Spring Application Context Files

The LAMS components are built using the Spring framework. It is set up so that each web-app has its own Application Context (that is, there isn't one overall context for all running software).

So each web-app will need to set up the context to include the context files for all components required for that web-app. This is done through the web.xml settings.

Location Of Context Files

Each "component" of LAMS should create at least one context file in their main package directory. For example, the content repository will have use the file /org/lamsfoundation/lams/contentrepository/applicationContent.xml.

The context files for core packages, services *and tools* should be named applicationContent.xml or <something>ApplicationContent.xml. This will make it easier to search for context files.

Each module must document the location and name of its content file in the javadoc package details. This documentation should include specify the name of bean(s) that may be accessed by other modules from the Spring context. This will usually be a single service bean.

Configuration Of Context Files

Context files will be configured using <context-param> and <listener> in each war's web.xml. Each context file required for the web-app need to be specfied.

For example, the IMS Content Package tool uses the content repository, so it needs to include both its own context file and the content repository's context file.

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                classpath:/org/lamsfoundation/lams/contentrepository/applicationContext.xml
                  classpath:/org/lamsfoundation/lams/tool/imscp/applicationContext.xml
        </param-value>
</context-param>
<listener>
        <listener-class>
                org.springframework.web.context.ContextLoaderListener
        </listener-class>
</listener>
```

Make sure any tool's applicationContext.xml file is in its own package. If the full name (path + name) of a file is the same as a file in another tool, then it may pick up the file *from the other tool*, even though the other tool's jar is not on the classpath. We cannot work out why this is happening, but it may be caused by the JBOSS unified class loader. We have tried the Tomcat setting to use the Tomcat class loader, but the problem still occurs.

Why Not ...
We cannot use the "classpath*:/applicationContext.xml" style configuration. This would load an applicationContext.xml file that is in the root of all jars on the classpath. This works, and would make things much simpler for the learning and monitoring modules. However it requires that all tool's context files are in the root of the jars, and then we end up with the wrong file being used when the tool's war is loaded. See previous point.

Why not use WEB-INF/spring/applicationContext.xml?
For tools, we could put the tool's context file in the WEB-INF directory. However putting it in the jar file allows all the context files to be easily accessed from another module should it be required. For example, the tool engine may wish to access the context file of one or more tools for testing.

It also keeps the location consistent with the context files in jar only modules.

Find out more*

There is very few examples of using the ContextLoaderListener for configuration. If you want to learn more, then the references maybe of interest:

http://www.jroller.com/page/raible?anchor=package_your_spring_config_files
http://www.springframework.org/docs/api/org/springframework/web/context/ContextLoader.html