

Aide en ligne du monitoring JavaMelody

Le monitoring JavaMelody est un outil de mesure et de statistiques sur le fonctionnement réel d'une application selon l'usage qui en est fait par les utilisateurs.

Le monitoring est en grande partie basé sur des statistiques de requêtes et sur des courbes d'évolution. Il permet ainsi d'améliorer les applications en recette et en production et d'aider à :

- factueliser les temps de réponse moyens et les nombres d'exécutions
- prendre des décisions quand les tendances sont mauvaises
- optimiser sur la base des temps de réponse les plus pénalisants
- trouver les causes à l'origine des temps de réponse
- vérifier l'amélioration réelle après des optimisations

Le rapport html de JavaMelody est optimisé pour [Firefox](#), [Chrome](#) ou MSIE8 (MSIE7 non recommandé).



Documentations

[Java SE Reference at a Glance](#)
[Troubleshooting Guide for Java SE 6](#)
[Troubleshooting Guide for Java SE 5](#)
[Monitoring and Managing Java SE 6](#)
[Java SE 6 Performance](#)
[Memory Management](#)



Synthèse

Dans la page du monitoring, la synthèse présente des courbes d'évolution sur différentes valeurs de mesures.

Ces mesures sont effectuées à un instant t, par exemple toutes les minutes. Chaque courbe suit l'évolution d'une valeur de mesure sur la période plus ou moins large choisie par l'intermédiaire des liens au-dessus de la synthèse : le jour, la semaine, le mois, l'année ou une période personnalisée. Dans chaque courbe sont suivies et indiquées en chiffres la valeur moyenne en vert et le maximum en bleu. Les courbes sont persistées : un redémarrage du serveur d'application n'a pas d'influence sur elles hormis un trou dans les mesures.

Chaque courbe peut être affichée en grand et redimensionnée en cliquant sur celle-ci dans la synthèse.

Les courbes présentées sont :

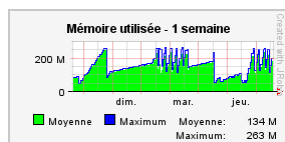
- la mémoire java utilisée, entre 0 et le maximum paramétré par Xmx dans la configuration du serveur
- le pourcentage cpu utilisé par le processus java, entre 0 et 100 même s'il y a plusieurs coeurs ou plusieurs processeurs
- le nombre de sessions http (ou nombre d'utilisateurs connectés)
- le nombre de threads actifs (ou nombre de requêtes http en cours)
- le nombre de connexions jdbc actives (ou nombre de requêtes sql en cours)
- le nombre de connexions jdbc utilisées (ou nombre de transactions sql ouvertes); dans le cas où il n'y a pas de datasource, c'est en fait le nombre de connexions jdbc ouvertes dans le pool de connexions
- pour chacun des compteurs de requêtes (http, sql et éventuellement ejb, spring, guice, interfaces, actions struts, pages jsp) :
 - le nombre de hits par minute (ou nombre d'exécutions de requêtes par minute)
 - le temps moyen en millisecondes
 - le pourcentage d'erreurs systèmes (ces valeurs pour un compteur représente une moyenne pour la dernière période de mesure)
- et aussi dans 'Autres courbes', des données qui sont globales pour la JVM ou pour l'OS :
 - si Tomcat ou JBoss, le nombre de threads actifs dans tout le serveur, le nombre d'octets reçus par minute et le nombre d'octets envoyés par minute par le serveur (le nombre d'octets reçus est souvent 0 pour la plupart des webapps)
 - le pourcentage cpu du ramasse-miettes pour la JVM
 - le nombre de threads pour la JVM
 - le nombre de classes chargées pour la JVM
 - la mémoire utilisée hors tas pour la JVM
 - la mémoire physique utilisée pour l'OS
 - l'espace swap utilisé pour l'OS
 - l'age moyen des sessions http actuellement valides en minutes
 - si linux : la charge système et le nombre de fichiers ouverts pour l'OS

Les heures dans les graphiques pour le jour dépendent de l'heure du serveur et du fuseau horaire du serveur (timezone, défini par l'OS ou spécifiquement pour le serveur d'application).

Le lien ' Actualiser' permet de rafraîchir la page et les courbes. Le lien ' PDF' affiche tout le rapport en format PDF pour Adobe Reader.

Si un serveur de collecte est utilisé pour afficher le monitoring de plusieurs serveurs en ferme ou en cluster, la courbe de mémoire est la somme des mémoires sur les différents serveurs, mais le pourcentage cpu est celui entre 0 et 100 pour tous les serveurs, et le nombre de sessions http est la somme des sessions sur les différents serveurs, de même que pour le nombre de threads actifs, le nombre de connexions jdbc actives ou utilisées, et les hits par minute ou le temps moyen en millisecondes.

Courbe mémoire : cas d'utilisation d'une application utilisée par intermittence (jour/nuit par exemple)



La courbe de la mémoire augmente rapidement quand l'application est utilisée, et elle augmente doucement mais augmente tout de même quand l'application n'est pas utilisée. Une courbe avec des pentes, comme dans l'exemple ci-dessus, est donc classique même si l'application n'est pas utilisée.

Dans tous les cas et selon la nécessité, la mémoire est finalement libérée d'un coup par le ramasse-miette (GC majeur) et revient à un niveau bas, avant de remonter plus ou moins vite. Si la résolution paramétrée est suffisamment fine, la courbe montre également les GC mineurs en petites dents de scie entre les GC majeurs. Ces GC mineurs libèrent également la mémoire mais en moins grande quantité que les GC majeurs. Il est possible de forcer un GC majeur par l'action ' Exécuter le ramasse-miette' dans la partie ' Informations systèmes' du rapport.

Courbe mémoire : cas d'utilisation d'une saturation mémoire

Si la courbe mémoire augmente jusqu'au maximum et si l'application ne peut en libérer avec le ramasse-miette, le serveur provoque éventuellement des erreurs "OutOfMemoryError: Java heap space" pour interrompre le traitement et libérer si possible la mémoire.

Tant que la mémoire est insuffisante, le cpu reste à 100% car le ramasse-miette s'exécute en permanence. Cela peut provoquer de très fortes lenteurs et un quasi-blocage du serveur d'application.

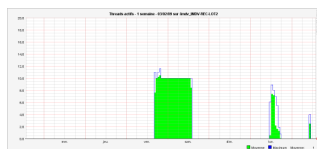
Les solutions peuvent être d'augmenter la mémoire java maximum (paramètre Xmx au lancement du serveur, 64 Mo par défaut), et éventuellement la mémoire physique du serveur, ou bien d'optimiser si possible la consommation mémoire de l'application.

Mémoire Perm Gen: cas d'utilisation d'une saturation perm gen

Si la valeur de la mémoire perm gen augmente jusqu'au maximum et si l'application ne peut en libérer, le serveur provoque éventuellement des erreurs "OutOfMemoryError: PermGen space" pour interrompre le traitement. Cela peut bloquer toutes les fonctionnalités dans l'application.

Les solutions peuvent être d'augmenter la mémoire PermGen maximum (paramètre XX:MaxPermSize au lancement du serveur), et éventuellement la mémoire physique du serveur, ou bien d'optimiser si possible le nombre de classes chargées par l'application.

Courbes de threads actifs et connexions jdbc actives : cas d'utilisation d'un plateau (requêtes longues)



Quand une application est peu ou pas utilisée, les courbes des threads actifs et connexions jdbc actives restent à 0. En effet, la plupart des requêtes sont heureusement courtes et la plupart des mesures (effectuées toutes les 2 minutes par exemple) sont prises sans requêtes en cours, sauf 1 ou 2 de temps en temps visibles sous forme d'un pic.

Par contre, quand il y a une forme de plateau dans la courbe des threads actifs comme dans l'exemple ci-dessus, cela veut dire qu'il y a une ou plusieurs requêtes qui sont longues (plusieurs minutes ou plusieurs heures) et qui potentiellement saturent le serveur d'application ou la base de données.

Pour trouver la cause :

- Si le plateau sur les threads actifs est reproduit à l'identique sur les connexions jdbc actives, cela indique que la cause est dans une ou plusieurs requête(s) sql et non dans l'exécution de code java.
- Si le(s) plateau(x) se poursuivent actuellement, vous pouvez trouver les requêtes http (et sql) en cause dans la partie '🔍 Requête en cours' du rapport.
- Si le(s) plateau(x) sont terminés, vous pouvez trouver les statistiques des requêtes http (et sql) en cause, dans les tableaux détaillés de '📊 Statistiques http' (et sql), sauf si le serveur d'application a été arrêté avant la fin des requêtes en cause.
- L'ouverture du "drill-down" dans le détail d'une requête http peut vous permettre de trouver les façades ou requêtes sql en cause.
- Si une base de données Oracle est utilisée, l'affichage du plan d'exécution dans le détail de la requête sql peut vous aider à trouver la cause du temps de cette requête (si l'affichage du plan d'exécution indique qu'une table "plan_table" doit être créée : vous pouvez utiliser le script @\$ORACLE_HOME/rdbms/admin/catplan.sql en tant qu'utilisateur SYS si v10g ou plus et sinon @\$ORACLE_HOME/rdbms/admin/utlxplan.sql et "grant all on plan_table to public" si v9i ou moins).
- Il peut être utile de vérifier avec le monitoring l'état de la mémoire et de l'utilisation cpu du serveur d'application. Il peut être utile également de vérifier par vos propres moyens l'état de la mémoire, du cpu et des accès disques de la base de données.

Courbes des connexions jdbc utilisées : cas d'utilisation d'une fuite de connexions jdbc

Les connexions jdbc sont généralement mutualisées dans un pool de connexions par l'intermédiaire d'une datasource jdbc, ce qui permet en particulier de ne pas réouvrir les connexions vers la base de données en permanence et ce qui permet également de définir un nombre maximum de transactions pouvant s'exécuter simultanément par instance du serveur d'application. Il est possible dans JavaMelody de visualiser les 2 courbes de connexions jdbc actives et de connexions jdbc utilisées. La courbe de connexions jdbc actives indique le nombre de connexions en cours d'exécution de requêtes sql. Et la courbe de connexion jdbc utilisées indique le nombre de connexions ouvertes et utilisées dans des transactions avec la base de données mais qui n'exécutent pas forcément de requêtes sql. Si jamais l'application monitorée contient une fuite de connexions jdbc se manifestant dans certaines conditions, alors des connexions sont inutilement utilisées et restent indisponibles en dehors du pool de connexions. La courbe de connexions utilisées montrera éventuellement que leur nombre augmente sans redescendre au minimum tel que configuré dans le pool de connexions de la datasource et ce même si le serveur n'a aucune activité, la nuit par exemple. Une fois au maximum, des erreurs surviennent dans le pool de connexions indiquant qu'il n'y a plus de connexion disponible. Si une fuite de connexions jdbc est suspectée, les heures et les stack traces indiquant où les connexions ont été ouvertes peuvent être affichées par le lien '🔗 Connexions jdbc ouvertes', dans la partie '📊 Informations systèmes' du rapport. (Ce lien est disponible à la condition que le paramètre 'system-actions-enabled' ait été positionné à 'true'). Une lecture attentive du code correspondant à ces stack traces confirmera ou non la fuite de connexions jdbc.

La courbe de connexions jdbc utilisées et ces stack traces d'ouverture sont toutefois peu utiles si un pool de connexions est utilisé directement en passant par un driver jdbc sans qu'il y ait de datasource, car dans ce cas on ne visualise dans cette courbe et ces stack traces que les ouvertures de connexions par le pool et non les utilisations réelles de ces connexions par l'application.

📊 Statistiques

Les statistiques d'un compteur présentent les statistiques des requêtes exécutées sur le serveur d'application. Il existe 7 compteurs de requêtes aujourd'hui :

- 📊 Statistiques http
- 📊 Statistiques sql (jdbc)
- 📊 Statistiques ejb (si façades ejb3 de JavaEE 5+)
- 📊 Statistiques spring (si façades spring)
- 📊 Statistiques guice (si façades guice)
- 📊 Statistiques d'interfaces façades (sans ejb3 ni spring ni guice)
- 📊 Statistiques d'actions struts
- 📄 Statistiques des pages jsp (ou plus précisément, des exécutions de include ou forward sur `HttpServletRequest.getRequestDispatcher`, c'est-à-dire les pages jsp en général)

Pour chaque compteur, une requête apparaît dans les statistiques quand elle est terminée; pour connaître les requêtes non terminées voir la partie '🔍 Requêtes en cours'. Comme pour les courbes, les statistiques des requêtes sont celles sur la période plus ou moins large choisie par l'intermédiaire des liens au-dessus des courbes : le jour (depuis 0h), la semaine, le mois ou l'année. Ainsi il est possible de voir les statistiques pour la seule journée en cours, ou par exemple pour une version de l'application déployée depuis un mois sans les statistiques des versions précédentes. Et pour chaque compteur, le rapport présente une synthèse des statistiques pour toutes les requêtes (global), des statistiques dépassant le seuil d'alerte de niveau 1 (warning) et de celles dépassant le seuil d'alerte de niveau 2 (severe). Les détails des statistiques pour chaque requête peuvent être affichés en cliquant le lien '+' Détails'.

Chaque statistique de requête indique :

- le contenu de la requête
- une courbe d'évolution des temps moyens de cette requête en tooltip selon la même période que les statistiques; cette courbe peut être agrandie en cliquant sur le contenu de la requête
- le pourcentage de temps cumulé (temps moyen * nb de hits) par rapport aux autres requêtes
- le nombre de hits (ou nombre d'exécutions)
- le temps moyen en millisecondes
- le temps maximum en millisecondes
- l'écart-type (ou déviation standard) entre les temps d'exécution : s'il est élevé, les temps sont très variés, s'il est faible les temps sont rapprochés entre eux
- le pourcentage de temps cpu cumulé (temps cpu moyen * nb de hits) par rapport aux autres requêtes
- le temps cpu moyen en millisecondes
- le pourcentage d'erreurs systèmes

et si il s'agit des statistiques http, ejb, spring, guice, interfaces, actions struts ou jsp :

- la taille moyenne en kilo-octets du flux de réponse (seulement pour http)
- le nombre moyen de hits sql, par exécution de requête http, ejb...
- le temps moyen passé à exécuter des requêtes sql, par exécution de requête http, ejb...

Les temps moyens, les temps maximums et les temps cpu moyens sont cumulatifs : par exemple, les temps http moyens incluent les temps ejb moyens et les temps sql moyens ; de plus, les temps cpu moyens pour http incluent les temps cpu moyens pour ejb et pour sql. Mais les temps d'attente comme 'sleep' ou attente pour I/O ne consomment pas de cpu, donc ils sont inclus dans les temps moyens mais pas dans les temps cpu moyens.

Comme tous les tableaux du rapport, les tableaux de statistiques sont triables par ordre ascendant ou descendant en cliquant sur les entêtes de colonnes. Les statistiques sont persistées pour chaque compteur : un redémarrage du serveur d'application n'a pas d'influence sur elles. Si un serveur de collecte est utilisé pour afficher le monitoring de plusieurs serveurs en ferme ou en cluster, les statistiques des compteurs sont globales pour tous les serveurs.

Remarque : Dans MS Windows (sauf Vista), la résolution de l'horloge utilisée est d'environ 16 ms (soit 0, soit 16, soit 32 ms). Ainsi le temps d'une exécution n'est pas précis en-dessous de 16 ms avec Windows. Cela est en partie corrigé par l'utilisation d'une moyenne quand il y a eu suffisamment d'exécutions. Mais cela n'est pas grave puisqu'en général, les requêtes problématiques feront beaucoup plus que 16 ms.

🚨 Statistiques d'erreurs systèmes http

Les statistiques d'erreurs systèmes http présentent les erreurs qui remontent jusqu'au filtre http du monitoring, soit sous forme d'exceptions lancées par l'application par l'intermédiaire d'une servlet, soit sous forme d'un code d'erreur http (404 "not found" ou 500 "internal server error" par exemple, [liste complète](#)). Ces statistiques indiquent la liste des 250 erreurs systèmes les plus fréquentes avec en particulier le nombre d'occurrences de chaque erreur sur la période choisie, ainsi que le temps moyen de la requête pour chaque erreur. Seule l'erreur la plus fréquente est affichée au départ, les autres erreurs étant affichées en cliquant le lien '+' Détails'. Les dates, heures, utilisateurs et requêtes http complètes des 100 dernières erreurs sont affichées en cliquant le lien '+' Dernières erreurs'. En cliquant sur le nom d'une erreur, il est possible de voir pour cette erreur la stack-trace java de l'erreur quand il s'agit d'une exception java. Ces statistiques d'erreurs permettent d'améliorer la fiabilité de l'application selon son utilisation réelle en production.

📊 Statistiques de logs d'erreurs systèmes

Les statistiques de logs d'erreurs systèmes présentent les logs 'warning' et 'error' inscrits par l'application (les librairies log4j d'apache et java.util.logging du jdk sont toutes les deux supportées). Ces statistiques indiquent la liste des 500 logs d'erreurs systèmes les plus fréquentes avec le nombre d'occurrences de chaque erreur sur la période choisie. Seule le log d'erreur le plus fréquent est affiché au départ, les autres logs étant affichés en cliquant le lien '+' Détails'. Les dates, heures, utilisateurs et le cas échéant requêtes http complètes des 100 derniers logs d'erreurs sont affichées en cliquant le lien '+' Dernières erreurs'. En cliquant sur le nom d'un log d'erreur, il est possible de voir pour ce log la stack-trace java de l'erreur quand une exception java a été inscrite avec le log. Ces statistiques de logs d'erreurs permettent également d'améliorer la fiabilité de l'application selon son utilisation réelle en production.

🔍 Requêtes en cours

Les requêtes en cours présentent à l'instant de génération du rapport les exécutions de requêtes non terminées. L'arbre des requêtes http, éventuellement ejb, spring et/ou sql est indiqué avec pour chacune des requêtes le temps déjà écoulé, le temps cpu écoulé, le nombre de requêtes sql déjà exécutées et le temps sql de ces requêtes. Les statistiques des requêtes sont rappelées avec les requêtes en cours : temps moyen, temps cpu moyen, hits sql moyens et temps sql moyen. Cela permet de comparer les requêtes en cours avec les statistiques moyennes. La stack-trace java courante est indiquée par un tooltip sur le thread (si jdk 1.6). Seule la requête la plus longue est affichée au départ, les autres sont affichées en cliquant sur le lien '+' Détails'.












Informations système

Les informations système indiquent à l'instant de génération du rapport des informations sur le serveur java, son état et sur le système d'exploitation du serveur.

Les principales informations sont affichées au départ : mémoire java utilisée, nombre de sessions http, nombre de threads actifs, nombre de connexions jdbc actives et nombre de connexions jdbc utilisées. Ce sont d'ailleurs des valeurs qui seront mesurées et utilisées pour les courbes.

Les autres informations, comme la version du serveur, du système d'exploitation ou de la base de données ou la mémoire du système d'exploitation, sont affichées en cliquant sur le lien '+ Détails'.

Dans cette partie des informations systèmes, des liens donnent accès aux actions systèmes :

-  'Exécuter le ramasse-miette', pour forcer une libération de mémoire
-  Si jdk 1.6, 'générer un heap dump', pour exporter tout le contenu de la mémoire dans un fichier du répertoire temporaire sur le serveur, que vous pouvez ouvrir ensuite avec [VisualVM](#) du JDK ou avec [Eclipse MAT](#)
-  Si jdk 1.6, 'voir l'histogramme mémoire', pour afficher le nombre d'instances en mémoire pour chaque classe java
-  'Invalider les sessions http', pour forcer la déconnexion de tous les utilisateurs
-  'Voir les sessions http', pour voir les attributs, les tailles sérialisées (généralement supérieure aux tailles en mémoire), la provenance et éventuellement l'utilisateur de chaque session (si authentification par JavaEE)
-  'Voir le descripteur de déploiement' : fichier web.xml de l'application
-  'MBeans': management beans comme dans jconsole ou jvisualvm, avec la configuration et des données techniques sur le serveur d'application et sur la JVM. Les valeurs sont consultables mais pas modifiables et les opérations ne peut être exécutées.
-  'Voir les processus de l'OS' : liste des processus du système d'exploitation (linux avec [ps](#) ou windows avec [tasklist](#)) avec pour chaque processus : utilisateur, mémoire et cpu
-  'Arbre JNDI' : Parcours du contexte JNDI, par exemple pour trouver le chemin et le nom d'une datasource JDBC.
-  Pour aider à diagnostiquer une éventuelle fuite de connexions jdbc, les 'connexions jdbc ouvertes' donnent la liste des stack traces où ces connexions ont été ouvertes.
-  'Base de données' : Informations et statistiques sur la base de données (si postgresql, mysql ou oracle), comme par exemple les requêtes sql en cours ou si oracle, les requêtes les plus longues en temps cumulé avec indication du temps cpu et du coût élémentaire (en buffer gets). Cette fonction affichera une erreur si l'utilisateur configuré dans l'application pour accéder à la base de données n'a pas les droits nécessaires. Sous oracle, la requête suivante accorde ce droit: "grant select any dictionary to monutilisateur".

Si un serveur de collecte est utilisé pour afficher le monitoring de plusieurs serveurs en ferme ou en cluster, les informations systèmes de chaque serveur sont affichées (agrégées pour ce qui est de la liste des sessions et de l'histogramme mémoire) et les actions s'exécutent successivement sur chacun des serveurs.

Threads

Ce tableau affiche simplement la liste de tous les threads dans le serveur, avec pour chacun la priorité, l'état et la stack-trace en tooltip (si jdk 1.6) entre autres.

Caches de données

Les caches de données affichent la liste des caches dans l'application java à condition que la librairie [ehcache](#) soit utilisée pour ceux-ci.

Pour chacun des caches, des statistiques sont indiquées : le nombre d'objets en mémoire et sur disque à cet instant, l'efficacité du cache mémoire (pourcentage depuis le démarrage du serveur des demandes satisfaites par le cache en mémoire par rapport à celles satisfaites par le cache sur disque), l'efficacité du cache au global (pourcentage des demandes satisfaites par le cache en mémoire ou sur disque par rapport à toutes les demandes y compris celles non satisfaites pour les données non présentes en cache). La configuration de chaque cache est également rappelée.

Un bouton permet de purger tous les caches.

Jobs

Les jobs affichent la liste des tâches (ou batchs) dans l'application java à condition que la librairie [quartz](#) soit utilisée pour ceux-ci.

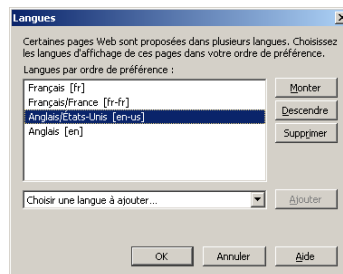
Pour chacun des jobs, le temps moyen, la dernière erreur avec stack-trace pour la période sélectionnée, ainsi que l'heure de la dernière exécution et de la prochaine exécution sont indiqués. Comme pour les requêtes http, les statistiques pour les jobs sont affichées avec temps moyen, temps cpu moyens, hits sql et temps sql pour la période sélectionnée.

Des boutons permettent de mettre en pause ou de redémarrer chacun des jobs ou tous les jobs.

Si vous avez des jobs quartz ordonnancés avec Spring mais qu'aucun job n'apparaît dans le rapport, avez-vous ajouté la propriété "exposeSchedulerInRepository" comme indiqué dans le guide utilisateur ?

Langue

Le monitoring est affiché en français ou en anglais selon la langue préférée de votre navigateur web. Si le monitoring est en langue anglaise au lieu d'être en langue française, il faut corriger la langue préférée. Par exemple, dans [Firefox](#), ouvrir le menu et la boîte de dialogue "Outils, Options, Contenu, Langue ... Choisir" et placer "Français [fr]" en premier comme ceci :



Crédits icônes

[Silk](#), [mini and flag icons](#) (Creative Commons)

[Tango icons](#) (GPL)