LAMS and the JBossCache

LAMS uses the JBossCache. It was primarily chosen as it is the cache that comes with JBoss. It also ties in with Hibernate. We are using the cache as an MBean. This may have to change if we were to change application servers.

When an object is added to the cache, it is added to a node. The nodes are a tree - starting with /, then /org, then /org/lamsfoundation and so on.

To use the JBoss cache requires four pieces of configuration:

- The jboss-cache.jar and jgroups.jar need to be copied from /server/all/lib to /server/default/lib.
- The configuration file local-service.xml needs to be copied to /server/default/deploy. This is done by the copy-files task in the master build script. The definition in the script gives the MBean name as "jboss.cache:service=TreeCache"
- The cacheManager property cacheObjectName should be "jboss.cache:service=TreeCache". This value must match the name in the local-service.xml file. If the parameter is not set, then the CacheManager class will default to this value.
- The commonContext.xml file (in lams_common) must configure Hibernate and the LAMS Cache Manager to use the same name (jboss.cache:service=TreeCache) as the local-service.xml file. This has already done.
    - The cacheManager.cacheObjectName should be "jboss.cache:service=TreeCache". If the parameter is not set, then the CacheManager class will default to this value.
    - The hibernateProperties.properties.hibernate.cache.provider_class should be "org.jboss.hibernate.cache.DeployedTreeCacheProvider". This configures Hibernate to use a cache already set up in JBoss.
    - The hibernateProperties.properties.hibernate.treecache.objectName should be "jboss.cache:service=TreeCache"

If you want to change the "name" of the cache from "jboss.cache:service=TreeCache" to something else, you will need to change it in three places - once in local-service.xml and twice in commonContext.xml.

Hibernate Second Level Cache

The easiest way to use the cache is to add your objects to the region setup in the local-service.xml. This will trigger the objects to be cached in a second level Hibernate cache. The objects will be added/removed from the cache by Hibernate.

To cache an object, add <cache usage="transactional"/> to the class' Hibernate mapping. This may be one by adding the xdoclet tag '@hibernate.cache usage = "transactional"' to the class level javadoc.

To cache objects that are related to cached object via 1 to many relationship, add the <cache usage="transactional"/> to the set definition in the mapping file. This may be done by adding the xdoclet tag '@hibernate.collection-cache usage = "transactional"' to the javadoc for the collection's get() method.

The last two paragraphs probably don't make much sense. Instead, have a look at CrNode and CrNodeVersion in org.lamsfoundation.lams.contentrepository, along with their hibernate mapping files (in conf/hibernate/mappings). You will see that this both classes are marked with the cache usage and the set that defines the relationship also is marked with the cache usage.

Programmatically Adding To The Cache

The cache can be used programmatically. The class org.lamsfoundation.lams.cache.CacheManager handles putting objects in and out of the cache. To access this class, set up a reference to the bean "cacheManager" in your Spring context.

The CacheManager expects all entries in the cache to be based on the class name of the object. Please use the methods in the CacheManager to ensure your nodes follow the expected format. If you do not, then the cached objects cannot be managed via the administration page.

For an example of how it may be used, see org.lamsfoundation.lams.usermanagement.service.UserManagementService.getUserById().

Manual Cache Management

An administration page, http://localhost:8080/lams/admin/cache.do, has been set up to allow manual cache management. This lists all the current nodes in the hibernate cache.

To clear the whole cache, use the Remove option at the top level ("/"). Alternatively, you can remove one of the subnodes, leaving objects from other nodes still in the cache.

The objects created by Hibernate appear a little differently to the objects cached manually. The Hibernate objects all go into the top node of the tree. This is not a problem - the only difficulty it creates is that if you need to clear one of the Hibernate entries, you have to clear all the Hibernate entries. As this is a cache this may cause a slight performance hit, but nothing more. If this becomes a major issue, then we can improve the administration page to allow the user to remove a particular child node.

Debugging Cache Entries

To monitor the addition and removal of objects from the cache, you can turn on the cache listener. This may be done by setting UseCacheDebugListener to true in the lams.xml. This should not be turned on in production as it may cause performance issues.

If you only want to monitor the object being used by Hibernate, say to check that Hibernate is using the objects from the cache rather than disk, turn on the DEBUG level for Hibernate in log4j.xml and check hibernatelams.log. The Hibernate debugging should show you how the cache is being used.