Production Checklist

Table of contents

Here's a list of points you should take care of when running Lenya in a production environment. It covers only the most obvious aspects, but it might prevent you from falling in the biggest traps in first real-world project.

1. General Deployment Recommendations

- Seperate your web application from your data. This applies to the content, access control, and work data (search index, cache).
- When creating backups of your data, make sure you'll still know which version of the application they're compatible with when you need them again.
- Always have two instances of Lenya ready, and make sure you can switch between them immediately (e.g., by changing a symlink to a proxy configuration file).
- Consider using vendor branches for Lenya, Cocoon etc. This helps you to stay flexible when you're faced with bugs, endorsed library issues etc.
- When you deploy a version of your application, **always** create a branch in your code versioning system. This way, you can merge essential bugfixes from the trunk and re-deploy the application. **Never** deploy an un-tagged development version.

2. During Development

- Run sophisticated and thorough load tests early and often.
- Run search engine crawlers on your site. Observe the performance behaviour and session handling.
- Test the site in various browsers, using various settings (disabling JavaScript etc.), and preferrably using different bandwidths.
- Make sure you don't create weak points for DoS attacks (e.g. by expensive dynamic generation of non-cached pages based on request parameters).

3. Preparation (Internal Pilot Phase)

Set the log level to *ERROR* in `externals/cocoon_2_1_x/src/webapp/WEB-INF/log4j.xconf` (source, needs rebuild to become active) or `build/lenya/webapp/WEB-INF/log4j.xconf` (deployed file):

```
<root>
  <priority value="error" />
  <appender-ref ref="COCOON_DEFAULT" />
</root>
```

Make sure that the logs stay clean. If exceptions occur, mercilessly track them down and eliminate their causes. Even if you consider some exceptions "normal" behaviour - they aren't.

At a later point in your testing process, disable the "DEBUG" setting for the Java compiler in `src/targets/properties-build` to speed up your bytecode:

```
<property name="debug" value="off"/>
```

4. Caching in Cocoon

There's a great presentation (http://wiki.apache.org/cocoon-data/attachments/GT2006Notes/attachments/10-caching.pdf) (PDF) about caching in Cocoon, held at the Cocoon Get-Together 2006.

Cocoon's default store implementation is an in-memory store backed by a disk store (based on EHCache). You can configure this store using a file called `ehcache.xml` which is located in `org/apache/cocoon/components/store/impl/ehcache.xml`. The default entries of the `defaultCache` are as follows:

```
<defaultCache
    maxElementsInMemory="10000"
    eternal="true"
    timeToIdleSeconds="0"
    timeToLiveSeconds="0"
    overflowToDisk="true"
    diskPersistent="true"
    diskExpiryThreadIntervalSeconds="120"
    />
```

So, when `overflow-to-disk` is set to true, `eternal` to true and `timeToIdleSeconds=0`, then once a cachekey is in memory/disk, it will never be removed which might lead to a cache file becoming very large (see also thread: http://java2.5341.com/msg/170235.html).

For production use these setting should be changed. A possible configuration might look like:

```
<defaultCache
    maxElementsInMemory="10000"
    eternal="false"
    timeToIdleSeconds="1800"
    timeToLiveSeconds="3600"
    overflowToDisk="true"
    diskPersistent="true"
    diskExpiryThreadIntervalSeconds="120"
    />
```

This cache contains a maximum in memory of 10000 elements, and will expire an element if it is idle for more than 30 minutes and lives for more than 60 minutes. If there are more than 10000 elements it will overflow to the disk cache.

5. Performance Tuning

The following hints can be considered to improve the performance of your application:

- Set XSLT caching (use-store) to true in `cocoon.xconf`.
- Turn off reloading of sub-sitemaps.
- Enable client-side caching by changing the `pipeline-expiration` parameter in `global-sitemap.xmap`, for instance:
  ```
  <global-variables>
    <pipeline-expiration>access plus 2 hours</pipeline-expiration>
  </global-variables>
  ```
- Configure the expiration time of your resource types. For more information, consult the resource types reference (../../../docs/2_0_x/reference/resource-types.html) .
- Tips about Cocoon performance tuning: http://wiki.apache.org/cocoon/CocoonPerformance
- Tips about Tomcat performance tuning: http://marc.theaimsgroup.com/?t=103598885300001&r=1&w=2

Doug Chestnut recommends to adapt the garbage collector to make use of all available CPU cores. This can be done by adding the following to your JAVA options (for a machine with 4 cores):

```
-XX:+UseParallelGC -XX:ParallelGCThreads=4
```

You can also increase stack and heap size to make better use of available memory (be careful not to starve other processes on the same machine, though):

```
-Xms512m -Xmx1024m
```

Lenya does XML prettyprinting by default, which comes with a performance penalty. You might want to disable the transformation for maximum speed:

```
<map:transform src="fallback://lenya/modules/prettyprinting/xslt/xml2nicexml.xsl"/>
```

6. General Measures for Production

- Double-check your access control settings.
- Remove the example publications (default and blog). If you use them as templates, don't forget to remove the example users or change their passwords.
- Set the log level to *FATAL* or at least *ERROR*. This makes the code faster and prevents the log files from growing very large while keeping them comprehensible. The log level can be configured in `$COCOON_HOME/src/webapp/WEB-INF/log4j.xconf` or ultimately in `$LENYA_HOME/build/lenya/webapp/WEB-INF/log4j.xconf`.
- Set the *debug* property in `src/targets/properties-build.xml` to *off*. This way, the generated byte code will run faster.
- Disable all modules which accept request to dynamically generate images to prevent DoS attacks.
- Consider disabling image upload.
- Set the session expiration time to the least acceptable value.
- Prepare for maintenance (updates etc.), either by switching the application or by showing a friendly information page.
- Prepare for a worst-case scenario. For instance, have a statically exported version of the site ready.

7. Cosmetics

In order to hide the default publication from the welcome page (if you're not going to delete it anyway), change `src/pubs/default/publication.xml` as follows:

```
--- lenya/      (revision 452350)
         <publication xmlns="http://apache.org/cocoon/lenya/publication/1.1"
lenya:show="false">
```