Manual de herramienta de validación HTML

Eduardo Fernández Casal
2º ciclo de Ingeniería en Informática

Jordi Ferrer Duran

06/2012

Índice

1. Introducción

El sistema operativo usado para el desarrollo de esta herramienta ha sido Ubuntu 11.10 64-bit sobre un procesador Intel® Core™ i5 CPU M 460 @ 2.53GHz × 4 y 4 GB de RAM.

Además, la herramienta ha sido probada en entornos PC con sistemas operativos Windows 7 Ultimate y Windows XP Professional.

2. Instalación

2.1 JDK

La versión de Java empleada para el desarrollo del validador XHTML5 es 1.6. Para instalarla en un sistema operativo Ubuntu basta con lanzar el gestor de paquetes Synaptic y seleccionar el paquete sun-java6-jdk. El gestor nos informará de todas las dependencias del paquete y procederá con la instalación.

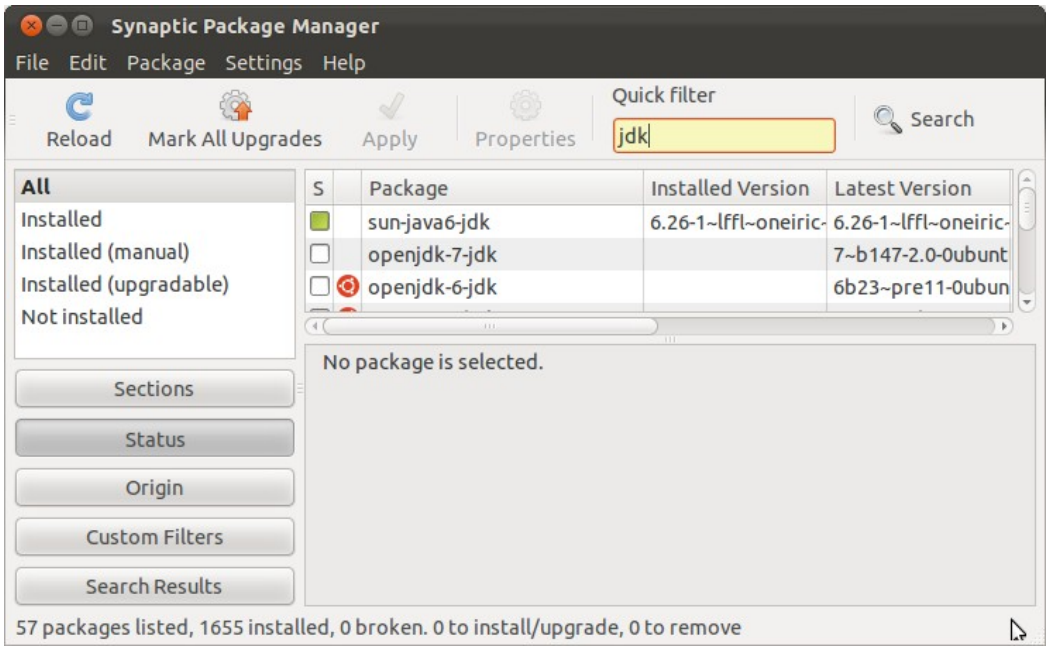

Illustration 1: Gestor de paquetes Synaptic

En caso de trabajar en un entorno Windows, será necesario descargar el JDK desde la página web de Oracle:

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

Una vez instalado el JDK, será necesario crear la variable de entorno JAVA_HOME y asignarle como valor la ruta del directorio donde se ha instalado.

2.2 JLex

Para instalar JLex en un sistema Ubuntu basta con acudir al gestor de paquetes Synaptic y seleccionar el paquete jlex. Como en el caso anterior, este detectará las dependencias, informará al usuario e instalará las librerías.

En caso de trabajar en sistemas Windows, será necesario descargar el archivo de la ubicación

http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java

y almacenarlo en una carpeta, que será añadida posteriormente a la variable de entorno CLASSPATH.

2.3 CUP

El *modus operandi* a seguir con CUP es similar al proceso anterior. En Ubuntu instalaremos el paquete cup con ayuda del gestor de paquetes Synaptic. En Windows almacenaremos el contenido del archivo:

http://www.cs.princeton.edu/~appel/modern/java/CUP/java_cup_v10k.zip

en la misma carpeta que JLex. En caso de escoger otra carpeta, será necesario incluirla también en la variable CLASSPATH.

2.4 JBoss

La instalación de Jboss en Ubuntu entraña un poco más de complejidad. En http://www.surajitray.com/coders-corner/jboss-5-1-on-ubuntu-10-04/ se encuentra una guía detallada de los pasos a seguir.

En entornos Windows descargaremos y descomprimiremos el archivo http://downloads.sourceforge.net/jboss/jboss-5.1.0.GA-jdk6.zip en la ubicación que deseemos. A continuación, será necesario modificar y añadir nuevas variables de entorno:

```
JBOSS_HOME = ruta_elegida
CLASSPATH =.;%JBOSS_HOME%\client\log4j.jar;
                %JBOSS_HOME%\client\jbossall-client.jar;
                %JAVA_HOME%\lib\tools.jar;...
```

2.5 Ant

La instalación de Ant en Ubuntu es tan sencilla como la de JLex y CUP. El nombre del paquete es ant.

En Windows hay que acudir a http://ant.apache.org y descargar la versión que deseemos. En nuestro caso hemos optado por la versión 1.8.1. Descomprimiremos el archivo en la ubicación deseada y añadiremos la variable de entorno ANT_HOME con dicha ruta.

2.6 Eclipse

La instalación de Eclipse en ambos entornos es trivial. En http://www.eclipse.org podremos elegir entre varias versiones.

Llegados a este punto, y conscientes de que vamos a trabajar con Jboss, puede resulta muy útil instalar el plugin de Jboss AS Tools. Seguiremos estas instrucciones:

- Seleccionar la opción *Install New Software* desde la pestaña *New*
- Desde *Available Software* seleccionar *Add a Work with*
- Añadir la ubicación:
  http://download.jboss.org/jbosstools/updates/stable/galileo
- De entre todos los elementos listados seleccionamos *All JBoss Tools*
- Aceptarlas condiciones de uso
- Finalmente reiniciar el Eclipse

3. Despliegue

El despliegue de la aplicación consiste en la generación y copia de un archivo .war en el directorio deploy del servidor de aplicaciones JBoss . Afortunadamente, no hay razón para realizar esta tarea de manera manual. El uso de Ant nos permite automalizarla. En el archivo build.xml se desglosan las instrucciones para llevar a cabo el proceso.

```
<project name="Proyecto de TFC" default="deploy" basedir=".">

<description>Constructor de la aplicaciÃ³n web</description>

<!-- propiedades -->
<property environment="env"/>
<property name="jboss.home" value="/usr/local/jboss"/>
<property name="source" value="./src"/>
<property name="sourceweb" value="."/>
<property name="lib" value="lib"/>
<property name="build" value="${sourceweb}/build"/>
<property name="buildJar" value="${build}/Jar"/>
<property name="buildWar" value="${build}/War"/>
<property name="deploy" value="$
{jboss.home}/server/default/deploy"/>
<property name="cup"     location="cup"     />
<property name="flex"    location="flex"    />



<!-- This is JFlex -->
  <taskdef name="jflex"
           classname="JFlex.anttask.JFlexTask"
           classpath="${lib}/JFlex.jar"
  />

  <!-- We also use CUP-TUM -->
  <taskdef name="cup"
           classname="java_cup.anttask.CUPTask"
           classpath="${lib}/java-cup-11a.jar"
  />


<!-- Copia el war en la carpeta deploy del servidor JBoss -->
<target name="deploy" depends="war">
      <copy file="${build}/validator.war" todir="${deploy}"/>
</target>


<target name="war" depends="jar">
      <copy todir="${buildWar}/css">
            <fileset dir="${sourceweb}/webapp/css"/>
      </copy>
      <copy todir="${buildWar}/images">
            <fileset dir="${sourceweb}/webapp/images"/>
      </copy>
      <copy file="${sourceweb}/webapp/index.jsp" todir="$
{buildWar}"/>
```

```
            <copy file="${sourceweb}/webapp/WEB-INF/web.xml" todir="$
{buildWar}/WEB-INF"/>
            <copy file="${sourceweb}/webapp/WEB-INF/jboss-web.xml"
todir="${buildWar}/WEB-INF"/>
            <copy file="${build}/validator.jar" todir="$
{buildWar}/WEB-INF/lib"/>
            <copy file="${lib}/java-cup-11a-runtime.jar" todir="$
{buildWar}/WEB-INF/lib"/>

            <jar jarfile="${build}/validator.war" basedir="$
{buildWar}" update="yes"></jar>
      </target>

      <target name="jar" depends="compileClass">

            <jar
                  jarfile="${build}/validator.jar"
                  basedir="${buildJar}"
                  update="yes"
                  includes="*validator/*.class">
            </jar>
      </target>

      <!-- It compiles all classes and puts them in the directory
build -->
      <target name="compileClass" depends="cup">
            <javac
                  includeantruntime="false"
                  srcdir="${source}"
                  destdir="${buildJar}"
                  classpath="${jboss.home}/client/jbossall-client.jar;$
{jboss.home}/common/lib/servlet-api.jar;${lib}/java-cup-11a-
runtime.jar;/usr/share/java/servlet-api-2.5.jar"
                  debug="on" debuglevel="lines,vars,source"
                  includes="**/*.java, ****/*.java, ***/*.java"/>
      </target>

        <target name="cup" depends="jlex">
          <cup srcfile="${source}/validator/final.cup"
               destdir="${source}"
               interface="true"
             parser="parser"
          />
        </target>

        <target name="jlex" depends="init">
          <jflex file="${source}/validator/final.jlex" destdir="$
{source}" />
        </target>



      <target name="init" depends="clean"
            description="it makes the boots pertinent: it creates the
structure
            of directories and copies the files ejb-jar.xml and
web.xml of each jar" >
          <!-- It creates time-stamp -->
          <tstamp/>
```

```
          <!-- It creates the structure of directories and the
describer copies of ejb -->
          <mkdir dir="${buildJar}"/>
          <mkdir dir="${buildWar}"/>
        </target>
      <!-- Clean the directory build  -->
      <target name="clean">
            <delete dir="${build}"/>
            <delete dir="${dist}"/>
      </target>

</project>
```

La primera parte del archivo se centra en la definición de variables. Esto permite poder adaptar su funcionamiento bajo otros sistemas operativos (por ejemplo, Windows), sin necesidad de modificar el resto del contenido. Para ejecutar Ant basta con situarse en la raíz del proyecto y ejecutar `sudo ant`. Detectará automáticamente la presencia de build.xml y ejecutará las órdenes allí especificadas. Previamente deberemos haber arrancado Jboss. Lo podemos hacer de manera automática desde Eclipse gracias al plugin instalado, o manualmente ejecutando la orden `run` dentro del subdirectorio bin del directorio donde se encuentre instalado Jboss.

4. Ejecución

Finalmente, una vez que la aplicación ha sido correctamente desplegada, podremos acceder a ella mediante la siguiente URL:

http://localhost:8080/validator/