Performance Tuning

Performance tuning

The following are some tips and tricks to squeeze the most performance out of Struts 2.

> ✅ **Important OGNL update**
>
> For Struts 2 versions before 2.3: the OGNL version 3.0.3 library is a drop-in replacement for older OGNL jars, and provides **much** better performance. See the following Jira issue for more information: https://issues.apache.org/jira/browse/WW-3580

Turn off logging and devMode.

devMode allows reloading of configuration and validation related files, but because they happen on each request, this setting will totally kill your performance.
When using logging, make sure to turn off logging (esp. Freemarker generates a LOT of logging), and check if a level is enabled before printing it, or you will get the cost of the String parsing/concatination anyways.

Use the Java Templates

If you use the simple theme, and do not overwrite any of the FreeMarker templates, consider using the java templates, which provide a drop in replacement for most tags, and are a lot faster than the regular tags.

Do not use interceptors you do not need.

If you do not require a full stack of interceptors for an Action, then try using a different one (basicStack), or remove interceptors you do not need. Remove the I18nInterceptor interceptor if you don't need it, as it can cause a session to be created.

Use the correct HTTP headers (Cache-Control & Expires).

When returning HTML views, make sure to add the correct headers so browsers know how to cache them.

Copy the static content from the Struts 2 jar when using the Ajax theme (Dojo) or the Calendar tag.

Struts 2 uses some external javascript libraries and cascading stylesheets for certain themes and tags. These by default are located inside the Struts 2 jar, and a special filter returns them when requesting a special path (/struts). Although Struts 2 can handle these requests, an application/servlet container is not optimized for these kind of requests. Consider moving these .js and .css files to a seperated server (Lighttpd, Apache HTTPD, ..).

Create a freemarker.properties file in your WEB-INF/classes directory.

Create the freemarker.properties file and add the following setting (or whatever value you deem fitting):

```
template_update_delay=60000
```

This value determines how often Freemarker checks if it needs to reloads the templates from disk. The default value is 500 ms. Since there is no reason to check if a template needs reloading, it is best to set this to a very large value. Note that this value is in seconds and freemarker will convert this value to milliseconds.

You can also use *struts.freemarker.templatesCache.updateDelay* constant to achieve the same effect.

See also: Freemarker configuration properties

Enable Freemarker template caching

As of Struts 2.0.10, setting the property *struts.freemarker.templatesCache* to true will enable the Struts internal caching of Freemarker templates. This property is set to false by default.

In Struts versions prior to 2.0.10, you had to copy the /template directory from the Struts 2 jar in your WEB_APP root to utilize Freemarker's built in chaching mechanism in order to achieve similar results.

The built in Freemarker caching mechanism fails to properly cache templates when they are retrieved from the classpath. Copying them to the WEB_APP root allows Freemarker to cache them correctly. Freemarker looks at the last modified time of the template to determine if it needs to reload the templates. Resources retrieved from the classpath have no last modified time, so Freemarker will reload them on every request.

When overriding a theme, copy all necessary templates to the theme directory.

There's a performance cost when a template cannot be found in the current directory. The reason for this is that Struts 2 must check for a template in the current theme first before falling back to the parent theme. In the future, this penalty could be eliminated by implementing a missing template cache in Struts 2.

Do not create sessions unless you need them.

Struts 2 does not create sessions unless asked to (for example, by having the createSession interceptor in your interceptor stack). Note that when you use SiteMesh however, a session will **always** be created (See http://forums.opensymphony.com/thread.jspa?messageID=5688 for details). The I18nInterceptor interceptor can create sessions, so make sure you remove it, if you don't need it.

When using Freemarker, try to use the Freemarker equivalent rather than using the JSP tags.

Freemarker has support for iterating lists, displaying properties, including other templates, macro's, and so on. There is a small performance cost when using the S2 tags instead of the Freemarker equivalent (eg. <s:property value="foo"/> should be replaced by ${foo}).

Next: Interceptors