Builds failing with OutOfMemoryErrors

Builds running out of memory?

As your project grows, and you use new tools to either build or analyze your code, you will inevitably exceed the memory settings which your JVM provides by default. This is especially true on 64 bit JVM's since they double the size of the reference pointer. This page aims to show you how to increase the memory available to your build process.

Heap or Permgen?

There are two OutOfMemoryErrors which people usually encounter. The first is related to heap space: `java.lang.OutOfMemoryError: Heap space` When you see this, you need to increase the maximum heap space. You can do this by adding the following to your JVM arguments `-Xmx200m` where you replace the number 200 with the new heap size in megabytes.

The second is related to PermGen: `java.lang.OutOfMemoryError: PermGen space`. When you see this, you need to increase the maximum Permanent Generation space, which is used for things like class files and interned strings. You can do this by adding the following to your JVM arguments `-XX:MaxPermSize=128m` where you replace the number 128 with the new PermGen size in megabytes.

Various Build Tools

This section demonstrates how to set these JVM switches depending on the kind of build you are running.

Maven2/3 Project Type

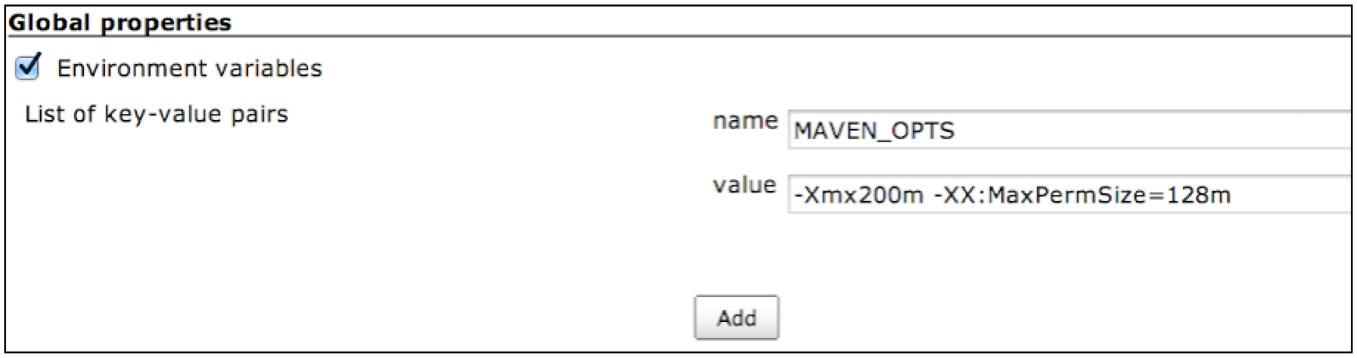
If you are using a Maven2/3 project type, you can set the `-Xmx` or `-XX:MaxPermSize` in your Jenkins global configuration. To do this, navigate to Jenkins global configuration page (Manage Jenkins -> Configure System) and look for the **Maven Project Configuration**. Add the necessary JVM settings to the *Gl obal MAVEN_OPTS* field and press save. Subsequent Maven2/3 builds will use these new settings.

You can adjust the MAVEN_OPTS individually per job by navigating to the job, then clicking "configure" on the job menu, then under the "Build" section clicking advanced, then specifying a setting for the MAVEN_OPTS option.

Freestyle projects with Maven Build Steps

If you have a Freestyle project with a "Invoke Top Level Maven Targets" build step, you can add the appropriate JVM switches for a given build by clicking the **Advanced** button on the build step and entering the JVM arguments in the *JVM Options* field.

Alternatively, you can affect all free style Maven build steps by adding a MAVEN_OPTS global environment variable in the Jenkins global configuration. To do this, click **Manage Jenkins**, then **Configure System**. In the **Global properties** section, click the **Environment Variables** checkbox, then add a new environment variable called MAVEN_OPTS with the value set appropriately:



Global properties

☑ Environment variables

List of key-value pairs

name: MAVEN_OPTS

value: -Xmx200m -XX:MaxPermSize=128m

Gradle build steps

You can set the `-Xmx` or `-XX:MaxPermSize` by adding a GRADLE_OPTS global environment variable in the Jenkins global configuration. To do this, click **Manage Jenkins**, then **Configure System**. In the **Global properties** section, click the **Environment Variables** checkbox, then add a new environment variable called GRADLE_OPTS with the value set appropriately, similar to the screen shot above regarding MAVEN_OPTS

Ant build steps

For Ant steps, there is no global environment variable; you must set the Ant options in each individual build step. In the configuration for the build, find the **I nvoke Ant** step, click **Advanced**, and enter the options into the **Java Options** box.