Deployment

Permalink to this page: https://cwiki.apache.org/confluence/x/5SklBg

Preface

This section of the FAQ discusses common questions related to web application deployment.

Questions

Answers

Why does tomcat 5 create context configuration files?

Tomcat 5, Tomcat 6:

Unlike tomcat 4.x, tomcat 5.x creates context configuration files for you in its `conf/[Engine name]/[Host name]` directory. This is part of the change in tomcat's configuration mechanism from version 4.x to make overall configuration more robust, flexible, and enterprise-friendly. Note, however, that this has changed the recommended deployment practices for web applications. These context configuration files are created by tomcat, but not removed by tomcat, because the user may have changed them or other files in the conf directory. The suggested practice for tomcat 5 is to place context configuration files as `META-INF/context.xml` in your webapp, and use Tomcat's Manager webapp to deploy/undeploy your applications. More details can be found here: MARC Archive

Tomcat 7 and later:

In Tomcat 7 the default behaviour has been changed to do not auto-create those context configuration files.

The recommended practice of using `META-INF/context.xml` files is still the same. Those files are discovered and processed in the same way. The difference is that they are not copied to the `conf/[Engine name]/[Host name]`directory.

This is convenient, as you do not need to care of those copied files when undeploying your application, and you do not need to care whether the `conf` directory is writeable. This change in behaviour is documented in the Migration Guide.

Why does the memory usage increase when I redeploy a web application?

That is because your web application has a memory leak.

A common issue are "PermGen" memory leaks. They happen because the Classloader (and the Class objects it loaded) cannot be recycled unless some requirements are met (*). They are stored in the permanent heap generation by the JVM, and when you redeploy a new class loader is created, which loads another copy of all these classes. This can cause `OufOfMemoryErrors` eventually.

(*) The requirement is that all classes loaded by this classloader should be able to be gc'ed at the same time.

Starting with Tomcat 6.0.25 there is a tool in the Manager webapp to help diagnose such misbehaving applications. See FAQ/Memory and MemoryLeakProtection.