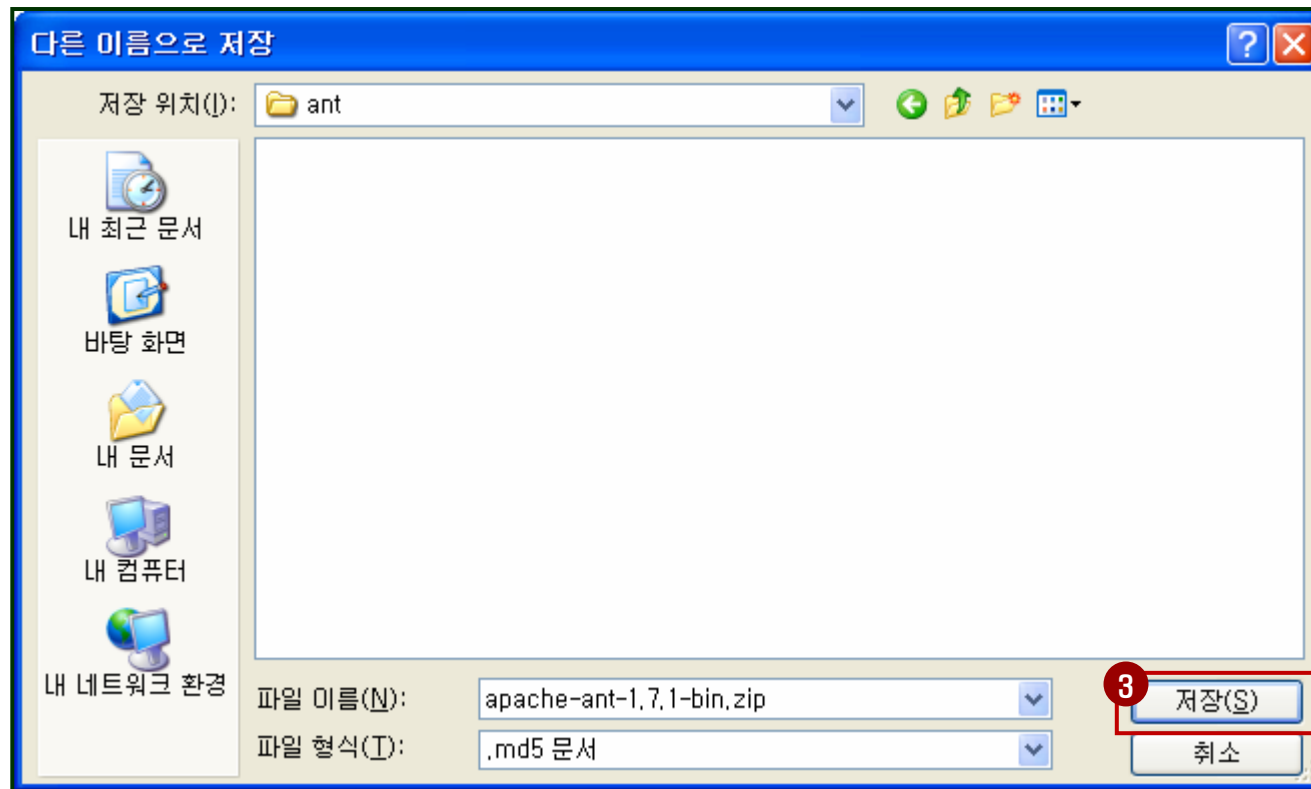
- .zip archive: [apache-ant-1.7.1-bin.zip](#) [PGP] [SHA1] [MD5]
- .tar.gz archive: [apache-ant-1.7.1-bin.tar.gz](#) [PGP] [SHA1] [MD5]
- .tar.bz2 archive: [apache-ant-1.7.1-bin.tar.bz2](#) [PGP] [SHA1] [MD5]



## 2. 도구 설치 방법

### 2.1 Ant 다운받기 (3/4)

- Ant를 다운로드 받습니다.
  - Ant 설치 파일을 저장할 곳을 선택 후 저장(S) 버튼을 누릅니다.
    - 여기서는 C:\want 라는 디렉터리에 다운로드 받도록 하겠습니다.



## 2. 도구 설치 방법

### 2.1 Ant 다운받기 (4/4)

- 다운받은 apache-Ant-1.7.1-bin.zip파일의 압축을 풉니다.
  - 압축을 푼 파일 구조는 다음과 같습니다.

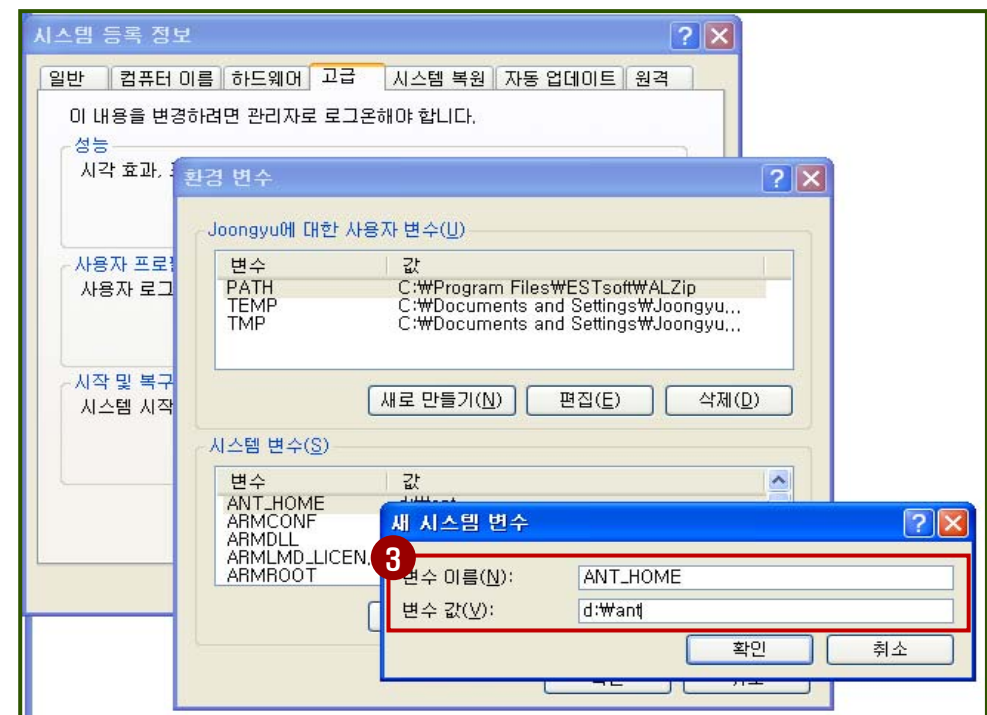
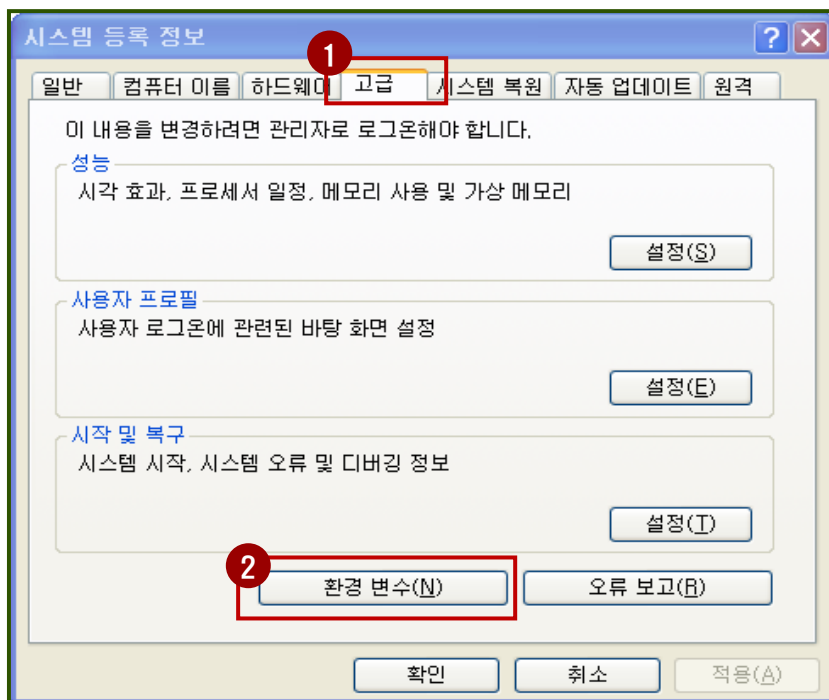
이름 ▲	크기	종류	수정한 날짜
bin		파일 폴더	2008-11-10 오후 ...
docs		파일 폴더	2008-11-10 오후 ...
etc		파일 폴더	2008-11-10 오후 ...
lib		파일 폴더	2008-11-10 오후 ...
fetch	7KB	XML 문서	2008-06-27 오전 ...
get-m2	5KB	XML 문서	2008-06-27 오전 ...
INSTALL	1KB	파일	2008-06-27 오전 ...
KEYS	51KB	파일	2008-06-27 오전 ...
LICENSE	15KB	파일	2008-06-27 오전 ...
NOTICE	2KB	파일	2008-06-27 오전 ...
README	5KB	파일	2008-06-27 오전 ...
WHATSNEW	144KB	파일	2008-06-27 오전 ...

- **bin** : Ant 1.7의 실행 파일입니다.
- **docs** : Ant 1.7의 문서파일, 사용자 매뉴얼과 FAQ 문서 등을 포함하고 있습니다.
- **etc** : XSL스타일 시트 파일이 포함되어 있습니다.
- **lib** : Ant 1.7의 핵심 패키지를 비롯하여 XML 파서 등 Ant를 실행할 라이브러리들을 포함하고 있습니다.

## 2. 도구 설치 방법

### 2.2 Ant 설치하기 (1/3)

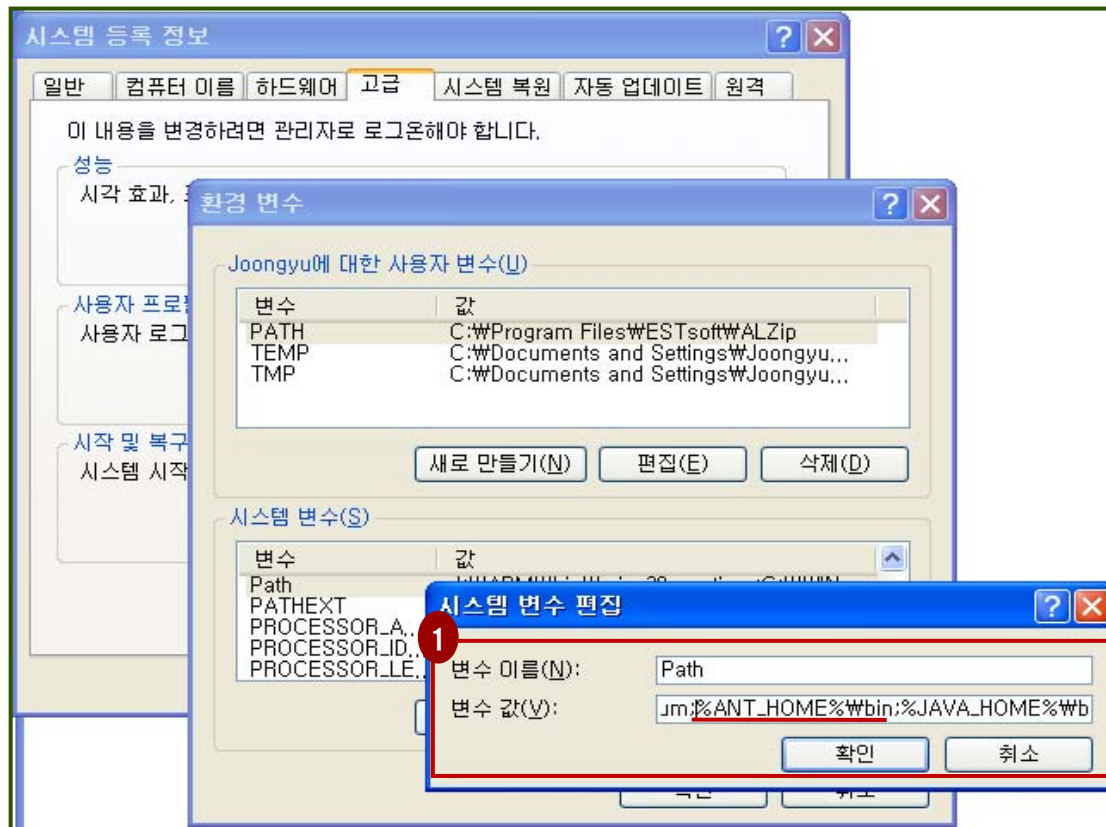
- Ant수행을 위해 환경 변수를 설정합니다.
  - 내 컴퓨터 -> 오른쪽 버튼 클릭 -> [고급]탭 선택 -> 환경 변수(N) 버튼을 클릭합니다.
  - 시스템 변수에 Ant\_HOME을 추가합니다.
    - 변수 이름 = Ant\_HOME
    - 변수 값 = Ant를 다운로드 받은 디렉터리 (여기서는 C:\WAnt로 하였습니다.)



## 2. 도구 설치 방법

### 2.2 Ant 설치하기 (2/3)

- 시스템 변수의 PATH에 Ant를 추가합니다.
  - 내 컴퓨터 -> 오른쪽 버튼 클릭 -> [고급]탭 선택 -> 환경 변수(N)버튼을 클릭합니다.
  - Path 변수를 편집합니다.
    - 시스템 변수(S) 아래의 PATH변수 선택 후 편집(E)을 클릭합니다.
    - 변수 값의 마지막에 ;%ANT\_HOME%\bin 을 추가해 넣고 확인 버튼을 클릭합니다.



## 2. 도구 설치 방법

### 2.2 Ant 설치하기 (3/3)

Ant

- Ant가 동작하는지 시험합니다.
  - 시작-> 실행 -> cmd 입력을 통해 콘솔 창을 띄웁니다.
  - 콘솔 창에 Ant -h를 입력 후 엔터 키를 입력해 Ant가 동작하는지 확인합니다.

```

C:\WINDOWS\system32\cmd.exe
C:\WINDOWS\system32\cmd.exe ant -h
ant [options] [target] [target2] [target3] ...]]
Options:
  -help, -h                print this message
  -projecthelp, -p         print project help information
  -version                 print the version information and exit
  -diagnostics             print information that might be helpful to
                           diagnose or report problems.
  -quiet, -q              be extra quiet
  -verbose, -v            be extra verbose
  -debug, -d              print debugging information
  -emacs, -e              produce logging information without adornments
  -lib <path>             specifies a path to search for jars and classes
  -logfile <file>         use given file for log
                           -l <file>
  -logger <classname>    the class which is to perform logging
  -listener <classname>  add an instance of class as a project listener
  -noinput                do not allow interactive input
  -buildfile <file>       use given buildfile
                           -file <file>
                           -f <file>
  -D<property>=<value>    use value for given property
  -keep-going, -k         execute all targets that do not depend
                           on failed target(s)
  -propertyfile <name>   load all properties from file with -D
                           properties taking precedence
  -inputhandler <class>  the class which will handle input requests
  -find <file>            <s>search for buildfile towards the root of
                           the filesystem and use it
                           -s <file>
  -nice number            A niceness value for the main thread:
                           1 <lowest> to 10 <highest>; 5 is the default
  -nouserlib              Run ant without using the jar files from
                           ${user.home}/.ant/lib
  -noclasspath            Run ant without using CLASSPATH
  -autoproxy              Java1.5+: use the OS proxy settings
  -main <class>           override Ant's normal entry point
C:\WINDOWS\system32\cmd.exe
  
```

### **3. 도구 기능 소개**

## 3. 도구 기능 소개

### 세부 목차

Ant

- 3.1 Ant 실행 주요 옵션 목록
- 3.2 Ant의 사용
- 3.3 Project
- 3.4 target
- 3.5 태스크
- 3.6 Property
- 3.7 예제 소개
- 3.8 Ant 빌드 파일 작성하기

## 3. 도구 기능 소개

### 3.1 Ant 실행 주요 옵션 목록

- Ant를 실행할 때 사용할 수 있는 주요 옵션 목록
  - Ant는 개발자가 명령어 프롬프트에서 옵션을 입력하여 여러 내용을 직접 설정할 수 있습니다.
  - Ant는 기본적으로 다음과 같은 형태로 명령행 인자를 입력 받습니다.
    - Ant [옵션] [타겟 [타겟2 [타겟3] ...]]
  - 여기서 [옵션]에는 빌드 파일, 로그 파일 그리고 프로퍼티 등의 값을 지정할 수 있는 옵션이 오게 되며 [타겟]에는 빌드 파일에 지정되어 있는 <target>태그의 name 속성의 값이 오게 됩니다. [타겟]을 지정하지 않을 경우 빌드 파일에서 <project:>태그의 default 속성의 값을 작업 대상으로 사용합니다.
  - 아래 표는 Ant를 실행할 때 사용할 수 있는 주요 옵션 목록입니다.

옵션	설명
-help	도움말을 출력합니다.
-projecthelp	프로젝트의 도움말 정보를 출력합니다.
-version	버전 정보를 출력합니다.
-quiet	적은 양의 메시지를 출력합니다.
-verbose	추가적인 메시지를 출력합니다.
-debug	디버깅 정보를 출력합니다.
-logfile file	로그 메시지를 file에 기록합니다.
-buildfile file	지정한 file을 빌드 파일로 사용합니다.
-Dproperty=value	이름이 property인 프로퍼티의 값을 value로 지정합니다.



## 3. 도구 기능 소개

### 3.2 Ant의 사용 [1/2]

Ant

- Ant를 올바르게 활용하기 위해서는 Ant가 사용할 빌드 파일을 알맞게 작성해야만 합니다.
- Ant의 빌드 파일은 XML 문서의 구조를 갖고 있으며 Ant가 작업을 수행할 프로젝트에 대한 정보를 담고 있으며, 다음과 같은 형태를 취하고 있습니다.

```
<project name = "프로젝트이름" default = "기본타겟이름" basedir = ".">

  <target name = "타겟이름">
    <property name = "프로퍼티이름1" value = "프로퍼티값1"/>
    <property name = "프로퍼티이름2" value = "프로퍼티값2"/>
  </target>

  <target name = "타겟이름1">
    <태스크명/>
    <태스크명1 dir = "${build}" />
    <property name = "프로퍼티이름3" value = "프로퍼티값3"/>
  </target>

  <target name = "타겟이름2" depends = "타겟이름1">
    <태스크명2 속성1 = "값1" 속성2 = "값2" />
  </target>
</project>
```

## 3. 도구 기능 소개

### 3.2 Ant의 사용 (2/2)

Ant

- 위 코드를 보면 크게 project, property, target, 태스크의 네 가지 요소로 빌드 파일이 구성되어 있는 것을 알 수 있습니다.
  - 이 네 가지 요소가 Ant의 모든 기능을 사용할 수 있도록 해 주며, 네 가지 요소를 알맞게 조합함으로써 개발자는 원하는 작업을 배치 작업으로 처리할 수 있게 됩니다.
  - Project, target, property 그리고 태스크 태그를 어떻게 설정해주는지 살펴보도록 하겠습니다.

```
<project name="project_name" default="all" basedir=".">
  <target name="all" depends="compile,dist,clean">
    ...
  </target>

  <target name="compile">
    ...
  </target>

  <target name="dist" depends="compile">
    ...
  </target>

  <target name="clean">
    ...
  </target>
</project>
```

## 3. 도구 기능 소개

### 3.3 Project

- Project

- <project> 태그는 빌드 파일의 루트 태그로서 프로젝트를 정의해줍니다. 모든 빌드 파일은 한 개의 project 태그를 가지며, project 태그는 프로젝트를 설명하기 위해서 다음과 같은 속성을 사용합니다.

속성	설명
Name	프로젝트의 이름
default	Ant.bat 파일을 실행할 때 [타겟]이 지정되지 않을 때 기본적으로 사용하는 타겟
basedir	경로 계산을 할 때 사용할 기본 디렉토리. basedir 프로퍼티를 지정하는 경우, 그 값을 이 속성에서 지정한 값으로 대체한다. 만약 이 속성도 basedir 프로퍼티도 지정하지 않았을 경우에는 빌드 파일이 디렉토리를 기본 디렉토리로 사용한다.

- Ant는 Ant를 실행할 때 가장 먼저 실행할 태스크를 지정할 수 있는 두 가지 방법을 갖고 습니다.
- 한 가지는 앞에서 Ant를 실행할 때 [타겟]을 지정해주는 것입니다.
- 다른 하나는 <project>태그의 default 속성을 사용하여 첫 번째로 실행할 태스크를 지정해 주는 것입니다.

## 3. 도구 기능 소개

### 3.4 target [1/2]

- target

- <target>태그는 <project> 태그에 포함되며 실제로 프로젝트가 수행하게 될 작업(태스크)를 지정합니다. <target>태그의 기본 구조는 다음과 같습니다.

```
<target name="타겟이름1">
  <태스크명/>
  <태스크명1 dir="${build}"/>
  <property name="프로퍼티이름3" value="프로퍼티값3"/>
</target>
```

- 하나의 <project> 태그는 여러 개의 <target>태그를 포함할 수 있으며, 각각의 <target> 태그는 그 타겟을 통해서 수행하고자 하는 태스크(작업)를 명시합니다. 즉, 실제 작업의 처리는 <target> 태그 내에 명시되어 있는 태스크를 통해서 이루어지는 것입니다.
- 이때 각각의 <target> 태그는 상호간 의존관계가 있을 수 있습니다.
- 예를 들어, 일반적인 어플리케이션의 빌드 순서는 '컴파일 -> API 문서 생성 -> 배포판 생성' 인데, 이 경우 빌드 파일에는 이 각각의 작업에 대한 <target>이 있을 것입니다.
- 즉, 컴파일을 위한 <target> 태그, API문서 생성을 위한 <target> 태그 그리고 배포판 생성을 위한 <target> 태그 등 3개의 <target> 태그가 존재하게 됩니다.

### 3. 도구 기능 소개

#### 3.4 target [2/2]

Ant

- target 간의 의존관계는 target 요소의 depends 속성으로 나타냅니다.
- 그림 [Ant 빌드 파일 구조] 에서 살펴본 dist 타겟은 compile 타겟에 의존합니다.

```
<project name="project_name" default="all" basedir=".">
  <target name="all" depends="compile,dist,clean">
    ...
  </target>

  <target name="compile">
    ...
  </target>

  ① <target name="dist" depends="compile">
    ...
  </target>

  <target name="clean">
    ...
  </target>
</project>
```

[Ant 빌드 파일 구조]

## 3. 도구 기능 소개

### 3.5 태스크 (1/8)

#### • 태스크

- 타겟이 각 수행할 작업간의 의존관계나 수행 조건 등을 표시한다면 태스크는 타겟 내에서 실제로 수행할 작업을 나타냅니다.
- 예를 들어, 태스크를 통해서 소스 코드를 컴파일하고 파일을 복사/삭제하고 API 문서를 생성할 수 있습니다.
- 태스크 <target> 태그에 중첩되어 표시되며 다음과 같은 구조를 가지고 있습니다.

```
<태스크명2 속성1="값1" 속성2="값2"/>
```

- ‘태스크명’은 태스크의 이름을 나타내고 ‘속성n’과 ‘값n’은 각각 태스크를 처리할 때 사용할 속성값을 나타냅니다.
- 태스크는 이미 만들어져 있는 built-in 태스크를 사용할 수도 있고, 추가적으로 제공되는 옵션 태스크를 사용할 수도 있습니다. 또한 직접 작성한 태스크를 사용할 수도 있습니다.
- 예를 들어, built-in 태스크인 javac를 사용하여 소스코드를 컴파일 할 때는 다음과 같이 <target> 태그에 javac 태스크를 중첩 시키면 됩니다.

```
<target name="compile" depends="init">
  <javac srcdir="${src}"
        destdir="${build}"
        classpath="jcorelogging.jar" />
</target>
```

## 3. 도구 기능 소개

### 3.5 태스크 (2/8)

Ant

- 태스크 속성의 사용방법을 알아보도록 하겠습니다.
- 본 예제에서는 각 태스크의 모든 속성을 살펴보기보다는 예제를 통해 대략의 사용 방법만을 설명하겠습니다.
- 아래의 표에 나온 7개 속성에 대해 알아보겠습니다.

mkdir	새로운 디렉터리를 생성합니다.
Copy	파일/디렉터리를 복사합니다.
Javac	자바 파일을 컴파일합니다.
jar	jar 파일을 생성합니다.
Javadoc	javadoc를 생성합니다.
Delete	파일/디렉터리를 삭제합니다.

## 3. 도구 기능 소개

### 3.5 태스크 (3/8)

Ant

- 태스크 mkdir은 새로운 디렉터리를 만들 때 사용됩니다.
- 사용방법

- 디렉터리 생성

```
<mkdir dir="${dist}">  
<mkdir dir="${dist}/lib/">
```

-> 위의 예는 속성 dist에 정의되어 있는 디렉터리 경로와 dist의 하위 디렉터리 lib를 생성하는 명령입니다.



## 3. 도구 기능 소개

### 3.5 태스크 (4/8)

Ant

- 태스크 copy는 디렉터리나 파일을 복사하는데 사용됩니다.
- 사용방법

- 하나의 파일을 복사

```
<copy file="myfile.txt" tofile="mycopy.txt" />
```

-> myfile.txt 파일을 mycopy.txt라는 파일로 복사합니다.

- 디렉터리를 다른 디렉터리로 복사

```
<copy todir="../new/dir">  
  <fileset dir="src_dir" />  
</copy>
```

-> src\_dir 디렉터리에 있는 파일들을 ../new/dir 이라는 디렉터리에 복사합니다

- 특정 디렉터리의 원하는 파일만 지정해 복사

```
<copy todir="../dest/dir">  
  <fileset dir="src_dir">  
    <exclude name="**/*.java">  
  </fileset>  
</copy>
```

-> src\_dir 디렉터리에서 확장자가 java인 파일들을 제외한 나머지 파일들을 ../dest/dir 디렉터리에 복사합니다.

## 3. 도구 기능 소개

### 3.5 태스크 (5/8)

Ant

- 태스크 javac는 java 소스파일을 컴파일 하는데 사용됩니다.
- 소스 디렉터리는 재귀적으로 탐색되며, class파일이 없거나 java소스파일 내용이 갱신된 경우에만 컴파일 합니다.
- 사용방법

- 컴파일

```
<javac srcdir="${src}:${src2}"
    destdir=${build} "
    include="mypackage/p1/**,mypackage/p2/**"
    exclude="mypackage/p1/testpackage/**"
    classpath="xyz.jar"
    debug="on"
/>
```

- > srcdir : 컴파일 할 소스 디렉터리를 지정합니다.  
(여기서는 \${src}와 \${src2}로 지정하였습니다.)
- > destdir : class파일이 생성될 디렉터리를 지정합니다
- > include : 컴파일 시 사용할 파일을 지정합니다.  
(여기서는 mypackage/p1,p2에 있는 파일만 사용합니다.)
- > exclude : 컴파일 시 사용하지 않을 파일을 지정합니다.
- > classpath : 클래스 패스를 지정합니다

## 3. 도구 기능 소개

### 3.5 태스크 (6/8)

Ant

- 태스크 jar은 지정된 파일을 jar로 묶습니다.
- 사용방법

- jar파일 생성

```
<jar destfile="${dist}/lib/app.jar"  
    basedir="${build}/classes"  
    include="mypackage/test/**"  
    exclude="**/Test.class"  
>
```

- >destfile : 생성할 jar파일명과 경로를 지정합니다. (여기서는 app.jar파일을 생성합니다.)
- >basedir : jar파일로 묶을 class 파일들이 위치한 경로를 지정합니다.
- >include : basedir에 지정한 경로에서 실제 사용할 파일을 지정합니다.
- >exclude : 제외할 파일을 지정합니다. (여기서는 Test.class파일을 제외합니다.)

## 3. 도구 기능 소개

### 3.5 태스크 (7/8)

Ant

- 태스크 javadoc은 javadoc문서를 생성합니다.
- 사용방법

- javadoc 문서 생성

```
<javadoc destdir="${doc}">  
  <fileset dir="${src}">  
    </fileset>  
  </javadoc>
```

-> \${src}에 지정한 디렉터리에서 javadoc문서를 생성하여 \${doc}디렉터리에 저장합니다.

## 3. 도구 기능 소개

### 3.5 태스크 (8/8)

Ant

- 태스크 delete은 하나의 파일 또는 디렉터리와 그 하위 디렉터리를 삭제합니다.
- 사용방법

- 하나의 파일 삭제

```
<delete file="/lib/ant.jar"/>
```

-> file에 지정한 하나의 파일을 삭제합니다.

- 디렉터리와 하위 디렉터리 삭제

```
<delete dir="lib"/>
```

-> dir에 지정한 하나의 파일을 삭제합니다

- 특정파일을 지정 후 삭제

```
<delete>  
  <fileset dir="." include="**/*.bak" />  
</delete>
```

-> fileset으로 지정한 디렉터리와 그 하위 디렉터리에서 include에 지정한 파일들을 삭제합니다.  
(여기서는 현재 디렉터리와 그 하위 디렉터리에서 확장자가 bak인 파일들을 삭제합니다.)

## 3. 도구 기능 소개

### 3.6 Property [1/2]

Ant

- Property

- 프로젝트를 진행하다보면 다양한 프로퍼티를 사용하게 됩니다.
- 컴파일할 소스 코드의 위치, 컴파일된 클래스 파일을 저장할 디렉토리, 압축한 파일을 위치시킬 디렉토리 등 유연한 개발을 위해서는 다양한 프로퍼티의 사용이 필수적이라 할 수 있습니다.
- Ant는 이처럼 빌드 과정에서 사용되는 다양한 프로퍼티를 지정할 수 있도록 하고 있으며, 또한 몇몇 개의 built-in 프로퍼티를 제공하고 있습니다.
- 프로퍼티의 지정은 property 태스크를 통해서 할 수 있으며, property 태스크는 아래의 표와 같은 속성을 갖고 있습니다.

속성	설명
name	설정할 프로퍼티의 이름입니다.
value	프로퍼티의 값입니다.
location	프로퍼티를 주어진 파일의 절대 파일명으로 지정합니다. 이 속성의 값이 절대 경로일 경우, '/'나 'W'와 같은 구분자는 현재 플랫폼에 알맞게 처리 됩니다. 절대 경로가 아닐 경우 프로젝트의 basedir에 상대적인 경로로 처리 됩니다.
refid	다른 곳에서 정의된 객체를 참조합니다.
resource	프로퍼티 파일을 나타내는 자원의 이름입니다.
file	프로퍼티 파일의 파일 이름입니다.
environment	환경 변수를 읽을 때 사용할 접두어, 만약 environment="myenv"라고 지정했다면, OS에 종속적인 환경변수인 PATH나 TEMP와 같은 값을 "myenv.PATH"나 "myenv.TEMP"와 같은 프로퍼티 이름을 사용하여 구할 수 있습니다
classpath	자원을 검색할 때 사용할 클래스패스 입니다.
classpathref	다른 곳에서 정의된 PATH에 대한 참조로서 주어진 자원을 검색할 때 사용할 클래스 패스 입니다.

## 3. 도구 기능 소개

### 3.6 Property [2/2]

- 예를 들어 “buildno” 속성의 값을 “356”으로 지정하고 싶다면 다음과 같이 하면 됩니다.
  - `<property name = “buildno” value=“365”/>`
- 특정한 파일에 저장된 프로퍼티를 사용하고 싶다면 다음과 같이 file 속성을 사용하면 됩니다.
  - `<property file=“build355.properties”/>`
- Ant의 빌드 파일에서는 자바에서 기본적으로 제공하는 “java.version”과 같은 시스템 프로퍼티를 기본적으로 사용할 수 있으며, 추가적으로 Ant 자체에서 기본적으로 제공하는 프로퍼티를 사용할 수 있습니다.
- 아래 표는 Ant가 기본적으로 제공하는 프로퍼티 목록입니다.

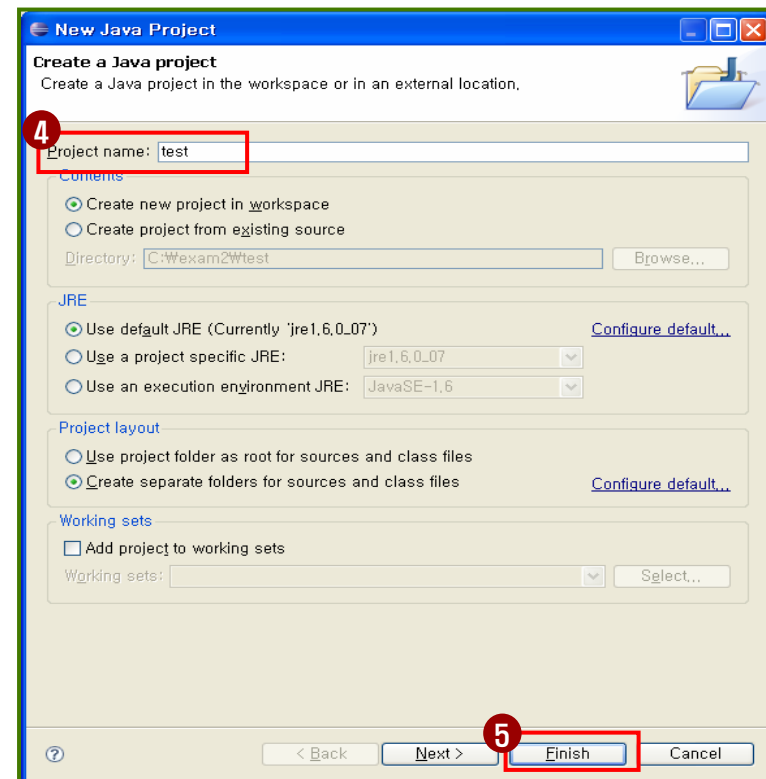
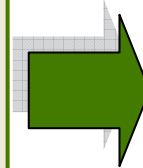
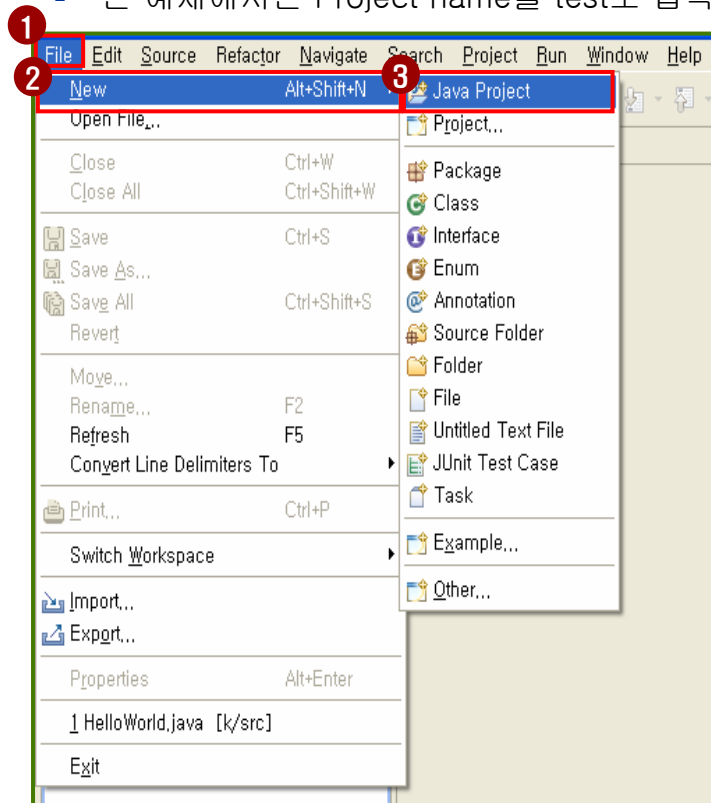
프로퍼티	설명
Basedir	프로젝트의 기본 디렉토리에 대한 절대 경로. <project> 태그의 base의 속성값을 사용합니다.
Ant.file	빌드 파일의 절대 경로 입니다.
ant.version	Ant의 버전을 표시합니다.
ant.project.name	현재 실행중인 프로젝트의 이름. <project> 태그의 name 속성값을 사용합니다.
Ant.java.version	Ant가 발견한 Java의 버전 입니다.

## 3. 도구 기능 소개

### 3.7 예제 소개 (1/3)

Ant

- 지금까지 설명한 내용을 바탕으로, Ant를 이용해보도록 하겠습니다.
- HelloWorld를 출력하는 간단한 프로그램을 이용하여 빌드파일을 작성하고, 실행해 보도록 하겠습니다.
- 먼저, Eclipse에서 자바 프로젝트를 생성합니다.
  - File -> New -> JavaProject -> Project name 입력 -> Finish
    - 본 예제에서는 Project name을 test로 입력하도록 하겠습니다.



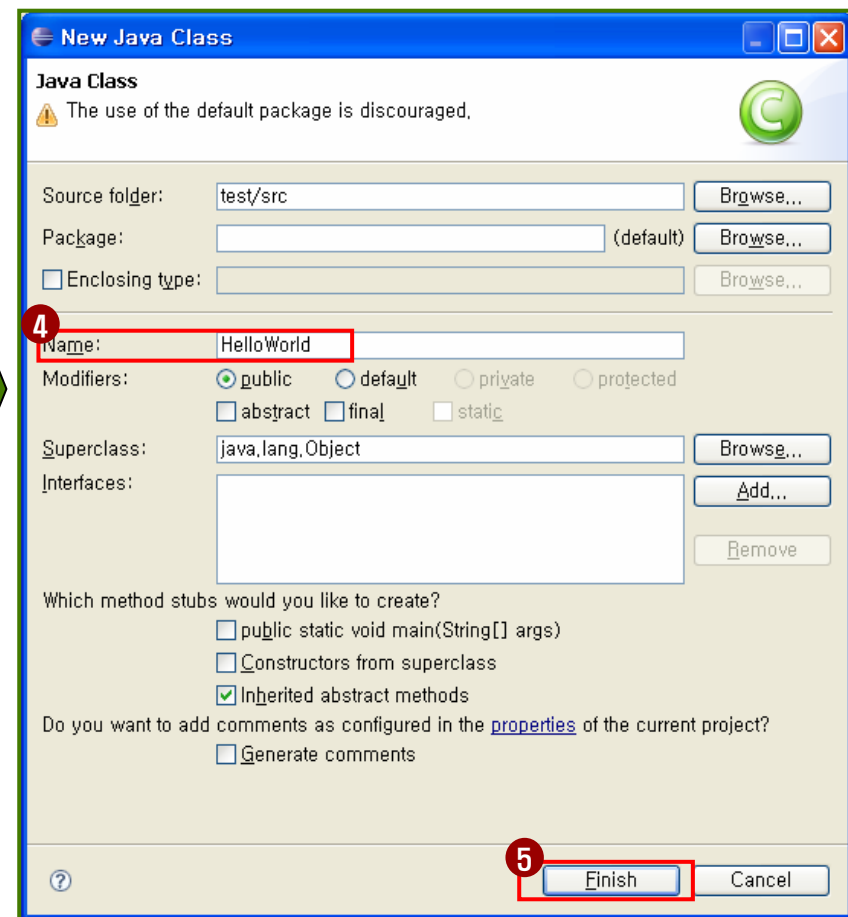
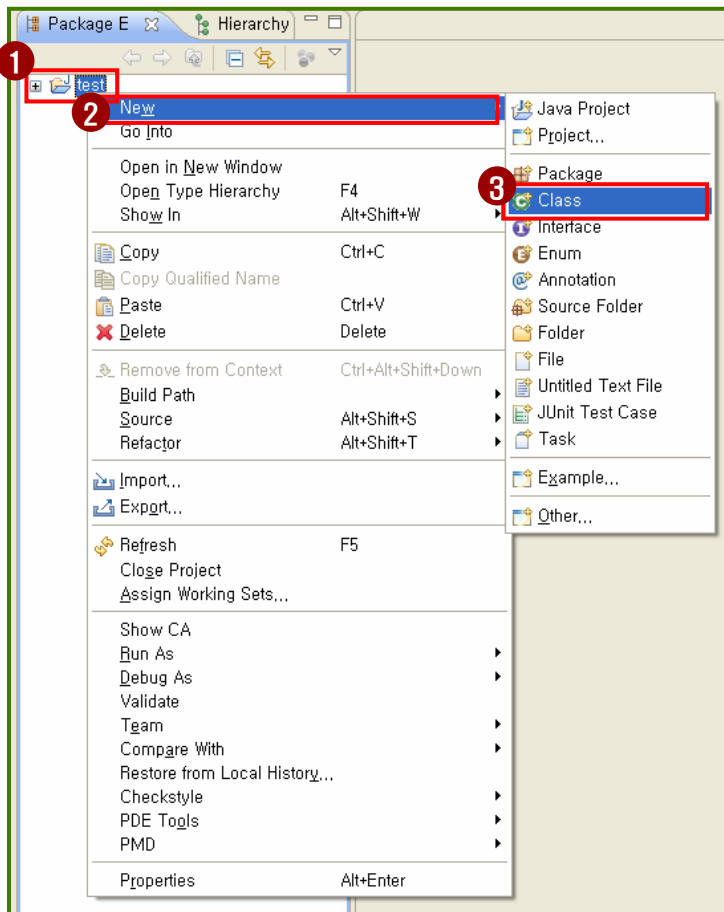


## 3. 도구 기능 소개

### 3.7 예제 소개 (2/3)

- Class 파일을 생성합니다.

- test 프로젝트 선택 후 마우스 오른쪽 버튼 클릭 -> Class -> Name 입력 -> Finish
  - 본 예제에서는 name을 HelloWorld로 입력하도록 하겠습니다.

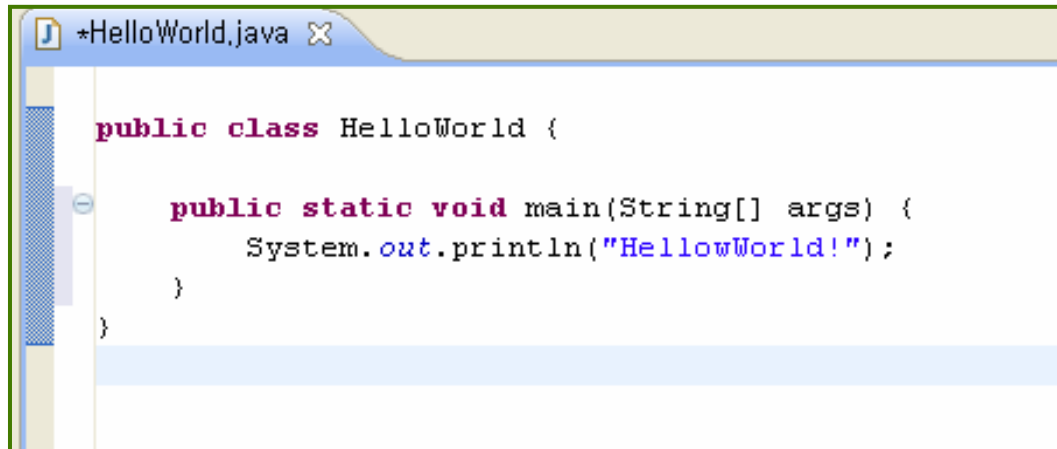


## 3. 도구 기능 소개

### 3.7 예제 소개 (3/3)

Ant

- HelloWorld 파일을 작성합니다.
  - 아래와 같이 작성합니다.



```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("HelloWorld!");  
    }  
}
```

## 3. 도구 기능 소개

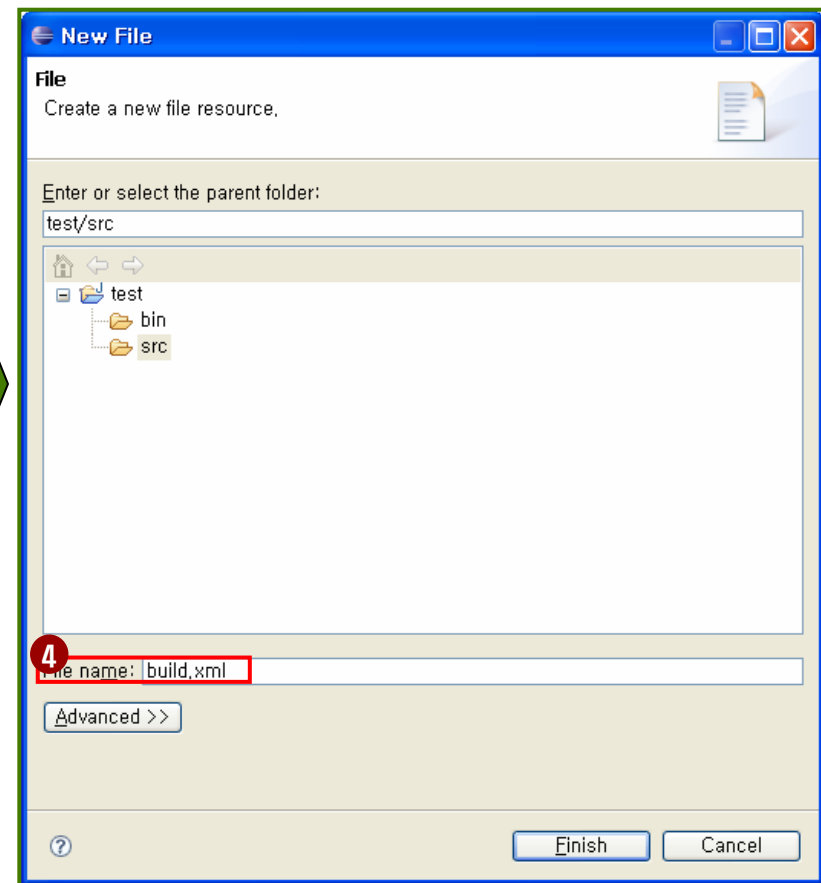
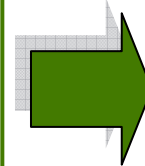
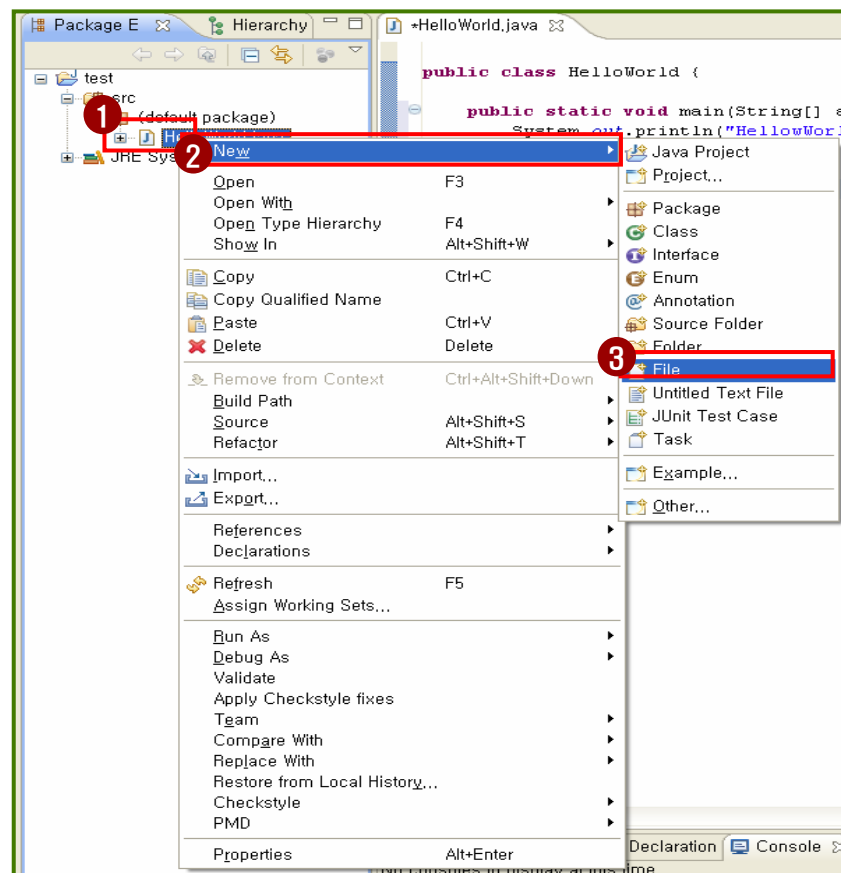
### 3.8 Ant 빌드 파일 작성하기 [1/13]

- 빌드파일을 작성하기 전에 빌드를 설계해야 합니다. 빌드 파일 역시 프로그래밍의 연속이므로 구체적으로 빌드를 어디에 사용하고 어떤식으로 진행할지를 결정해야 합니다.
- 이번 예제에서는 다음과 같이 빌드를 진행할 것입니다.
  - 배포 기능
    - Clean : 컴파일 디렉토리에 클래스가 존재하면 지웁니다.
    - Compile : 소스를 컴파일하며, 이때 클래스 패스를 명시합니다.
    - Mkjar : 컴파일된 클래스를 jar로 묶습니다.
    - Dist : 배포에 필요한 디렉토리 및 파일을 zip 파일로 묶습니다.
  - 실행기능
    - Run : HelloWorldApplication을 실행

### 3. 도구 기능 소개

#### 3.8 Ant 빌드 파일 작성하기 [2/13]

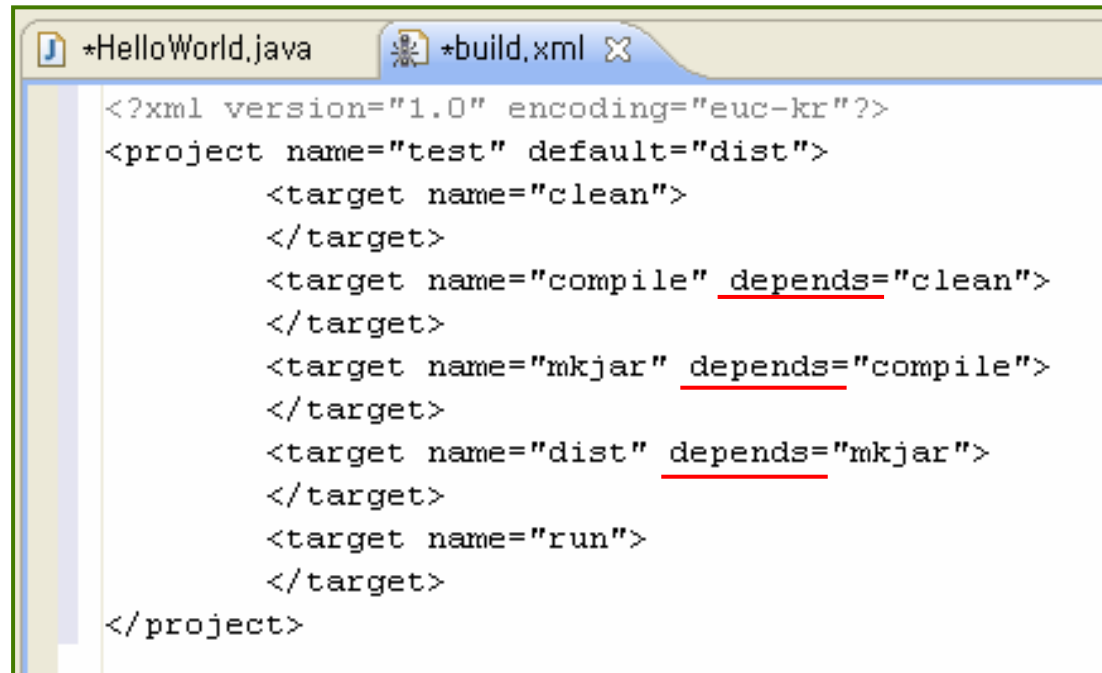
- 빌드파일을 작성합니다.
  - HelloWorld.java 선택 후 마우스 오른쪽 버튼 -> New -> File -> build.xml 생성



### 3. 도구 기능 소개

#### 3.8 Ant 빌드 파일 작성하기 [3/13]

- 빌드 파일을 위에서 정의한 필요한 타겟으로 분리하여 기본 골격을 작성합니다.



```
<?xml version="1.0" encoding="euc-kr"?>
<project name="test" default="dist">
  <target name="clean">
  </target>
  <target name="compile" depends="clean">
  </target>
  <target name="mkjar" depends="compile">
  </target>
  <target name="dist" depends="mkjar">
  </target>
  <target name="run">
  </target>
</project>
```

Depends의 의미는 그 Target이 실행되기 전에 depends에 있는 Target이 먼저 실행되어야 한다는 의미입니다.

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [4/13]

Ant

- 빌드시에 사용할 프로퍼티를 지정하도록 하겠습니다.
  - build.xml의 윗부분에 다음과 같이 property를 추가 합니다.
  - 아래는 build.xml의 현재까지 내용입니다.

```
<?xml version="1.0" encoding="euc-kr"?>
<project name="test" default="dist">
  <!-- 프로퍼티를 지정 합니다. -->
  <property file="build.properties"/>
  <property name="base.dir" value="."/>
  <property name="dist.dir" value="dist"/>
  <property name="build.dir" value="build"/>
  <property name="src.dir" value="src"/>
  <property name="jar.file" value="${version}_helloworld.jar"/>
  <property name="dist.file" value="${version}_helloworld.zip"/>
  <target name="clean">
  </target>
  <target name="compile" depends="clean">
    </target>
  <target name="mkjar" depends="compile">
  </target>
  <target name="dist" depends="mkjar">
  </target>
  <target name="run">
  </target>
</project>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 (5/13)

Ant

- build.xml을 추가로 작성 합니다.
- 최초 빌드시 시간을 출력하기 위한 부분을 추가 합니다.
  - Target 맨 윗부분에 prepare Target을 추가 합니다.

```
<!-- 빌드 초기 단계에 날짜 및 시간을 출력 -->
<target name="prepare">
    <tstamp>
        <format property="DSTAMP" pattern="yyyy.mm.dd"/>
        <format property="TSTAMP" pattern="HH:mm"/>
    </tstamp>
    <echo message="Build Start!! =====> ${DSTAMP}-${TSTAMP}"/>
</target>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [6/13]

Ant

- build.xml을 추가로 작성 합니다.
  - 먼저 만들어진 클래스가 있을지 몰라 삭제합니다.

```
<!-- 이미 있는 배포 파일 및 디렉토리를 삭제 -->
<target name="clean" depends="prepare">
    <delete dir="${dist.dir}"/>
    <delete dir="${build.dir}"/>
</target>
```



## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [7/13]

Ant

- 소스를 컴파일 하며 클래스 패스 및 소스 디렉토리를 지정하고 < javac> 태그를 이용하여 컴파일 합니다.

```
<!-- 소스 컴파일 -->
<target name="compile" depends="clean">
  <mkdir dir="${build.dir}"/>
  <javac deprecation="off" srcdir="${src.dir}" destdir="${build.dir}" listfiles="
    <classpath>
      <pathelement path="${base.dir}/lib" />
      <fileset dir="${base.dir}/lib">
        <include name="*.jar" />
      </fileset>
    </classpath>
  </javac>
</target>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [8/13]

Ant

- Mkdir Target을 완성합니다.

```
<!-- 빌드 디렉토리에 컴파일이 끝난 클래스를 jar 파일로 묶음 -->
<target name="mkjar" depends="compile">
    <mkdir dir="${dist.dir}"/>
    <jar destfile="${dist.dir}/${jar.file}"
        basedir="${build.dir}" />
</target>
```

### 3. 도구 기능 소개

#### 3.8 Ant 빌드 파일 작성하기 [9/13]

Ant

- Dist Target을 완성합니다.

```
<!-- 배포할 라이브러리와 소스등을 배포용 디렉토리로 복사하고 배포용 디렉토리를 zip으로 묶음 -->
<!-- jar와 소스, lib 아래의 jar를 배포용 디렉토리에 복사 후 zip으로 묶음 -->
<target name="dist" depends="mkjar">
    <copy todir="${dist.dir}/lib">
        <fileset dir="lib" />
    </copy>
    <copy todir="${dist.dir}/src">
        <fileset dir="src" />
    </copy>
    <zip destfile="${DSTAMP}_${dist.file}">
        <fileset dir="${dist}/.." />
    </zip>
</target>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 (10/13)

Ant

- run Target을 완성 합니다.

```
<!-- HelloWorld 클래스를 호출하여 실행 -->
<!-- <java> 태스크에 실행 파일을 입력하고 클래스 패스를 지정 -->
<target name="run">
    <java classname="test.HelloWorld">
        <classpath>
            <pathelement location="${dist.dir}/${jar.file}" />
            <pathelement path="${base.dir}/lib" />
            <fileset dir="${base.dir}/lib">
                <include name="*.jar" />
            </fileset>
        </classpath>
    </java>
</target>
</project>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [11/13]

Ant

- 완성된 build.xml 파일 입니다.

```
<?xml version="1.0" encoding="euc-kr"?>
<project name="test" default="dist">
  <!-- 프로퍼티를 지정 합니다. -->
  <property file="build.properties"/>
  <property name="base.dir" value="."/>
  <property name="dist.dir" value="dist"/>
  <property name="build.dir" value="build"/>
  <property name="src.dir" value="src"/>
  <property name="jar.file" value="${version}_helloworld.jar"/>
  <property name="dist.file" value="${version}_helloworld.zip"/>

  <!-- 빌드 초기 단계에 날짜 및 시간을 출력 -->
  <target name="prepare">
    <tstamp>
      <format property="DSTAMP" pattern="yyyy.mm.dd"/>
      <format property="TSTAMP" pattern="HH:mm"/>
    </tstamp>
    <echo message="Build Start!! ===== ${DSTAMP}-${TSTAMP}"/>
  </target>

  <!-- 이미 있는 배포 파일 및 디렉토리를 삭제 -->
  <target name="clean" depends="prepare">
    <delete dir="${dist.dir}"/>
    <delete dir="${build.dir}"/>
  </target>

  <!-- 소스 컴파일 -->
  <target name="compile" depends="clean">
    <mkdir dir="${build.dir}"/>
    <javac deprecation="off" srcdir="${src.dir}" destdir="${build.dir}" listfiles="
      <classpath>
        <pathelement path="${base.dir}/lib" />
        <fileset dir="${base.dir}/lib">
          <include name="*.jar" />
        </fileset>
      </classpath>
    </javac>
  </target>
```

```
<!-- 빌드 디렉토리에 컴파일이 끝난 클래스를 jar 파일로 묶음 -->
<target name="mkjar" depends="compile">
  <mkdir dir="${dist.dir}"/>
  <jar destfile="${dist.dir}/${jar.file}"
    basedir="${build.dir}" />
</target>

<!-- 배포할 라이브러리와 소스등을 배포용 디렉토리로 복사하고 배포용 디렉토리를 zip으로 묶음 -->
<!-- jar와 소스, lib 아래의 jar를 배포용 디렉토리에 복사 후 zip으로 묶음 -->
<target name="dist" depends="mkjar">
  <copy todir="${dist.dir}/lib">
    <fileset dir="lib" />
  </copy>
  <copy todir="${dist.dir}/src">
    <fileset dir="src" />
  </copy>
  <zip destfile="${DSTAMP}_${dist.file}">
    <fileset dir="${dist.dir}/.." />
  </zip>
</target>

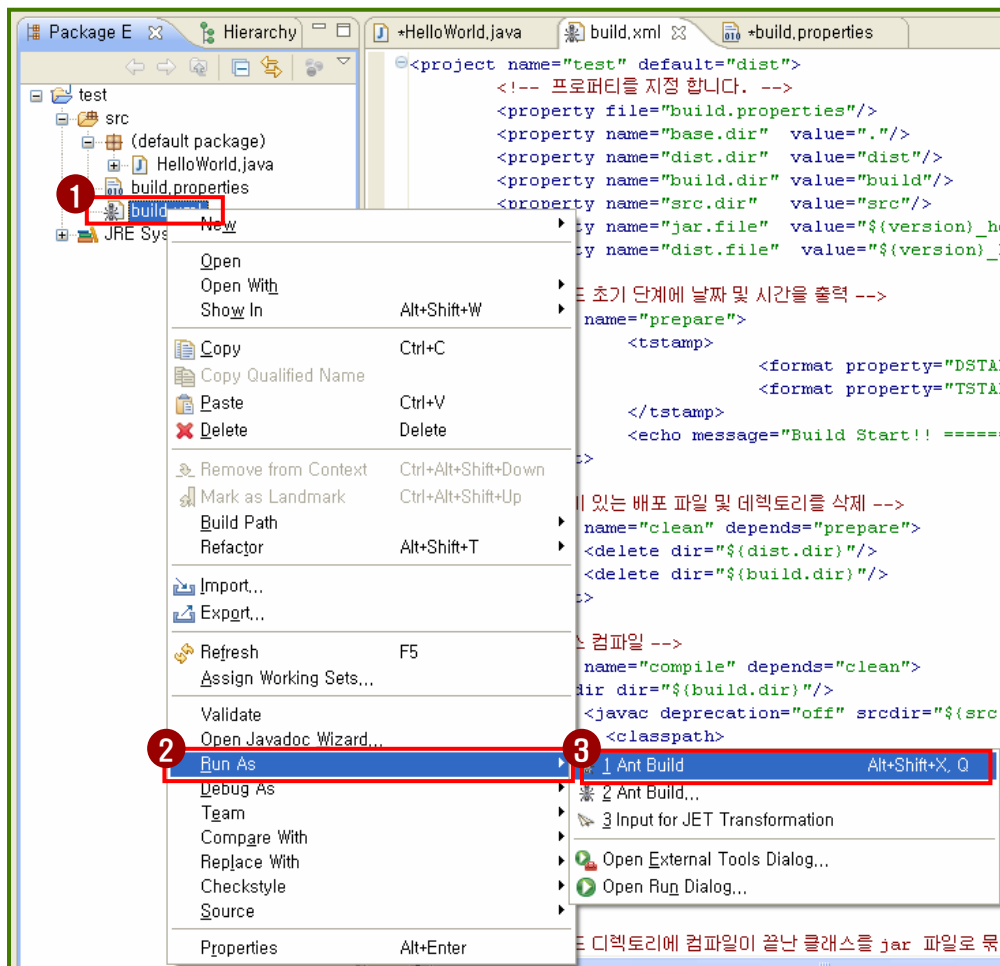
<!-- HelloWorld 클래스를 호출하여 실행 -->
<!-- <java> 태스크에 실행 파일을 입력하고 클래스 패스를 지정 -->
<target name="run">
  <java classname="test.HelloWorld">
    <classpath>
      <pathelement location="${dist.dir}/${jar.file}" />
      <pathelement path="${base.dir}/lib" />
      <fileset dir="${base.dir}/lib">
        <include name="*.jar" />
      </fileset>
    </classpath>
  </java>
</target>
</project>
```

## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 [12/13]

#### • Ant로 빌드하기

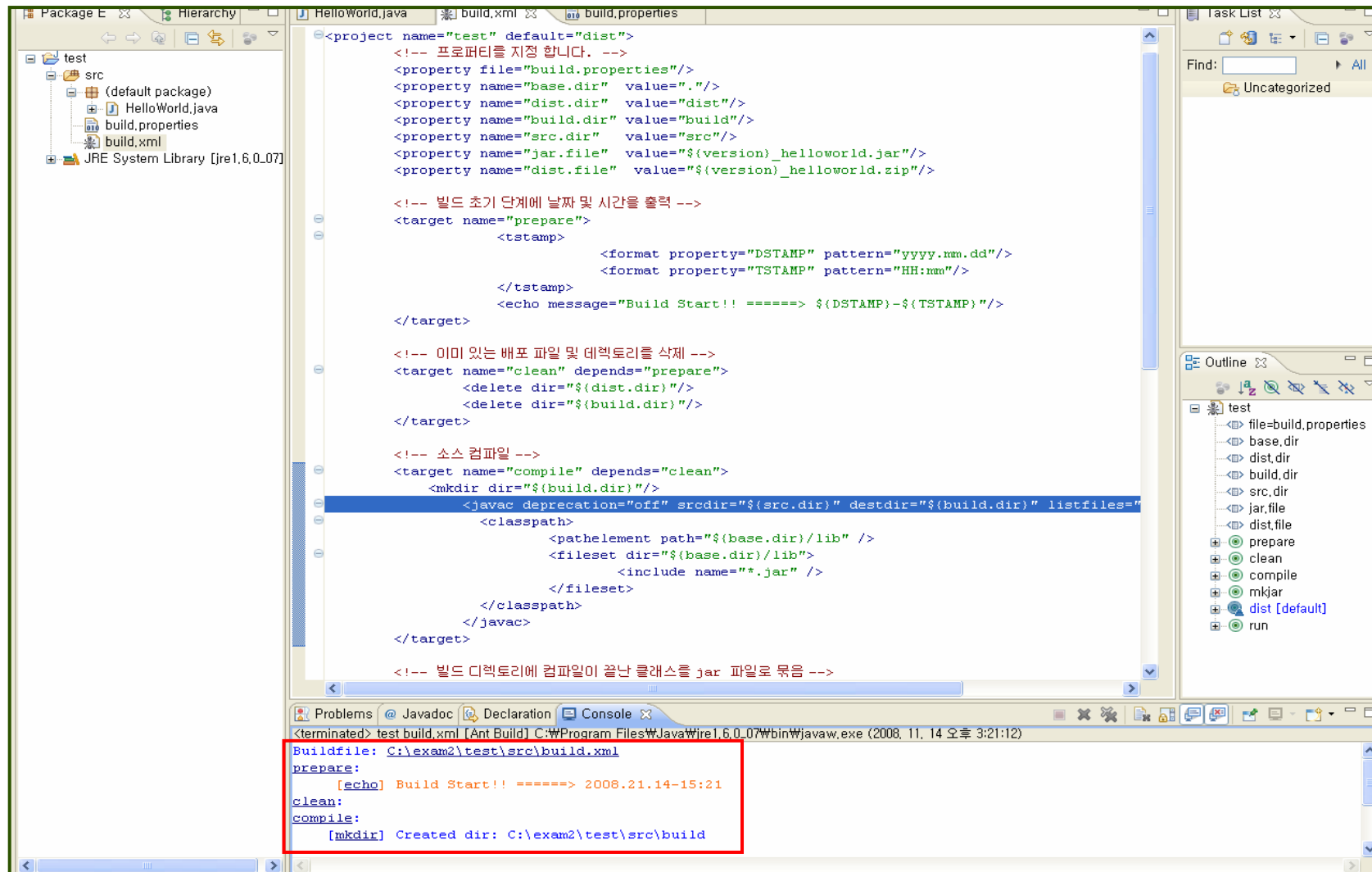
- build.xml 선택 후 마우스 오른쪽 버튼 -> Run As -> Ant Build



## 3. 도구 기능 소개

### 3.8 Ant 빌드 파일 작성하기 (13/13)

- 빌드 결과입니다.



## **4. 도구 활용 예제**



## 4. 도구 활용 예제

### 세부 목차

Ant

- 4.1 예제 소개
- 4.2 프로젝트 생성
- 4.3 Ant 빌드 스크립트의 제작
- 4.4 Ant의 동작 확인

## 4. 도구 활용 예제

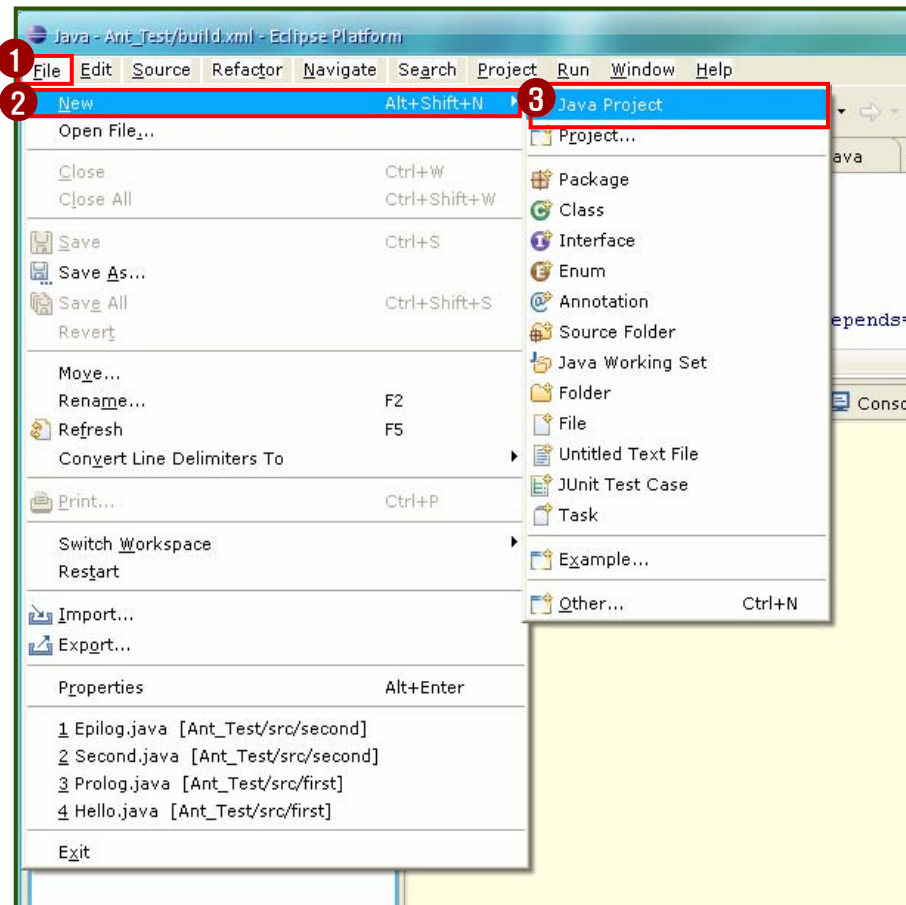
### 4.1 예제 소개

- 실제 프로젝트 빌드를 통해 Ant의 각 기능을 소개하기 위해 간단한 프로젝트를 생성합니다.
  - Ant\_Test라는 프로젝트를 생성합니다.
  - Ant\_Test 프로젝트 내에 first, second 패키지를 생성합니다.
  - First패키지 내에 Hello.java, Prolog.java파일을 생성합니다.
  - Second패키지 내에 Bye.java, Epilog.java파일을 생성합니다.
- Ant 빌드 스크립트 파일을 작성합니다.
  - Ant의 기능을 확인해 보기 위한 스크립트 파일을 작성합니다.
  - Outline를 통해 각 태스크들을 수행합니다.

## 4. 도구 활용 예제

### 4.2 프로젝트 생성 [1/9]

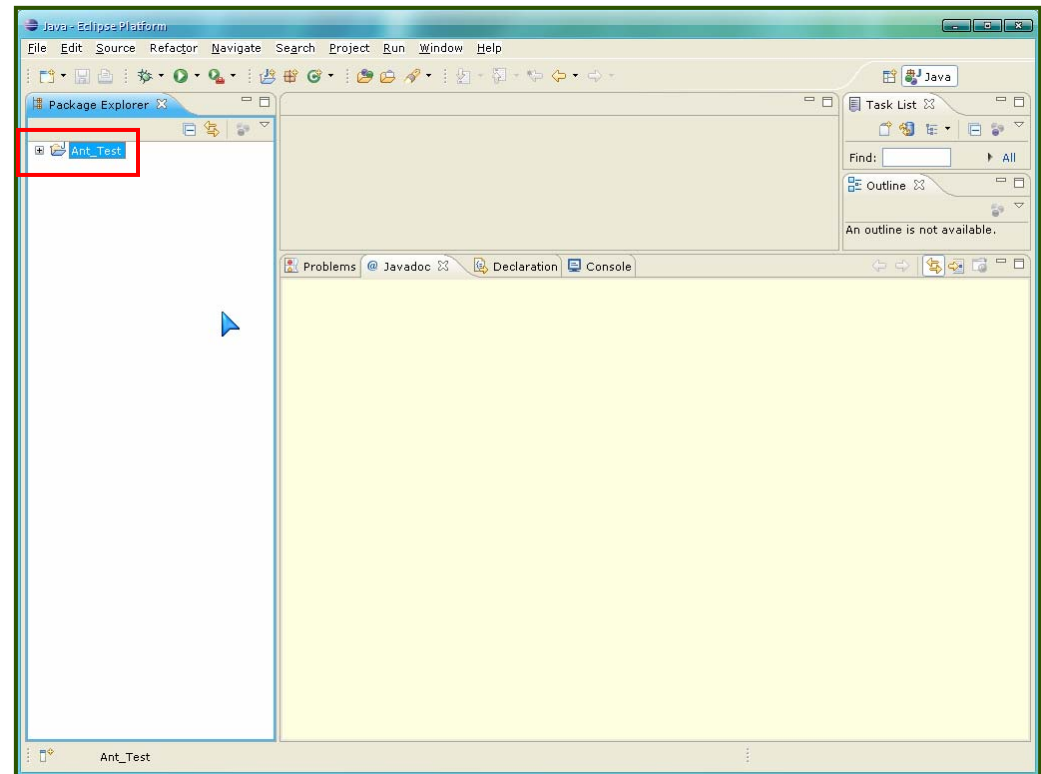
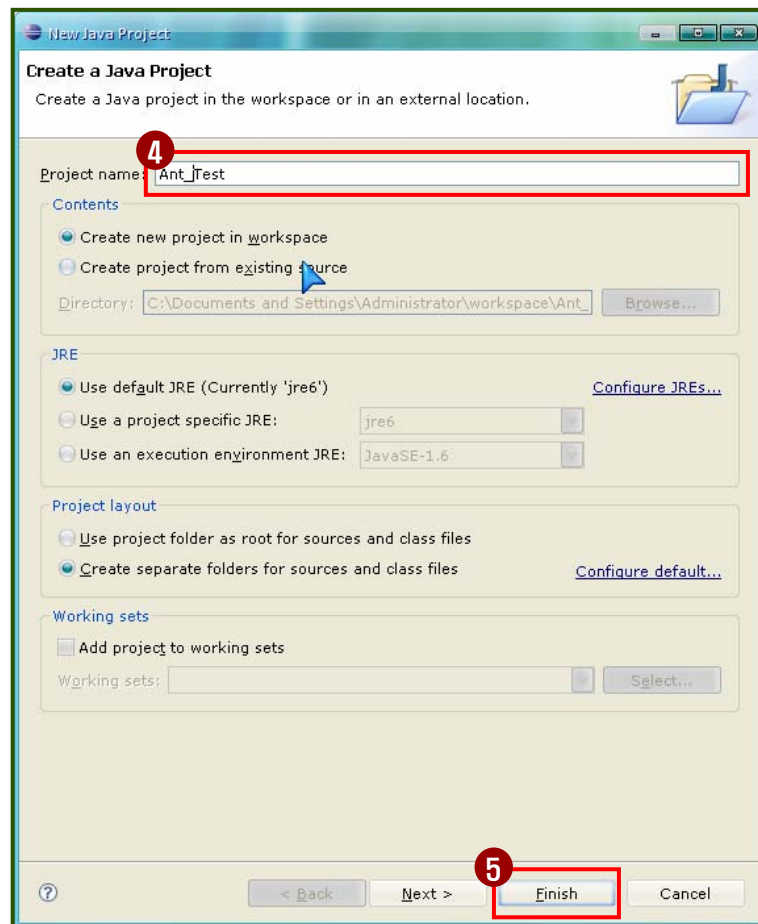
- 실제 Ant의 기능을 살펴보기 위해 간단한 프로젝트를 생성합니다.
- Ant\_Test라는 프로젝트를 생성해 보도록 하겠습니다.
  - Eclipse에서 File->New->Java Project를 선택합니다.



## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (2/9)

- Project name에 Ant\_Test라 입력합니다
- Finish 버튼을 클릭해 프로젝트를 생성합니다.



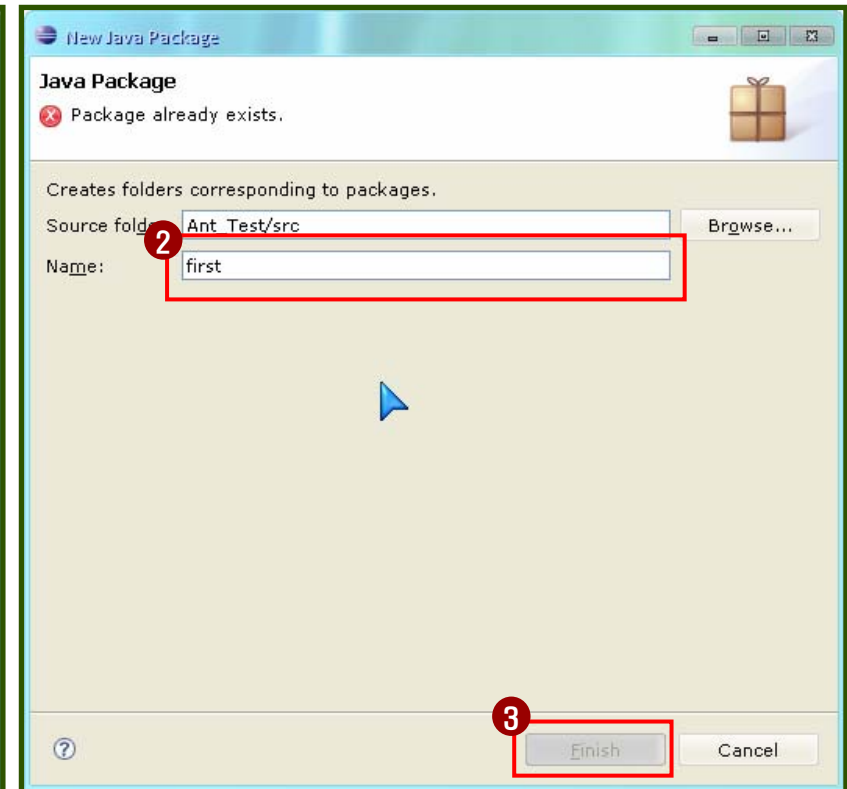
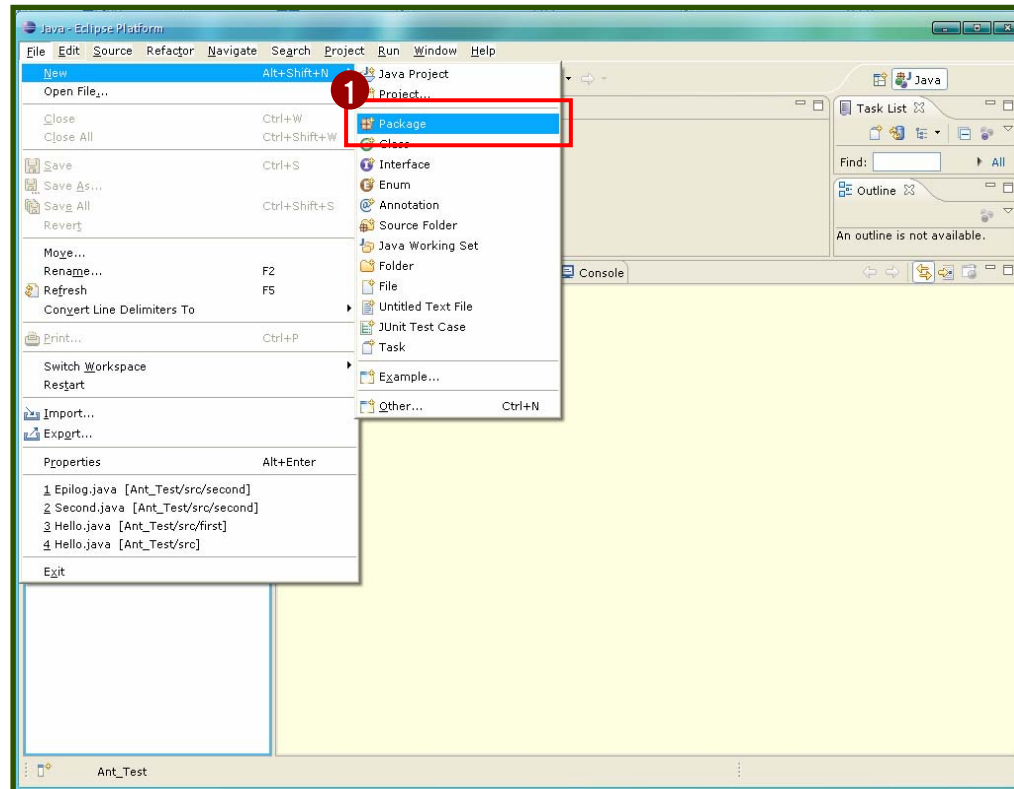
[Ant\_Test프로젝트가 생성된 모습]

## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (3/9)

Ant

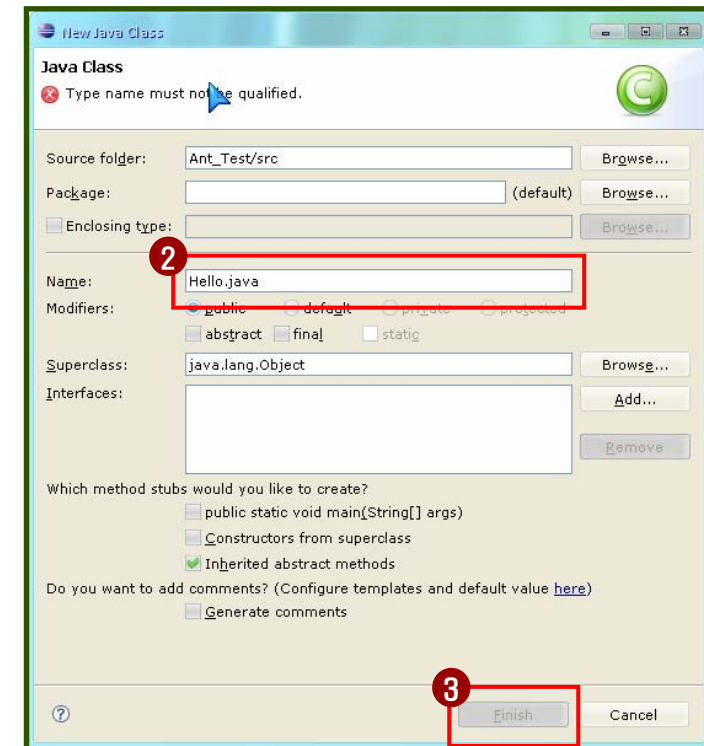
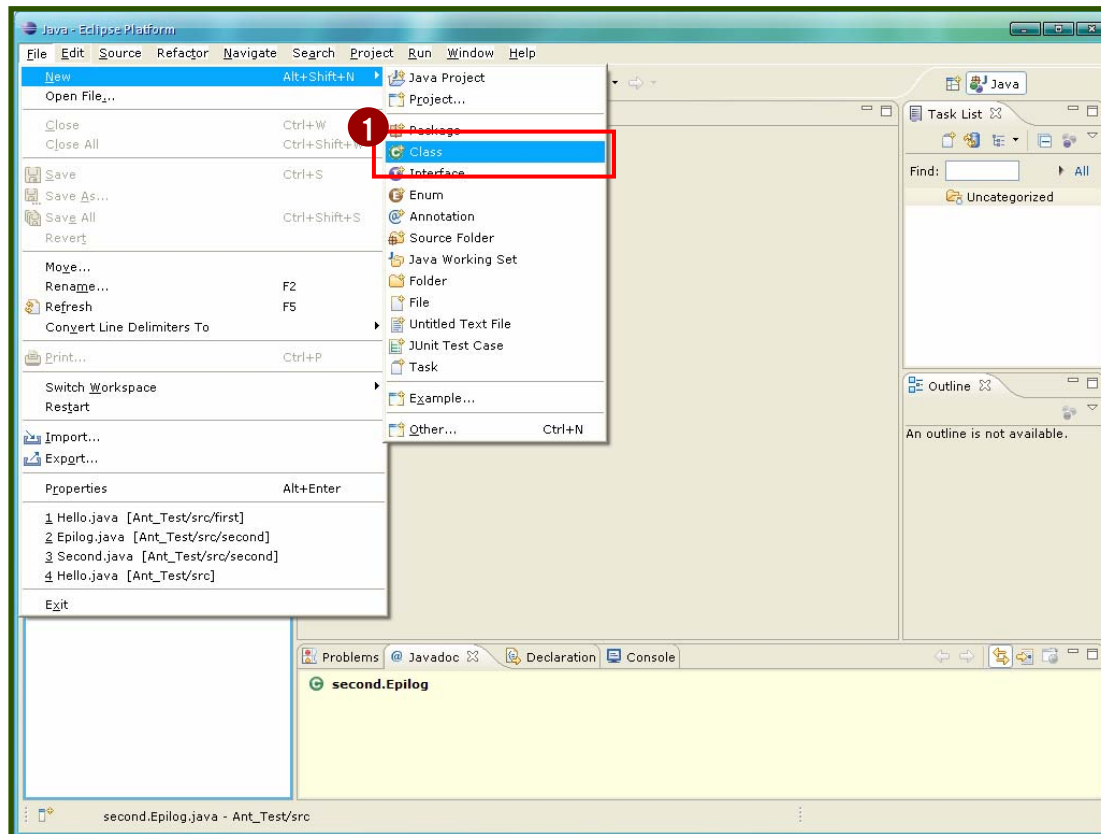
- Ant\_Test 프로젝트 내에 first와 second라는 패키지를 생성합니다.
  - File->New->Package를 선택합니다.
  - 우측에 보이는 창이 뜨게 되면 Name에 첫 번째 패키지이름인 first를 입력 후 Finish버튼을 클릭합니다.
  - 두 번째 패키지인 second도 동일한 방법을 사용해 생성합니다.



## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (4/9)

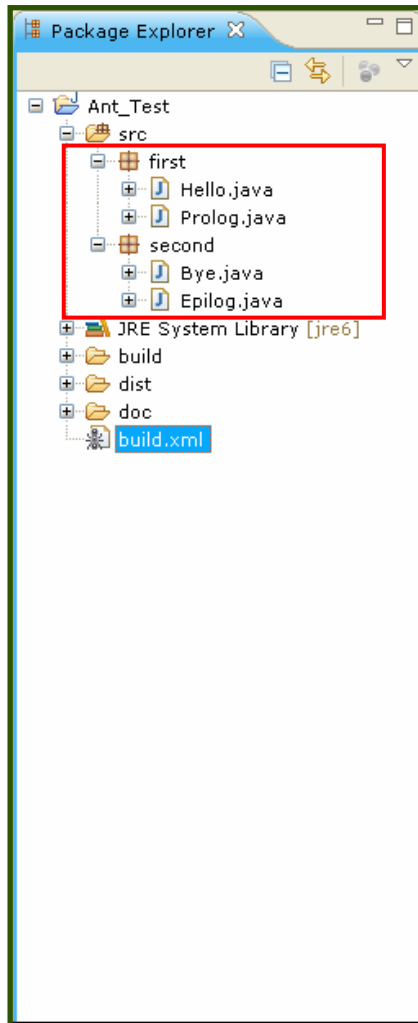
- 예제 클래스 파일들을 생성합니다.
  - first 패키지 안에 Hello.java, Prolog.java 파일을 생성합니다.
    - File->new->Class를 선택하면 우측에 보이는 창이 뜨게 됩니다.
    - 각각의 자바 파일명을 입력하고 Finish버튼을 클릭합니다.
  - 마찬가지로 second패키지 안에 Bye.java, Epilog.java 파일을 생성합니다.



## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (5/9)

- Package Explorer창에서 각 패키지와 자바 클래스 파일들이 생성된 것을 확인합니다.



## 4. 도구 활용 예제

### 4.2 프로젝트 생성 [6/9]

Ant

- 이제 각 자바 클래스 파일에 다음과 같이 입력합니다.
- 먼저 Hello.java파일의 내용입니다.

```
package first;

public class Hello {
    public void main(String[] args){
        String helloMsg = Prolog.getMsg();
        System.out.println(helloMsg);
    }
}
```



## 4. 도구 활용 예제

### 4.2 프로젝트 생성 [/7/9]

Ant

- 다음은 Prolog.java파일의 내용입니다.

```
package first;

public class Prolog {
    private static String helloMsg = "Hello, welcome to Ant world";

    public Prolog() {}

    static String getMsg() {
        return helloMsg;
    }
}
```

## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (8/9)

Ant

- 다음은 Bye.java파일의 내용입니다.

```
package second;

public class Bye {
    public void main(String[] args){
        String byeMsg = Epilog.getMsg();
        System.out.println(byeMsg);
    }
}
```

## 4. 도구 활용 예제

### 4.2 프로젝트 생성 (9/9)

Ant

- 다음은 Epilog.java파일의 내용입니다.

```
package second;

public class Epilog {
    private static String byeMsg = "Bye, see you soon";

    public Epilog() {}

    static String getMsg() {

        return byeMsg;
    }
}
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (1/9)

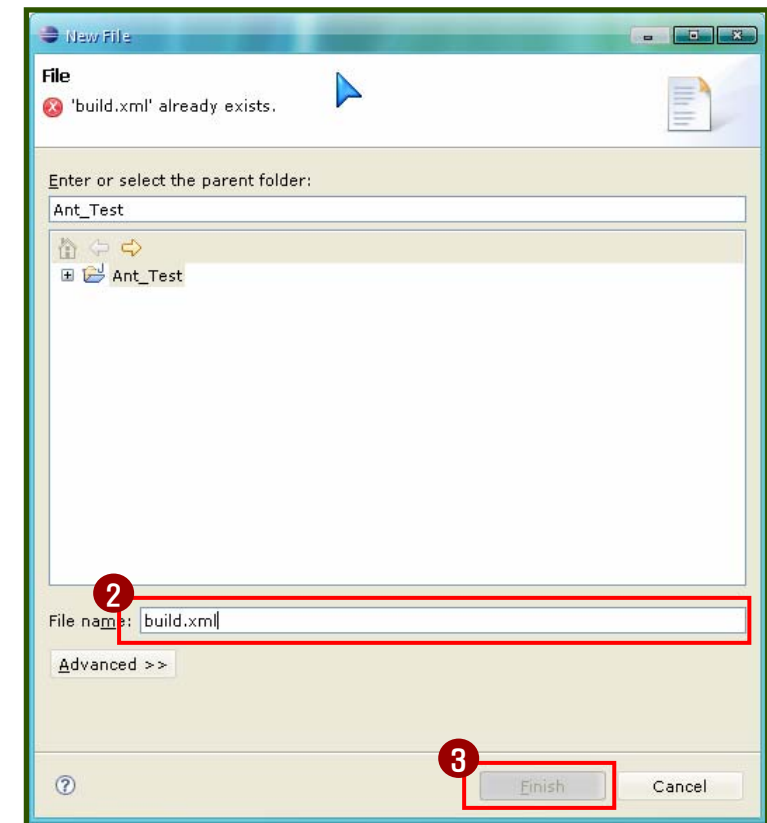
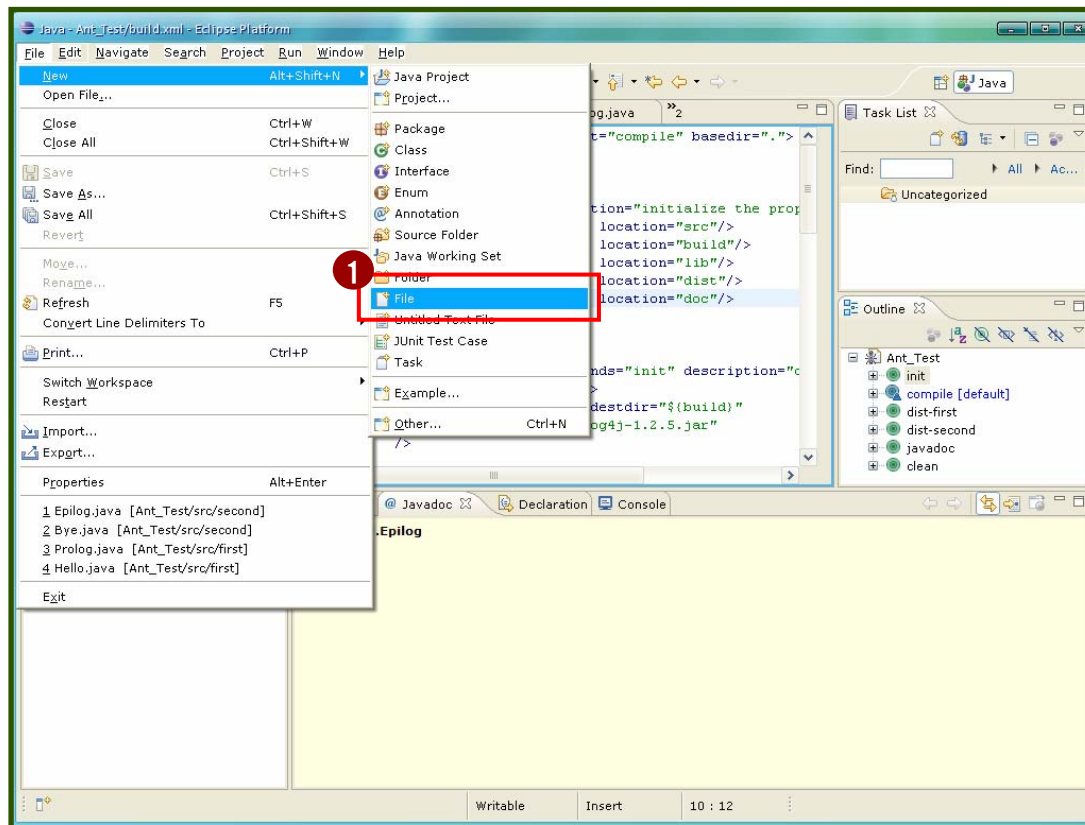
Ant

- 마지막으로 Ant의 빌드 스크립트를 작성합니다.
- 이번 예제에서는 다음과 같은 태스크들을 지정합니다.
  - init : 각 경로에 대한 속성값들을 지정합니다.
  - compile : 자바 파일들을 컴파일 합니다.
  - dist-first : first 패키지 내의 자바 파일들을 빌드 합니다.
  - dist-second : second 패키지 내의 자바 파일들을 빌드 합니다.
  - javadoc : 각 자바 파일들로부터 javadoc 문서를 생성합니다.
  - clean : 빌드를 통해 생성했던 결과물들을 제거합니다.

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (2/9)

- Ant 빌드 스크립트 파일인 Build.xml 파일을 생성합니다.
  - Ant는 디폴트로 build.xml이라 명명된 파일을 빌드 스크립트 파일이라 인식합니다.
  - File->New->File을 선택합니다.
  - 우측에 보이는 창이 뜨면 build.xml이라 입력하고 Finish버튼을 클릭합니다.



## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (3/9)

Ant

- 이제 build.xml의 내용을 기술하도록 하겠습니다.
- 먼저 init 태스크의 동작을 기술합니다.
  - Init태스크에서는 빌드에 사용할 각 디렉터리에 대한 별칭을 지정합니다.

```
<!-- init -->
<target name="init" description="initialize the properties">
  <property name="src"    location="src"/>
  <property name="build"  location="build"/>
  <property name="lib"    location="lib"/>
  <property name="dist"   location="dist"/>
  <property name="doc"    location="doc"/>
</target>
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (4/9)

Ant

- 다음은 compile태스크의 동작을 기술합니다.
  - compile태스크에서는 자바 소스 코드의 경로와 출력 클래스 파일이 위치할 경로, 사용할 classpath등을 지정합니다.
  - Compile동작은 init태스크를 의존관계로 갖기 때문에, compile수행 전에 init 태스크가 수행되지 않았다면 init을 먼저 수행하고 compile동작을 하게 됩니다.

```
<!-- compile -->
<target name="compile" depends="init" description="compile the source files">
  <mkdir dir="${build}" />
  <javac srcdir="${src}" destdir="${build}"
    classpath="${lib}/log4j-1.2.5.jar"
  />

  <copy todir="${build}">
    <fileset dir="${src}">
      <include name="**/*.properties"/>
    </fileset>
  </copy>
</target>
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (5/9)

Ant

- 첫 번째 패키지를 jar로 묶는 dist-first 태스크의 동작을 기술합니다.
  - Dist-first에서는 컴파일 된 첫 번째 패키지의 클래스 파일들만을 사용해 first.jar파일을 생성합니다.
  - 이 동작은 compile을 의존관계로 갖기 때문에, dist-first수행 전에 compile이 수행되지 않았다면 compile을 먼저 수행합니다

```
<!-- dist-first -->
<target name="dist-first" depends="compile" description="first package" >
  <mkdir dir="${dist}" />
  <jar jarfile="${dist}/first.jar">
    <fileset dir="${build}">
      <exclude name="second/*.*" />
    </fileset>
  </jar>
</target>
```



## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (6/9)

Ant

- 두 번째 패키지를 jar로 묶는 dist-second 태스크의 동작을 기술합니다.
  - dist-second에서는 컴파일 된 두 번째 패키지의 클래스 파일들만을 사용해 second.jar파일을 생성합니다.
  - 이 동작 역시 dist-first와 마찬가지로 compile을 의존관계로 갖기 때문에, dist-second수행 전에 compile이 수행되지 않았다면 compile을 먼저 수행합니다.

```
<!-- dist-second -->
<target name="dist-second" depends="compile" description="second package" >
  <mkdir dir="${dist}" />
  <jar jarfile="${dist}/second.jar">
    <fileset dir="${build}">
      <exclude name="first/*.*" />
    </fileset>
  </jar>
</target>
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (7/9)

Ant

- Javadoc문서를 생성하는 javadoc태스크의 동작을 기술합니다.
  - Javadoc태스크에서는 지정된 경로에 위치한 자바 파일로부터 javadoc문서를 생성합니다.

```
<!-- javadoc -->
<target name="javadoc" depends="init" description="create java doc">
  <mkdir dir="${doc}" />
  <javadoc destdir="${doc}">
    <fileset dir="${src}">
    </fileset>
  </javadoc>
</target>
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (8/9)

Ant

- 마지막으로 clean태스크의 동작을 기술합니다.
  - Clean태스크에서는 빌드 스크립트를 통해 생성한 결과물들을 제거합니다.

```
<!-- clean -->  
<target name="clean" depends="init" description="clean up">  
    <delete dir="${build}"/>  
</target>
```

## 4. 도구 활용 예제

### 4.3 Ant 빌드 스크립트 제작 (9/9)

- 다음은 예제에서 사용하려는 Ant 빌드 스크립트가 모두 기술된 build.xml 내용입니다.
- 첫 줄과 마지막 줄에 프로젝트 지정과 관련된 스크립트가 추가되었습니다.

1

```

<project name="Ant_Test" default="dist-first" basedir=".">

  <!-- init -->
  <target name="init" description="initialize the properties">
    <property name="src" location="src"/>
    <property name="build" location="build"/>
    <property name="lib" location="lib"/>
    <property name="dist" location="dist"/>
    <property name="doc" location="doc"/>
  </target>

  <!-- compile -->
  <target name="compile" depends="init" description="compile the source files">
    <mkdir dir="${build}" />
    <javac srcdir="${src}" destdir="${build}"
      classpath="${lib}/log4j-1.2.5.jar"
    />

    <copy todir="${build}">
      <fileset dir="${src}">
        <include name="**/*.properties"/>
      </fileset>
    </copy>
  </target>

  <!-- dist-first -->
  <target name="dist-first" depends="compile" description="first package">
    <mkdir dir="${dist}" />
    <jar jarfile="${dist}/first.jar">
      <fileset dir="${build}">
        <exclude name="second/*.*" />
      </fileset>
    </jar>
  </target>

```

[build.xml (1)]

2

```

    <!-- dist-second -->
    <target name="dist-second" depends="compile" description="second package">
      <mkdir dir="${dist}" />
      <jar jarfile="${dist}/second.jar">
        <fileset dir="${build}">
          <exclude name="first/*.*" />
        </fileset>
      </jar>
    </target>

    <!-- javadoc -->
    <target name="javadoc" depends="init" description="create java doc">
      <mkdir dir="${doc}" />
      <javadoc destdir="${doc}">
        <fileset dir="${src}">
        </fileset>
      </javadoc>
    </target>

    <!-- clean -->
    <target name="clean" depends="init" description="clean up">
      <delete dir="${build}" />
    </target>
  </project>

```

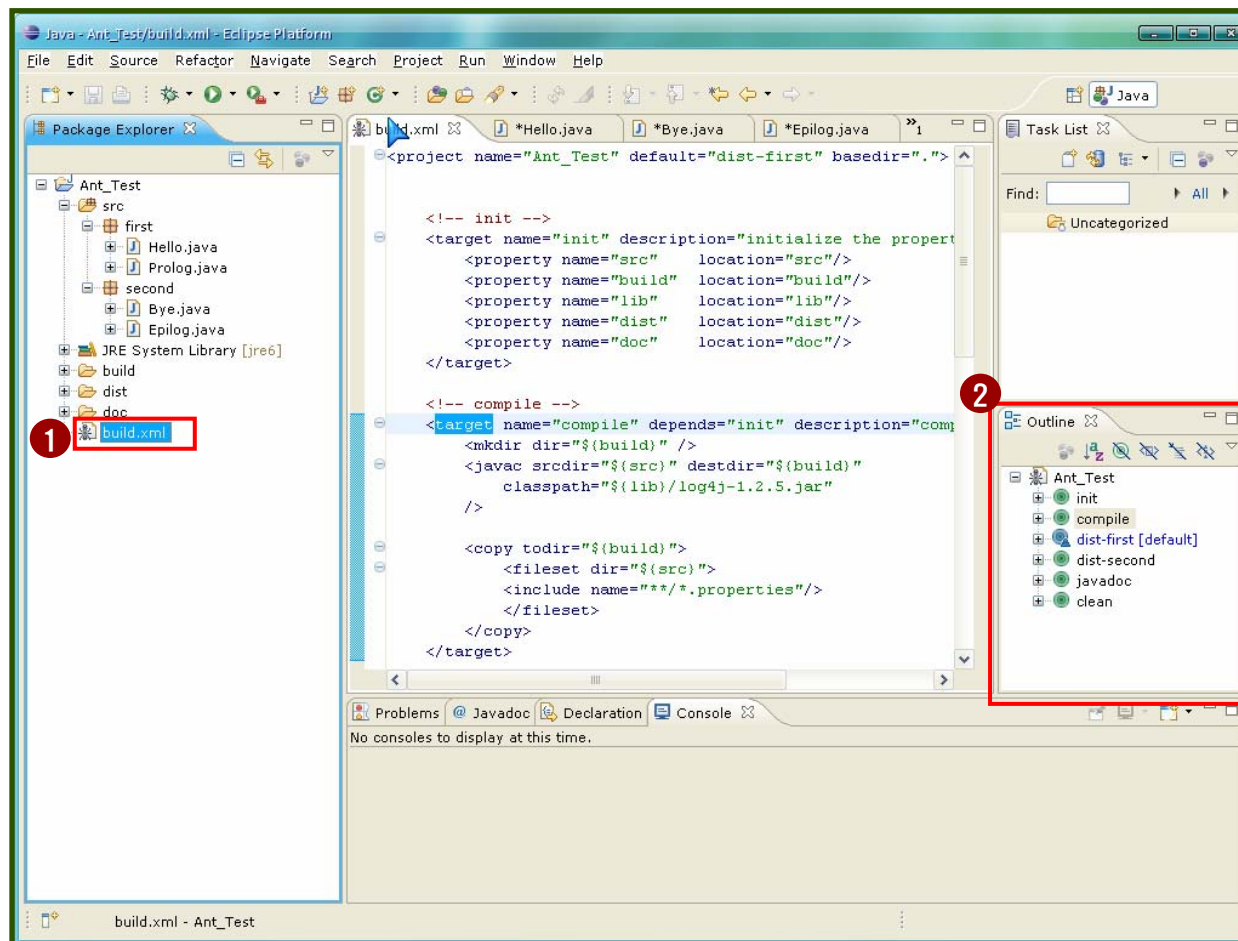
[build.xml (2)]

## 4. 도구 활용 예제

### 4.4 Ant 의 동작 확인 (1/4)

Ant

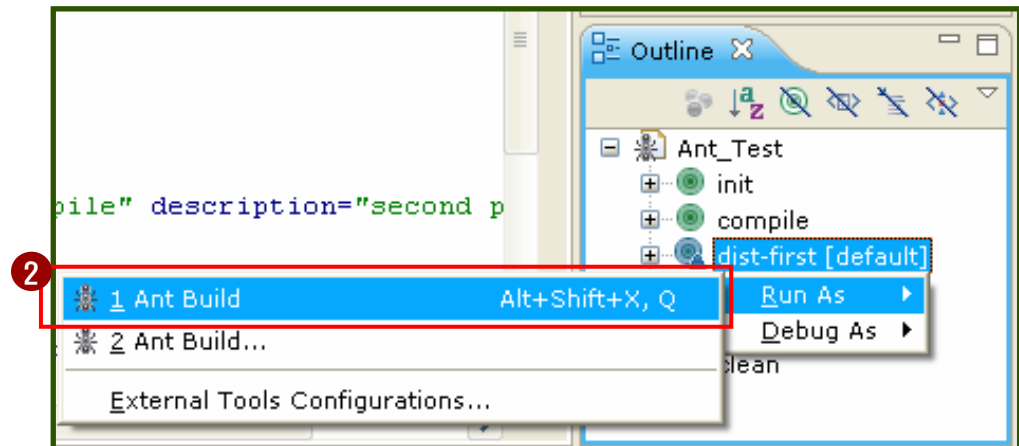
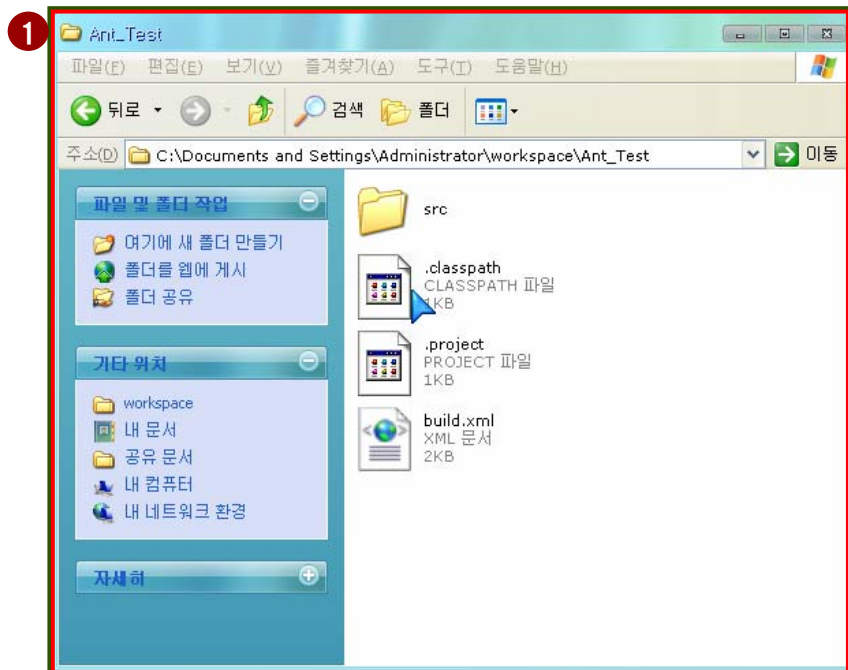
- 프로젝트 생성과 Ant 빌드 스크립트 작성이 완료되었습니다.
- 각 태스크의 동작을 확인해 보기 위해 Package Explorer의 build.xml파일을 선택해 봅니다.
- 오른쪽의 outline창에 우리가 작성했던 태스크들의 목록이 뜨는 것을 확인할 수 있습니다.



## 4. 도구 활용 예제

### 4.4 Ant 의 동작 확인 (2/4)

- 태스크들을 동작시켜 보도록 하겠습니다.
  - 먼저 현재 프로젝트 소스 트리 경로의 내용을 확인해 봅니다.
    - 현재 자바 파일이 위치한 src 디렉터리와 build.xml파일이 있는 것을 확인 할 수 있습니다.
  - 태스크 dist-first를 수행해 보도록 하겠습니다.
  - Outline창의 dist-first에서 마우스 오른쪽 버튼 클릭->Run As->Ant Build를 클릭합니다.



## 4. 도구 활용 예제

### 4.4 Ant 의 동작 확인 (3/4)

- Eclipse의 콘솔 창에 수행 결과가 출력됩니다.
  - 의존관계에 따라 init -> compile -> dist-first 순으로 수행되는 것을 확인 할 수 있습니다..
  - 자바 파일이 컴파일 되고, dist-first에 정의된 동작에 따라 first.jar파일이 생성되는 것을 확인 할 수 있습니다.
  - BUILD SUCCESSFUL이 출력되면 태스크 수행이 완료된 것입니다.

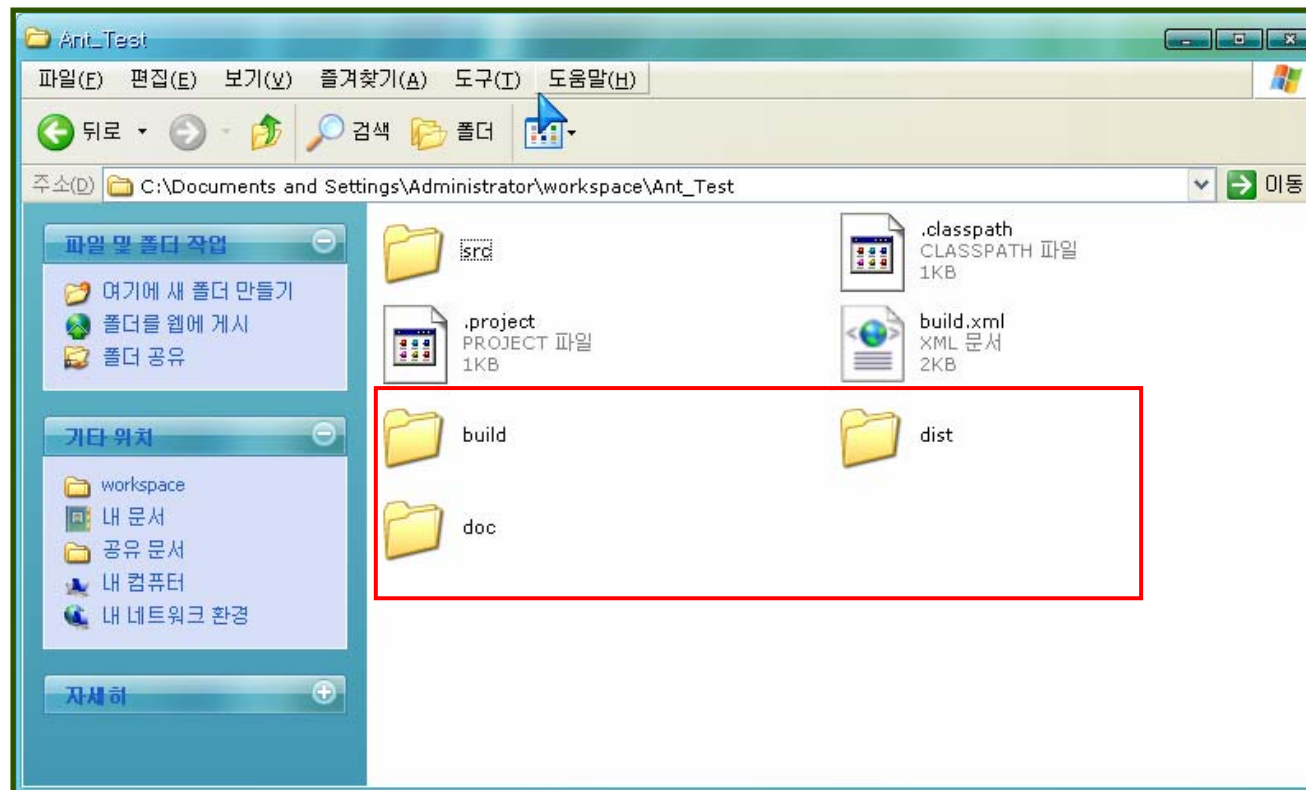
```

<terminated> Ant_Test build.xml (2) [Ant Build] C:\Program Files\Java\jre6\bin\javaw.exe (2008. 10. 27. 오후 2:13:23)
Buildfile: C:\Documents and Settings\Administrator\workspace\Ant_Test\build.xml
1 init:
2 compile:
   [mkdir] Created dir: C:\Documents and Settings\Administrator\workspace\Ant_Test\build
   [javac] Compiling 4 source files to C:\Documents and Settings\Administrator\workspace\Ant_Test\build
3 dist-first:
   [mkdir] Created dir: C:\Documents and Settings\Administrator\workspace\Ant_Test\dist
   [jar] Building jar: C:\Documents and Settings\Administrator\workspace\Ant_Test\dist\first.jar
BUILD SUCCESSFUL
Total time: 562 milliseconds
  
```

## 4. 도구 활용 예제

### 4.4 Ant 의 동작 확인 (4/4)

- dist-second, javadoc 태스크도 동일한 방법으로 수행합니다.
- 위의 태스크들이 성공적으로 수행되면, 프로젝트 소스 트리가 다음과 같이 변한 것을 확인 할 수 있습니다.
  - build : 컴파일 된 클래스 파일들이 저장되는 디렉터리
  - dist : 패키지들의 jar파일이 저장되는 디렉터리
  - doc : javadoc 문서가 저장되는 디렉터리





## **5. FAQ**

## 5. FAQ

Ant

질문1) Ant은 무료입니까?



답1) 네. 무료입니다. Eclipse설치 시 기본으로 설치되며 <http://Ant.apache.org>에서 무료로 다운받을 수 있습니다.

질문2) 설명된 기능들 이외에 다른 기능들도 더 있습니까?



답2) 네. 설명된 것보다 더 많은 기능들이 있습니다. Ant는 프로젝트 빌드를 위한 스크립트를 지원하기 때문에 보다 많은 표현이 가능합니다. 더 구체적인 기능 설명은 공식 홈페이지에 가시면 자세한 정보를 얻을 수 있습니다.

## 6. 도구 평가

## 6. 도구 평가

Ant

- XML을 사용해본 경험이 있다면 문법이 매우 간단하여 쉽게 배울 수 있습니다.
- 쉽게 사용할 수 있으므로, 대규모의 Make 기반 소프트웨어 프로젝트에서 당연히 존재해 왔던 makefile 전달 기술자를 제외할 수 있습니다.
- 크로스 플랫폼을 지원하고 있기 때문에 코딩이 가능한 방법으로 Java의 클래스패스와 파일 디렉토리 구조를 다룰 수 있습니다.
- 실행 속도가 매우 빠르며, Java 컴파일러나 JAR 파일을 만드는 것과 같은 Java 루틴은 Ant의 JVM 상에서 실행할 수 있습니다. 이 방법은 별도의 JVM을 사용하여 구동할 때 소요되는 지연 문제를 해소할 수 있습니다.
- eXtreme Programming 스타일의 유닛 테스트를 위해, JUnit 테스트 프레임워크와 밀접하게 통합되어 있습니다.

## 7. 용어집

## 7. 용어집

- 본 매뉴얼에서 사용하고 있는 용어가 정리되어 있습니다.

용어	설명
XML	XML(Extensible Markup Language)은 W3C에서 다른 특수 목적의 마크업 언어를 만드는 용도에서 권장되는 다목적 마크업 언어입니다.
빌드 스크립트 파일	빌드 스크립트 파일이란 여러 개의 파일 또는 기타 프로젝트에 필요한 것들로 구성되는 복잡한 소프트웨어 프로젝트에서 사용자가 손수 프로젝트에 필요한 결과물을 만들어 내는 것이 아니라, 미리 기술된 프로젝트 빌드 순서에 따라 자동으로 결과물을 생성 해주기 위한 파일입니다.
JAR	JAR 파일은 웹 브라우저에서 빠르게 다운로드 할 수 있도록, 자바 애플릿을 위한 클래스, 이미지 및 사운드 파일들을 하나의 파일에 압축하여 담고 있는 파일입니다. jar 유틸리티가 JAR 파일을 만들거나, 또는 JAR 파일의 목록을 보여주고, 거기에서 개별 파일들로 추출하는 등의 일을 할 수 있게 해 줍니다 .