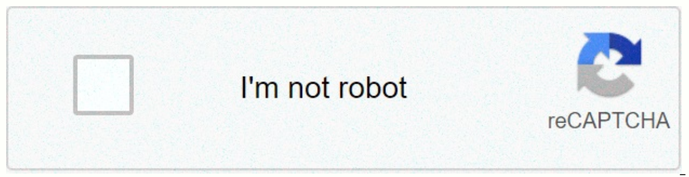


Continue

# Java heap size for tomcat

Apache Tomcat is a Java servlet container, and is performed on a Java Virtual Machine, or JVM. Tomcat uses the Java serviette specification to run servlets generated by requests, often using JSP pages, so dynamic content can be generated much more efficiently than with a CGI script. If you want to run a high-performing installation of Tomcat, taking some time to learn about your JVM is essential. In this article, we learn how Tomcat and the JVM interact, look at some of the different JVMs that are available, explain how to vote the JVM for better performance, and provide information about some of the tools available for monitoring your JVM's performance. How Tomcat interacts with the JVMUtilizing servlets allows the JVM to handle each request within a separate Java thread, since each servlet is in fact a standard Java class, with special elements that allow it to respond to HTTP requests. Tomcat's main function is to pass HTTP requests to the correct components to serve them, returning the dynamically generated results to the correct location after the JVM processes it. If the JVM can't efficiently serve the requests Tomcat makes to it, Tomcat's performance will be negatively affected. Choosing the right JVMThere is many JVMs to choose from, and Tomcat runs on many of them, from open source projects such as Sun Microsystem's HotSpot or Apache Harmony, to own JVMs like Azul VM.Despite the wide range of available JVM flavors, the majority of Tomcat users favor Sun Microsystem's HotSpot JVM, because its just-in-time composition and adaptive optimization features are particularly suitable for efficiently handling Tomcat's servlet requests. So, for the majority of tomcat users, HotSpot is the JVM to use. However, if you're attracted a feature that's specific to a certain JDK, there's nothing wrong with installing Tomcat on two different JVM's and managing some benchmarks to see which solution is best for your needs. In the end, it's a balancing act. Select the JVM that provides the best balance of performance and features for your site. How to configure Tomcat's Default JVM preferencesOnsOnce you decide on a JVM and install it on your server, configuring Tomcat to run on them, is a very simple process. Simply edit catalina.sh, found in Tomcat's bin folder, and change the JAVA_HOME environment variable to the folder of your chosen JVM's JDK. When you restart Tomcat, it will run on your new JVM. Optimizing your JVM for best performanceThe better your JVM performs, the better your installation of Tomcat will perform. It's as simple as that. Getting the most out of your JVM is a matter of setting up its settings to suit your actual performance needs as closely as possible. Update your JVM to the latest version, establish some benchmarks so you have a way to quantify any changes you make, and then get down to business. Effective memory managementThe most important thing to consider when voting your JVM for Tomcat performance how to avoid wasting memory and draining your server's power to process requests. Certain automated JVM processes, such as garbage collection and memory reallocation, can chew through memory if they occur more frequently than necessary. You can make sure these processes occur only when they need to by using the JAVA_OPTS -Xmx and -Xms switch to control how JVM handles its heap memory. If your JVM insults garbage collection too often, use the -Xmx switch to start the JVM with a higher maximum heap memory. This will free up CPU time for the processes you really care about. To get even more out of this change, you can include the -Xms switch. This switch makes the JVM's initial heap of memory size equal to the maximum assigned memory. This means the JVM will never have to process more memory, an expensive process that can end up power you want to be used to serve incoming requests. If your web applications can handle a lower garbage collection throughput, you can also experiment with the -Xincgc switch, enbyting incremental garbage collection. This means that rather than being in place to perform garbage collection tasks, the JVM will perform garbage collection in small phases. It can be difficult to determine the most balanced configuration for the needs of your website. Fortunately, there's an easy way to capture data on how your JVM handles garbage collection. Simply use the -verbose: gc switch to generate logs that you can use to help you get to the best solution. Configure threadsNext, let's look at the way your JVM threads handle. There are two types of Java threads - green and native. Native threads are scheduled by your Os, while green threads are completely managed within the userspace of your Java Virtual Machine. If you support JVM both, you should try both models to determine the best choice for your website. In general, native threads offer the best performance, especially if you run a lot of I/O bound applications (which is very likely, since you run Tomcat). However, green threads perform native threads in some specific areas, such as synchronization and thread activation. Try both and see which option gives you the biggest performance boost. Managing your JVMTuning for performance is not a limited process. Usage situations change over time, and problems that are not immediately clear can expose themselves over a longer period of time. There are a number of tools available to help you keep an eye on your JVM's performance. One of the most convenient solution is VisualVM, a tool that's packaged with the JDK, and you can provide good performance statistics. Other commonly used JVM monitoring tools included in the SDK include console, jps, and jstack. Run regular tests on your JVM to make sure its configuration still suits your needs, and you can be sure that your always will perform at their best! ×Sorrie to InterruptCSS Error This topic provides instructions for setting the Java Java Machine (JVM) memory configuration on Windows, Linux, and OSX operating systems. What do the JVM memory settings do? LabKey Server is Java web application that runs on Tomcat. The Tomcat server is run within a Java Virtual Machinex (JVM). This JVM that controls the amount of memory available to LabKey Server. LabKey recommends configuring the Tomcat web application to have a maximum Java Hope size of at least 2GB for a test server, and at least 4GB for a production server. <CATALINA_HOME>: Installation location of the Apache Tomcat Web Server. If you follow our recommended folder configuration, the location will be (where #.#.## the specific version is installed): <LABKEY_ROOT>/apps/apache-tomcat-#.#.## On Linux or OSX, you may have also created a symbolic link/usr/local/tomcat to this location. Locate the tomcat.service file, typically located on /etc/systemd/system/tomcat.service Open the file and edit the CATALINA_OPTS parameter. For a production server, use: -Xms4g -Xmx4g -XX:-HeapDumpOnOutOfMemoryError For a test server, use: -Xms2g -Xmx2g -XX:-HeapDumpOnOutOfMemoryError Find the JSVC service script. On Linux servers it is usually in the /etc/init.d folder and named either tomcat or tomcat # On OSX servers it can in /usr/local/jsvc/Tomcat# .sh Open the JSVC service script using your favorite editor Find the line for setting up CATALINA_OPTS and add one of the following settings within the dual quotes. For a production server, use: -Xms4g -Xmx4g -XX:-HeapDumpOnOutOfMemoryError For a Test Server, usage: -Xms2g -Xmx2g -XX:-HeapDumpOnOutOfMemoryError Save the file Restart LabKey Server The starting script is located in <CATALINA_HOME>/bin/catalina.sh Make the Start script using your favorite text editor Above the line reading # OS specific support. $var _must_ to either true or false.: add one of the following: For a production server, use: JAVA_OPTS=$JAVA_OPTS -Xms4g -Xmx4g -XX:-HeapDumpOnOutOfMemoryError For a test server, usage: JAVA_OPTS=$JAVA_OPTS -Xms2g -Xmx2g -XX:-HeapDumpOnOutOfMemoryError Save the file Restart LabKey Server Tomcat is usually started as a service on Windows, and this includes a dialog to configure the JVM. The maximum total memory allocation is configured in its own text box, but other settings are configured in the common JVM options using the Java command-line parameters syntax. If you changed the name of the LabKey Windows service, you must use Method 2. Open Windows Explorer. Go to <CATALINA_HOME>the /bin folder. Find and run the file tomcat #w.exe (true # is the Tomcat version number). Start this as an administrator by right-clicking the .exe selecting a file and selecting Run as an administrator. The command will open an app window. If it produces an error that says: The specified service does not exist on server, then go to Method 2. Go to the Java tab in the new window. In the Java Options box, scroll to the bottom of the properties, and set the following property:</CATALINA_HOME> </CATALINA_HOME> </LABKEY_ROOT> </CATALINA_HOME> </CATALINA_HOME> Change the Initial memory pool to 2000 MB for a test server or 4000 MB for a production server. Change the Maximum memory pool to the same value. Click OK Restart Lab Key Server. You will need to use this method if you have adjusted the name of the LabKey Windows service. Open a command prompt. (Click Start, type cmd, and press the Enter key.) Navigate to the <CATALINA_HOME>bin folder. Execute the following command, which makes appropriate changes if you are running a different version of Tomcat. This example is for Tomcat 9: tomcat9w.exe // ES/LabKeyTomcat9 The command will open an app window. If it produces an error that says: The specified service does not exist on the server, then see the note below. Go to the Java tab in the new window. In the Java Options box, scroll to the bottom of the properties, and set the following property: -XX:-HeapDumpOnOutOfMemoryError Change the initial memory pool to 2GB for a test server or 4GB for a production server. Change the Maximum memory pool to the same value. Clicking the OK button Restart LabKey Server NOTE: The text to the // ES // must exactly match the name of the Windows service used to start/stop your LabKey Server. You can determine the name of your Windows service by taking the following actions: Open the Windows Services Pane. (Click Start, type Services, and press the Enter key.) In the Services pane, find the entry for LabKey Server. It can be called something like Apache Tomcat or LabKey Double click on the service to open the Properties dialog box. In the command above, replace the text LabKeyTomcat # with the text running next to ServiceName in the Properties dialog. Back to installation: Tomcat configuration </CATALINA_HOME>