

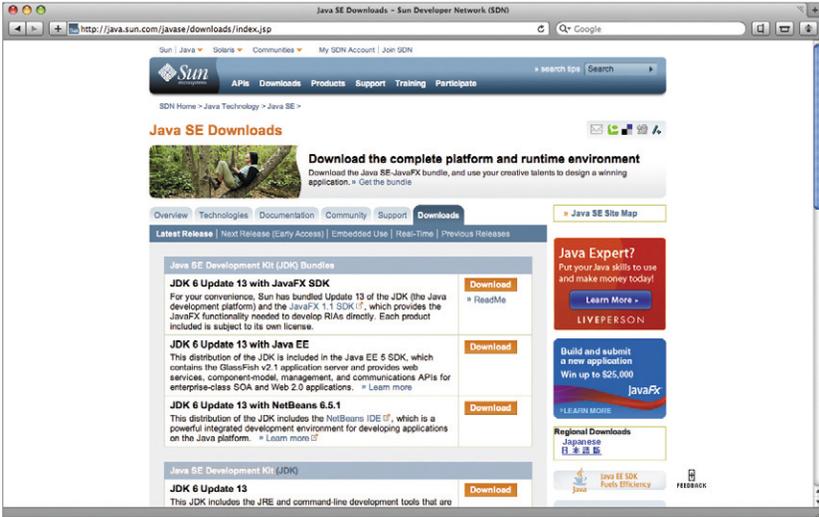


완전 초보 개발자를 위한 자바와 이클립스 설치 및 사용법

‘개발자를 위한 테스트 책에 무슨 이클립스 설명인가?’ 라고 하는 사람이 분명히 있을 것이다. 물론 이 책은 개발자를 위해서 쓴 책이지만, 이 책을 읽는 여러분 중에는 반드시 자바를 아직 잘 모르는 학생이나, 품질을 담당하는 사람이 포함되어 있을 것이다. 특히 이 책에서 이클립스와 연계된 내용이 많기 때문에 (막상 알면 별거 아니지만) 이클립스 기본 사용법을 모르는 독자 대상으로 가볍게 정리해 놓았다. 참고로 여기에 있는 화면 캡처는 Mac으로 수행했지만, Windows 버전과 메뉴가 대부분 비슷하여 불편함이 없을 것이다.

자바 설치

당연한 이야기지만 이클립스를 사용하려면 자바가 설치되어 있어야만 한다. 자바는 <http://java.sun.com/javase/downloads>에서 다운로드한다. 반드시 ‘Java SE Development Kit(JDK)’로 되어 있는 것을 다운로드하기 바란다.



Sun의 Java SE 다운로드 페이지

파일을 다운로드한 후 설치를 시작한다. 설치 시에는 긍정적인 단어의 버튼을 클릭 하여야만 설치가 정상적으로 완료된다. 설치 완료 후에는 커맨드 창을 띄워서 (Windows 시작→ 실행... → cmd라고 입력하면 창이 나타남) 다음의 명령어를 사용하여 제대로 설치가 되었는지 확인해 보기 바란다.

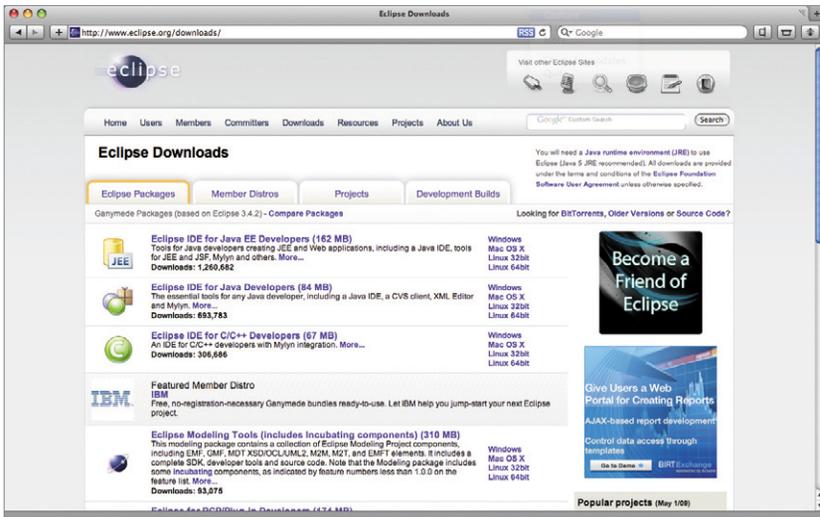
```
C:\>java -version
java version "1.6.0_XX"
Java(TM) SE Runtime Environment (build 1.6.0_XX-b03)
Java HotSpot(TM) Client VM (build 11.0-b16, mixed mode,
sharing)

C:\>
```

만약 이 명령어를 수행했을 때 다운로드 및 설치한 버전과 동일하지 않다면, 환경변수에 JDK를 설치한 bin 디렉토리를 추가해야만 한다(환경변수를 추가하는 방법은 익히 알고 있으리라 생각하고 넘어가겠다. 혹시 모른다면 필자에게 이메일을 주면 답변해 주겠다^^).

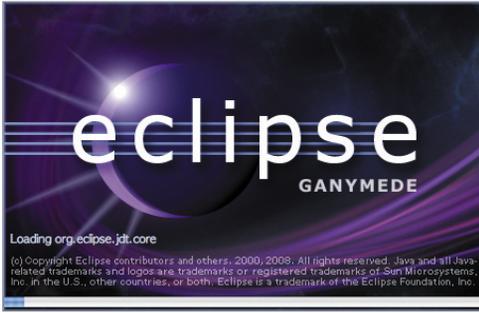
이클립스 다운로드 및 설치

자바가 정상적으로 설치되었다면, 이클립스를 다운로드하자. 이클립스는 <http://www.eclipse.org/downloads/>에서 다운로드하면 된다. 이후 선택의 기로에 놓이는데, 너무 걱정하지 말고 'Eclipse IDE for Java Developers'를 선택하면 된다. 이클립스 버전이 바뀌면 해당 이클립스 패키지의 이름이 변경될 수 있으니, 그때그때 적절한 이름의 이클립스를 선택하면 된다.



이클립스 다운로드 페이지

이클립스 다운로드가 완료되면 압축을 해제한다(현재까지 나온 이클립스는 별도의 설치 과정이 따로 없다. 즉 압축 해제만 하면 된다). 압축 해제된 디렉토리에 있는 eclipse.exe 파일을 수행하면 다음과 같은 화면이 나타날 것이다.



이클립스 시작 화면

만약 이 화면이 나타나지 않는다면, 이클립스의 메모리 설정이나 자바가 정상적으로 설치되지 않은 것이다. 하지만 이러한 오류는 인터넷에서 오류 시 발생하는 메시지로 검색해 보면 해결책을 쉽게 찾을 수 있다. 시작하면 workspace(작업 경로)의 위치를 지정하는 화면이 나타나는데, 이 작업 경로에 새로 생성하는 프로젝트의 파일이 생성된다. 원하는 위치를 선택한 후 다음으로 넘어가자.

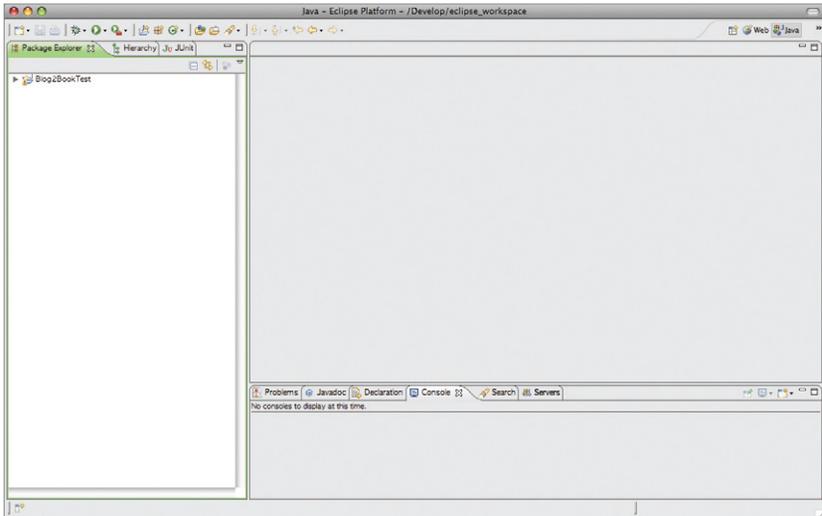
이클립스 기본 화면 설명

이클립스가 시작되면 다음과 같은 화면이 나타난다. 이 화면 중간 중간에 있는 각 그림들을 선택하면, 이클립스를 처음 시작하는 사용자를 위한 설명을 볼 수 있다. 이클립스의 기본적인 것부터 잘 알고 싶다면 이 화면에 있는 각 아이콘을 눌러 설명서를 잘 읽어보기 바란다.



이클립스 시작 화면

이 화면을 닫으면 다음과 같은 화면이 나타난다.

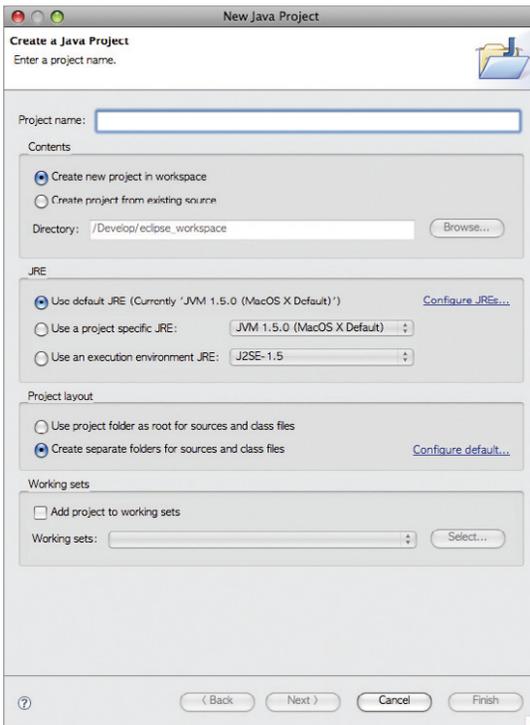


이클립스 메인 화면

메인 화면에 대해서 간단히 설명하자면, 화면의 좌측은 프로젝트 및 클래스를 탐색하고, 기타 여러 툴에서 제공하는 기능을 사용할 수 있는 창이다. 우측 상단에 있는 화면은 소스 및 각종 파일 등을 수정하는 창이다. 화면 하단에 있는 화면은 자바 파일을 수행한 결과를 보여주는 콘솔 창을 비롯하여 등록된 서버 관리, 문제점 등에 대한 정보를 제공하는 창이다. 참고로 각 정보의 위치는 여러분 마음대로 옮길 수 있다.

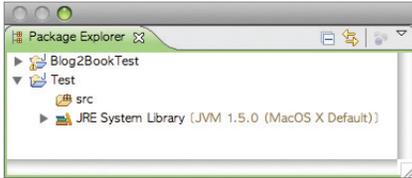
프로젝트 생성 및 라이브러리 설정

새로운 자바 프로젝트를 생성하기 위하여 'File → New → Java Projects' 를 선택한다. 이후 다음과 같은 화면이 나타난다.



새로운 프로젝트 생성 화면

가장 쉽게 프로젝트를 생성하려면, 프로젝트 이름만 지정하고 ‘Finish’ 버튼을 클릭한다. 그러면 해당 화면은 닫히고, 좌측에 있는 탐색 창에 프로젝트가 추가된 것을 확인할 수 있다.



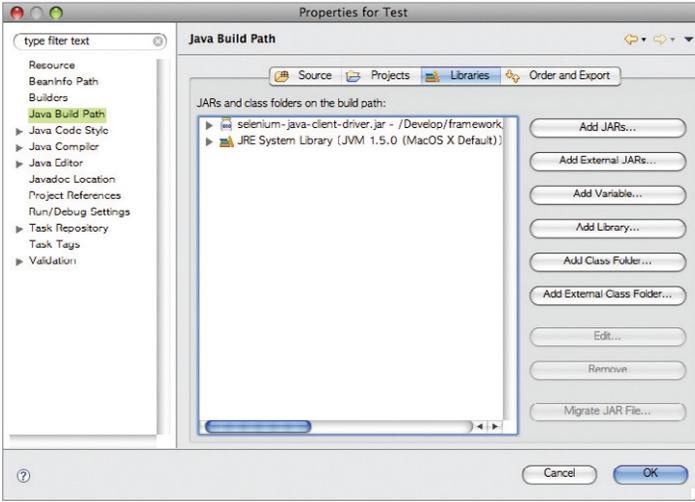
패키지 탐색기

여기서 ‘Test’ 라는 새로운 프로젝트를 만들었다. 그림과 같이 src라는 폴더와 JRE System Library가 자동으로 생성된다. 새로운 자바 클래스를 생성할 경우 src 폴더 하단에 클래스가 표시된다. 기본 설정으로 프로젝트를 생성하면, 실제 OS에 생성한 자바 파일은 src 폴더와 클래스 파일은 bin 폴더에 구분되어 저장된다.

라이브러리(jar) 파일 추가하기

이렇게 만든 프로젝트에 관련된 라이브러리(jar) 파일을 등록하는 절차는 다음과 같다. 프로젝트 이름을 클릭한 상태(여기서는 ‘Test’)에서 마우스 우측을 클릭하고, 메뉴의 가장 하단에 있는 ‘Properties’ 를 클릭한다. 이 속성 화면에서 ‘Java Build Path’ 를 선택하면, 4개의 탭이 존재하는데, 그 중 ‘Libraries’ 를 선택한다.

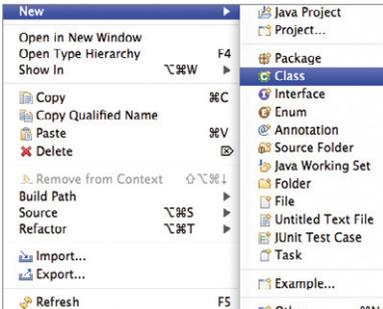
OS에 있는(외부에 있는) 라이브러리를 추가해야 하므로, ‘Add External Jars’ 를 선택하고 원하는 라이브러리 파일을 선택한다. 여기서는 Selenium 관련 라이브러리를 추가했다. 정상적으로 추가되었다면, 다음과 같이 목록에 관련 라이브러리가 있는 것을 확인할 수 있다.



라이브러리 파일 추가

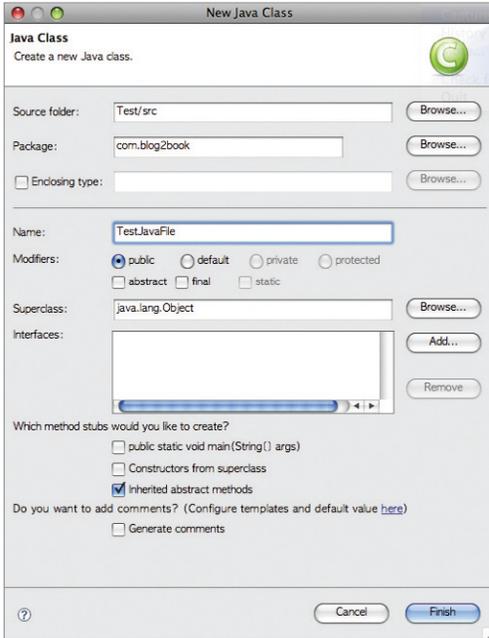
클래스 추가 및 이름 변경하기

새로운 클래스를 추가하려면 아래 그림과 같이 src나 추가로 생성하려는 패키지를 선택한 후 마우스 우측을 클릭하여 'New → Class' 를 선택하면 된다.



클래스 추가

이렇게 메뉴를 선택하면, 자바 클래스 설정 화면이 나타나는데 기본적으로는 'Package' 라고 되어 있는 부분에 패키지 이름과 'Name' 으로 표시되어 있는 부분에 클래스 이름을 지정하면 된다.



클래스 파일 생성

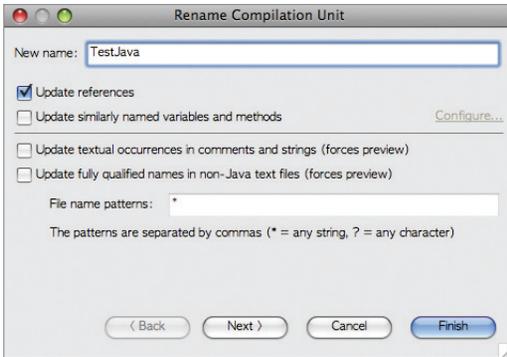
이 그림처럼 클래스 파일을 생성하면, 다음 클래스 소스가 우측 상단의 창에 나타난다.

```
package com.blog2book;

public class TestJavaFile {

}
```

그럼 마지막으로 클래스 파일의 이름을 변경하는 방법을 알아보자. 이클립스에서 소스에 있는 클래스 이름을 바꾼다고, 실제 클래스의 이름이 변경되지 않는다(자바로 개발하지 않고, 이클립스에 익숙하지 않은 개발자들이 어렵게 생각하는 것도 이 부분이다). 클래스의 이름을 변경하려면 좌측의 탐색 창에서 변경하려는 클래스를 선택한 상태에서 'Refactor → Rename'을 클릭한다. 그러면 새로운 클래스 이름을 입력하는 창이 나타나고, 여기서 자바 클래스 명명 규칙에 맞는 클래스 이름을 지정하면 된다.



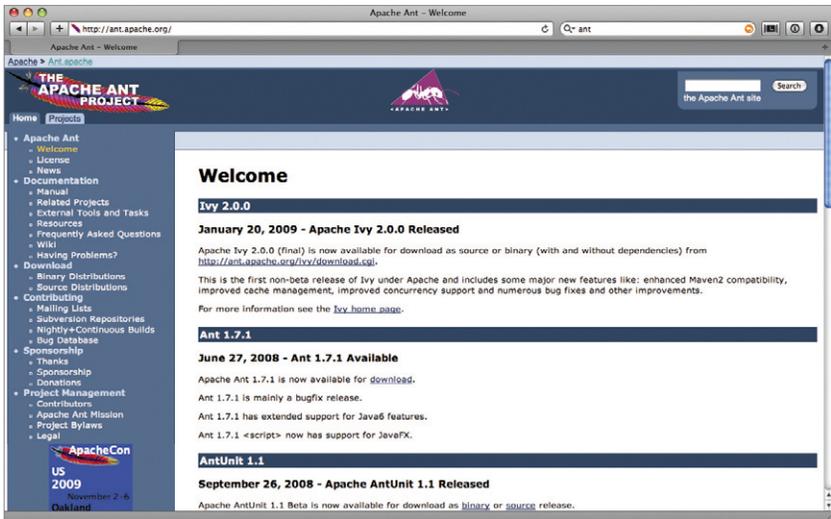
클래스 이름 변경 화면

이클립스에 대한 사용법은 여기까지 알아보겠다. 추가로 이클립스에 대한 사용법을 알아보려면, 이클립스만 다룬 전문 서적을 참조하기 바란다.



Ant 설치 및 사용법

Ant 다운로드 및 설치



Ant 홈페이지

Ant는 apache 그룹에서 제공하는 오픈 소스이며, 본문에서도 이야기했듯이 Windows의 배치 파일이나 Linux 및 Unix의 셸 스크립트와 얼핏 보면 비슷하지만, 실제로 많은 차이가 있는 툴이다. 어쨌든 확장성도 뛰어나고 잘 작성하면 OS에 종속적이지 않게 할 수 있어서 유용하다. 참고로 이 부록에서 다루는 Ant 설명은 전체 Ant 기능 중 '극히 일부분'이며, 여러분이 테스트를 수행하기 위한 Ant 스크립트를 작성할 수 있을 정도만 정리되어 있다는 것을 미리 밝혀두겠다(Ant도 제대로 다루려면 책 한 권 분량도 모자랄만큼 방대하다).

자바 기반인 Ant가 나오게 된 것은 Tomcat 개발자가 반복적이고, 귀찮은 소스 빌드 작업에 지쳐서 만들었기 때문이라고 한다. Ant 다운로드는 Ant 홈페이지인 <http://ant.apache.org/>에서 하면 된다. Ant는 별도의 설치 과정이 없으며, 다운로드받은 파일의 압축만 해제하면 된다. 이후 쉽게 실행할 수 있도록 Path에 Ant의 bin 디렉토리만 추가하면 모든 설치가 마무리된다(참고로 이 책에서는 Windows 기반에서의 Ant 사용법을 정리해 놓았으니 이 점 양해해 주기 바라며, 혹시 Path를 어떻게 잡는지 모르겠다면 필자에게 이메일을 보내거나 네이버 지식인에 물어보기 바란다).

Path가 정상적으로 잡혔는지 확인하려면, 커맨드 창을 띄워 'ant' 라는 명령어를 수행했을 때, 다음과 같은 메시지가 나타나는지 확인하면 된다.

```
C:\>ant
Buildfile: build.xml does not exist!
Build failed
C:\>
```

 참고로 이클립스를 사용하는 개발자는 Ant가 기본적으로 이클립스에 추가되어 있으므로, 별도의 설치 없이 이용할 수 있다. 하지만 커맨드 창에서 Ant를 수행해 보면 가장 쉽고 확실하게 이해할 수 있기 때문에 이클립스에서 Ant를 직접 실행해보기 바란다(정말 어렵지 않다).

Ant로 작업을 수행할 디렉토리의 구조

이 부록에서 예제로 컴파일하고 실행할 소스와 디렉토리 구조를 살펴보면 아마 C:\AntSample이라는 디렉토리에 다음과 같은 파일이 담겨 있을 것이다.

| 디렉토리명 | 위치하는 파일 | 비고 |
|------------------------|----------------|------------------------------|
| jars | JUnit.jar | 테스트 파일을 컴파일하기 위한 JUnit 라이브러리 |
| src/com/blog2book | Hello.java | 테스트 대상 프로그램 |
| src/test/com/blog2book | HelloTest.java | 테스트 프로그램 |

이 파일들을 대상으로 수행할 작업은 다음과 같다.

- ① 빌드 디렉토리 초기화(이전에 만들어진 빌드 디렉토리 삭제)
- ② 빌드 디렉토리 생성(결과 디렉토리 등 필요한 디렉토리 생성)
- ③ 소스코드 컴파일
- ④ 컴파일한 파일을 대상 디렉토리로 복사
- ⑤ 복사된 파일 수행
- ⑥ JUnit 테스트 수행 및 리포트 작성

그렇다면 테스트할 대상 프로그램과 테스트 프로그램은 어떻게 되어있는지 Hello.java부터 확인해보자.

```
package com.blog2book;
public class Hello {
    public static void main(String args[]) {
        Hello hello=new Hello();
        int[] data=new int[5];
        data[0]=1; data[1]=2; data[2]=3; data[3]=4; data[4]=5;
        int result=hello.getSum(data);
        System.out.println(result);
    }
    public int getSum(int[] data) {
        int returnValue=0;
        for(int temp:data) {
            returnValue+=temp;
        }
        return returnValue;
    }
}
```

이 예는 배열로 넘어온 값을 더해서 결과를 리턴하는 간단한 예이다. 이 코드를 테스트하기 위한 HelloTest.java 파일은 다음과 같이 되어 있다.

```

package com.blog2book;
import junit.framework.TestCase;

public class HelloTest extends TestCase {
    int[] data;
    Hello hello;
    public void setUp() {
        data=new int[5];
        data[0]=2; data[1]=4; data[2]=6; data[3]=8; data[4]=10;
        hello=new Hello();
    }
    public void testGetSum() {
        assertEquals(30,hello.getSum(data));
    }
}

```

그럼 본격적으로 빌드 파일을 구성해 보자.

Ant의 기본적인 사용 방법

기본적인 경로가 설정되어 있는 상황에서 Ant를 사용하기 위해서 가장 먼저 해야 할 것은 build.xml(이하 빌드 파일)이라는 xml 파일을 만드는 것이다(물론 파일 이름이 꼭 build.xml일 필요는 없다)

빌드 파일은 기본적으로 다음과 같이 구성되어 있다.

```

<project name="Blog2Book Test-Ant sample" default="test"
basedir=". ">
    <target name="target1" >
    </target>
    <target name="target2" >
    </target>
    <target name="target3" >
    </target>
</project>

```

즉 가장 상위 태그는 project 태그이고, 그 하위 태그는 기본적으로 target의 집합으로 되어 있다. 이 같이 target 내부에는 여러 종류의 태그가 존재할 수 있다. 그리고 project 태그의 하위 태그는 target만 가능한 것은 아니라 property나 path 등의 태그도 사용 가능하다.



Ant의 각 작업의 attribute를 상세하게 알고 싶다면, [Ant홈/docs/manual/index.html](http://ant.apache.org/docs/manual/index.html)에 있는 문서를 참조하면 된다.

그러면 간단한 Ant 예를 살펴보자.

```
<project name="Blog2Book Test-Ant sample" default="makedir"
basedir=".">
  <property file="build.properties"/>
  <target name="clean" >
    <delete dir="${build.dir}"/>
    <delete dir="${server.dir}"/>
  </target>

  <target name="makedir" depends="clean">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${server.dir}"/>
    <mkdir dir="${result.dir}"/>
    <mkdir dir="${junit.result.dir}" />
  </target>
</project>
```

위 코드는 특정 디렉토리를 삭제하고(clean), 디렉토리를 생성하는(makedir) 작업을 수행하는 예인데, 가장 먼저 project 태그를 보면 빌드 파일의 프로젝트의 이름과 default 값, basedir 값을 지정하도록 되어 있다는 것을 알 수 있다. default 값은 빌드를 아무런 옵션 없이 할 때 수행되는 작업을 지정하도록 되어 있다. 여기서는 default 값이 makedir로 되어 있기 때문에 makedir라는 이름의 작업을 수행한다.

그렇다면 clean이라는 작업은 수행하지 않을까? makedir인 작업의 태그에 보면,

depends라는 값이 있다. depends는 해당 작업이 수행되기 전에 먼저 수행되어야 하는 작업을 선언하도록 되어 있다. 그래서 이 예에서는 mkdir만 수행하라고 지정해 놓아도, 해당 작업을 수행하기 전에 알아서 clean 작업을 수행한다.

그럼 두 번째 줄을 보자. 두 번째 줄에는 property라는 태그와 함께 build.properties라는 파일을 지정해 놓았다. build.properties 파일 내부는 다음과 같다.

```
source.dir=./src
build.dir=./build
server.dir=./server/webapps
result.dir=./result
results.file=result.txt
tomcat.url=localhost
tomcat.port=8080
junit.result.dir=./result.junit
```

build.properties 파일은 key=value 형식을 사용하는 일반적인 .properties 파일과 동일한 형식으로 구성되어 있다. 즉 좌측 값은 key, 우측 값은 value인 형식의 파일이며, 이 파일에서 설정해 놓은 source.dir나 build.dir 같은 key 값을 빌드 파일에서 사용하기 위하여 \${키}의 형식을 사용하면 된다. 만약 \${a} 같이 사용했는데, a라는 값이 어떤 설정에도 없을 경우에는 Ant에서 에러를 발생하지 않고, 해당 부분을 그대로 \${a}로 지정한다.

clean 작업에 있는 delete 태그는 파일이나 디렉토리를 지울 때 사용하고, mkdir 작업에 있는 mkdir 태그는 디렉토리를 생성할 때 사용한다. 그럼 이 빌드 파일을 수행해보자.

```
C:\AntSample>ant
Buildfile: build.xml

clean:
  [delete] Deleting directory C:\AntSample\build
  [delete] Deleting directory C:\AntSample\server\webapps
```

```

mkdir:
    [mkdir] Created dir: C:\AntSample\build
    [mkdir] Created dir: C:\AntSample\server\webapps

BUILD SUCCESSFUL
Total time: 0 seconds
C:\AntSample>

```

다행스럽게도 Build Successful이라는 메시지가 나타나고, 디렉토리도 정상적으로 생성한 것으로 보인다. 해당 디렉토리로 이동하여 디렉토리들이 정말 생성되었는지 확인도 해보기 바란다. 만약 clean만 수행하고자 할 경우에는 명령어 창에서 'ant clean' 으로 설정하면 된다.

Ant를 이용한 컴파일과 실행

디렉토리를 생성하는 것도 완료되었으니 이번엔 컴파일을 하고, 대상 디렉토리로 복사하는 작업을 해보자. 여기서 대상 디렉토리라는 것은 Web 기반의 개발을 할 경우 WEB-INF/classes로 컴파일된 파일들이 옮겨져 수행될 위치를 말한다. 이 작업이 귀찮다면 대상 디렉토리에서 바로 컴파일할 수도 있지만, 기존 파일과 중복되어 컴파일 오류가 발생할 수도 있으니 별도의 build 디렉토리를 만들어 컴파일할 것을 권장한다.

```

<project ...>
  <path id="build.classpath">
    <fileset dir="./jars">
      <include name="*.jar"/>
    </fileset>
    <pathelement location="\${server.dir}"/>
  </path>
  <!?Clean, mkdir tasks -->

  <target name="compile" depends="mkdir">

```

```

    <javac srcdir="\${source.dir}" destdir="\${build.dir}">
      <classpath refid="build.classpath"/>
    </javac>
    <echo message="Compile Succeed"/>
  </target>

  <target name="build" depends="compile">
    <copy todir="\${server.dir}">
      <fileset dir="\${build.dir}">
        <include name="**/*.class"/>
      </fileset>
    </copy>
  </target>
</project>

```

일반적으로 컴파일을 수행할 때에는 classpath를 지정해야 한다. classpath를 지정하기 위해서는 위와 같이 path 태그를 사용하여 미리 선언한 후 재사용하는 것이 좋다. 왜냐하면 컴파일, 수행, 테스트를 할 때 모두 동일한 classpath를 참조해야 하기 때문이다. fileset이라는 태그는 관련 파일들의 목록을 지정할 때 사용되는데 이 태그 내에서 include 태그를 사용하여 원하는 파일 목록을 추가할 수도 있고, exclude 태그를 사용하여 원하지 않는 파일 목록을 제외할 수도 있다.

컴파일을 수행하는 작업을 살펴보면 javac라는 태그가 있는 것을 확인할 수 있다. javac 태그에는 기본적으로 srcdir로 소스 파일이 있는 디렉토리를 지정하고, destdir로 컴파일한 파일이 저장될 디렉토리를 지정한다. javac 태그 안에 원하는 여러 옵션을 추가할 수 있으며, 이 예에서는 classpath 태그만 추가하였다. 그 다음엔 컴파일한 파일이 있는 위치에서 대상 디렉토리로 복사하는 작업이 선언된다.

그럼 이 결과를 실행하기 위한 코드를 어떻게 작성하면 되는지 알아보자.

```

<project ...>
  <!?path, Clean, mkdir, compile, build tasks -->

  <target name="run" depends="build">

```

```

<property name="results.file" value="results.txt"/>
<tstamp/>
<property name="results.file.name" value="\${DSTAMP}-
\${TSTAMP}-\${results.file}"/>
<java classname="com.blog2book.Hello">
  <classpath path="\${server.dir}"/>
  <redirector output="\${result.dir}/\${results.file.name}"/>
</java>
<echo message="Result is save at \${result.dir}/\${DSTAMP}-
\${TSTAMP}-\${results.file}"/>
</target>

</project>

```

Ant에서 자바를 실행할 때에는 java 태그를 사용하여 수행하면 된다. 하지만 이렇게 사용할 경우 결과로 출력되는 내용이 저장되지 않고, 화면에만 출력되기 때문에 redirector 태그를 사용하여 파일에 결과를 저장해야만 한다. 추가로 동일한 이름으로 지정하여 저장할 경우 기존 결과에 덮어 쓰기 때문에, tstamp라는 태그를 사용하여 수행한 시간을 결과 파일 이름에 추가하여 위와 같이 사용하면 좋다. 여기서는 \\${DSTAMP}와 \\${TSTAMP}를 사용했는데 만약 tstamp를 그 전에 선언하지 않으면, 해당 변수 값은 사용할 수 없다. 그리고 출력되는 시간의 형식을 원하는 대로 변경하고 싶다면, tstamp 태그 내에 포맷을 정의해 주면 된다.

Ant를 이용한 JUnit 테스트 수행

```

<project ...>
<!?path, Clean, mkdir, compile, build tasks -->
  <target name="test" depends="build" description="Test the
  application">
    <echo message="Testing the application"/>
    <junit>
      <classpath refid="build.classpath"/>

```

```

    <formatter type="xml"/>
    <test todir="${result.dir}" name="com.blog2book.
      HelloTest"/>
  </junit>
</target>
</project>

```

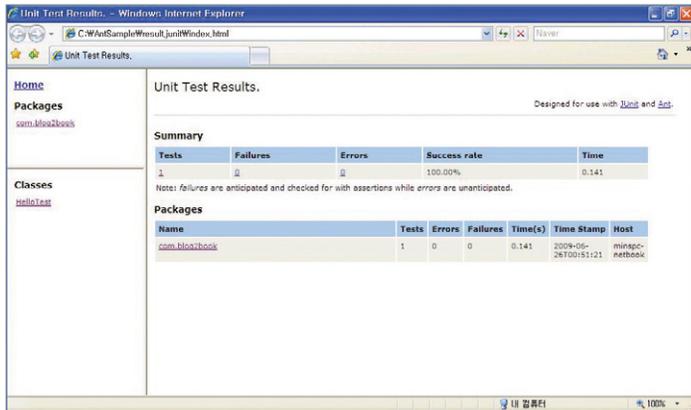
JUnit으로 작성한 테스트 스크립트를 Ant에서 수행하려면, junit 태그를 사용하면 된다. 그리고 결과 포맷을 xml로 지정하여 결과를 저장하도록 했다. 테스트를 수행할 때 todir를 지정하지 않으면, build.xml이 있는 위치에 결과 xml 파일이 저장되기 때문에 디렉토리가 지저분해질 수 있다. 그래서 todir를 지정하여 결과가 저장되는 디렉토리를 지정하는 것이 좋다. 하지만 이렇게 사용하면 일반적인 개발자들이 결과를 보기 힘들기 때문에 junitreport라는 태그를 사용하여 결과를 확인하면 효율적이다. junitreport 태그를 추가한 test 작업은 다음과 같다.

```

...
<target name="test" depends="run" description="Test the
  application">
  <echo message="Testing the application"/>
  <junit>
    <classpath refid="build.classpath"/>
    <formatter type="xml"/>
    <test todir="${result.dir}" name="com.blog2book.
      HelloTest"/>
  </junit>
  <junitreport todir="${junit.result.dir}">
    <fileset dir="${result.dir}">
      <include name="TEST-*.xml"/>
    </fileset>
    <report format="frames" todir="${junit.result.dir}"/>
  </junitreport>
</target>
...

```

이렇게 설정하고 Ant를 다시 수행하면, result.junit이라는 디렉토리에 여러 개의 HTML 파일이 생성된 것을 확인할 수 있다. 여기서 index.html 파일을 수행하면, 다음과 같은 화면이 나타난다.



Ant Report

지금까지 간단하게나마 Ant 사용법을 모르는 독자들을 위해서 Ant 사용법을 설명했다. 앞서 말했듯이 여기서 설명한 것은 아주 기본적인 내용이다. Ant에서는 이 외에도 더 많은 옵션과 작업을 제공하기 때문에 관심이 있는 독자들은 관련 API를 더 학습하여 필요에 따라 옵션을 확장하여 사용하기 바란다.

마지막으로 Tomcat 서버를 Ant에서 수행시킬 수 있는데, 다음의 스크립트를 보고 여러분이 직접 이해해 보기 바란다.

```
<project name="Blog2Book Test-Ant sample" default="start-
tomcat" basedir=". ">
  <property file="build.properties"/>
  <path id="build.classpath">
    <fileset dir="./jars">
      <include name="*.jar"/>
    </fileset>
  <pathelement location="\${server.dir}"/>
```

```
</path>

<target name="check-tomcat-port">
  <condition property="tomcat.running">
    <socket server="${tomcat.url}" port="${tomcat.port}"/>
  </condition>
  <echo message="Check Tomcat is running-${tomcat.running}"/>
</target>

<target name="start-tomcat" depends="check-tomcat-port"
unless="tomcat.running">
  <echo message="Start Tomcat"/>
  <property name="CATALINA_HOME" location="C:/tomcat5.5" />
  <exec executable="C:/tomcat5.5/bin/startup.bat" spawn=
"true" vmlauncher="false" >
    <env key="CATALINA_HOME" value="C:/tomcat5.5"/>
  </exec>
  <sleep seconds="5"/>
</target>

</project>
```



Subversion 설치 및 사용법

Hudson을 사용할 때에는 WAS와 Subversion이 설치된 환경에서 사용하는 것이 좋다. 대부분의 개발자는 손쉽게 Subversion 설치하고 설정할 수 있겠지만, 아쉽게도 그렇지 못한 독자들이 있을 것이라 판단하여 Subversion 설치 및 사용법을 간단하게 정리해 놓았다. Hudson을 사용하기 위한 Tomcat 설치 방법은 부록 '웹 테스트를 위한 간단한 페이지 준비하기'를 참조하기 바란다.

Subversion의 설치 및 사용

Hudson을 사용할 때 반드시 필요한 형상관리 서버 중 요즘 가장 많이 사용되는 Subversion(서브버전)의 설치법을 알아보자. 이 책에서는 자세한 설명은 제외하고, Hudson을 구동할 수 있는 기본적인 환경 설정 부분만 설명할 것이다.

Subversion의 홈페이지는 <http://subversion.tigris.org/>이며, 여기서 Windows용 Subversion을 선택하여 설치하자(Subversion도 Tomcat과 마찬가지로 Windows 기준으로 설명할 것이고, 설치 시에는 항상 그렇듯이 긍정적인 버튼을 선택을 하면 된다)

설치가 완료되었으면, 이제 저장소를 생성하자. 커맨드 창을 띄워서 C:\로 위치한 후 다음과 같은 명령어를 입력한다.

```
C:\>mkdir blog2book
C:\>svnadmin create c:\blog2book\test
C:\>
```

일단 아무런 메시지도 나오지 않는다. 이와 같이 저장소를 만들 때는 상위 디렉토리를 만들어 놓아야만 한다. 즉 지금과 같은 경우는 C:\blog2book\test에 생성을 했기 때문에 C:\blog2book이라는 디렉토리가 존재해야만 저장소 생성이 가능하다는 말이다(생성된 저장소에 어떤 파일이 있는지 한 번 확인해 보는 것도 좋다).

앞의 Ant 부록에서 사용한 소스를 저장소에 추가하기 위하여, svn의 import라는 명령을 다음과 같이 사용하면 된다. 참고로 여기서는 file://로 저장소에 접근했지만, 소스를 저장한 이후에 서버로 접근하는 방법을 확인해보자.

```
C:\>svn import antsample file:///blog2book/test/antsample -m
"initial import"
추가          AntSample\jars
추가 (bin)    AntSample\jars\junit.jar
// 중간 생략
추가          AntSample\build.properties
추가          AntSample\build.xml

커밋된 리버전 1.

C:\>
```

한편 저장소를 생성했다고, 모든 과정이 끝나는 것이 아니다. 반드시 Subversion 서버를 띄워야 한다. 그냥 로컬 서버에서 개인 용도로만 사용한다면 소스 추가할 때처럼 파일로 직접 접근해도 상관 없지만, 대부분 다른 서버에 있는 Subversion에 접근하는 용도로 사용하기 때문에 서버를 띄우는 방법도 알아야 한다. 커맨드 창에 다음과 같이 입력하면, Subversion 서버의 준비가 완료된다.

```
C:\>svnserve -d -r C:\blog2book\test
```

역시 아무런 메시지도 없이 다음 줄에 커서만 깜빡이고 있을 것이다. Subversion 서버는 원격지에서 사용할 수 없기 때문에 이 창은 띄워놓아야 한다. 혹시 커맨드 창을 띄워 놓는 것이 귀찮다면 Windows 서비스에 등록하는 것을 권장한다.

Subversion에 파일이 제대로 추가되었는지, svn의 list 명령을 사용하여 확인해보자.

```
C:\>svn list svn://localhost/antsample
build.ex.xml
build.properties
build.xml
build_tomcat.xml
jars/
src/

C:\>
```

만약 해당 장비의 IP가 할당되어 있다면, 그 IP로 접근해도 무방하다. 참고로 기본 port는 3690이다. 그럼 저장소에 저장된 것도 확인했으니, 저장된 파일을 받아보자. 형상관리 툴에서 Checkout이라는 것은 ‘내가 소스를 수정하려고 가져간다’ 는 말을 서버에 하고, 소스를 받아오는 작업을 말한다. 파일을 받아오는 작업을 수행할 디렉토리를 C:\blog2book work라고 만들고, 커맨드 창에서 그 디렉토리로 이동하여 아래의 checkout명령어를 수행해보자.

```
C:\blog2bookwork>svn checkout svn://localhost/antsample
A   antsample\jars
A   antsample\jars\junit.jar
// 중간 생략
A   antsample\build.properties
A   antsample\build.xml
체크아웃된 리비전 1.

C:\blog2bookwork>
```

체크아웃을 했다면, 파일을 수정하고 다시 저장소에 넣어 두어야 한다. 만약 build.xml 파일을 수정했을 경우에는 commit 명령어를 수행하여 소스를 저장해 보도록 하자.

```
C:\blog2bookwork\antsample>svn commit build.xml -m "change
build files"
전송중          build.xml
파일 데이터 전송중 .
커밋된 리비전 2.

C:\blog2bookwork\antsample>
```

만약 파일이 수정되지(변경되지) 않았다면, 아무런 메시지도 없이 작업을 하지 않는다. 그리고 commit했을 때 다음과 같은 메시지가 나타날 수 있다.

```
C:\blog2bookwork\antsample>svn commit build.xml -m "change
build files"
svn: 커밋이 실패하였습니다:
svn: 인증 실패
```

이러한 상황이 발생하는 이유는 권한 설정 때문이다. 이럴 때에는 당황하지 말고, 다음과 같은 절차를 따라서 사용자를 추가하면 된다.

1. svn 서비스를 제공하는 c:\blog2book\test\conf 디렉토리에 있는 passwd 파일과 svnserve.conf 파일을 연다.
2. passwd 파일에 있는 [users] 아랫줄에 blog2book=test라고 추가한다.
3. svnserve.conf 파일의 password-db=passwd라고 되어 있는 라인의 주석(#)을 제거한다.
4. svn 데몬을 다시 시작한다.
5. commit 명령을 다음과 같이 수행한다.

```
C:\blog2bookwork\antsample>svn commit build.xml -m "change
build files" --username blog2book --password test
```

여기서 유념해야 할 것은 username과 password 옵션은 -를 두 개 입력해야 한다는 점이다.

지금까지 간단하게 svn에 대해서 알아보았다. 또한 이렇게 입력하는 방법 외에도 svn Windows용 클라이언트는 종류가 굉장히 많기 때문에 원하는 것을 다운로드하여 사용하면 된다. Windows용 클라이언트는 이 같은 명령어를 일일이 입력할 필요가 없기 때문에 더욱 편리할 것이다. 만약 Windows 버전의 svn 서버를 설치했다면, 프로그램 목록에 있는 Subversion 가이드 문서를 통해서 자세한 명령어를 확인할 수 있으니 참조하기 바란다(단 영어다).



Internet Explorer Developer Toolbar 설치 및 사용법

Internet Explorer Developer Toolbar 다운로드 및 설치

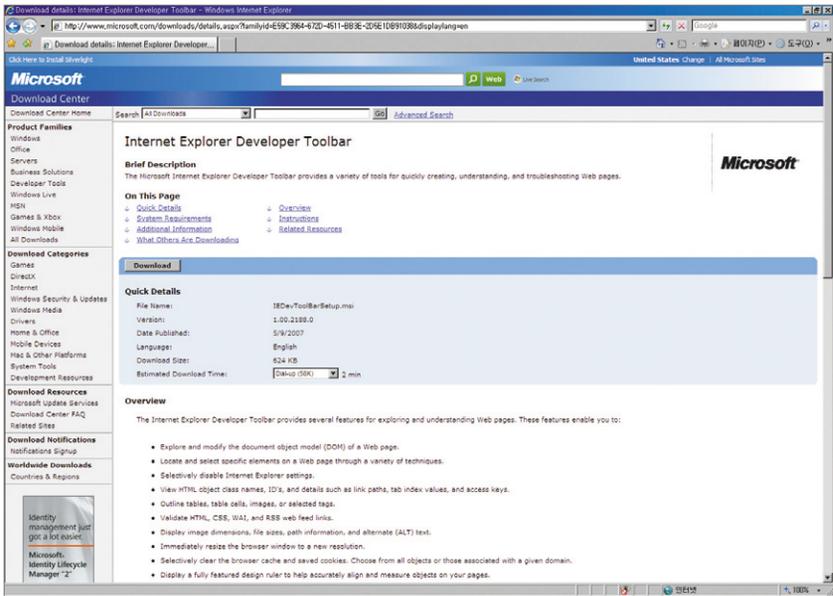
설치 및 사용법에 대해서 알아보기 전에 이 툴의 주요 특징을 알아보자.

- 웹 페이지의 DOM의 내용을 확인하고 수정이 가능하다.
- 웹 페이지를 통해서 특정 HTML 개체의 위치를 지정하고 선택할 수 있다.
- 인터넷 익스플로러의 설정을 변경할 수 있다.
- HTML 요소의 아이디와 이름, 클래스명 등의 속성 값을 볼 수 있다.
- 테이블, 이미지 등 선택한 태그의 외곽선을 표시할 수 있다.
- HTML, CSS, WAI, RSS 등의 오류를 검출할 수 있다.
- 이미지의 크기, 파일 크기, 경로, ALT 문자 등을 화면에 나타낼 수 있다.
- 지정한 해상도로 브라우저의 크기를 즉시 변경할 수 있다.
- 브라우저 캐시나 쿠키를 지울 수 있다. 모든 대상을 지울 수도 있고, 주어진 도메인과 관련된 것만 지울 수도 있다.
- 요소의 크기를 측정하기 위해서 디자인용 자(ruler)를 화면에 표시할 수 있다.
- 특정 스타일 값을 각 요소에 지정하기 위해서 스타일 규칙을 찾을 수 있다.
- HTML과 CSS의 소스를 읽기 쉽게 포맷해준다.

(참고로 위 내용은 마이크로소프트 홈페이지에 있는 설명을 번역해 놓은 것이다.)

이러한 기능들은 여러분의 필요에 따라서 다양한 분야에서 사용할 수 있다. 예를 들면 개발하고 있는 웹 페이지의 디자인상 오류를 찾거나, HTML 태그의 오류를 찾는 것 등이다. 하지만 이 책에서 개발자용 툴바에 대해서 설명하는 이유는 Selenium 같은 툴로 테스트를 할 때 HTML 태그를 훑어보면서 해당 태그의 아이디 및 이름을 찾는 것보다 신속하고 편하게 찾을 수 있고, 다른 곳에도 쓸모가 많기 때문이다.

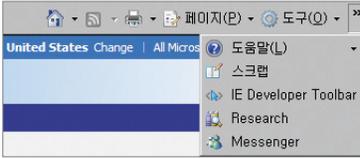
그럼 개발자용 툴바를 설치해보자. Internet Explorer Developer Toolbar(이하 개발자용 툴바)는 마이크로소프트 홈페이지에서 다운로드할 수 있는데 홈페이지 검색에서 'Internet Explorer Developer Toolbar'를 치고 검색하면 쉽게 찾을 수 있을 것이다. 검색 결과 목록에서 관련 다운로드 링크를 선택하면 아래의 화면이 나타난다.



IE 개발자 툴바 다운로드 화면

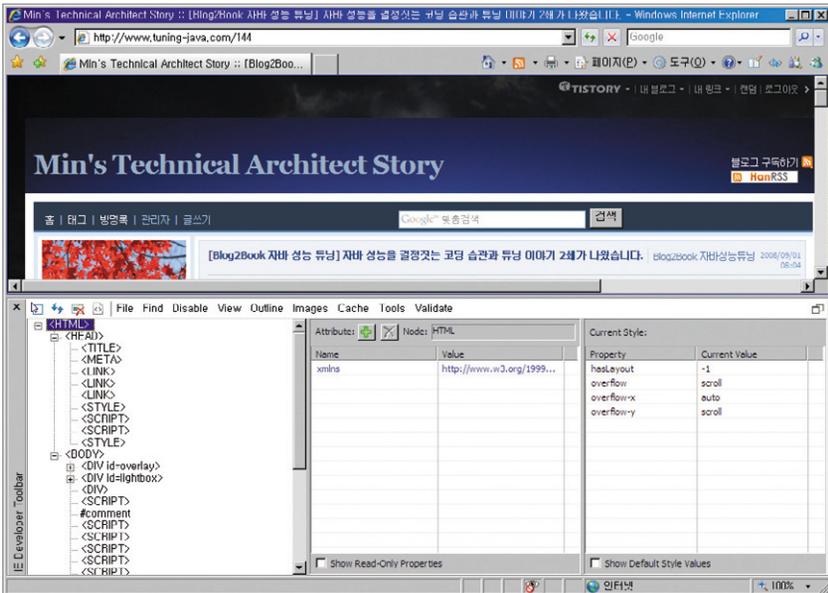
개발자용 툴바 사용하기

다운로드한 개발자용 툴바를 설치한 이후 모든 인터넷 익스플로러를 닫았다가 다시 열면 이 툴바를 사용할 수 있다. 아래 그림과 같이 도구 모음을 확장해보면 중간에 IE Developer Toolbar가 추가되어 있을 것이다.



개발자용 툴바 실행

툴바가 실행되면 아래와 같이 메뉴와 좌측의 태그 탐색기, 중간 및 우측에 각각 정보를 나열하기 위한 화면이 구성되어 있다. 참고로 여기서 샘플로 띄운 페이지는 필자의 블로그 사이트이다.



툴바 첫 화면

가장 먼저 좌측 상단에 있는 메뉴 버튼에 대해서 알아보자.



메뉴 버튼

그림처럼 메뉴 버튼이 존재하는데, 가장 좌측에 있는 것은 HTML 요소를 선택하는 기능을 한다. 이 버튼을 클릭하고 화면에서 마우스를 움직여 보면, 각 요소의 외곽에 틀이 표시된다. 이때 해당 요소를 클릭하면, 중앙에 있는 속성 창에 선택한 요소의 속성 값이 표시된다. 중앙의 속성 창에서 원하는 속성을 추가할 수도 있다.

두 번째 있는 버튼은 새로 고침 버튼인데, 화면을 새로 고치는 것이 아니라 가장 좌측에 있는 태그 트리를 새로 고친다. 세 번째 버튼은 캐시 삭제 버튼이며, 브라우저에서 다운로드받아 놓은 캐시들을 모두 지운다. 가장 우측에 있는 버튼은 HTML 소스를 여는 버튼으로 소스를 아래 그림처럼 보여주는데 메모장에서 소스를 보는 것보다 훨씬 가독성이 좋다.



```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML Strict//EN"><META http-equiv="Content-Type"
2
3 <HTML xmlns="http://www.w3.org/1999/xhtml">
4 <HEAD><STYLE>
5 /* Rule 32 of http://cfs.tistory.com/custom/blog/16/160600/skin/style.css?_T=122904
6 .container {
7     WIDTH: 1240px
8 }
9
10 /* Rule 1 of http://cfs.tistory.com/custom/blog/16/160600/skin/style.css?_T=1229049
11 BODY {
12     PADDING-RIGHT: 20px;
13     BACKGROUND-POSITION: 0% 0%;
14     PADDING-LEFT: 20px;
15     BACKGROUND-IMAGE: url(./images/IMG_1708_2.jpg);
16     PADDING-BOTTOM: 50px;
17     MARGIN: 0px;
18     FONT: small/1.5 verdana;
19     COLOR: #666;
20     PADDING-TOP: 50px;
21     BACKGROUND-REPEAT: repeat;
22     LETTER-SPACING: -1px
23 }
24
25 /* Rule 33 of http://cfs.tistory.com/custom/blog/16/160600/skin/style.css?_T=122904
26 .header {
27     PADDING-RIGHT: 10px;
28     BACKGROUND-POSITION: center top;
29     PADDING-LEFT: 10px;
30     BACKGROUND-IMAGE: url(./images/IMG_1708_2.jpg);
```

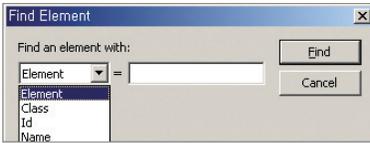
소스보기

그럼 이제 메뉴에 대해서 알아보자.

File

이 툴에는 다른 일반 툴과 달리 File 메뉴에 별다른 기능이 없고, 수정한 모든 내용을 취소하는 기능만 존재한다. 수정한 내용에 대한 저장기능 역시 존재하지 않는다.

Find



검색 창 화면

검색 메뉴에서는 태그 엘리먼트나, 클래스, 아이디, 이름으로 원하는 대상을 찾는 기능을 제공한다. 키워드로 검색해서 찾을 수도 있으며, 대상을 화면에서 클릭해서 찾을 수도 있다.

Disable

말 그대로 비활성화 대상을 지정할 수 있다. 자바 스크립트, CSS, 팝업을 선택적으로 비활성화한다.

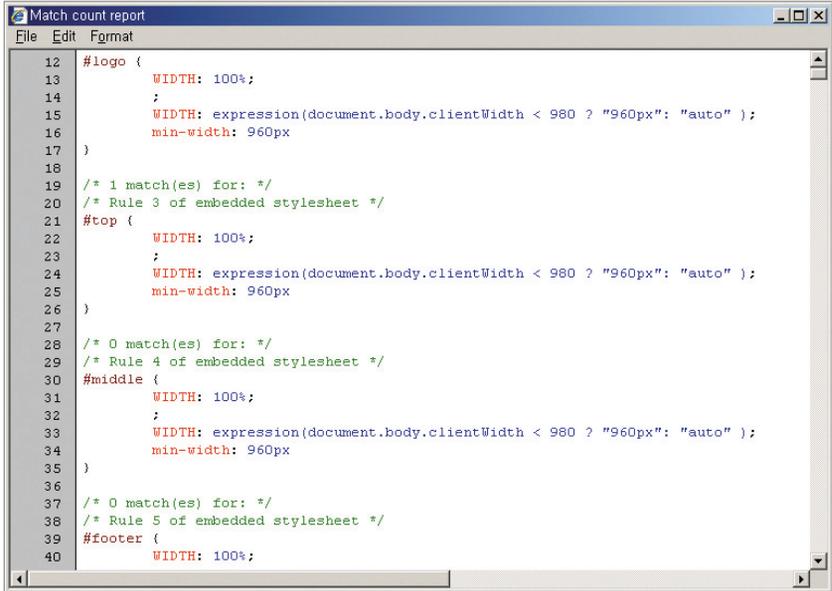
View

몇몇 메뉴 항목을 제외하고, 이 메뉴에서 선택한 대상은 즉시 화면에 표시된다. 이 메뉴에서 선택할 수 있는 대상은 클래스와 ID 정보, 링크 경로, 탭 인덱스, 단축키 등이 있다.

만약 링크 리포트를 선택하면, 현재 분석 중인 화면에 존재하는 링크 목록이 정리되어 출력된다. 만약 보고 있는 화면이 여러 HTML이나 JSP 등으로 구성되어 있다면 각 화면별 링크 정보를 보여주고, 링크가 자바 스크립트 함수로 연결되어 있다면, 해당

함수까지 표시해준다.

CSS Selector Matches를 선택하면, 해당 화면에서 사용되는 CSS에서 해당 CSS 스타일이 몇 번이나 사용되었는지를 정리해서 보여준다. 그런데 이 기능은 PC의 성능에 따라 다르겠지만, 굉장히 오래 소요되는 작업이므로 여러분의 인내심을 시험할 수도 있다.



```
12 #logo {
13     WIDTH: 100%;
14     ;
15     WIDTH: expression(document.body.clientWidth < 980 ? "960px": "auto" );
16     min-width: 960px
17 }
18
19 /* 1 match(es) for: */
20 /* Rule 3 of embedded stylesheet */
21 #top {
22     WIDTH: 100%;
23     ;
24     WIDTH: expression(document.body.clientWidth < 980 ? "960px": "auto" );
25     min-width: 960px
26 }
27
28 /* 0 match(es) for: */
29 /* Rule 4 of embedded stylesheet */
30 #middle {
31     WIDTH: 100%;
32     ;
33     WIDTH: expression(document.body.clientWidth < 980 ? "960px": "auto" );
34     min-width: 960px
35 }
36
37 /* 0 match(es) for: */
38 /* Rule 5 of embedded stylesheet */
39 #footer {
40     WIDTH: 100%;
```

CSS 일치 갯수 리포트

Outline

이 메뉴에 있는 각각의 항목을 클릭하면, 화면에 틀을 표시해준다. 디자이너가 사용하면 굉장히 유용할 것으로 생각된다. 기본적으로 제공되는 대상은 테이블, div, 이미지만인데 필요하다면 해당 태그를 추가하여 표시할 수도 있다.

Image

이미지 메뉴 항목을 선택하면 이미지와 관련된 정보를 표시해준다. 이미지의 경로, 크기, 파일 크기 등을 보여주고, 모든 이미지를 비활성화할 수도 있다. 추가로 링크 리

포트 같이 이미지 관련 리포트도 제공한다.

Cache

캐시를 지우거나 항상 다시 데이터를 받도록 설정할 수 있으며, 쿠키 관련 설정도 가능하다.

Tools

툴 메뉴에는 여러 가지 기능이 있다. 크기 변경 기능은 화면의 크기를 원하는 크기로 보이도록 변경한다. 화면에서 각 요소의 크기를 확인하기 위한 자 기능도 제공하며, 컨트롤 키를 누르고 드래드를 하면 여러 개의 자를 화면에 표시할 수도 있다. 이 기능을 사용시 **Cmd**-**M** 키를 누르면 돋보기가 표시되어 보다 편하게 원하는 위치를 선택할 수 있는 기능을 제공한다. 화면에 있는 특정 픽셀의 색 정보를 보여주는 컬러 픽커도 이 메뉴에 포함되어 있다.

Validate

각 항목(HTML, CSS, Feed, Link)에 대한 검증 리포트를 제공해준다. 여기에서 제공되는 리포트를 실행해보면, 엄청나게 많은 에러 목록을 보여주는데 이 때문에 너무 걱정하지 말기 바란다. 단순하게 태그를 대문자로 써도 에러라고 표시하기 때문에 사이트에 심각한 문제가 있다는 의미는 아니다. 단지 표준을 따르는지 여부를 점검한다고 생각하면 된다.



Firebug 설치 및 사용법

Firebug의 특징

Firebug는 Firefox에서 작동하도록 만들어졌다. 물론 인터넷 익스플로러나 사파리나 같은 브라우저에서도 사용할 수 있는 Firebug Lite라는 버전이 있긴 하지만, Firefox에서 사용하면 기본적인 기능에 추가 기능까지 사용할 수 있으므로 되도록이면 Firefox 버전을 사용하기 바란다.

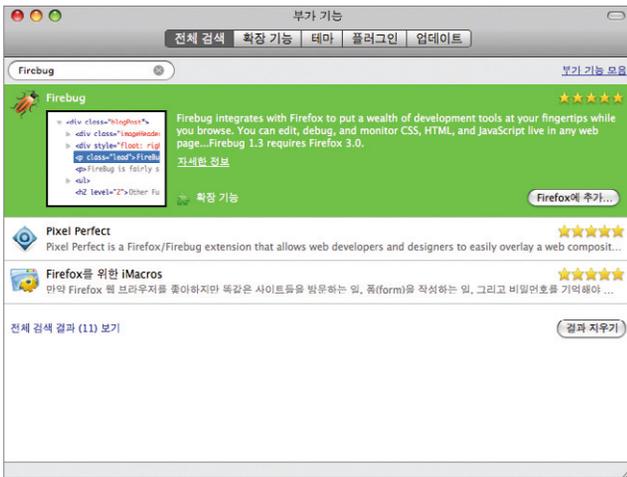
Firebug를 설치하기 전에 간단히 Firebug에서 제공되는 기능을 알아보자.

- 간편한 HTML 편집 : Firebug를 이용하여 간편하게 HTML 소스를 수정하고, 그 수정된 결과를 실시간으로 확인할 수 있다.
- 간편한 CSS 편집 : HTML과 마찬가지로 CSS도 실시간으로 수정하고, 그 결과를 실시간으로 확인할 수 있다.
- 네트워크 모니터링 : 페이지를 구성하는 각 요소가 네트워크로 데이터를 받을 때 얼마나 시간이 소요되는지, 관련된 HTTP 헤더 정보는 어떤지 확인할 수 있다.
- 자바 스크립트 디버그 : 자바 스크립트를 편집하고, 디버그할 수 있는 기능을 제공한다. 기본적인 디버깅 기능만 제공하는 것이 아니라 Stack trace도 볼 수 있고, 성능 측정까지 가능하다.
- 쉬운 에러 찾기 : 인터넷 익스플로러에서 제공하는 간단한 에러 찾기 경고 기능을 뛰어 넘어서 상세하게 에러가 발생한 것을 알려준다.
- DOM 구조의 소스 검색 : DOM(Document Object Model) 구조로 되어 있는 페이지의 구성요소를 확인할 수 있다.

이 외에도 여러 기능이 많이 있으니, 제공되는 기능을 더욱 상세히 알고자 한다면 <http://getfirebug.com/>에서 확인해 보기 바란다.

Firebug 설치

Firebug를 Firefox에 추가하는 방법은 다른 부가기능을 추가하는 것과 동일하다. Firefox를 띄우고, 메뉴에서 도구 → 부가기능을 선택한다. 부가기능의 가장 좌측에 있는 전체 검색을 선택하면 검색 창이 나타나고, 검색 창에서 'Firebug'를 입력하여 검색하면 다음과 같은 화면이 나타난다.



Firebug 검색 결과

Firebug를 선택 후 'Firefox에 추가...'를 클릭하여 설치한다. 설치 완료 후 Firefox를 재기동하면 아무런 변화가 없는 것으로 보이지만, Firefox 도구 메뉴를 선택하면 다음과 같이 Firebug가 추가된 것을 확인할 수 있다.



Firebug가 추가된 도구 메뉴

Firebug 메뉴의 'Firebug 새 창에서 열기' 를 선택하면 브라우저 화면 하단에 다음과 같이 Firebug 창이 나타난다.



Firebug 초기 화면

이처럼 별도 화면에서 띄우는 방법도 있지만, HttpWatch 같이 브라우저 화면 하단에 나타나게 할 수도 있다. 그럴 때에는 'Firebug 열기' 를 선택하면 된다.

Firebug 사용법

Firebug는 매우 많은 기능을 제공하지만, 여기서는 독자들이 쉽게 Firebug를 이

용할 수 있도록 기본적인 사용법 위주로 설명하겠다. 먼저 메뉴를 알아보자. 기본 메뉴는 다음과 같다.

- 콘솔 : 각종 이벤트 및 결과를 보여주는 창이다. 가장 중심이 되는 창이라고 생각하면 된다.
- HTML : 페이지의 구성 요소가 HTML 소스의 어느 부분인지 쉽게 찾고, HTML을 수정할 수 있는 기능을 제공한다. '검사' 버튼을 클릭한 후 브라우저 페이지를 이동해 보면, 해당 페이지의 구성 요소가 소스의 어디에 위치해 있는지를 HTML 창에 나타낸다.
- CSS : 페이지를 구성하는 CSS를 분석하고, 수정하는 기능을 제공한다.
- 스크립트 : 자바 스크립트를 수정하고, 디버깅할 수 있다. 중단점(Break point)을 선택하면, 자바 스크립트 수행 시 해당 부분에서 멈추었다가 다시 실행할 수도 있다.
- DOM : 페이지의 현재 상황을 DOM으로 보여준다. 자바 스크립트 등에서 사용하는 각종 변수의 현재 값을 확인할 수 있다.
- Net : 페이지를 로딩할 때 어떤 요소를 받는데 얼마나 소요되었는지를 보여주고, 각 요소의 크기 및 내용, HTTP 헤더 정보까지 확인할 수 있다.
- Reference : 해당 페이지의 각 태그 및 스타일의 브라우저 호환성을 보여준다.
- Cookies : 쿠키 정보를 확인한다.

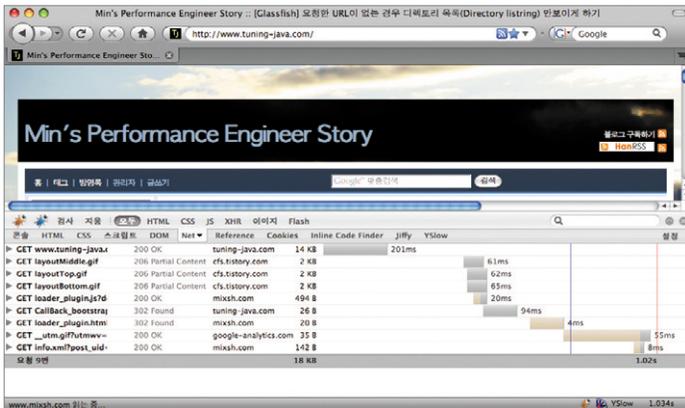
기본적으로 콘솔, 스크립트, Net, Cookies 메뉴를 선택하면 다음과 같은 화면이 나타날 것이다.



콘솔 기본 화면

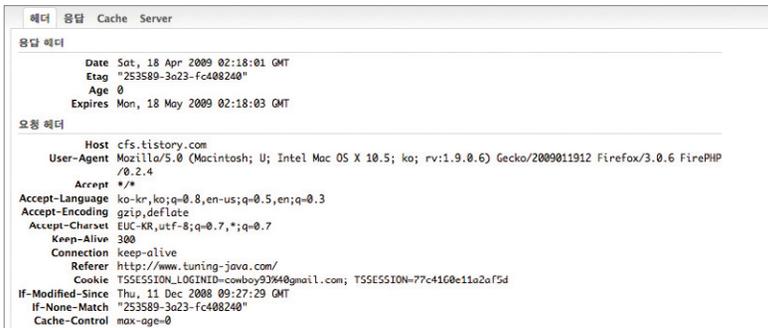
여기서 당황하지 말고(여러분이 툴을 잘못 설치한 것이 아니다), 각 메뉴 이름 옆에 있는 체크박스를 선택한 후 하단에 있는 'Enable selected panels for xxxx'를 누르면 각 메뉴를 선택했을 때 해당 기능이 정상적으로 작동을 할 것이다.

HttpWatch라는 툴을 사용하는 가장 주된 이유도 네트워크 상태를 확인하기 위함이니, Firebug의 Net 기능을 자세히 알아보자. Net 기능을 켜 놓은 상태에서 화면을 네비게이션하면, 다음과 같다.



네트워크 응답속도 측정

물론 여기서 Net 기능은 끝이 아니다. 가장 좌측에 있는 삼각형을 누르면, 해당 요청에 대한 상세정보를 제공한다. 요청 상세정보는 다음과 같은 형식으로 제공한다.



요청 상세정보

그림과 같이 헤더정보, 응답 받은 데이터, 캐시, 서버 정보를 보여준다. 만약 받은 콘텐츠가 이미지라면, 데이터 크기에 마우스를 갖다 대면 이미지의 썸네일을 보여준다. 그리고 우측의 상태 바에 마우스를 갖다 대면 얼마나 대기기를 했는지, 데이터를 받는데 소요된 시간이 얼마나 되는지를 확인할 수 있다.

기능에 대한 설명의 마지막으로 자바 스크립트의 성능 프로파일링 방법을 알아보자. 사용법은 매우 간단하다. 콘솔 메뉴를 클릭하고, 프로파일이라는 버튼을 누른다.



자바 스크립트 프로파일링

프로파일을 누른 후 그냥 다시 프로파일 버튼을 누르면 아무런 결과도 나오지 않는다. 페이지에서 자바 스크립트를 수행해야만 수행된 자바 스크립트에 대한 프로파일 결과가 나오므로, 자바 스크립트가 수행되도록 화면을 동작시킨다. 그러면 다음과 같은 프로파일링 결과를 제공한다.

| 함수 | 호출 횟수 | 퍼센트 | 고유 시간 | 시간 | 평균 | 최소 | 최대 | 파일 |
|---------------------------|-------|--------|----------|----------|---------|---------|---------|------------------------------|
| Mozilla_deRainbowAnchor | 160 | 44.29% | 58.855ms | 58.855ms | 0.368ms | 0.033ms | 1.688ms | rainbow.js (132째 줄) |
| ChangeColor | 47 | 21.96% | 29.179ms | 30.852ms | 0.656ms | 0.42ms | 0.783ms | rainbow.js (172째 줄) |
| toggleFolder | 13 | 13.61% | 18.089ms | 23.485ms | 1.807ms | 1.387ms | 4.358ms | www.tutnl...java.com (92째 줄) |
| Mozilla_stopRainbowAnchor | 160 | 10.01% | 13.299ms | 13.299ms | 0.083ms | 0.001ms | 0.675ms | rainbow.js (156째 줄) |
| tt_hideLayer | 9 | 2.76% | 3.663ms | 3.663ms | 0.407ms | 0.348ms | 0.474ms | common.js?v=2.122 (238째 줄) |
| tt_showLayer | 4 | 1.3% | 1.733ms | 1.733ms | 0.433ms | 0.359ms | 0.479ms | common.js?v=2.122 (234째 줄) |
| makeColor | 47 | 1.26% | 1.673ms | 1.673ms | 0.036ms | 0.02ms | 0.069ms | rainbow.js (182째 줄) |
| onclick | 1 | 0.92% | 1.228ms | 2.963ms | 2.963ms | 2.963ms | 2.963ms | puA3lwCK...1Gq== (1째 줄) |
| onclick | 9 | 0.87% | 1.154ms | 15.648ms | 1.739ms | 1.427ms | 2.599ms | 16laRmVI...F6g== (1째 줄) |
| onclick | 1 | 0.78% | 1.039ms | 2.55ms | 2.55ms | 2.55ms | 2.55ms | KZR9xPCa...8Yq== (1째 줄) |

프로파일링 결과

각각의 자바 스크립트가 어디에 속해있으며, 메소드별 소요 시간, 평균, 최소, 최대 응답 시간 등을 아주 상세하게 보여준다.

Firebug에 추가 가능한 부가 기능

Firebug의 기능이 강력하긴 하지만, 추가 가능한 부가 기능의 종류도 많다. 필자가 이 책을 쓸 당시에 나온 부가 기능 중 개발자들에게 유용한 것은 다음과 같다.

- Firecookie : 확장된 쿠키 정보를 볼 수 있는 기능을 제공한다.
- FirePHP : PHP 디버그를 할 수 있는 기능을 제공한다.
- Inline Code Finder for Firebug : 자바 스크립트로 통제되는 모든 컴포넌트를 페이지에 표시하고, 관련된 통계 정보를 보여준다.
- Jiffy : 페이지를 처리하는 응답속도를 상세하게 구분하여 보여준다. 자세한 내용은 <http://billwscott.com/jiffyext/>를 참조하기 바란다.
- Yslow : 페이지의 성능 및 구성요소를 분석하는데 굉장히 유용한 툴이다. 대부분의 튜닝 작업이 서버의 응답속도를 줄이기 위함이라면, 이 툴은 화면에서 렌더링하는데 소요되는 시간을 측정하고, 줄이는 방법을 가이드하기 위한 툴이다.