

Anàlisi d'Utilització de Memòria en Aplicacions Canigó

A qui va dirigit

Aquest how-to va dirigit a tots aquells que al desenvolupar una aplicació amb Canigó tinguin necessitat d'analitzar la utilització de memòria.

Versió de Canigó

Els passos descrits en aquest document són d'aplicació a les versions 2.x de Canigó.

Introducció

En aquest document es presenten algunes recomanacions, en relació amb els mecanismes disponibles per analitzar la utilització de memòria en aplicacions web desenvolupades amb Canigó.

En primer lloc, es presenta un cas particular, que consisteix en analitzar els objectes emmagatzemats en la Sessió http.

A continuació es descriu una eina que permet analitzar la utilització de memòria en una aplicació en execució.

Objectes guardats en la Sessió http

Tot i que constitueixen una petita part dels objectes involucrats en la execució d'una aplicació web, els que es guarden en la Sessió http tenen una especial rellevància degut a que

- la quantitat d'objectes guardats creix amb el número de sessions actives i, per tant, té un efecte negatiu sobre la escalabilitat de l'aplicació que no es posarà de manifest a menys que es facin proves de càrrega degudament dimensionades
- tenen una vida potencialment llarga i no seran eliminats pel Garbage Collector mentre duri la sessió
- quan l'aplicació s'executa en servidors muntats en clúster, els objectes guardats en la sessió http s'han de replicar a tots els nodes del clúster, amb unes operacions de serialització, transmissió per xarxa i des-serialització que són relativament costoses

Per facilitar l'anàlisi de quins són els objectes guardats en la sessió i quin tamany tenen, en l'aplicació plantilla de Canigó s'ha inclòs la classe `net.gencat.ctti.canigo.services.VerificacioSessioSerialitzable` que realitza un bolcat de la sessió http, calculant el tamany serialitzat de cada un dels objectes que conté.

Una forma d'utilitzar aquesta classe és afegir en alguna de les pàgines de l'aplicació (es proposa `footer.jsp`)

```
<%  
new net.gencat.ctti.canigo.services.VerificacioSessioSerialitzable(request.getSession());  
%>
```

Anàlisi d'Utilització de Memòria en Aplicacions Canigó

llavors, a mida que es va navegant per l'aplicació, apareix una traça del contingut de la sessió

```
Atributs de sessió:
- fr.improve.struts.taglib.layout.util.FormUtils.FORM_MODE_KEY (java.util.HashMap) - 473
- javax.servlet.jsp.jstl.fmt.request.charset (java.lang.String) - 17
- paramsSubmit12566638377271575975729 (java.util.HashMap) - 293
- CATEGORY (java.util.HashMap) - 174
- ACEGI_SECURITY_CONTEXT (net.sf.acegisecurity.context.SecurityContextImpl) - 981
- fr.improve.struts.taglib.layout.util.FormUtils.FIELD_MODE_KEY (java.util.HashMap) - 421
- org.springframework.web.servlet.i18n.SessionLocaleResolver.LOCALE (java.util.Locale) - 127
- acegiPermissionsAdapter
(net.gencat.ctti.canigo.services.web.struts.menu.AcegiPermissionsAdapter) - 259
- org.apache.struts.action.LOCALE (java.util.Locale) - 127
- javax.servlet.jsp.jstl.fmt.locale.session (java.util.Locale) - 127
- paramsSubmit12566638377271575975729%%VALIDATORNAME (java.lang.String) - 61
- ACEGI_SECURITY_LAST_USERNAME (java.lang.String) - 21
tamany total dels atributs de la sessió: 3081
```

aquesta traça indica la llista de tots els atributs guardats a la sessió i, per cada un, la classe i el tamany total de l'objecte serialitzat (i per tant incloent tot l'arbre d'objectes que pugui contenir).

Es recomana **no** utilitzar aquesta eina en els entorns de producció ja que cada vegada realitza una serialització en memòria de tots els atributs de la sessió i, per tant, pot consumir una quantitat significativa de recursos.

És difícil donar un valor a partir del qual el tamany de la sessió es pugui considerar excessiu, ja que depèn de diversos factors que s'han de tenir en compte

- número màxim de sessions concurrents
- freqüència de les peticions realitzades en una sessió i que realitzin modificacions en els atributs de la sessió
- si l'aplicació s'executa o no en un clúster i, en cas afirmatiu, quants nodes constitueixen el clúster

Aquests factors s'hauran d'haver tingut en compte al realitzar el disseny de l'aplicació i dimensionar els servidors.

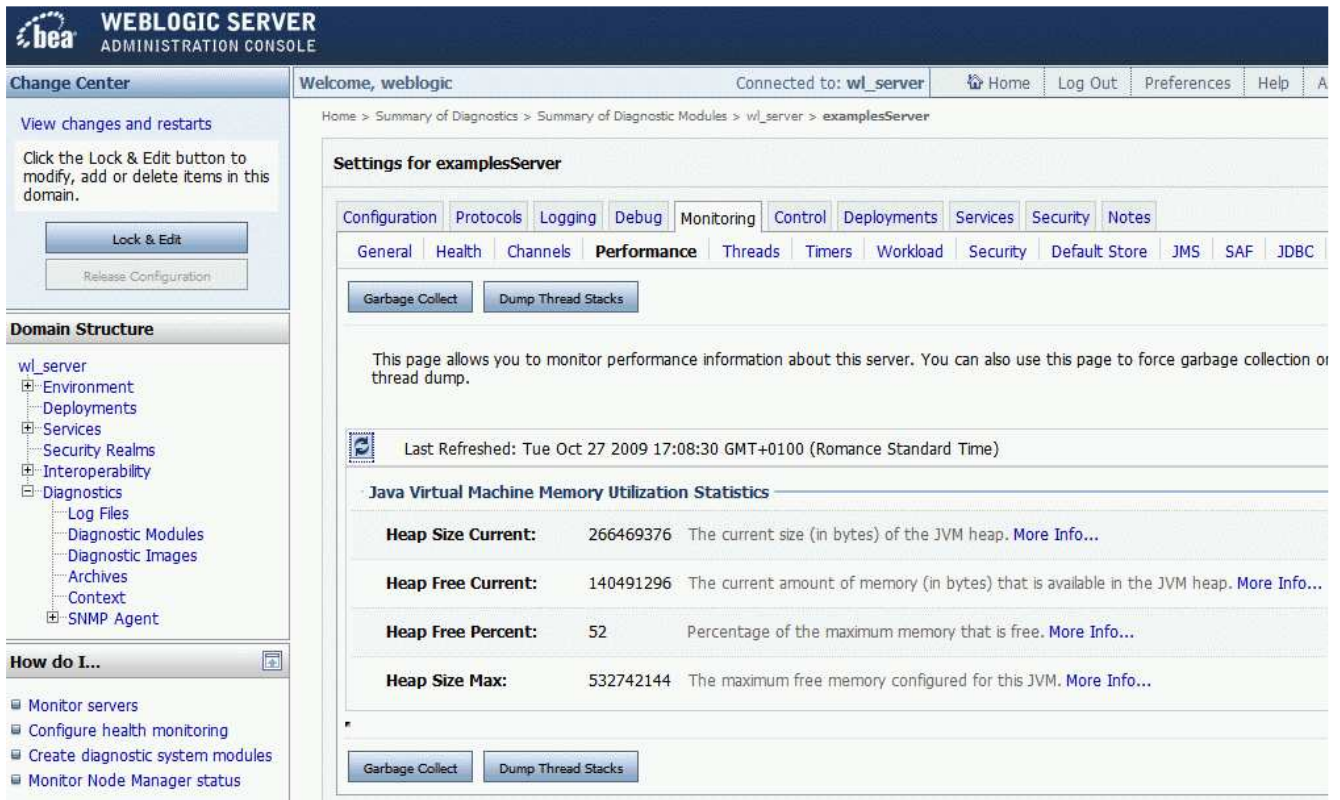
Anàlisi de la utilització de memòria d'una aplicació

Utilització de la consola del servidor d'aplicacions

Tot i ser una de les principals causes dels problemes de rendiment que es detecten en les aplicacions J2EE, especialment en els entorns de Producció, els servidors d'aplicacions no ofereixen eines d'anàlisi de la utilització de la memòria per part de les aplicacions i del mateix servidor.

Com exemple, a continuació es mostra la informació que dona la consola de Weblogic 9 sobre la utilització de memòria en el servidor:

Anàlisi d'Utilització de Memòria en Aplicacions Canigó



The screenshot shows the WebLogic Administration Console interface. The main content area is titled "Settings for examplesServer" and is under the "Performance" tab. It displays "Java Virtual Machine Memory Utilization Statistics" with the following data:

Statistic	Value	Description
Heap Size Current:	266469376	The current size (in bytes) of the JVM heap. More Info...
Heap Free Current:	140491296	The current amount of memory (in bytes) that is available in the JVM heap. More Info...
Heap Free Percent:	52	Percentage of the maximum memory that is free. More Info...
Heap Size Max:	532742144	The maximum free memory configured for this JVM. More Info...

Com es pot veure, la informació és mínima i es limita als valors totals del tamany actual / màxim / lliure del Heap i dona la possibilitat de forçar una execució del Garbage Collector.

Aquesta informació és insuficient a l'hora de determinar si la causa d'un problema de rendiment està relacionada amb la utilització de memòria i no dona cap detall de quina de les aplicacions desplegades al servidor pot ser responsable d'un consum excessiu de memòria.

Utilització d'eines de monitorització

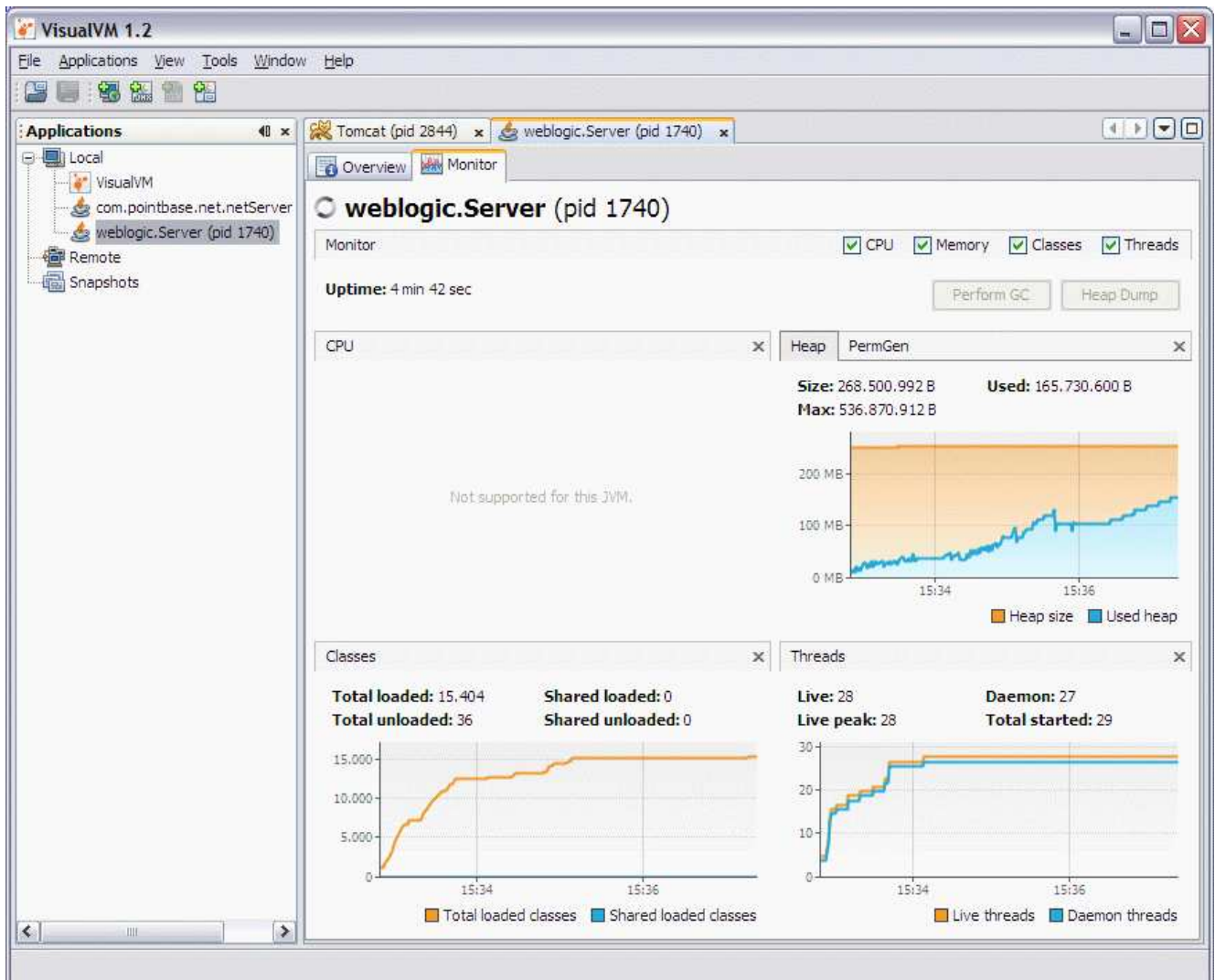
Hi ha varies eines propietàries per monitoritzar el comportament d'aplicacions Java / J2EE però com exemple de la informació que es pot obtenir, en aquest document proposem l'ús del Java VisualVM, que s'inclou de forma gratuïta amb les darreres versions del JSDK o pot descarregar-se de

<https://visualvm.dev.java.net/>

Pels exemples s'ha utilitzat la versió 1.2, descarregada de l'adreça anterior, que amplia la funcionalitat de la inclosa en el Java 6.

Aquesta eina permet realitzar un primer anàlisi del comportament del servidor d'aplicacions connectant-se directament a la màquina virtual Java que executa el Weblogic, tal com es mostra en la següent figura:

Anàlisi d'Utilització de Memòria en Aplicacions Canigó



Els Weblogic 9.x s'executen en una màquina virtual Java 5, de la que el VisualVM només pot obtenir la informació agregada del us de la memòria, el número de classes carregades i el número de threads actius.

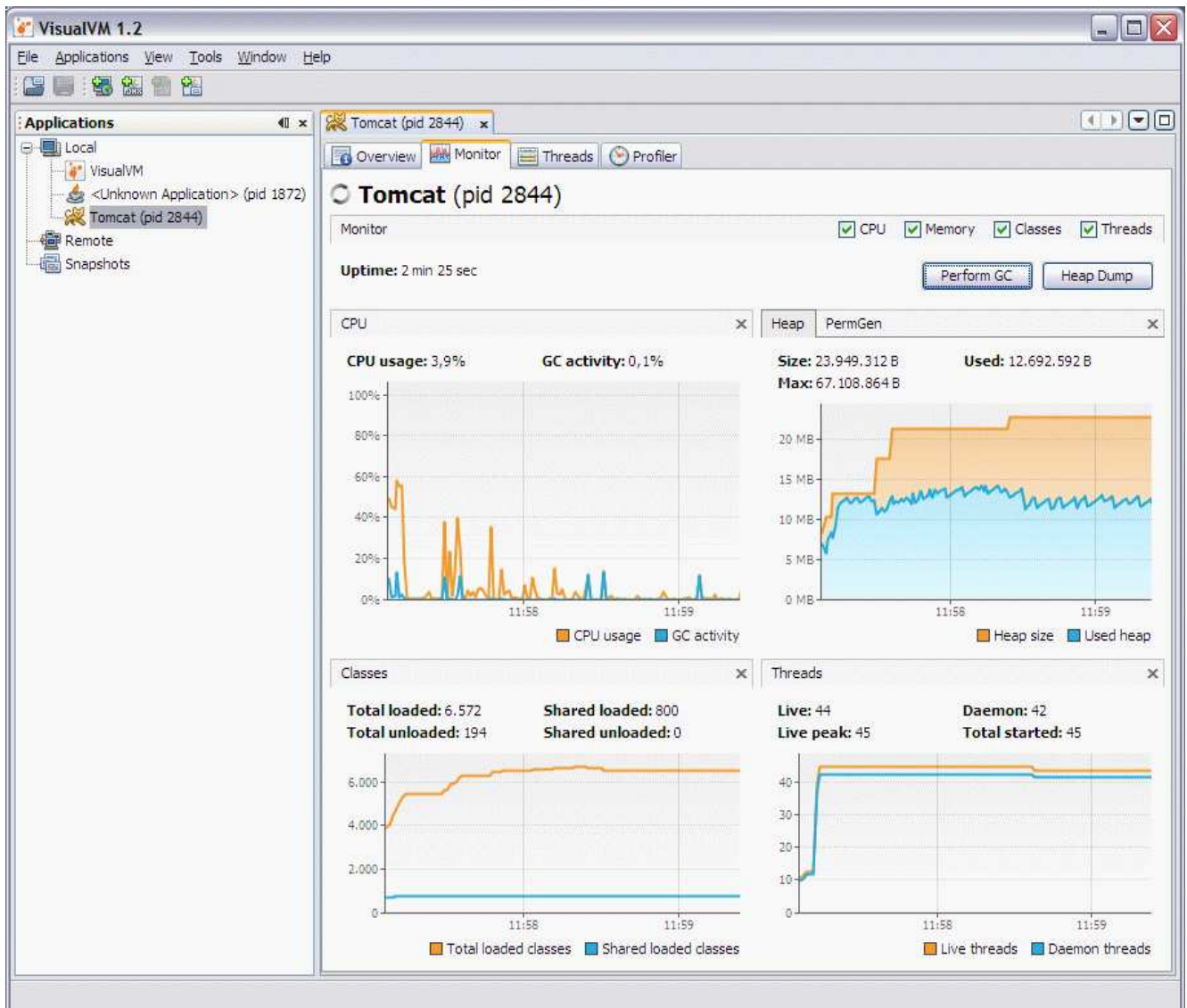
Tot i no ser una informació gaire detallada, el fet de poder veure-la en temps real mentre s'està utilitzant l'aplicació pot donar pistes sobre els possibles punts problemàtics i que requereixen un anàlisi més detallat.

Per realitzar aquest anàlisi més detallat cal executar l'aplicació (i per tant el servidor d'aplicacions) en una màquina virtual Java 6. Això és preferible fer-ho en un entorn de desenvolupament i utilitzant Tomcat com a servidor.

Al monitoritzar amb VisualVM el Tomcat executant-se en un Java 6, s'activa la monitorització de la CPU consumida, indicant quina part del consum es dedica a la execució del Garbage Collector.

S'activa també la possibilitat de forçar la execució del Garbage Collector i de realitzar un bolcat de memòria (Heap Dump) que es pot guardar en un arxiu per a un posterior anàlisi.

Anàlisi d'Utilització de Memòria en Aplicacions Canigó



Amb aquesta monitorització es poden realitzar un conjunt d'observacions. Per exemple

- Un cop carregada i "escalfada" l'aplicació (s'han compilat les JSP, s'ha accedit a les diverses funcionalitats que requereixin inicialització i càrrega de llibreries, etc.) el consum de memòria s'hauria d'estabilitzar (per un número fix de sessions obertes) i els augments haurien de ser puntuals, deguts a operacions conegudes (com la execució d'una consulta Hibernate que retorni un número elevat d'objectes, la generació d'un PDF, etc.) i la memòria s'hauria d'alliberar un cop acabada la operació.
- Un cop carregada i "escalfada" l'aplicació també s'haurien d'estabilitzar tant el número de classes carregades com el número de threads actius.
- L'augment en el consum de memòria degut a la creació d'una nova sessió concurrent hauria d'estar dintre dels límits esperats (i la memòria total disponible hauria d'estar dimensionada pel número màxim de sessions concurrents esperades). Al tancar-se la sessió s'hauria d'alliberar la memòria que consumia.
- Un consum excessiu de CPU per part del Garbage Collector indica que s'ha exhaurit la memòria disponible i és necessari ampliar-la (amb els paràmetres corresponents a l'arrancar la màquina virtual) o detectar en quin punt de l'aplicació hi ha un consum excessiu que calgui reduir.
- En el cas de que s'hagi arrancat la màquina virtual amb uns paràmetres de memòria molt grans (normalment de l'ordre de gigabytes) i si la gràfica de consum de memòria indica que no s'ha arribat al màxim, un consum excessiu de CPU per part del Garbage Collector podria indicar que la capacitat de procés de l'ordinador no és suficient per realitzar la gestió de tanta memòria / tants objectes.

Anàlisi d'Utilització de Memòria en Aplicacions Canigó

Bolcat de memòria

Si en l'anàlisi inicial s'ha identificat algun possible problema amb la utilització de la memòria, es pot realitzar un Heap Dump per analitzar-lo en més detall.

En el bolcat de memòria es pot analitzar el número d'objectes de cada classe i el tamany de memòria que ocupen, lo que permet identificar les causes d'un consum excessiu.

The screenshot shows the VisualVM 1.2 interface. The 'Applications' pane on the left shows the process tree with 'Tomcat (pid 2844)' selected. The main window displays the 'Heap Dump' for 'Tomcat (pid 2844)'. The 'Classes' tab is selected, showing a table of classes with their instance counts and sizes.

Class Name	Instances [%]	Instances	Size
net.gencat.ctti.canigo.services.web.struts.DefaultFormDisplayResolver	5 (0%)	5	40 (0%)
net.gencat.ctti.canigo.services.web.struts.DispatchActionSupport	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.ExtendedDelegatingTilesRequestProcessor	1 (0%)	1	32 (0%)
net.gencat.ctti.canigo.services.web.struts.FormDisplayResolver	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.SpringBindingActionForm	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.SpringBindingActionForm\$SpringBindingAwar...	1 (0%)	1	20 (0%)
net.gencat.ctti.canigo.services.web.struts.action.ActionSupport	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.helper.BindingHelper	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.menu.AcegiPermissionsAdapter	1 (0%)	1	12 (0%)
net.gencat.ctti.canigo.services.web.struts.menu.AcegiPermissionsFilter	1 (0%)	1	8 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.configuration.ConfigurationTag	3 (0%)	3	141 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.ActionImageTag	1 (0%)	1	270 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.FormTag	9 (0%)	9	2349 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.SubmitTag	22 (0%)	22	5412 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.AjaxValidableTag	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.AjaxValidableTagHelper	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.CheckboxFieldTag	1 (0%)	1	295 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.FieldValidatorTag	1 (0%)	1	85 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.FieldValidatorTagHelper	0 (0%)	0	0 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.FileFieldTag	1 (0%)	1	312 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.IconErrorTag	1 (0%)	1	49 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.LabelTag	6 (0%)	6	318 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.OptionsFieldTag	3 (0%)	3	240 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.PagedSelectFieldTag	1 (0%)	1	375 (0%)
net.gencat.ctti.canigo.services.web.struts.taglib.forms.fields.PasswordFieldTag	1 (0%)	1	303 (0%)

Un cop identificades quines classes son les responsables del consum excessiu, l'anàlisi dels objectes concrets (i en especial de les referències a ells des d'altres objectes) permeten analitzar una causa molt comú de pèrdues de memòria ocasionades per mantenir referències a objectes que ja no es fan servir impedit així que siguin eliminats pel Garbage Collector.

Referències

<https://visualvm.dev.java.net/>

<http://java.dzone.com/announcements/visualvm-12-great-java>