Problem

How can components use static variables and Singletons? Are there Classloader issues?

Solution

Components should be written without making deployment assumptions, such as which container they run in, which other components will be collocated in that container etc. References to other components are identical for remotely or locally deployed components, thus there cannot be any interference on the component interface level.

Inside the implementation though, Java components running in the same container and thus inside the same Java VM, could normally cross-talk through static class variables, which are also the basis of the Singleton pattern (private s_instance). In the container implementations prior to ACS 5.0, Singletons were possibly shared among components in the same container. This was potentially dangerous, as it created dependencies on a particular deployment scenario.

Following other Java container frameworks (e.g. Tomcat, EJB servers), with ACS 5.0 we introduced full Makefile support for separate Java ClassLoaders for individual components (c.f. Makefile directive `COMPONENTS_JARFILES` and the ACS 5.0 release notes).

This allows the container to eliminate component interaction through class variables, as is described in http://developer.java.sun.com/developer/TechTips/2000/tt1027.html. [Link broken. This tech tip is currently also not available from http://java.sun.com/developer/JDCTechTips/ or http://blogs.sun.com/CoreJavaTechTips/]. Static variables cannot be shared among components.

Related articles

- How can more people do development with ACS on the same machine without disturbing each other?
- Which ports are used by ACS?
- Problems connecting to ACS servers on a remote machine: bad /etc/hosts
- Why does the getComponent method of ZLegacy/ACS.ContainerServices return an object of type None?
- Why are some of my print statements not showing up in the container output section of acscommandcenter?