Configuring Tomcat To Avoid Errors

Most issues with XNAT that have root causes in Tomcat configuration relate to various issues in memory usage and configuration. If you're seeing issues that are not covered on this wiki page, you can submit a question to the XNAT discussion group or check out the following resources:

- The Apache Tomcat site
- Stack Overflow

Heap Errors

If your heap memory is configured too low (or not configured at all, which uses the JVM default heap size), you may see an error like this:

```
Horrible Exception: java.lang.OutOfMemoryError: Java heap space
```

The “horrible exception” verbiage is Tomcat-specific, but the rest of the error is a standard message from the JVM when the heap space blows out. This is easily handled, but the method for fixing it depends on the platform on which you’re running Tomcat:

- On Linux, add a file named **setenv.sh** into the **bin** folder of your Tomcat installation. Tomcat runs this script on server start up when present. This gives you an opportunity to set values in the environment. When you create this script (or modify an existing instance), pay attention to script ownership and file permissions: you’ll want these settings to be the same as the other scripts in your Tomcat **bin** folder. Add something like this to your script:

  ```
  CATALINA_OPTS=$CATALINA_OPTS -Xms512m -Xmx1024m
  ```

- Another option on some Linux and Tomcat platforms is to modify the **tomcat.conf** configuration file to include these options on service start-up. There is usually already a line configuring the JAVA_OPTS setting, although it may be commented out. You can uncomment and/or set the appropriate values on this line of the configuration file.

Once you've made one of these configuration changes, you'll need to restart Tomcat for the changes to go into effect.

> ⚠ As of XNAT 1.7, we have discontinued support for Windows-hosted instances of XNAT. If you are running Windows, we recommend running XNAT inside a virtual machine.

About these options:

The **`-Xms`** option sets the starting heap space available to the JVM. Setting too large a minimum value can slow Tomcat start-up time, as well as take physical and virtual memory space from other processes on your server, with no guarantee that your process will ever use it. That said, this can be a valuable performance enhancement when XNAT is initializing. Try different values for this to help speed XNAT start-up without degrading the performance of other services and applications on your server.

The **`-Xmx`** option sets the maximum heap space available to the JVM. If you're getting a Java heap space error, this is really the figure that you’re interested in. Increasing this number increases the maximum amount of memory that Tomcat can consume while running. The default value for this depends on the version of the JVM you're using, the platform on which you're running, and the physical memory available on the hardware. However, the default value is often too small for computationally intensive applications like XNAT, so try increasing the value of this setting until your server can run under production loads.

PermGen Errors

A problem that is similar to and sometimes related to heap errors is running out of PermGen space:

```
Horrible Exception: java.lang.OutOfMemoryError: PermGen space
```

> ⚠ With the release of Java 8, permanent generation was removed. If you're running Tomcat and XNAT under Java 8, the MaxPermGen option will be ignored if configured.

This is related to the JVM object allocation and garbage collection mechanism. A technical discussion of the perm gen space and tuning JVM garbage collection can be found on the Oracle Java site. Generally, though, you won't need to know all of the in-depth technical information and will just need to set an option to increase the perm gen space. You can use any of the techniques described in the previous section to set the option. The actual text for the option is:

```
-XX:MaxPermSize=256m
```

Even more than the **-Xmx** option, it's very important that you experiment with different values for this option. Too small a perm gen size can obviously lead to the out-of-memory error above. Too *large* a perm gen size can lead to significant degradation in Tomcat performance over time as objects accumulate in the perm gen pool. However, the perm gen pool relates directly to the heap, so increasing your heap size can lead to perm gen errors, necessitating changes in your perm gen configuration.

For other good tips and options for configuring your perm gen space, as well as a little background in what's going on with these settings, check out this post by Mark Kolich.