

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

Control de autenticación y
acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)



NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución



[¡NUEVO!] 2008-03-25



2008-03-17



2008-03-11



2008-03-06

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la sección [comic](#).

Tutorial desarrollado por



Raúl Expósito Díaz

Consultor tecnológico de desarrollo de proyectos

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosAlTrabajo.com es el Web de difusión de conocimiento de Autentia.



Catálogo de cursos

Ingeniero Técnico en Informática de Gestión por la Universidad de Alcalá e Ingeniero en Informática por la Universidad Carlos III de Madrid. [Perfil XING](#)

Puedes encontrarme en [Autentia](#)

Somos expertos en Java/J2EE

Descargar este documento en formato PDF: [crap4jant.pdf](#)

Fecha de creación del tutorial: 2008-03-26

Cómo lanzar Crap4j desde Ant

1. Introducción

En [adictosaltrabajo](#) ya os hemos hablado de [Crap4j](#), una implementación de la métrica CRAP (Change Risk Analysis and Predictions), algo que en castellano podríamos traducir como "*Análisis y Predicciones sobre los Riesgos de realizar Cambios*", os hemos explicado en qué consiste dicha métrica y os hemos explicado cómo instalar y utilizar el [plugin de Crap4j para Eclipse](#). Podeis acceder al tutorial donde ya expliqué todo ello [pulsando aquí](#).

Sin embargo, ¿sólo podemos utilizar [Crap4j](#) y todas sus ventajas si tenemos Eclipse?, ¿no podemos encontrar código "*difícilmente mantenible*" si usamos otro IDE, o incluso si no usamos ninguno?. Por suerte el equipo de [Crap4j](#) pensó en ello y permite la ejecución de su aplicación a través del archiconocido [Ant](#).

En este tutorial explicaré cómo lanzar [Crap4j](#) desde [ant](#) en un proyecto desarrollado con [maven](#), y menciono lo de [maven](#) simplemente por la estructura de directorios que tendrá el proyecto. Si queréis saber más sobre [Crap4j](#) visitad [el tutorial enlazado anteriormente](#) o [el blog de Aberto Savoia](#).

2. Entorno

- [Debian GNU/Linux 4.1 \(Lenny\)](#)
- [JDK 6 Update 1](#)
- [Crap4j 1.1.6](#)
- [Ant 1.7.0](#)

3. Instalación

Lo primero que hacemos es bajar la distribución para Ant de [Crap4j](#) desde esta URL: http://crap4j.org/downloads/ant_download.html. Como podeis ver es un jar y ocupa unos 7 MB.

Una vez descargado lo descomprimos con cualquier aplicación que pueda descomprimir zip: 7-Zip, WinZip, Ark, FileRoller, Winrar, WinAce, etc. y lo dejamos en el directorio donde queramos instalarlo, por ejemplo en `/home/raul/crap4j`

Catálogo de servicios Autentia (PDF 6,2MB)



En formato comic...



☐ Web

☒ www.adictosaltrabajo.com

Buscar

Últimos tutoriales

2008-03-26
[Cómo lanzar Crap4j desde Ant](#)

2008-03-17
[Fuentes de Documentación para OpenCms 7](#)

2008-03-16
[Cómo Exportar email de KMail a otros clientes de correo electrónico](#)

2008-03-10
[Visualización de código HTML como un grafo](#)

2008-03-04
[Introducción a JPivot](#)

2008-03-03
[Tablas dinámicas online](#)

2008-02-29
[Generación automática de gráficas en un web](#)

2008-02-28
[Manual de instalación de OpenCms 7](#)

2008-02-28
[Creación de un proyecto en SourceForge.net](#)

2008-02-22
[Lucene: Analyzers, stemming y búsqueda de documentos similares.](#)

Últimas ofertas de empleo

2008-03-19
[Otras - Construcción - BARCELONA.](#)

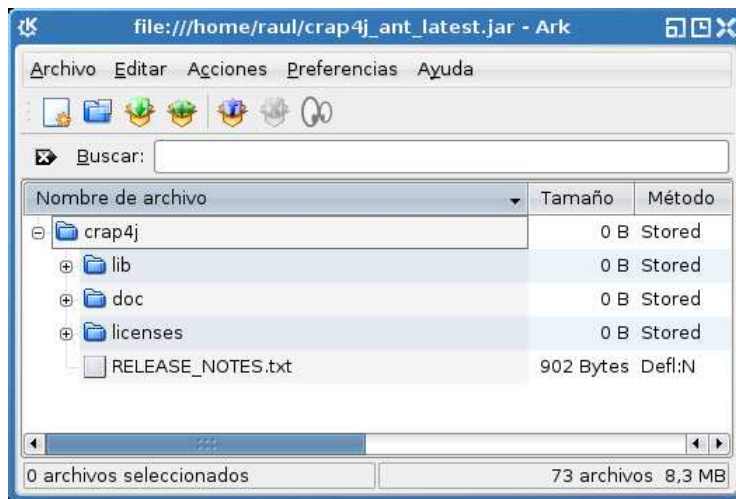


Imagen 1. Contenido del JAR



2. JAR descomprimido en el directorio /home/raul/crap4j

2008-03-18
T. Información -
Administrador Sistemas UNIX
/ NT - BARCELONA.

2008-03-18
Contabilidad - Especialista
Contable - BARCELONA.

2008-03-18
T. Información - Analista /
Programador - BARCELONA.

2008-02-28
Otras - Arquitecto /
Aparejador - MADRID.

Anuncios Google

Con esto ya tenemos descargado e instalado [Crap4j](#), ahora sólo faltaría instalar [Ant](#), cuya instalación presupongo y no cubro en este tutorial.

4. Ejecución

Si mirais dentro del directorio doc/ dentro de la instalación de [Crap4j](#) vereis un fichero llamado `example_build.xml` que servirá de plantilla para que creamos nuestros propios `build.xml` con los que lanzar [Crap4j](#) a través de [Ant](#). Su contenido es el siguiente:

```

view plain print ?
01. <?xml version="1.0" ?>
02. <project name="crap4j" default="run-crap4j" basedir="..">
03.
04.     <!-- #1 redefine crap4j_home to point to the absolute directory where you unzipped crap4j -->
05.     <property name="CRAP4J_HOME" value="/Users/bobevans/Documents/projects/crap4j/org.crap4j/dist/crap4j" />
06.
07.     <!-- Define the Crap4j ant task -->
08.     <taskdef name="crap4j" classname="org.crap4j.anttask.Crap4jAntTask" >
09.         <classpath>
10.             <fileset dir="${CRAP4J_HOME}/lib">
11.                 <include name="**/*.jar" />
12.             </fileset>
13.         </classpath>
14.     </taskdef>
15.
16.     <!-- #2 Setup your project base directory. In most cases, it can just be basedir.
17.         The following is a fancy version -- >
18.     <!-- Use of ${project.dir.crap4j} allows for the resolution of relative paths
19.         even when this file is imported in another build file -- >
20.     <dirname file="${ant.file.crap4j}" property="import.dir.crap4j" />
21.     <condition property="my.project.root.dir" value="${import.dir.crap4j}/.." else="${basedir}">
22.         <isset property="import.dir.crap4j" />
23.     </condition>
24.
25.     <!-- #3 Use crap4j on the project -->
26.     <target name="run-crap4j">
27.         <crap4j projectdir="${my.project.root.dir}"
28.             outputDir="agitar/reports/crap4j"
29.             dontTest="false" debug="false">
30.
31.             <classes>
32.                 <!-- put the project's class directories here. These are the classes you want crap numbe
33.                 <pathElement location="${my.project.root.dir}/bin" />
34.             </classes>
35.
36.             <srces>
37.                 <!-- The project's src folders -->
38.                 <pathElement location="${my.project.root.dir}/src" />
39.             </srces>
40.
41.             <testClasses>
42.                 <!-- The project's test classes.
43.                     Note, separating it from the project classes makes life so much easier
44.                     but if you didn't, then you probably just want to repeat the classDir entry here
45.                 <pathElement location="${my.project.root.dir}/test_bin" />
46.                 <pathElement location="${my.project.root.dir}/integration_tests_bin" />
47.                 <pathElement location="${my.project.root.dir}/agitar/test_bin" />
48.             </testClasses>
49.
50.             <libClasspath>
51.                 <!-- Put the dependent paths and libraries here.
52.                     This is for stuff necessary to your project, but not stuff you want analyzed --
53.                 <fileset dir="${my.project.root.dir}/lib">
54.                     <include name="**/*.jar" />
55.                 </fileset>
56.                 <!-- if you get errors loading any agitar classes, like the mock classes, add this entry
57.                 <!--
58.                 <fileset dir="${CRAP4J_HOME}/lib/com.agitar.eclipse.api_4.2.0.401405/lib/ext" >
59.                     <include name="**/*.jar" />
60.                 </fileset>
61.                 -->
62.             </libClasspath>
63.
64.         </crap4j>
65.     </target>
66. </project>

```

Sin embargo este fichero xml tiene cosas que, a priori, no nos interesan, pero sin embargo es muy util tenerlo como referencia.

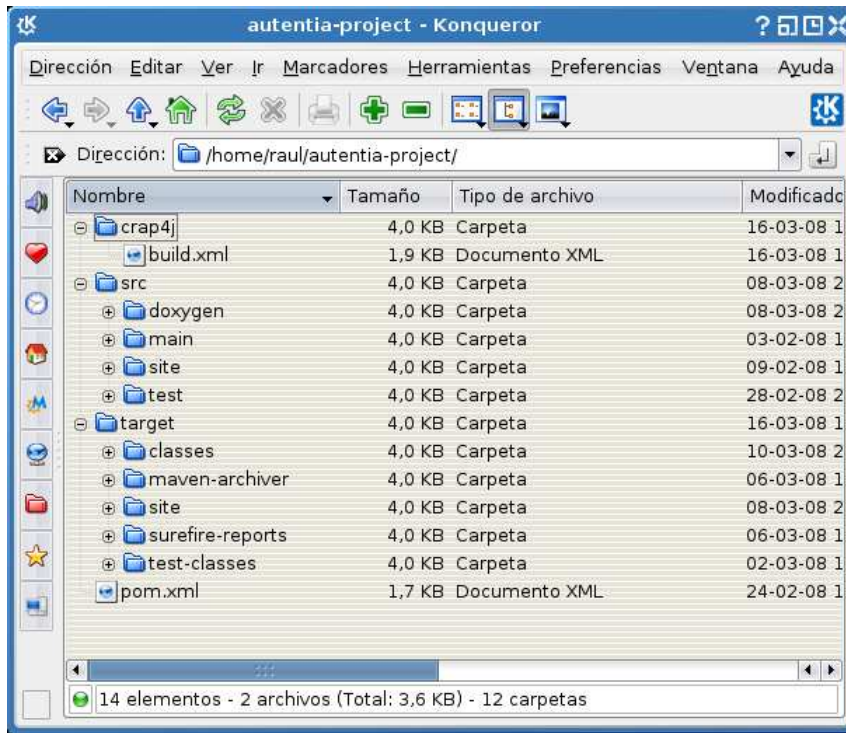
Como ya comenté anteriormente en este tutorial vamos a lanzar Crap4j sobre un proyecto que utilice maven. De nuevo indico que, aunque maven sea totalmente independiente a Crap4j, lo menciono porque la estructura de directorios del proyecto va a ser [la propuesta por maven](#):

src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/site	Site
LICENSE.txt	Project's license
README.txt	Project's readme

Es decir:

- El código fuente del proyecto estará en src/main/java
- Los test en src/test/java.
- Las librerías necesarias para la compilación del proyecto estarán en el repositorio local de maven, en mi caso /home/raul/.m2/repository/

Aparte de estos directorios vamos a crearnos uno propio en el raíz de proyecto y lo llamaremos 'crap4j' y ahí crearemos el fichero build.xml que utilizará ant para lanzar Crap4j, quedando la jerarquía de directorios de la siguiente manera;



Teniendo en cuenta las rutas a los directorios que he mencionado anteriormente, el fichero build.xml quedará de esta forma:

```

view plain print ?
01. <?xml version="1.0" ?>
02. <project name="crap4j" default="run-crap4j" basedir="..">
03.
04.     <!-- #1 directorio de instalacion de crap4j -->
05.     <property name="CRAP4J_HOME" value="/home/raul/crap4j"/>
06.
07.     <!-- definicion de la tarea ant -->
08.     <taskdef name="crap4j" classname="org.crap4j.anttask.Crap4jAntTask" >
09.         <classpath>
10.             <fileset dir="${CRAP4J_HOME}/lib">
11.                 <include name="**/*.jar" />
12.             </fileset>
13.         </classpath>
14.     </taskdef>
15.
16.     <!-- #2 Configuracion de crap4j para nuestro proyecto -->
17.     <target name="run-crap4j">
18.         <crap4j projectdir="${basedir}"
19.             outputDir="agitar/reports/crap4j"
20.             dontTest="false" debug="false">
21.
22.             <!-- directorios donde encontrar las clases compiladas -->
23.             <classes>
24.                 <pathElement location="${basedir}/target/classes" />
25.             </classes>
26.
27.             <!-- Directorios donde encontrar el codigo fuente del proyecto -->
28.             <srces>
29.                 <pathElement location="${basedir}/src/main/java" />
30.             </srces>
31.
32.             <!-- Directorios donde encontrar las clases de test compiladas -->
33.             <testClasses>
34.                 <pathElement location="${basedir}/target/test-classes" />
35.             </testClasses>
36.
37.             <!-- Directorios donde encontrar las librerias necesitadas por nuestro proyecto -->
38.             <libClasspath>
39.                 <fileset dir="/home/raul/.m2/repository">
40.                     <include name="**/*.jar" />
41.                 </fileset>
42.             </libClasspath>
43.
44.         </crap4j>
45.     </target>
46. </project>
47.

```

Donde:

Aquí se debe indicar la ruta donde hayais descomprimido Crap4j

```

view plain print ?
01. <!-- #1 directorio de instalacion de crap4j -->
02. <property name="CRAP4J_HOME" value="/home/raul/crap4j"/>
03.

```

No lo tocamos, es la definicion de la tarea ant

```
view plain print ?
01. <!-- definicion de la tarea ant -->
02. <taskdef name="crap4j" classname="org.crap4j.anttask.Crap4jAntTask" >
03.     <classpath>
04.         <fileset dir="${CRAP4J_HOME}/lib">
05.             <include name="**/*.jar" />
06.         </fileset>
07.     </classpath>
08. </taskdef>
```

Se configura la ejecución de Crap4j:

1. Se indica cuál es el directorio base (que está configurado como {basedir})
2. Dónde se van a dejar los resultados de la ejecución de Crap4j ("agitar/reports/crap4j")
3. Si se deben omitir los test (por defecto a false)
4. Si se deben mostrar las trazas de debug al ejecutar Crap4j (por defecto a false)

```
view plain print ?
01. <!-- #2 Configuracionde crap4j para nuestro proyecto -->
02. <target name="run-crap4j">
03.     <crap4j projectdir="${basedir}"
04.         outputDir="agitar/reports/crap4j"
05.         dontTest="false" debug="false">
06.     ...
07.     </crap4j>
08. </target>
```

Se indica cuál es el directorio o directorios donde encontrar las clases compiladas (sigue la jerarquia de maven)

```
view plain print ?
01. <!-- directorios donde encontrar las clases compiladas -->
02. <classes>
03.     <pathElement location="${basedir}/target/classes" />
04. </classes>
```

Se indica cuál es el directorio o directorios donde encontrar el código fuente del proyecto (sigue la jerarquia de maven)

```
view plain print ?
01. <!-- Directorios donde encontrar el código fuente del proyecto -->
02. <srces>
03.     <pathElement location="${basedir}/src/main/java" /><strong></strong>
04. </srces>
```

Se indica cuál es el directorio o directorios los test a ejecutar sobre el código compilados (sigue la jerarquia de maven)

```
view plain print ?
01. <strong></strong>
02. <!-- Directorios donde encontrar las clases de test compiladas -->
03. <testClasses>
04.     <pathElement location="${basedir}/target/test-classes" />
05. </testClasses>
```

Se indica cuál es el directorio o directorios donde encontrar las librerías utilizadas por nuestro proyecto (repositorio de maven)

```
view plain print ?
01. <!-- Directorios donde encontrar las librerías necesitadas por nuestro proyecto -->
02. <libClasspath>
03.     <fileset dir="/home/raul/.m2/repository">
04.         <include name="**/*.jar" />
05.     </fileset>
06. </libClasspath>
```

Esta es una configuración de ejemplo que debereis adaptar si vuestro proyecto sigue otra jerarquía de directorios.

4. Ejecucion

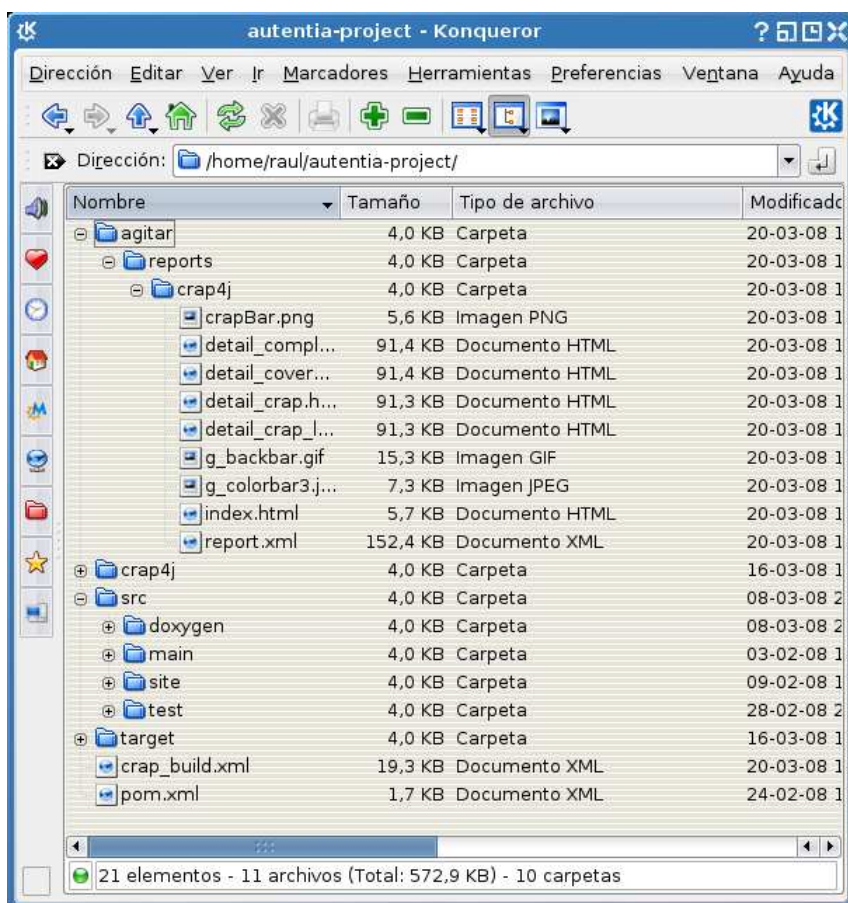
Nos dirigimos al directorio /home/raul/autentia-project/crap4j, donde hemos creado nuestro fichero build.xml y ejecutamos ant:

```
$ ant
Buildfile: build.xml

run-crap4j:
[crap4j] Buildfile: /home/raul/autentia-project/crap_build.xml
[crap4j]
[crap4j] run-tests:
[crap4j] [super-runner] Running com.autentia.project.entity.CodeTest
[crap4j] [super-runner] Tests run: 4, Failures: 0, Errors: 0, Time elapsed: 0,193 sec
[crap4j] [super-runner] Running com.autentia.project.entity.DetectionTest
[crap4j] [super-runner] Tests run: 11, Failures: 0, Errors: 0, Time elapsed: 0,419 sec
[crap4j] [super-runner] Running com.autentia.project.entity.DetectionTypeGroupTest
[crap4j] [super-runner] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 0,582 sec
[crap4j] [super-runner] Running com.autentia.project.entity.DetectionTypeTest
[crap4j] [super-runner] Tests run: 6, Failures: 0, Errors: 0, Time elapsed: 0,278 sec
[crap4j] [super-runner] Running com.autentia.project.entity.EngineTest
[crap4j] [super-runner] Tests run: 6, Failures: 0, Errors: 0, Time elapsed: 0,184 sec
[crap4j] [super-runner] Running com.autentia.project.entity.ProjectReportFileUrlTest
[crap4j] [super-runner] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 0,424 sec
[crap4j] [super-runner] Running com.autentia.project.entity.ProjectTest
[crap4j] [super-runner] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 0,367 sec
[crap4j] [super-runner] Running com.autentia.project.entity.ScanTest
[crap4j] [super-runner] Tests run: 7, Failures: 0, Errors: 0, Time elapsed: 0,244 sec
[crap4j] [super-runner] Running com.autentia.project.entity.SourceTest
[crap4j] [super-runner] Tests run: 10, Failures: 0, Errors: 0, Time elapsed: 0,448 sec
[crap4j] [super-runner] Running es.raulexposito.jaro.core.test.util.EntityUtilTest
[crap4j] [super-runner] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0,202 sec
[crap4j]
```

```
[crap4] agitar-all:
[crap4] BUILD SUCCESSFUL
[crap4] Total time: 17 seconds
[crap4] Ant exited with status 0
```

Tras ello, podeis ver que crea un par de cosas:



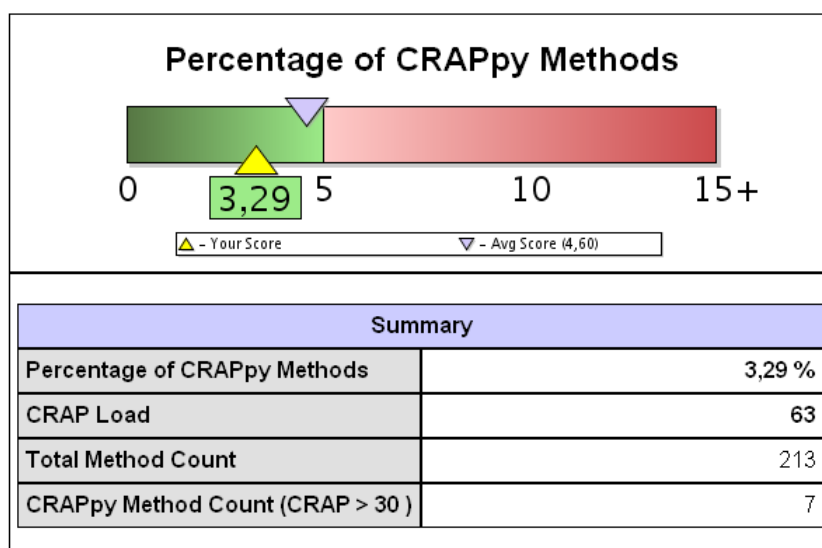
1. Un fichero crap_build.xml en el raíz del proyecto que podemos ignorar.
2. Los resultados de la ejecución de Crap4j en el directorio agitar/reports/crap4j

Para ver los resultados solo deberemos abrir con un navegador el fichero index.html del directorio agitar/reports/crap4j

CRAP Report

Project: /home/raul/autenticia-project

Generated at 20/03/08 16:54



Si estais usando linux y no pudieseis generar los informes por culpa de este error:

```
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified
```

```
[crap4j] java.lang.InternalError: Can't connect to X11 window server using ':0' as the value of the DISPLAY
variable.
[crap4j] at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
[crap4j] at sun.awt.X11GraphicsEnvironment.access$000(X11GraphicsEnvironment.java:53)
[crap4j] at sun.awt.X11GraphicsEnvironment$1.run(X11GraphicsEnvironment.java:142)
[crap4j] at java.security.AccessController.doPrivileged(Native Method)
```

lanzad este comando antes de invocar ant:

```
$ export ANT_OPTS=-Djava.awt.headless=true
```

5. Conclusiones

Como podeis ver no estamos "condenados" a usar Eclipse para poder utilizar la información proporcionada por Crap4j, ya que los autores de esta util herramienta nos permiten usarla de manera sencilla y configurable desde Ant.

Espero que os sea de utilidad.

- Puedes opinar sobre este tutorial [haciendo clic aquí](#).
- Puedes firmar en nuestro libro de visitas [haciendo clic aquí](#).
- Puedes asociarte al grupo AdictosAlTrabajo en XING [haciendo clic aquí](#).
- Añadir a favoritos Technorati. 



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#)). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com



Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subscripción a novedades:

E-mail

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	pdf
Introducción a ANT	En el mundo Java, la compilación, verificación e instalación de aplicaciones se ha normalizado con este potente paquete llamado ANT.	2003-06-14	23625	pdf
Callisto, nunca antes resultó tan fácil desarrollar con Eclipse	En este tutorial os enseñamos a instalar y utilizar Callipso: una aplicación que permite instalar de manera fácil y cómoda plugins y sus dependencias en Eclipse	2006-07-06	9989	pdf
Desarrollo Gráfico Scripts Ant	Os mostramos como crear y ejecutar scripts Ant (para automatizar tareas en el mundo Java) con las herramientas gratuitas Antelope y NetBeans	2004-06-15	10881	pdf
Cómo Crear una Tarea Ant	Este tutorial tiene como objetivo explicar cómo crear una tarea nueva Ant y utilizarla en un script.	2007-07-23	1746	pdf
Como integrar tareas Ant en Maven	En este tutorial nuestro compañero nos enseña como complementar maven usando ant y obtener mayor potencia en su conjunto	2008-01-17	874	pdf
Integración de Struts y eclipse	Alejandro Perez nos enseña como construir un entorno de alta eficiencia de desarrollo on Struts a través de plugins de eclipse	2003-11-28	47232	pdf
Optimización Java con Eclipse Profiler Plugin	Alejandro Pérez nos enseña como analizar el rendimiento de nuestras aplicaciones con Eclipse Profiler Plugin.	2004-07-21	17668	pdf
Subir el contenido de una carpeta por ftp mediante un script de Ant	Con este pequeño tutorial se pretende enseñar como poder subir todo el contenido de una carpeta por ftp, usando un script Ant.	2006-11-15	3608	pdf
Crap4j, ¿es tu código difícilmente mantenible?	En este tutorial os presentamos Crap4j, un plugin de eclipse que trata de localizar ese código con el que todos nos hemos encontrado alguna vez al auditar, mantener o continuar con un desarrollo y que resulta tan difícil de mantener...	2008-01-20	749	pdf

Nota:

Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento. Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores. En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo. Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.