



Eusko Jaurlaritzaren Informatika Elkarte
Sociedad Informática del Gobierno Vasco

AUS:

Manual de usuario

Fecha: 08/09/2006

Referencia:

EJIE S.A.
Mediterráneo, 3
Tel. 945 01 73 00*
Fax. 945 01 73 01
01010 Vitoria-Gasteiz
Posta-kutxatila / Apartado: 809
01080 Vitoria-Gasteiz
www.ejie.es

Control de documentación

Título de documento: AUS

Histórico de versiones

Código:

Versión: 1.0

Fecha: 08/09/2006

Resumen de cambios:

Cambios producidos desde la última versión

Primera versión.

Control de difusión

Responsable: Ander Martínez

Aprobado por: Ander Martínez

Firma:

Fecha:

Distribución:

Referencias de archivo

Autor: Consultoría de áreas de conocimiento

Nombre archivo: AUS. Manual de usuario v1.0.doc

Localización:

Contenido

Capítulo/sección	Página	
1	Introducción	4
2	Conceptos básicos	4
2.1	Dependencias	4
2.1.1.	Concepto de dependencia en AUS	4
2.1.2.	Dependencias frágiles	4
3	Funcionamiento	5
3.1	Ejecución de AUS como una tarea de <i>Ant</i>	5
3.1.1.	Ejemplos de ejecución como tarea de Ant	5
3.2	<i>Archivo de reglas</i>	6
3.2.1.	Especificar targets y locations usando comparadores (matchers)	7
3.2.2.	Default excludes	8
3.2.3.	Ejemplo de definición de reglas	9
3.2.4.	Evaluación de las reglas	9
3.3	Informes	10
3.3.1.	El informe resumen	10
3.3.2.	El informe de reglas	11
3.3.3.	Los informes de <i>uso</i> (dependencias encontradas) por cada archivo "escaneado"	12
3.4	"Escanear" páginas JSP	13
4	Utilidad práctica	14
5	Anexo 1 – Ejemplo en Ant	14

1 Introducción

El presente documento describe cuáles son los conceptos básicos y el proceso de ejecución de la herramienta AUS como tarea de *Ant* (API Usage Scanner).

2 Conceptos básicos

AUS (API Usage Scanner) es una herramienta J2EE, desarrollada en Java por IBM alphaWorks, que identifica dependencias de APIs de un producto en otro para crear un conjunto de detallados informes, mostrando dónde se encuentran dichas dependencias y cómo están siendo usadas. Por tanto, ayuda a los desarrolladores a detectar dependencias frágiles en sus aplicaciones Java.

Para obtener información adicional sobre el producto se puede acceder a su página Web:

http://www.alphaworks.ibm.com/tech/aus?open&S_TACT=105AGX59&S_CMP=GR&ca=dgr-jw26awaus

La última versión es la 3.1.3 (*aus-3.1.3-20051206.zip*) y se puede descargar desde:

<http://www.alphaworks.ibm.com/tech/aus/download>

El presente documento cubre el uso de AUS y su ejecución como tarea de *Ant*.

2.1 Dependencias

Una dependencia es una relación directa entre dos productos o componentes.

En una relación de dependencia siempre existe el producto que ofrece el soporte (A) y el producto dependiente (B) de dicho soporte (B depende de A, es decir, B no puede funcionar correctamente sin la presencia de A).

2.1.1. Concepto de dependencia en AUS

Una dependencia, en el ámbito de AUS, es una referencia explícita de Java en el código. Referencia perteneciente a un componente, una clase Java, un método o campo de otro componente.

2.1.2. Dependencias frágiles

Las siguientes condiciones hacen que una dependencia sea considerada frágil:

- La dependencia no es conocida por ambas partes (producto que oferta el soporte y el producto dependiente de dicho soporte). Ser conocida implica que dicha dependencia esté explícitamente documentada y consentida por ambas partes.
- El producto dependiente hace uso de una API privada del producto soporte.
- El producto dependiente necesita de una API en desuso (*deprecated*) del producto soporte.

3 Funcionamiento

AUS (API Usage Scanner) “escanea” código compilado en busca del uso de paquetes y clases específicos y muestra los resultados en informes ampliamente detallados.

Usado frecuentemente para encontrar en un producto el uso de APIs internas de otros productos y para descubrir dependencias frágiles (con alto riesgo de romperse cuando tanto el producto dependiente como el que ofrece el soporte sea modificado o actualizado).

La especificación de lo que se busca está contenida en un *archivo de reglas*. Siendo entendido este *archivo de reglas* como un conjunto de declaraciones (proporcionadas en un fichero de XML) de paquetes y clases a ser “escaneadas” (incluyendo la posibilidad del uso de expresiones regulares en dichas declaraciones).

Dos tipos de ficheros son “escaneados” por AUS:

- class files: ficheros .class.
- archive files (ficheros conjuntos): ficheros con extensiones .zip, .ear, .war, .rar (J2EE Resource Adapter Archive) o .jar.

3.1 Ejecución de AUS como una tarea de Ant

AUS puede ser invocado desde:

- línea de comandos,
- una tarea de *Ant*
- desde una API de Java

Este apartado se centrará únicamente en el segundo supuesto, es decir, en la ejecución de AUS como una tarea de *Ant*.

El manual de instalación de AUS en servidor incluye los requisitos previos, la configuración y el *script* que define la tarea de *Ant* para su ejecución.

3.1.1. Ejemplos de ejecución como tarea de Ant

Hay que recordar que los tipos de ficheros “escaneados” por AUS son:

- **class** files: archivos .class.
- **archive files** (ficheros conjuntos): archivos con extensiones .zip, .ear, .war, .rar (J2EE Resource Adapter Archive) o .jar.

Teniendo esto en cuenta y considerando el *script* definido en el manual de instalación de AUS, se muestran los siguientes ejemplos de ejecución de AUS en el servidor como una tarea de *Ant*:

- Valida las dependencias de los ficheros ".class" o *Archive files* del subdirectorio “classes” (aplicación completa) y considerando el *archivo de reglas* por defecto *rules.xml*:

```
ant aus
```

- Valida las dependencias de los ficheros ".class" o *Archive files* del subdirectorio "classes/s73a"

(se introduce la ruta a partir del subdirectorio classes) considerando el *archivo de reglas* por defecto *rules.xml*:

```
ant -Dpath=s73a aus
```

- Valida las dependencias de los ficheros ".class" o *Archive files* del subdirectorio "s73aPedidosWar" considerando el *archivo de reglas aus_rules.xml* situado en el subdirectorio *audit* (este parámetro aceptaría rutas absolutas):

```
ant -Dpath=s73aPedidosWar -Drules=WebRoot/test/audit/aus_rules.xml aus
```

3.2 Archivo de reglas

El *archivo de reglas* es obligatorio a la hora de ejecutar AUS puesto que configura las funciones a realizar, como pueden ser:

- Buscar tipos particulares de Java.
- Mirar dentro de tipos específicos de Java (que se encuentren dentro del conjunto de archivos pasados con el parámetro *-input*).
- Especificar cómo informar de los usos de los tipos encontrados (mensajes y colores).

El *archivo de reglas* de AUS es un documento XML que debe seguir la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="ibm.com/software/namespaces/apiscanner"
  xsi:schemaLocation="ibm.com/software/namespaces/apiscanner aus30-rules.xsd"
  name="Nombre para el ruleset"
>

  <!-- Policy declaration -->

  <!-- Group declaration -->

  <!-- Rules -->

</ruleset>
```

Un elemento `<policy>` recoge 1..n grupos de reglas (`<groups>`). Un *archivo de reglas* debe tener 1..n elementos `<policy>` pero sólo se usa uno para una ejecución simple del escáner. Por defecto, el primer elemento `<policy>` es "escaneado". Un elemento `<policy>` está formado por:

- Un atributo *name* (único para el *archivo de reglas*).
- El atributo opcional *defaultExcludes* que puede tomar el valor *true* o *false* para incluir/excluir dependencias encontradas en *locations* y que también son *targets*.
- 1..n elementos `<group>`.
- 0..n elementos `<target>`.
- 0..n elementos `<excludetarget>`.
- 0..n elementos `<location>`.
- 0..n elementos `<excludelocation>`.

Un elemento <group> puede presentar la siguiente estructura:

```
<group>
  <description>
  <excludetarget>
  <target>
  <location>
  <excludelocation>
  <ruleref>
```

Es un elemento usado para agrupar juntos elementos *rule* individuales del *archivo de reglas*. Cada elemento <group> está formado por:

- Un atributo *refname* (único para el archivo de reglas).
- El atributo opcional *defaultExcludes* que puede tomar el valor *true* o *false* para incluir/excluir dependencias encontradas en *locations* y que también son *targets*.
- El elemento <description> que muestra el propósito de la regla.
- 0..n elementos <target>.
- 0..n elementos <excludetarget>.
- 0..n elementos <location>.
- 0..n elementos <excludelocation>.

El atributo *name* se usa para referirse a la regla en los informes. Una regla puede contener un conjunto de elementos <target>, <excludetarget>, <location> y <excludelocation>. Estos elementos definen lo que la regla está buscando (*targets*) y qué clases (*locations*) deberían ser "escaneadas".

Se pueden definir múltiples *targets* (tipos a buscar) y múltiples *locations* (rutas de clases a buscar) dentro de un elemento <rule> usando los elementos <target>, <excludetarget>, <location> y <excludelocation>. El orden dentro de una regla individual (elemento <rule>) de los múltiples elementos *target* y *location* es importante e influirá en el resultado de las estadísticas (se evaluará posteriormente).

Los elementos *target* y *location* utilizan elementos de comparación (*matcher elements*) para definir patrones sobre los que el escáner operará. El nombre de un tipo de Java se representará usando dos comparadores (*matchers*). Estos comparadores se usan para comparar secuencias de caracteres, identificando patrones que son comparados contra tipos Java.

3.2.1. Especificar targets y locations usando comparadores (matchers)

Un elemento comparador simplemente compara una cadena contra otra cadena o patrón. Cada elemento comparador se define en el *archivo de reglas* a través de dos atributos:

- El valor (*value*) del elemento comparador (lo que comparar).
- El tipo (*type*) del elemento comparador (cómo interpretar el valor).

Los elementos de comparación (*matchers*) pueden ser especificados en elementos <rule>, <policy> y <group>. Esto posibilita un mecanismo válido para aplicar elementos de comparación a múltiples reglas usando una simple declaración del elemento comparador.

Los elementos de comparación especificados dentro de un elemento <policy> se consideran antes que los elementos de comparación de los elementos <group> incluidos dentro de estos elementos <policy>. Los

elementos de comparación especificados en un elemento <group> (incluidos los heredados del elemento <policy>) por tanto son añadidos al principio de la lista de elementos de comparación especificados para cada elemento <rule> dentro del elemento <group>.

Existen ocho atributos para definir en elemento de comparación (matcher element):

- packageName
- packageType
- className
- classType
- methodName
- methodType
- fieldName
- fieldType

Los siguientes tipos de elementos de comparación (*matcher element types*) están permitidos:

- **packageOrSubpackage (default)**: Se usa este tipo de comparación cuando se omite el atributo *packageType*. Esto produce que la comparación sea hecha para el paquete especificado o para un subpaquete del mismo.

El valor *com.ibm.* encaja con *com.ibm*, *com.ibm.websphere* y con *com.ibm.a.b.c* pero no con *com.ibmwas*.

- **classOrNestedClass (default)**: Se usa este tipo de comparación cuando se omite el atributo *classType*, lo que causa que la comparación se realice para la clase especificada o para alguna clase anidada de la misma.

El valor *Outer* encaja con las clases *Outer*, *Outer\$Inner* y *Outer\$Inner\$MoreInner* pero no con la clase *OuterClass*.

- **regexp**: Se usa este tipo de comparación para mandar a AUS tratar el valor de comparación como una expresión regular. Este es el tipo de comparación más potente y que ofrece la mayor flexibilidad. Se espera que un valor sea especificado usando el nombre JVM de la clase. Por ejemplo, la clase anidada *Inner* de la clase *Outer* en el paquete *com.ibm* podría expresarse como *com/ibm/Outer\$Inner*.

Un valor puede ser *com/ibm/.**

Un valor puede ser *com/[ibm|lotus]/.**

- **exact**: Se usa este tipo de comparación para que el valor que desea compara encaje exactamente con el valor de un *target* o *location*.

El valor del nombre de la clase a comparar *EntryPoint* deberá encajar solamente con la clase *EntryPoint*, no encajando con *ProgramEntryPoint* o *EntryPoint1*.

3.2.2. **Default excludes**

El atributo <defaultexcludes> de los elementos <policy>, <group> y <rule> toma el valor *true* o *false*. Si no se especifica, se toma el valor del elemento padre <group> o <policy>. Y en caso de que estos tampoco lo

especifiquen se asume el valor *false*.

Cuando el valor es true manda a AUS tratar cada elemento *target* como un elemento *excludelocation*. Esto hace que AUS no "escanee" ninguna clase donde el nombre de la clase es un *target* para el escáner. Esto es útil para "escanear" productos donde algunas clases *target* pueden ser incluidas en el "escaneo" (esto puede suceder para productos apilados que no pueden ser separados fácilmente para su "escaneo").

3.2.3. Ejemplo de definición de reglas

Normalmente existen varias maneras de expresar las mismas reglas usando distintos tipos de elementos de comparación (*matcher elements*). Parece obvio indicar que, por razones de rendimiento, se debería usar siempre el tipo disponible más sencillo.

Encontrar código perteneciente a la Apache Software Foundation (ASF) puede servir como un ejemplo sencillo de cómo definir reglas (tipos Java declarados en el paquete *org.apache* o en subpaquetes pertenecientes a la ASF).

Para encontrar todos los usos del código de ASF se pueden usar las siguientes reglas:

```
<rule name="Apache">
  <target packageName="org.apache" />
  <description> Reporting use of Apache code</description>
</rule>
```

Se recomienda utilizar este tipo de expresiones para definir reglas. Debería proporcionar un mayor rendimiento durante el "escaneo". Omitir el atributo *packageType* consigue que dicho "escaneo" tome un valor implícito de *packageOrSubpackage*.

El siguiente ejemplo utiliza una expresión regular para el nombre de paquete que se debe buscar con esta regla (*package/class\$inner*):

```
<rule name="Apache">
  <description> Reporting use of Apache code</description>
  <target packageType="regexp" packageName="org/apache/.*" />
</rule>
```

3.2.4. Evaluación de las reglas

Los elementos definidos en una regla son evaluados en orden inverso. Como antes se ha visto, estos elementos son *<target>*, *<location>*, *<excludetarget>* y *<excludelocation>*.

Es importante entender el orden de evaluación y su efecto en la precisión de los informes generados por AUS. La lógica que hay detrás de la evaluación en orden inverso se ilustra con el siguiente ejemplo:

```
<excludelocation packageName="com.ibm"/>
<location packageName="com.ibm.product"/>
<excludelocation packageName="com.ibm.product.component"/>
```

La clase se "escanea" si está dentro del paquete *com.ibm.product* pero no está dentro del paquete *com.ibm.product.component*. Y si no está en este paquete sólo se "escanea" si está fuera del paquete *com.ibm*.

3.3 Informes

Este apartado explica brevemente cómo se estructuran los informes y cómo interpretarlos correctamente.

Los informes se pueden dividir en tres secciones:

- El informe resumen (*summary report*).
- El informe de reglas (*rule report*).
- Los informes de *uso* (dependencias encontradas) por cada archivo "escaneado".

3.3.1. El informe resumen

El informe resumen es un fichero sencillo (summary.xml o summary.html) que contiene una visión general del "escaneo" llevado a cabo. En este informe se muestra:

- Información sobre el "escaneo" realizado: La versión de AUS usada, la URL del *archivo de reglas* usado, la fecha de realización, la duración en milisegundos, información sobre el XSLT usado para generar el HTML, el número total de archivos "escaneados", el número total de dependencias encontradas, etc.

Scan Information		
Scan Title	Untitled	Number of archives scanned
AUS Information	Version 3.1.3 built on December 6 2005	Loose classes scanned
Ruleset / Policy	Test Rules / default	Number of classes scanned
Scan Date	mié, 29 ago 2007 12:21:56 +0200	Number of usages found
Scan Duration	3.437 seconds	Usages on distinct lines (approx.)
XSL Engine Vendor & Version	Apache Software Foundation (http://xml.apache.org/xalan-j/) Version 1	

- Una tabla resumen de las dependencias encontradas para los archivos "escaneados" y agrupadas en función de cada grupo de reglas considerado (Input Location).

Usages by Input Location							
Archive	Rule matches						
	Summary	DOU	JavaXML	Logging	Removable	SHCode	Total
asm-1.5.3.jar	link	-	-	-	-	-	0
aus.jar	link	168	162	1324	200	50	1904
commandlineutil.jar	link	-	9	75	57	-	141
Loose Classes	link	-	-	-	-	-	0
Grand Totals		168	171	1399	257	50	2045

- Una tabla resumen con las distintas clases (Target) que se buscaban en el "escaneo" (estadísticas de las dependencias encontradas).

Usages by Target

Target Class	Occurrences	
	%age	Total
java.util.logging.Logger	34.47%	705
java.util.logging.Level	33.06%	676
org.objectweb.asm.Type	5.04%	103
org.xml.sax.Attributes	2.69%	55
org.xml.sax.SAXException	2.25%	46
javax.xml.transform.Transformer	2.05%	42
org.w3c.dom.Element	1.91%	39
org.w3c.dom.Document	1.71%	35
org.objectweb.asm.Label	1.52%	31

- Un conjunto de tablas, una por cada archivo "escaneado", para mostrar las dependencias encontradas individualmente por cada clase.

aus.jar

Archive Path

C:\aus\lib\aus.jar

Java Class/Interface	Rule matches					
	DOU	JavaXML	Logging	Removable	SHCode	Total
com.ibm.kamilos.report.XMLUtilities	-	70	116	10	-	196
com.ibm.kamilos.report.FileUtilities	-	-	140	-	-	140
com.ibm.kamilos.report.ReportGenerator	-	8	109	20	-	137
com.ibm.kamilos.rule.parser.RuleFileSAXHandler	-	-	82	49	-	131
com.ibm.kamilos.Kamilos	-	-	90	-	34	124
com.ibm.kamilos.StyleReports	-	17	68	9	16	110

3.3.2. El informe de reglas

Como parte del "escaneo" llevado a cabo, el *archivo de reglas* se copia en el directorio donde se hayan generado los informes bajo el nombre de *rules.xml* así como una versión HTML denominada *rules.html*.

AUS 3.x Rules Report

Back to [Summary Report](#)

Ruleset title	Test Rules
Policy Name	default
Ruleset URL	jar:file:C:\aus\bin\..\lib\aus.jar!/resources/ivt.xml
Number of active rule groups	5
Number of active rules	7

Policy	default			
Rule Groups	Group Name	Description	Rule Name	Description
	DOU	AUS dependencies on open source code	ASM Apache	Report direct use of ASM code Reporting use of Apache Software Foundation code
	JavaXML	AUS dependencies on the Java XML extensions.	XML	Reporting use of JAXP interfaces
	SHCode	Uses of code owned by System House.	CmdLine AUS	Reporting use of Command Line Utility Report use of classes in AUS
	Logging	Uses of the JSR47 Logging APIs	Logging	Report use of the Java logging APIs within AUS
	Removable	Dependencies on other products	Proprietary	Uses of anything outsource of java, javax, org.apache, com.ibm.kamilos and com.ibm.commandline

Este archivo HTML contiene información sobre la política, los grupos de reglas y sobre las propias reglas utilizadas durante el "escaneo":

- Política:

Policy	
Policy Name	default
<i>No targets or locations specified</i>	

- Grupos de reglas:

Rule Groups		
Rule Group	DOU	AUS dependencies on open source code
<i>No targets or locations specified</i>		
Rule Group	JavaXML	AUS dependencies on the Java XML extensions.
<i>No targets or locations specified</i>		
Rule Group	SHCode	Uses of code owned by System House.
<i>No targets or locations specified</i>		
Rule Group	Logging	Uses of the JSR47 Logging APIs
<i>No targets or locations specified</i>		
Rule Group	Removable	Dependencies on other products
<i>No targets or locations specified</i>		

- Reglas:

Rules							
Rule	ASM		Report direct use of ASM code				
Matchers	Incl/Exc		Package	Class	Field	Method	Description
Targets	include	Regular expression	org/objectweb/.*	-	-	-	All objectweb packages
Rule	Apache		Reporting use of Apache Software Foundation code				
Matchers	Incl/Exc		Package	Class	Field	Method	Description
Targets	include	Package or subpackage	org.apache	-	-	-	All Apache Packages
Rule	XML		Reporting use of JAXP interfaces				
Matchers	Incl/Exc		Package	Class	Field	Method	Description
Targets	include	Regular expression	javax/xml/.*	-	-	-	All Java XML Packages

3.3.3. Los informes de uso (dependencias encontradas) por cada archivo "escaneado"

Éstos son los informes más detallados que AUS genera. Existe uno por cada uno de los archivos "escaneados" y describe las dependencias encontradas.

En el caso concreto de *commandlineutil.jar* el informe obtenido presenta el siguiente aspecto, mostrando en primer lugar un resumen de las estadísticas obtenidas:

commandlineutil.jar - AUS 3.x Scan Results

[Back to Summary Report](#)

Product Information	Version 3.1.3 built on December 6 2005
Input Path	C:\aus\lib\commandlineutil.jar
Ruleset / Policy	Test Rules / default
Usage Instances Found	141
UIs on distinct lines (approx.)	101
Number of classes scanned	42
Classes found with usage instances	12 (28.57%)
Classes found with no usage instances	30 (71.43%) click here to view
Show or Hide Detail	<input type="button" value="Show All"/> <input type="button" value="Hide All"/>
Rule Toggles	<input checked="" type="checkbox"/> Apache <input checked="" type="checkbox"/> ASM <input checked="" type="checkbox"/> AUS <input checked="" type="checkbox"/> CmdLine <input checked="" type="checkbox"/> Logging <input checked="" type="checkbox"/> Proprietary <input checked="" type="checkbox"/> XML

Y, como ya se ha comentado, muestra una amplia información sobre cada una de las dependencias encontradas:

Classes containing usage instances			
com.ibm.commandline.specification.parser.CommandXMLParser			Usage Count: 76 (53 unique)
Rule	Line	Method	Target Class
Proprietary	-		org.xml.sax.helpers.DefaultHandler
Logging	-		java.util.logging.Logger
Logging	30	void <init>()	java.util.logging.Logger
Logging	30	void <init>()	java.util.logging.Logger
Proprietary	42	void <init>()	org.xml.sax.helpers.DefaultHandler
Proprietary	89	com.ibm.commandline.schema.Commandspec.parse(java.io.InputStream, java.lang.String)	org.xml.sax.InputSource

3.4 “Escanear” páginas JSP

AUS sólo puede "escanear" código binario (class files y Java archives como ya se ha comentado). Por tanto es necesario convertir estas páginas JSP a sus correspondientes archivos .class, es decir, es necesario compilarlas (*JSP batch compiler*).

Por tanto, habrá que seguir estos pasos para detectar las dependencias (frágles) de las JSP's:

1. Recoger todas las JSP's a "escanear" en un único directorio, por ejemplo, *jspdir*.
2. Compilar estas JSP's.
3. Ejecutar AUS utilizando como atributo de entrada *-input jspdir*.
4. Puesto que estas clases no están contenidas dentro de ningún *Java archive* específico, para observar los resultados obtenidos habrá que acceder al apartado *Loose Classes* que se encuentra dentro del informe resumen (*summary.html*).

4 Utilidad práctica

Puede resultar interesante el uso de AUS para la búsqueda del uso de dependencias no homologadas en el código de nuestras aplicaciones.

Para ello testeara el código que indiquemos en busca de las dependencias elegidas.

5 Anexo 1 – Ejemplo en Ant

Un ejemplo sencillo para encontrar el uso de dependencias que se encuentren fuera del conjunto de librerías homologadas podría ser el reflejado en el siguiente *archivo de reglas (rules.xml)*:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="ibm.com/software/namespaces/apiscanner"
  xsi:schemaLocation="ibm.com/software/namespaces/apiscanner aus30-rules.xs"
  name="Archivo de reglas">

  <policy name="default">
    <group name="Evitables" reportcolor="red">
      <description>Dependencias de otros productos</description>
      <ruleref refname="DepEvitables" />
    </group>
  </policy>

  <rule name="DepEvitables">
    <description>
      Uso de clases que no pertenezcan a los paquetes
      java, javax, org.apache, s73a y es.ejie
    </description>

    <target packageName=".*" packageType="regexp"
      description="Todos los paquetes" />

    <excludetarget packageName="java"
      description="Paquetes de Java" />
    <excludetarget packageName="javax"
      description="Paquetes de Javax" />
    <excludetarget packageName="org.apache"
      description="Paquetes de Apache" />
    <excludetarget packageName="s73a"
      description="Paquetes de s73a" />
    <excludetarget packageName="es.ejie"
      description="Paquetes de EJIE" />

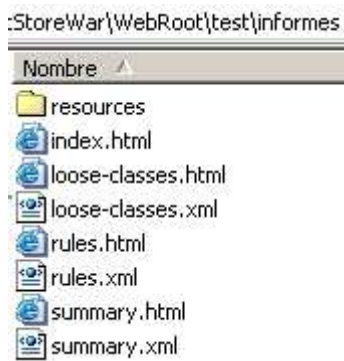
  </rule>
</ruleset>
```

Se puede observar que se realizará una búsqueda de todas las dependencias que se estén utilizando en

nuestras aplicaciones (o en parte de ellas) obviando la búsqueda de las dependencias incluidas en los paquetes (y sus correspondientes subpaquetes) indicados en los atributos *packageName* (paquetes de java, de la propia aplicación, etc.) de los elementos *excludetarget*.

En este ejemplo se considerarán como dependencias “no homologadas” las que se encuentren fuera de los paquetes considerados en los elementos `<excludetarget>` y así se tratará de reflejar en los informes.

Tras lanzar la ejecución de AUS, a través de esta tarea de *Ant*, el resultado obtenido presentará, dentro de la carpeta destinada para tal fin, una estructura como la siguiente:



Abriendo la página *summary.html* (o bien, a través de *index.html*) se podrá acceder a toda la información recogida en los informes generados por parte de AUS:

Untitled - AUS 3.1.3 Scan Summary Report

Scan Information

Scan Title	Untitled	Number of archives scanned	0
AUS Information	Version 3.1.3 built on December 6 2005	Loose classes scanned	0
Ruleset / Policy	Archivo de reglas / default	Number of classes scanned	25
Scan Date	vie, 14 sep 2007 08:29:37 +0200	Number of usages found	0
Scan Duration	1.203 seconds	Usages on distinct lines (approx.)	0
XSL Engine Vendor & Version	Apache Software Foundation (http://xml.apache.org/xalan-j) Version 1		

Usages by Input Location

Archive	Rule matches		
	Summary	Evitables	Total
Loose Classes	link	-	0
Grand Totals		0	0

Usages by Target

Target Class	Occurrences	
	%age	Total
Grand total for 0 classes		0

Obsérvese que se han “escaneado” 25 clases y que no se han encontrado usos de dependencias “no homologadas” (reflejado en la tabla o sección “Scan Information”). El número de usos de estas dependencias quedaría reflejado en la columna “Evitables” (nombre utilizado para el elemento `<group>` que hace referencia a la regla considerada) de la tabla “Usages by Input Location”.

En este caso las clases “escaneadas” han sido introducidas dentro del apartado *Loose Classes*. Se puede acceder a su información relacionada a través del enlace encontrado en esta última tabla “Usages by Input Location” (página *loose-classes.html*):

loose classes on the filesystem - AUS 3.x Scan Results

Back to [Summary Report](#)

Product Information	Version 3.1.3 built on December 6 2005
Ruleset / Policy	Archivo de reglas / default
Usage Instances Found	0
Uls on distinct lines (approx.)	0
Number of classes scanned	25
Classes found with usage instances	0 (0%)
Classes found with no usage instances	25 (100%) click here to view
Show or Hide Detail	<input type="button" value="Show All"/> <input type="button" value="Hide All"/>
Rule Toggles	<input checked="" type="checkbox"/> DepEvitables

Classes containing usage instances

None.

Classes containing no usage instances (25)

```
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aContainsItemAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aDispatchLoginAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aEditAccountFormAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aImprimirDocuAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetAccountUserAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetAccountUserPassAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetCategoryAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetCategoryListAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetItemAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetItemListProductAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetListCategoryIdAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aGetProductAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aInsertAccountAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aAlterInStockAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aListCategoryAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aSignOn1Action
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aSignOn2Action
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aUpdateAccountAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aOperacionesInternas/s73aViewItemAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aRemoveItemCartAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aUpdateCartQuantitiesAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aViewCartAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aViewCategoryAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aViewProductAction
s73a/s73aCapaWeb/s73aPetStore/s73aAction/s73aWelcomeAction
```

Se muestran todas las clases “escaneadas” separadas en dos grupos:

- Clases conteniendo usos de dependencias “no homologadas”: En el informe obtenido reflejadas en la sección “Classes containing usage instances” (en este caso concreto, se puede observar en la imagen superior, que no hay ninguna).

- Clases que no contienen usos de dependencias “no homologadas”: En el informe obtenido reflejadas en la sección “Classes containing no usage instances” (por tanto las restantes, es decir, las 25 clases “escaneadas”).

Desde la página *summary.html*, además, se podrá acceder al *archivo de reglas (rules.xml)*:

AUS 3.x Rules Report

[Back to Summary Report](#)

Ruleset title	Archivo de reglas
Policy Name	default
Ruleset URL	file:/C:/PETworkspace/s73aPetStoreWar/WebRoot/test/audit/rules.xml
Number of active rule groups	1
Number of active rules	1

Policy	default			
Rule Groups	Group Name	Description	Rule Name	Description
	Evitables	Dependencias de otros productos	DepEvitables	Uso de clases que no pertenezcan a los paquetes java, javax, org.apache.s73a y es.ejie

Policy

Policy Name	default
<i>No targets or locations specified</i>	

Rule Groups

Rule Group	Evitables	Dependencias de otros productos
<i>No targets or locations specified</i>		

Rules

Rule	DepEvitables							Uso de clases que no pertenezcan a los paquetes java, javax, org.apache, s73a y es.ejie		
Matchers	Incl/Exc	Package			Class	Field	Method	Description		
Targets	include	Regular expression	.	*	-	-	-	-	-	Todos los paquetes
	exclude	Package or subpackage	java		-	-	-	-	-	Paquetes de Java
	exclude	Package or subpackage	javax		-	-	-	-	-	Paquetes de Javax
	exclude	Package or subpackage	org.apache		-	-	-	-	-	Paquetes de Apache
	exclude	Package or subpackage	s73a		-	-	-	-	-	Paquetes de s73a
exclude	Package or subpackage	es.ejie			-	-	-	-	-	Paquetes de EJIE

Resulta interesante ver la información que refleja esta última tabla (“Rules”). Se muestra la descripción de lo que se incluye o excluye en la definición de la regla para realizar el escáner. En este ejemplo, entonces, se buscará el uso de cualquier tipo de dependencias pero excluyendo las relativas a los paquetes (o subpaquetes) de **java**, **javax**, **org.apache**, **s73a** y **es.ejie**.

Para terminar con este ejemplo será interesante mostrar cómo se reflejará el uso de ciertas dependencias. Para ello bastará simplemente con eliminar el siguiente elemento `<excludetarget>` del *archivo de reglas*:

```
<excludetarget packageName="es.ejie" description="Paquetes de EJIE" />
```

Por tanto, en este caso, la páginas *summary.html* presentará el siguiente aspecto:

Untitled - AUS 3.1.3 Scan Summary Report

Scan Information

Scan Title	Untitled	Number of archives scanned	0
AUS Information	Version 3.1.3 built on December 6 2005	Loose classes scanned	25
Ruleset / Policy	Archivo de reglas / default	Number of classes scanned	25
Scan Date	vie, 14 sep 2007 10:01:00 +0200	Number of usages found	321
Scan Duration	1.781 seconds	Usages on distinct lines (approx.)	170
XSL Engine Vendor & Version	Apache Software Foundation (http://xml.apache.org/xalan-j) Version 1		

La imagen superior muestra que se han “escaneado” 25 clases, como en el caso anterior, pero ahora se han encontrado 321 usos de dependencias en aproximadamente 170 líneas distintas de código.

Usages by Input Location

Archive	Rule matches		
	Summary	Evitables	Total
Loose Classes	link	321	321
Grand Totals		321	321

Usages by Target

Target Class	Occurrences	
	%age	Total
es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE	45.79%	147
es.ejje.frmk.negocio.operacioninterna.Q70OperacionInterna	18.07%	58
es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors	18.07%	58
es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean	8.72%	28
es.ejje.frmk.presentacion.operacion.paso.Q70ActionPasoOP	6.23%	20
es.ejje.frmk.infraestructura.traza.Q70TraceLevel	1.87%	6
es.ejje.frmk.listeners.base.Q70ListenerUtils	1.25%	4
Grand total for 7 classes		321

La tabla “Usages by Target” muestra las diferentes dependencias encontradas relativas al paquete **es.ejje** tanto en número como en porcentaje.

Y en la imagen siguiente se muestran en cuáles de las clases “escaneadas” (en este caso todas) han sido encontrados los usos de dependencias buscados y en qué número.

Archive Summaries

Loose Classes

Java Class/Interface	Rule matches	
	Evitables	Total
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aListCategoryAction	24	24
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aInsertAccountAction	19	19
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aUpdateAccountAction	19	19
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetAccountUserPassAction	18	18
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetItemAction	18	18
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetItemListItemAction	17	17
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetListCategoryIdAction	17	17
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aSignOn1Action	17	17
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetCategoryAction	16	16
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetCategoryListAction	16	16
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetProductAction	16	16
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aViewItemAction	16	16
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aSignOn2Action	15	15
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aAlterInStockAction	14	14
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aGetAccountUserAction	13	13
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aWelcomeAction	13	13
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aImprimirDocuAction	10	10
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aContainsItemAction	7	7
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aUpdateCartQuantitiesAction	7	7
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aDispatchLoginAction	6	6
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aRemoveItemCartAction	6	6
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aEditAccountFormAction	5	5
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aViewCartAction	4	4
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aViewCategoryAction	4	4
s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aViewProductAction	4	4

El aspecto que presentará ahora la página *loose-classes.html* será el siguiente:

loose classes on the filesystem - AUS 3.x Scan Results

[Back to Summary Report](#)

Product Information	Version 3.1.3 built on December 6 2005
Ruleset / Policy	Archivo de reglas / default
Usage Instances Found	321
Uls on distinct lines (approx.)	170
Number of classes scanned	25
Classes found with usage instances	25 (100%)
Classes found with no usage instances	0 (0%)
Show or Hide Detail	<input type="button" value="Show All"/> <input type="button" value="Hide All"/>
Rule Toggles	<input checked="" type="checkbox"/> DepEvitables

Classes containing usage instances

s73a.s73aCapaWeb.s73aPetStore.s73aAction.s73aOperacionesInternas.s73aListCategoryAction			
Rule Toggles <input checked="" type="checkbox"/> DepEvitables			
Rule	Line	Method	Target Class
DepEvitables	-		es.ejje.frmk.negocio.operacioninterna.Q70Operacio
DepEvitables	29	void <init>()	es.ejje.frmk.negocio.operacioninterna.Q70Operacio
DepEvitables	42	void preProcessMappingEntrada (es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors)	es.ejje.frmk.presentacion.sesiondatos.Q70Contexto
DepEvitables	42	void preProcessMappingEntrada (es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors)	es.ejje.frmk.negocio.operacioninterna.Q70Operacio
DepEvitables	42	void preProcessMappingEntrada (es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors)	es.ejje.frmk.negocio.operacioninterna.Q70Operacio
DepEvitables	42	void preProcessMappingEntrada (es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors)	es.ejje.frmk.presentacion.sesiondatos.Q70Contexto
DepEvitables	43	void preProcessMappingEntrada (es.ejje.frmk.presentacion.sesiondatos.Q70ContextoEJIE, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaConfigBean, es.ejje.frmk.negocio.operacioninterna.Q70OperacionInternaErrors)	es.ejje.frmk.presentacion.sesiondatos.Q70Contexto

Se puede observar, en la imagen superior, que por cada clase en la que se haya encontrado un uso de las dependencias buscadas se mostrará la información relativa a:

- La línea de código en la que aparece el uso de la dependencia.
- El método en el que aparece.
- La clase a la que hace referencia la dependencia.
- La manera en la que es utilizada dicha dependencia.