**Author:** Eric Nicosia

**Version:** 44

**Date:** 2012.06.20

**Pôle Architecture technique**

**Utilisation du profilage java sur une application web**

0

Table of ContenTs

[1 Introduction 3](#_Toc256000000)

[2 pré-requis 4](#_Toc256000001)

[3 Mode opératoire pour une mise en place manuelle 5](#_Toc256000002)

[4 Les différents paramètres et leurs définition 6](#_Toc256000003)

[5 Annexes 10](#_Toc256000004)

[5.1 Activation de JMX sur le serveur Tomcat 10](#_Toc256000005)

[5.2 Exemple de fichier aop.xml 11](#_Toc256000006)

[5.3 Exemple de fichier context.xml 12](#_Toc256000007)

[5.4 Exemple de fichier context-profiling.xml 13](#_Toc256000008)

[5.5 Exemple de fichier log4j.xml pour séparer la log de profiling de la log applicative 14](#_Toc256000009)

[6 Utilisation de la console JMX 18](#_Toc256000010)

* [Introduction](#scroll-bookmark-1)
* [pré-requis](#scroll-bookmark-2)
* [Mode opératoire pour une mise en place manuelle](#scroll-bookmark-3)
* [Les différents paramètres et leurs définition](#scroll-bookmark-4)
* [Annexes](#scroll-bookmark-5)
  + [Activation de JMX sur le serveur Tomcat](#scroll-bookmark-6)
  + [Exemple de fichier aop.xml](#scroll-bookmark-7)
  + [Exemple de fichier context.xml](#scroll-bookmark-8)
  + [Exemple de fichier context-profiling.xml](#scroll-bookmark-9)
  + [Exemple de fichier log4j.xml pour séparer la log de profiling de la log applicative](#scroll-bookmark-10)
* [Utilisation de la console JMX](#scroll-bookmark-11)

Introduction

Le profilage java utilise les capacités d'AspectJ et du Load Time Weaving pour intercepter l'exécution des méthodes java lors de leur appel dans l'application et de Spring qui fournit la librairie spring-tomcat-weaver-2.5.5.jar qui permet le weaving dans Tomcat en intervenant au moment du chargement des classes.

pré-requis

La mise en place du profiling java nécessite les pré-requis suivants :

* L'application doit utiliser le framework Spring pour cette procédure.
* Librairie spring-tomcat-weaver-2.5.5.jar pour Tomcat5 et Spring 2.x (disponible ici : <http://sunic.mediapost.fr:9998/archiva/browse/org.springframework/spring-tomcat-weaver/2.5.5>)
  + ou librairie spring-instrument-tomcat-3.1.3.RELEASE.jar pour Tomcat 7.x et java 7 (disponible ici : <http://sunic.mediapost.fr:9998/archiva/browse/org.springframework/spring-instrument-tomcat/3.1.1.RELEASE> )
* Libraire aspectjweaver-1.6.12.jar (pour java 5 ou 6), aspectjweaver-1.7.1 (pour java 7)
* Fichier aop.xml (Définition des aspects et méthodes à profiler)
* Fichier context.xml (Définition du classLoader Spring permettant le load Time weaving dans Tomcat)
* Fichier context-profiling.xml (définition des beans Spring du profiler java et exposition de ceux-ci en JMX pour permettre le paramétrage à chaud des options de profiling)

Mode opératoire pour une mise en place manuelle

* Arrêter le serveur Tomcat
* Copier la librairie spring-tomcat-weaver-2.5.5.jar dans le répertoire server/lib du Tomcat 5.x hébergeant l'application
  + ou, pour Tomcat 7, copier la librairie spring-instrument-tomcat-3.1.3.RELEASE.jar dans le répertoire lib du Tomcat hébergeant l'application.
* Copier le fichier context.xml dans le répertoire META-INF de l'application.
* Copier le fichier aop.xml dans le répertoire classes/META-INF de l'application. (Attention à ne pas confondre avec le répertoire META-INF qui se trouve à la racine de l'application)
* Copier le fichier context-profiling.xml, contenant le paramétrage du profiling, dans le répertoire WEB-INF/classes
* mettre à jour le fichier de contexte (Spring) de l'application en ajoutant l'import du fichier context-profiling.xml
* vérifier la présence de la librairie aspectjweaver en version 1.6.2 au moins dans le répertoire WEB-INF/lib de l'application  
  Si une version antérieure est présente, la remplacer par une version >= 1.6.2 pour Java5 ou 1.6.5 pour Java 6
* Pour pouvoir utiliser la console JMX, assurez-vous que celui-ci est bien activé lors du lancement de Tomcat. (Voir procédure ici)

Les différents paramètres et leurs définition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Paramètre | Depuis version | Type | Valeur par défaut | Définition | Exemple |
| active | 1.0.0 | boolean | false | Active ou désactive le loggueur des méthodes interceptées | <property name="active" value="false"/> |
| minMillisForLogging | 1.0.0 | long | 0 | Spécifie la durée d'exécution minimale pour que la méthode soit loguée. Valeurs à 0 par défaut (toutes les méthodes seront loguées) | <property name="minMillisForLogging" value="0" /> |
| countExecution | 1.0.1 | boolean | false | Indique si le profiler doit mémoriser les temps d'exécution de chaque méthode interceptée pour pouvoir ensuite fournir des statistiques. (temps min, max, moyenne des temps, etc...) | <property name="countExecution" value="true" /> |
| filterId | 1.1.0 | boolean | false | Indique au profiler d'activer ou non le filtre de session. | <property name="filterSession" value="true" /> |
| logCalledMethods | 1.0.1 | boolean | false | Indique si le profiler doit logger tous les appels de méthode dans la log. Attention, si cette option est activée, la log peut devenir très rapidement volumineuse. | <property name="logCalledMethods" value="true" /> |
| idToFilter | 1.1.0 | String | null | Identifiant de session qui servira de filtre pour les méthodes interceptées par le profiler. Attention, cela ne fonctionnera qu'en conjonction avec l'utilisation du filtre de requête ProfContextFilter qui positionnera dans le contexte de chaque requête les données nécessaires au profiler. | Utilisation de JMX. Vide par défaut. |
| showMemory | 1.0.1 | boolean | false | A chaque éxecution de méthode, affiche les informations de mémoire de la JVM (libre, totale, utilisée) avant et après exécution. Vivement déconseillé en environnement de production ! | <property name="showMemory" value="false" /> |
| showCallLevel | 1.0.1 | boolean | false | Active l'affichage du niveau d'appel de la méthode. | <property name="showCallLevel" value="true" /> |
| IndentCallLevel | 1.0.1 | boolean | true | Si cette option est activé, le tag contenant le niveau d'appel sera indenté en fonction du niveau d'appel. Cela facilite la lecture de la log car toutes les méthodes d'un même niveau d'appel sont alignées. | <property name="indentCallLevel" value="true" /> |
| showDetailedInfos | 1.0.1 | boolean | true | Si cette option est activée, le profiler rapportera le nom complet de la méthode intercepté, en incluant les packages et la classe. Dans le cas contraire, seule la classe et le nom de méthode seront rapportés. | <property name="showDetailedInfos" value="true" /> |
| showThreadNumber | 1.0.4 | boolean | true | Affiche, pour les méthodes capturées, le numéro du Thread dans lequel elles se sont exécutées. Utile pour pouvoir suivre des traitements multiThreadés. |  |

Annexes

Activation de JMX sur le serveur Tomcat

Pour que la console JMX puisse accéder aux informations JMX lorsque le serveur Tomcat est actif, il faut s'assurer que les options suivantes sont bien paramétrées dans le fichier setenv.sh, présent dans le répertoire bin du serveur Tomcat

Tabelle 1 setenv.sh

|  |
| --- |
| CATALINA\_OPTS="$CATALINA\_OPTS -Dcom.sun.management.jmxremote" CATALINA\_OPTS="$CATALINA\_OPTS -Dcom.sun.management.jmxremote.ssl=false" CATALINA\_OPTS="$CATALINA\_OPTS -Dcom.sun.management.jmxremote.authenticate=false" # Utilisez un numero de port disponible sur la machine cible en lieu et place de "PORT", par exemple 9081 CATALINA\_OPTS="$CATALINA\_OPTS -Dcom.sun.management.jmxremote.port=PORT" CATALINA\_OPTS="$CATALINA\_OPTS -Djava.rmi.server.hostname=HOSTNAME\_DU\_SERVER" export CATALINA\_OPTS # Controle de la generation des fichiers dump d aspectJ JAVA\_OPTS="$JAVA\_OPTS -Dorg.aspectj.dump.directory=/tmp" JAVA\_OPTS="$JAVA\_OPTS -Dorg.aspectj.weaver.Dump.exception=false" export JAVA\_OPTS |

|  |
| --- |
| On doit utiliser CATALINA\_OPTS et non pas JAVA\_OPTS. CATALINA\_OPTS n'est utilisé qu'au démarrage du serveur, alors que JAVA\_OPTS est utilisé au démarrage et a l’arrêt. Si on utilise JAVA\_OPTS, lors de l’arrêt du Tomcat (shutdown.sh), la JVM va tenter de recréer un port d'écoute JMX, déjà alloué lors du démarrage, et l’arrêt du serveur échouera. |

|  |
| --- |
| Par défaut, aspectj est configuré pour créer des dumps (fichiers ajcore) en cas d'exception. Afin d'éviter d'avoir des fichiers de dump en production, répartit n'importe où en fonction du répertoire de lancement, il est préférables d'ajouter deux paramètres au fichier setenv.sh  JAVA\_OPTS="$JAVA\_OPTS -Dorg.aspectj.dump.directory=/tmp" JAVA\_OPTS="$JAVA\_OPTS -Dorg.aspectj.weaver.Dump.exception=false"  Le premier indique a aspectJ de créer les fichiers de dump dans le répertoire /tmp Le deuxième indique à aspectJ de ne pas créer de dump en cas d'exception. Le premier paramètre est néanmoins nécessaire car avant la version 1.7.x d'aspectJ, les dump seront tout de même générés. |

Exemple de fichier aop.xml

|  |
| --- |
| <!DOCTYPE aspectj PUBLIC  "-//AspectJ//DTD//EN" "http://www.eclipse.org/aspectj/dtd/aspectj.dtd">  <aspectj> <!-- Afin de weaver les packages javax (meme chose pour java.\*) il est obligatoire d'ajouter l'option Xset:weaveJavaxPackage (ou weaveJavaPackage...) -->  <weaver options="-verbose -showWeaveInfo -Xset:weaveJavaxPackages=true -Xset:weaveJavaPackages=true">   <include within="fr.mediapost.profiling.mdpprof.aspect.ConcreteProfiler"/> <include within="fr.mediapost.commedia..service..\*"/> <include within="fr.mediapost.commedia..servlet..\*"/>  <include within="fr.mediapost.commedia..integration..\*"/>  <include within="fr.mediapost.commedia..backingbean..\*"/>  <include within="fr.mediapost.commedia.commun.util..\*"/>  <include within="fr.mediapost.www.schema.logistique.webservices..\*"/>  <include within="org.apache.axis.client.Call"/>  <include within="com.oracle.xmlns..\*.\*"/>  <include within="fr.mediapost.commedia.metier.commun.domaine.utilitaires.SQLUtils..\*"/>  <exclude within="fr.mediapost.commedia.metier.commun.service.impl.ThreadLocalInformationUtilisateurContext" />    </weaver>    <aspects>      <concrete-aspect name="fr.mediapost.profiling.mdpprof.aspect.ConcreteProfiler"  extends="fr.mediapost.profiling.mdpprof.aspect.AbstractProfiler">  <pointcut name="thePointcut" expression="execution( \* fr.mediapost.commedia..service..\*.\*(..))   || execution( \* org.apache.axis.client.Call.invoke(..))  || execution( \* fr.mediapost.commedia..backingbean..\*.\*(..))   || execution( \* fr.mediapost.commedia..integration..\*.\*(..))   || execution( \* fr.mediapost.www.schema.logistique.webservices..\*.\*(..))   || execution( \* fr.mediapost.commedia..servlet..\*.\*(..))   || execution( \* fr.mediapost.commedia.metier.commun.domaine.utilitaires.SQLUtils.\*(..))  "/>    </concrete-aspect>  <!-- Of the set of aspects declared to the weaver  use aspects matching the type pattern "" for weaving. -->  <include within="fr.mediapost.profiling.mdpprof.aspect.ConcreteProfiler"/>  </aspects>    </aspectj> |

Exemple de fichier context.xml

Tabelle 2 context.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--  \* Copyright 2002-2005 The Apache Software Foundation.  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License. --> <!--  Context configuration file for the Tomcat Manager Web App   antiResourceLocking is needed for proper redeploy under Windows. -->  <Context antiResourceLocking="true" antiJARLocking="true" reloadable="true">  <Loader loaderClass="org.springframework.instrument.classloading.tomcat.TomcatInstrumentableClassLoader" useSystemClassLoaderAsParent="false"/> </Context> |

Exemple de fichier context-profiling.xml

Tabelle 3 context-profiling.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation=" http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd ">    <bean id="AbstractProfiler" class="fr.mediapost.profiling.mdpprof.aspect.AbstractProfiler"  abstract="true">  </bean>    <bean id="ConcreteProfiler" class="fr.mediapost.profiling.mdpprof.aspect.ConcreteProfiler"  factory-method="aspectOf">  <property name="active" value="false"/>  <property name="minMillisForLogging" value="0" />  <property name="minMillisForWarning" value="4000" />  <property name="countExecution" value="true" />   <property name="showMemory" value="false" />  <property name="showCallLevel" value="true" />  <property name="indentCallLevel" value="true" />  </bean>   <bean class="org.springframework.jmx.export.MBeanExporter">  <property name="beans">  <map>  <entry key="Commedia Etablissement:service=JavaProfiler" value-ref="ConcreteProfiler" />  </map>  </property>  </bean>  <context:load-time-weaver aspectj-weaving="on" weaver-class="org.springframework.instrument.classloading.ReflectiveLoadTimeWeaver"/> </beans> |

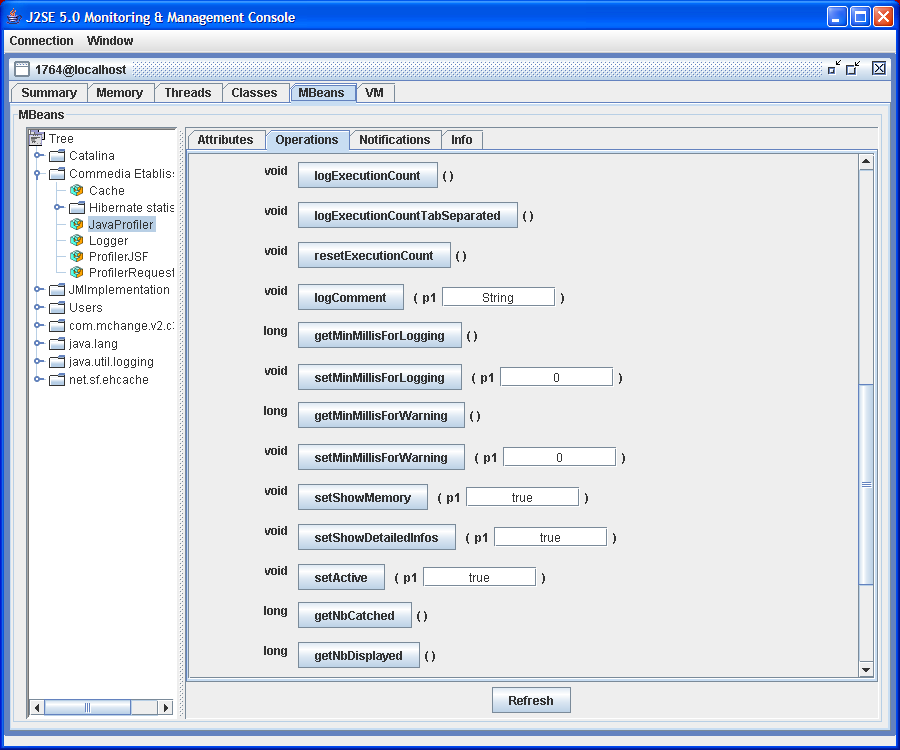
Exemple de fichier log4j.xml pour séparer la log de profiling de la log applicative

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">  <log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"  debug="false">   <!-- ============================== -->  <!-- Append messages to the console -->  <!-- ============================== -->  <appender name="consoleAppender" class="org.apache.log4j.ConsoleAppender">  <layout class="org.apache.log4j.PatternLayout">  <!-- The default pattern: Date Priority [Category] Message\n -->  <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c] %m%n" />  <!--<param name="ConversionPattern" value="%d %x %p [%c] - %m%n" />-->  </layout>  </appender>   <!-- ================================= -->  <!-- Preserve messages in a local file -->  <!-- ================================= -->  <appender name="fileAppender" class="org.apache.log4j.RollingFileAppender">  <param name="File" value="${catalina.home}/logs/commedia.log" />  <param name="Append" value="true" />  <param name="MaxFileSize" value="20MB" />  <param name="MaxBackupIndex" value="10" />  <layout class="org.apache.log4j.PatternLayout">  <param name="ConversionPattern" value="%d %x %-5p [%c] %m%n" />  </layout>  </appender>    <!-- Loggeur specifique pour le profiling -->  <!-- ================================= -->  <!-- Preserve messages in a local file -->  <!-- ================================= -->  <appender name="profilingAppender" class="org.apache.log4j.RollingFileAppender">  <param name="File" value="${catalina.home}/logs/profConsole-profiling.log" />  <param name="Append" value="true" />  <param name="MaxFileSize" value="20000KB" />  <param name="MaxBackupIndex" value="10" />  <layout class="org.apache.log4j.PatternLayout">  <param name="ConversionPattern" value="%-5p %m%n" />  </layout>  </appender>    <!-- Loggeur specifique pour le profiling de methodes -->  <!-- ================================= -->  <!-- Preserve messages in a local file -->  <!-- ================================= -->  <appender name="profilingMethodsAppender" class="org.apache.log4j.RollingFileAppender">  <param name="File" value="${catalina.home}/logs/profConsole-profiling-methods.log" />  <param name="Append" value="true" />  <param name="MaxFileSize" value="20000KB" />  <param name="MaxBackupIndex" value="10" />  <layout class="org.apache.log4j.PatternLayout">  <param name="ConversionPattern" value="%-5p %m%n" />  </layout>  </appender>  <logger name="mdpprof" additivity="false">  <level value="INFO" />  <appender-ref ref="profilingAppender" />  </logger>     <logger name="mdpprof.java" additivity="false">  <level value="INFO" />  <appender-ref ref="profilingMethodsAppender" />  </logger>     <logger name="mdpprof.java.methods" additivity="false">  <level value="INFO" />  <appender-ref ref="profilingMethodsAppender" />  </logger>     <logger name="mdpprof.java.statistics" additivity="false">  <level value="INFO" />  <appender-ref ref="profilingAppender" />  </logger>   <!-- ================ -->  <!-- Limit categories -->  <!-- ================ -->   <category name="org">  <priority value="ERROR" />  </category>   <category name="org.springframework">  <priority value="INFO" />  </category>   <category name="org.springframework.transaction.jta">  <priority value="DEBUG" />  </category>    <root>  <priority value="INFO" />  <appender-ref ref="consoleAppender" />  <appender-ref ref="fileAppender" />  </root>  </log4j:configuration> |

Utilisation de la console JMX

La console JMX permet de piloter en temps réel l'aspect de profiling, d'en modifier le paramétrage et de logguer des données, sans avoir à redémarrer l'application sur laquelle il est activé.

Ci-après une copie d'écran de quelques commandes disponibles par le biais de la console JMX pour l'aspect de profiling Java.



On peut modifier l'ensemble des paramètres positionnés au lancement de l'application, mais également déclencher à la demande la log des informations collectées durant l'exécution de l'application.

La principale méthode à connaitre et utiliser est la méthode logExecutionCountTabSeparated(), qui fournira, dans la log, le récapitulatif de l'ensemble des méthodes interceptés, avec pour chacune d'elle les statistiques de temps d'exécution, exprimées en nanoseconde (10-9 secondes). Un exemple de sortie est fournit ci-après. (Les paramètres executionCount et active doivent être positionnés à true pour que la collecte d'information se fasse).

Tabelle 4 Résultat de la commande logExecutionCountTabSeparated()

|  |
| --- |
| INFO Count CumulativeTimeNanos CumulativeUnitTimeNanos MinTimeNanos MinUnitTimeNanos MaxTimeNanos maxUnitTimeNanos MeanTimeNanos MeanUnitTimeNanos Method INFO 6 187433 161812 24075 21007 59819 51524 31238 26968 execution(protected void org.apache.axis.client.Stub.setAttachments(org.apache.axis.client.Call)) INFO 4 6112701 6112701 1388658 1388658 1781237 1781237 1528175 1528175 execution(public static java.io.InputStream fr.mediapost.commedia.commun.util.ResourceUtils.readResource(java.lang.String)) INFO 7 44362 44362 1747 1747 31608 31608 6337 6337 execution(public void fr.mediapost.commedia.metier.article.service.sig.ZoneTarification.setCodeTranscode(int)) INFO 25 14964 14964 358 358 1838 1838 598 598 execution(public java.lang.String org.apache.axis.schema.SchemaVersion2000.getXsiURI()) INFO 6 99605 93966 13085 12354 31741 29575 16600 15661 execution(public org.apache.axis.constants.Style org.apache.axis.client.Call.getOperationStyle()) INFO 3 121648793 121648793 9690 9690 121628033 121628033 40549597 40549597 execution(private fr.mediapost.commedia.metier.commande.domaine.tarif.RemiseRow fr.mediapost.commedia.metier.commande.service.impl.ServiceDetailTarifaireImpl.buildRemise(com.oracle.xmlns.Commande.XXMD\_AJUSTEMENT\_REC\_TYPE)) INFO 440 249988 249988 475 475 2030 2030 568 568 execution(public boolean org.apache.axis.encoding.SerializationContext.shouldSendXSIType()) INFO 127 1868399 1809445 12893 12462 29443 29007 14711 14247 execution(public boolean fr.mediapost.commedia.web.etablissement.backingbean.compte.CreationCompteBean.isCoclicoValid()) INFO 12 11811 11811 692 692 3055 3055 984 984 execution(public void org.apache.axis.Message.setMessageType(java.lang.String)) INFO 73 41729 41729 379 379 2181 2181 571 571 execution(public fr.mediapost.commedia.metier.sig.domaine.arbre.Noeud fr.mediapost.commedia.web.etablissement.backingbean.sig.SessionArbreBean.getNoeud()) INFO 1 94871919 5358053 94871919 5358053 94871919 5358053 94871919 5358053 execution(public java.util.List fr.mediapost.commedia.metier.accordcadre.service.impl.ServiceAccordCadreImpl.findAccordCadreForCommande(fr.mediapost.commedia.metier.client.domaine.Client, java.util.Date, boolean, boolean)) INFO 1 203492 176458 203492 176458 203492 176458 203492 176458 execution(public java.lang.String fr.mediapost.commedia.metier.sig.service.impl.ServicePointVenteImpl.getNumeroDepartement(java.lang.String)) INFO 20 15184 15184 541 541 2568 2568 759 759 execution(public boolean fr.mediapost.commedia.web.etablissement.backingbean.AbstractFluxClientDataHolder.isModeCreation()) INFO 24 2495063 292150 91825 11049 122491 22620 103960 12172 execution(public void org.apache.axis.message.SAXOutputter.startPrefixMapping(java.lang.String, java.lang.String)) INFO 12 460886 153323 30995 9985 77572 25338 38407 12776 execution(public org.apache.axis.message.SOAPBodyElement org.apache.axis.message.SOAPEnvelope.getFirstBody()) INFO 15 29313 29313 418 418 15865 15865 1954 1954 execution(public fr.mediapost.commedia.metier.sig.service.IServiceZone fr.mediapost.commedia.metier.sig.service.ServiceFactory.getServiceZone()) INFO 1 3789 3789 3789 3789 3789 3789 3789 3789 execution(public javax.faces.component.UISelectOne fr.mediapost.commedia.web.etablissement.backingbean.compte.CreationCompteBean.getFacturationPaysBinding()) INFO 22 2676480 333054 75172 13468 962034 30423 121658 15138 execution(public javax.xml.namespace.QName org.apache.axis.encoding.TypeMappingDelegate.getTypeQName(java.lang.Class)) INFO 3 141448 137626 41338 40156 55759 54463 47149 45875 execution(public boolean fr.mediapost.commedia.web.etablissement.backingbean.simulation.AssistantSimulationBean.getZoneDejaExistante()) INFO 3 4469 4469 894 894 2520 2520 1489 1489 execution(public boolean fr.mediapost.commedia.web.etablissement.backingbean.client.SessionClientBean.isEmailRenseigne()) INFO 17 656330 539853 27133 24315 48887 41873 38607 31756 execution(public java.lang.String fr.mediapost.commedia.web.etablissement.backingbean.UserBean.getUserName()) INFO 3 3811 3811 662 662 2226 2226 1270 1270 execution(public java.lang.String fr.mediapost.commedia.metier.campagne.service.impl.ServiceConstanteCampagneImpl.getCodeChampNbRepiquage()) INFO 2 17618 17618 2111 2111 15507 15507 8809 8809 execution(public boolean fr.mediapost.commedia.metier.campagne.service.impl.ServiceCycleVieImpl.buildMepLot(fr.mediapost.commedia.metier.campagne.domaine.Campagne)) INFO 12 163345 148340 2373 2373 125175 110170 13612 12361 execution(public static fr.mediapost.commedia.metier.commun.service.data.UniteMesure fr.mediapost.commedia.metier.commun.service.data.UniteMesure.valueOf(java.lang.String)) INFO 42 37427363 1513147 6971 6712 23516286 773229 891127 36027 execution(public void org.apache.axis.encoding.ser.BeanDeserializer.onStartElement(java.lang.String, java.lang.String, java.lang.String, org.xml.sax.Attributes, org.apache.axis.encoding.DeserializationContext)) INFO 20 26794 26794 637 637 11422 11422 1339 1339 execution(private void org.apache.axis.types.NonNegativeInteger.checkValidity()) INFO 31 19792 19792 409 409 3378 3378 638 638 execution(public fr.mediapost.commedia.metier.structure.integration.IUniteSelectDAO fr.mediapost.commedia.metier.structure.integration.DAOFactory.getUniteSelectDAO()) INFO 6 313290 87453 26249 11804 174951 25118 52215 14575 execution(public org.apache.axis.Handler org.apache.axis.configuration.FileProvider.getGlobalRequest()) INFO 8 10897 10897 746 746 4698 4698 1362 1362 execution(public void fr.mediapost.commedia.metier.campagne.service.impl.util.front.ChampsValorisesArticleBean.setListChampValorise(java.util.List)) INFO 2 3997 3997 692 692 3305 3305 1998 1998 execution(public void fr.mediapost.commedia.web.etablissement.backingbean.recherche.client.RechercheClientBean.setTraitement(fr.mediapost.commedia.web.commun.backingbean.recherche.ITreatmentAfterSearch)) INFO 5 5288 5288 605 605 2445 2445 1057 1057 execution(public fr.mediapost.commedia.metier.commande.domaine.admin.ListeProduit fr.mediapost.commedia.web.etablissement.backingbean.CommonDataHolderBean.getFormule()) INFO 1 4990 4990 4990 4990 4990 4990 4990 4990 execution(public void fr.mediapost.commedia.web.etablissement.backingbean.sig.ContexteEvaluableBuilderBean.setSessionArbre(fr.mediapost.commedia.web.etablissement.backingbean.sig.SessionArbreBean)) INFO 1 3636 3636 3636 3636 3636 3636 3636 3636 execution(public void fr.mediapost.commedia.web.etablissement.backingbean.sibp.SibpBean.setSessionClient(fr.mediapost.commedia.web.etablissement.backingbean.client.SessionClientBean)) INFO 32 19653 19653 415 415 2105 2105 614 614 execution(public org.apache.axis.MessageContext org.apache.axis.encoding.SerializationContext.getMessageContext()) INFO 6 102405 97506 13951 13175 25314 24460 17067 16251 execution(public void org.apache.axis.SimpleTargetedChain.PivotIndicator.invoke(org.apache.axis.MessageContext)) |

|  |
| --- |
| Les données étant logguées avec une tabulation comme séparateur, il suffit de les copier/coller dans un document excel pour pouvoir les manipuler facilement. |