

CACHING

AVAILABILITY & SCALABILITY

GX WEBMANAGER 9

Date

April 18, 2008

Target Audience

Application Manager

Developer

Target GX WebManager version

GX WebManager 9.2.0 and later

Document version

1.0

SUMMARY

Caching is essential for high traffic websites. This document will give more insight in the caching mechanism in WebManager 9. It also features best practices and a trouble shooting guide.

PREREQUISITES

Content Management

- Basic Content Management
- Design API
- Debugging

RELATED TOPICS

Design API

VERSION CONTROL

| Version | Date | Description |
|---------|----------------|-----------------|
| 1.0 | April 18, 2008 | Initial version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

TABLE OF CONTENTS

| | | |
|-------|--|----|
| 1 | Caching Overview | 5 |
| 1.1 | Introduction | 5 |
| 1.2 | Architecture and overview | 5 |
| 1.3 | Pages and Server Side Includes | 6 |
| 1.4 | An SSI example: the Poll Element | 7 |
| 2 | Working with the Caching module | 9 |
| 2.1 | Enabling caching | 9 |
| 2.2 | Cache configuration settings | 9 |
| 2.3 | Caching optimization in the edit environment | 10 |
| 2.4 | Clearing the cache | 10 |
| 2.5 | Timestamps in the database | 12 |
| 3 | Performance checklist..... | 15 |
| 3.1 | Infrastructure | 15 |
| 3.1.1 | Application Server | 15 |
| 3.1.2 | Database Server | 15 |
| 3.2 | Application | 16 |
| 3.2.1 | Configuration | 16 |
| 3.2.2 | JSPs/programming | 16 |
| 4 | FAQ | 17 |

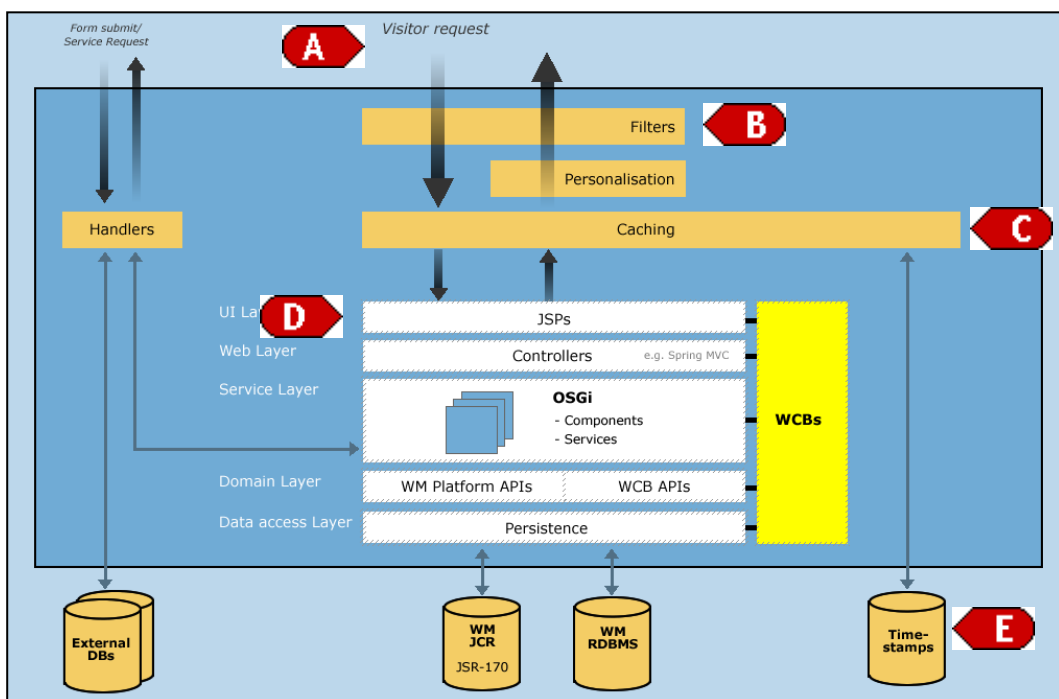
1 CACHING OVERVIEW

1.1 Introduction

Caching is essential for high traffic websites. It will handle the load created by many page requests by using an intelligent mechanism that returns pages without having to regenerate them completely. The WebManager caching module is also tailored not to interfere with customer interaction and personalization, so customers visiting the site won't notice that they are looking at a largely static website.

1.2 Architecture and overview

Caching occurs in the front end server processes and has links with JSP rendering [D] and the time stamps table [E], as you can see in the image below. A page request [A] from a website visitor goes through some filters [B] and arrives at the caching module [C].

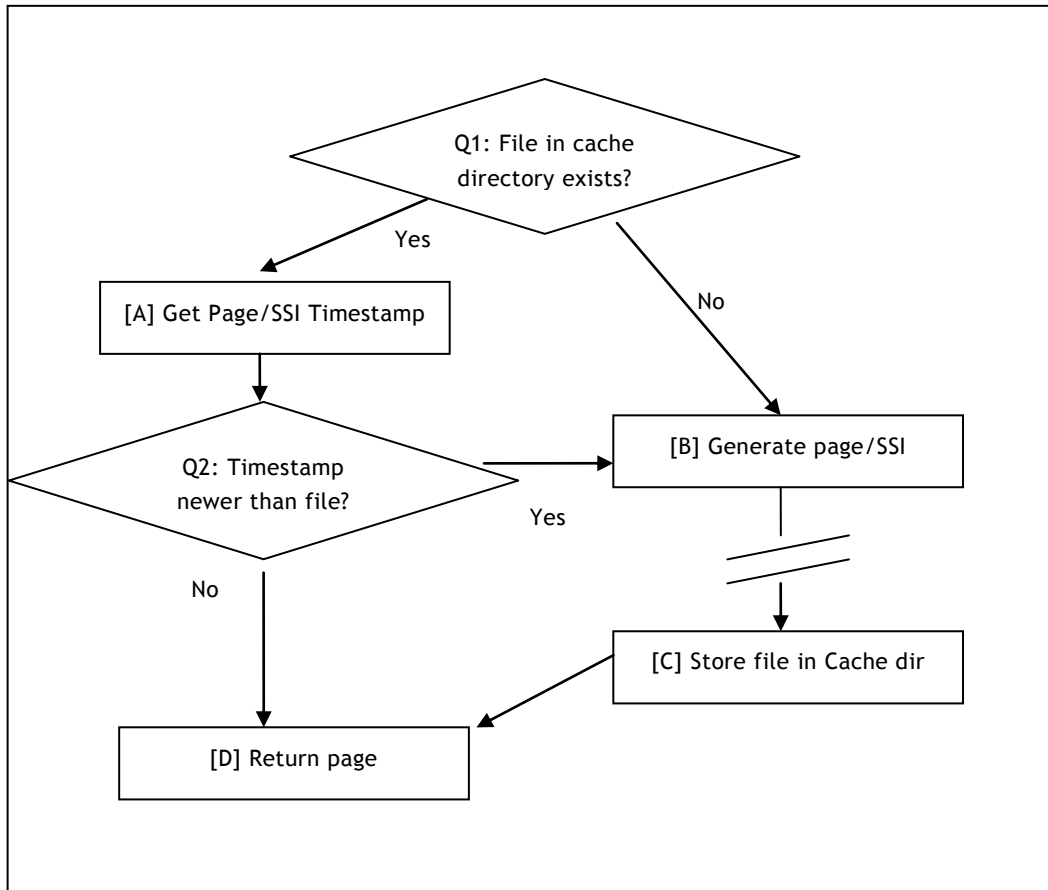


The actions inside the caching module [C] can be summarized as follows (see image below):

Q1: The caching module first determines whether a cached file exists of the page request. If the file exists go to [A], otherwise go to [B]

- ⇒ [A] Get the timestamp of the page (or SSI) by performing a query on the Timestamp database. (Note that the database is not always queried but that the timestamps are also cached in the caching module). The timestamp is compared [Q2] with the date of the file in the cache. If the timestamp is newer then the file is returned [D], otherwise the page has to be rendered (continue at 'If No').

- ⇒ [B] The page must be (re-)generated. Therefore a request goes from the caching module to the backend servlet to render the page. This takes some time and it's the slowest scenario. After the page has been rendered it is stored as a file in the cache directory [C] and the page is returned.



Workflow in the caching module

Note: this is a basic workflow scheme and that it doesn't deal with exceptions like too many page render requests or when the page rendering takes too long.

1.3 Pages and Server Side Includes

In a very basic setting the caching module could save local cache files based on every page request e.g. every URL. But in several cases this may not really speed up the page because parts of the page might be changing very often. Polls, database pages and media collections are examples of page parts that can be very dynamic.

Therefore in GX WebManager not only the entire page can be cached but also parts of the page. This happens with Server Side Includes (SSI's). This is a common mechanism where tags are placed inside the (HTML/JSP) source code. These so called SSI tags refer to other pieces of code and can

be referenced by local file path or URL. When the application server encounters an SSI tag it will request the SSI and add the generator code to the page. An example of an SSI tag:

```
<!--#include
virtual="/web/show?id=68644&langid=43&contentid=5&elementHolder=68637&jelly=true&ssiObjectClassName=nl.gx.webmanager.cms.mediarepository.MediaCollectionElement&ssiObjectId=68647&webid=26098" -->
```

This SSI tag will render a Media Collection Element with the Element ID 68647. There are more parameters that describe the element's context, e.g. Page ID, Media Item ID, Web Initiative ID etc.

1.4 An SSI example: the Poll Element

Polls are elements that can change quite often when a lot of people vote. Therefore it is better to use a SSI for the Poll Element. In the default presentation that comes with WebManager this has been included. In a basic situation where you add a poll element to a page the `page.jsp` is used as a starting point. The `page.jsp` will require the `/pagepart/content.jspf`, which will call the `/element/pollElement.jspf`. Before we look into this `.jspf` let's examine the `/element/pollElement.xml` file:

```
<presentation>
  <name>WM pollelement</name>
  <display-name>WM Poll element</display-name>
  <scope>PollElement</scope>
  <include>WM pollelement form</include>
  <include>WM pollelement result</include>
  <property>
    ...
  </property>
  <ssi>
    <presentation>pollElementSsi</presentation>
  </ssi>
</presentation>
```

An extra SSI node is included that points to a special SSI presentation. In this case it points to the `/element/pollElementSsi.jspf` file (see marked lines). Because of this SSI presentation the default `pollElement.jspf` is not used, but instead the `/pollElementSsi.jspf` is rendered.

```
<wm:link var="ssiLink" ssiObjectId="{pollElement.id}"
ssiObjectClassName="nl.gx.webmanager.cms.element.PollElement"
passOn="{passOn}" cachetimeout="120" />

${ssiLink.ssiTag}
```

To see the SSI in action on a page you have to take the following steps:

- ⇒ First make sure you can login to your application server as an administrator. For Tomcat this involves editing the file `/conf/tomcat-users.xml` in the Tomcat directory.
Example:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
<role rolename="wmadmin"/>
<user username="wmadmin" password="123456"
      roles="wmadmin"/>
</tomcat-users>
```

Restart Tomcat and go to the URL

`http://<edit-environment-hostname>:<port>/web/admin`. You should get a username/password popup. Enter 'wmadmin' and your password (see marked text). You will see a blank page.

- ⇒ Create a page in GX WebManager and add a Poll element. Enter some information in the poll element and set the status of the page to 'Published'.
⇒ Open the page on the **backend** hostname: open a new tab/window in your browser and go to `http://<edit-environment-hostname>:<port>/web/show`. This will show the homepage of your website without using the cache. Navigate to the page with the poll element.
⇒ Add the following string to the URL of the page: `/ssidebug=true` and refresh the page. This should show the string we have seen in the previous paragraph, something like

```
<!--#include
virtual="/web/show?id=26111&langid=43&cachetimeout=120&elementHolder
=26114&ssiObjectClassName=nl.gx.webmanager.cms.element.PollElement&ssiObjectid=72418" -->
```

- ⇒ It's possible to paste the marked part after your backend hostname and port number and open it in a browser. You will then see the code of the poll element and nothing else.

This exercise shows that individual SSIs can be rendered on their own by passing the right parameters (such as `elementHolder` and `ssiObjectClassName` to a `/web/show` command.

2 WORKING WITH THE CACHING MODULE

2.1 Enabling caching

- ⇒ To enable caching go to the `/web/setup` tool, under ‘frontend_system_settings’ and make sure the ‘allow_cache’ checkbox is checked.
- ⇒ You can test if it works by requesting several pages on the frontend. Verify if there are subdirectories with numbers created in the cache directory (default is `/work/cache` for local installations, `/cache` for production servers)
- ⇒ If you want to make sure that most pages are generated (and therefore in the cache) before a website goes live you can run the GX WebManager search engine.

2.2 Cache configuration settings

You can find most of the caching settings in the `/web/setup` tool, under ‘frontend_system_settings’.

CONFIGURATION SET DEFINITION: FRONTEND_SYSTEM_SETTINGS

default

| | |
|--|---|
| allow_cache | <input checked="" type="checkbox"/> |
| application_filter_definition_url_base | <code>http://localhost\${NOT_EMPTY}\${website_settings.frontend_portnr}:\${website_settings.frontend_portnr}/w</code> |
| cache_cacheable_response_codes | <input type="text" value="200"/> |
| cache_directory | <code>\${startup_settings.base_directory}/cache</code> |
| cache_directory_depth | <input type="text" value="3"/> |
| cache_max_stale_time | <input type="text" value="0"/> |
| cache_number_of_directories | <input type="text" value="30"/> |
| cache_statement_timeout | <input type="text" value="2"/> |
| cache_timestamp_expire | <input type="text" value="0"/> |

The cache settings in the `/web/setup` tool

The relevant cache options are:

| Field | Explanation |
|---|---|
| <code>allow_cache</code> | Indicates whether the proxy should cache. |
| <code>application_filter_definition_url_base</code> | Base URL for application filter functionality. |
| <code>cache_cacheable_response_codes</code> | Comma-separated list of cachable response codes. Default this is set to ‘200’, but can be extended with ‘,301’ if the website contains a lot of redirect pages. |
| <code>cache_directory</code> | Folder used for the cache. |
| <code>cache_directory_depth</code> | Depth of folder tree used for caching. Recommended value: 3. |
| <code>cache_max_stale_time</code> | The number of seconds that the frontend will serve an old version of a page from the cache, while the page is being regenerated by the backend. Recommended value: 0. |

| | |
|------------------------------------|---|
| <i>cache_number_of_directories</i> | Maximum number of subfolders that will be generated for the caching directory structure. Recommended value: 30. |
| <i>cache_statement_timeout</i> | Number of seconds that a query to the Timestamp database is allowed to take. If there is no response from the database within this time period, then the cached version of the page is returned. Recommended value: 2. |
| <i>cache_timestamp_expire</i> | Number of seconds that a timestamp is cached in Java memory. This prevents a lot of queries to the database from very popular pages. This might, however, cause a little delay in updating the page if something has changed. Recommended value: 30 (if the delay of max. 30 seconds is unacceptable, then this value can be set to 0). |

2.3 Caching optimization in the edit environment

Some caching optimization can also be done in the edit environment of GX WebManager. This is part of presentation management and can be found under `Format > Presentation > Page Parts`. Every page part can have a caching timeout value. This will prevent the rendering of the page part more than once in the time span specified as 'caching timeout'. Common page parts that can benefit from a caching timeout value are page parts that contain navigation such as menus.

Example: suppose there is a page part that contains a 'top 10 articles of the month'. This is not likely to change very fast (say every minute), so it's reasonable to set the 'caching timeout' for this page part to '3600', which means this page part is only generated at most once per hour.

Watch out with unnecessary using the 'caching timeout' parameter, because it can also slow down performance if used incorrectly. Before adding a 'caching timeout' value, consider if the page part is very static or updated more frequently. Static page parts like logos, headers and footers don't have to contain a caching timeout because sometimes they change only once per year, or less. In that case you can clear the timestamps of all the pages, which will also regenerate the page parts with the logos, headers and footers. If you set the timeout for a logo to '3600' it might be generated every hour, which is not necessary.

On the other hand be careful with very dynamic page parts such as 'WM content'. If you set a caching timeout for these types of page parts you might frustrate site editors because it takes too long for new content or content changes to show up.

2.4 Clearing the cache

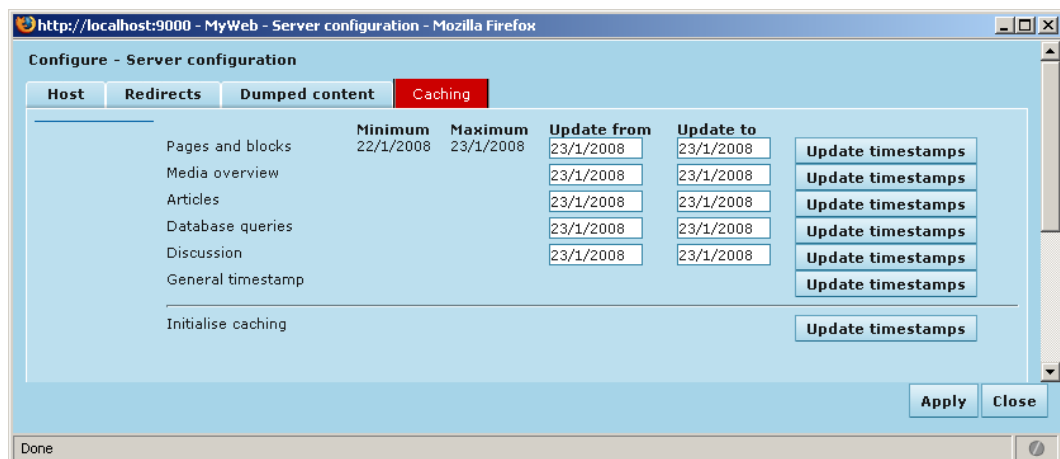
In a normal situation GX WebManager takes care of the required actions to clear the cache when content is changed, added or removed. When an editor adds an element to a page the timestamp for the page will be set to the current date and time. The caching mechanism will notice this

when the page is requested and the page will be regenerated (for more information see chapter 1.2).

But there are more complex situations where several page sections could be involved, or situations where a lot of caching timeouts are set for individual parts of a page, or a situation where the load on the server is very high and there's no time to regenerate pages. For example one page section could be included on a lot of pages and if you create a new version of a page section all these pages have to be regenerated. This will normally happen automatically, but in some situations you don't want this to wait until someone retrieves the page, but you want to force all the pages to be regenerated.

GX WebManager has the option to clear the timestamps in a safe and controlled manner. There are two ways to do this: updating the timestamps and 'soft timestamping'. Updating the timestamps sets the timestamps of objects within a certain range ('update from'-'update to') to the current date and time. This means that when a visitor requests one of the updated objects the object is regenerated and then returned to the user.

To control the cache you can go to `Configure > Server configuration > [Caching]`. The panel looks like this:



To update the timestamps select a time range for the objects you want to update and click on the button 'Update timestamps'. This can be done for the various items, so use common sense to clear the cache only for the relevant items.

If you are dealing with a live server with a significant load you have to be a bit more careful with updating the timestamps. For this purpose 'soft timestamping' is available. By setting a soft timestamp a special entry with id '0' is created in the 'sw_object' table (see next paragraph). When a visitor requests a page the timestamp of this page is compared with the '0' object. If the timestamp is older a request is sent to regenerate the page, but at the same time the cached version of the page is returned to the user. Therefore the user doesn't have to wait until the page is generated but he will get the cached page. The next visitor will get the regenerated page. To use soft timestamping you have to click the 'Update timestamps' button next to 'General timestamp'.

When you launch a new website you can initialize the caching module by clicking on the 'update timestamps' button next to the 'Initialize caching' label. This will create a timestamp entry for each page.

2.5 Timestamps in the database

For debugging purposes it might be necessary to look at the timestamps more closely. Timestamps are stored in the WebManager external database in seven tables:

| Table | Explanation |
|----------------------------------|--|
| <i>Tsdatabaseentitytimestamp</i> | Used for database detail pages. Timestamps are identified by the 'dbid' parameter. |
| <i>Tsdatabasetimestamp</i> | Used for entire databases. Timestamps are identified by the database id. |
| <i>Tsdiscussiontimestamp</i> | Used for discussion timestamps. Timestamps are identified by the thread id. |
| <i>Tsjellycontenttimestamp</i> | Used for media repository items. Timestamps are identified by media repository content id. |
| <i>Tsjellyfishtimestamp</i> | Used for an entire media repository of a web initiative. Timestamps are identified by web initiative id. |
| <i>Tsubjecttimestamp</i> | Used for various WebManager objects such as pages and page sections. Timestamps are identified by object id. For pages this is also the id in the url of a page, for example <code>/web/show/id=26111</code> . '26111' is the object id for this page. |
| <i>tstermtimestamp</i> | Used for media repository terms. When terms are changed they could influence media overviews so they have an own table. Timestamps are identified by term id. |

3 PERFORMANCE CHECKLIST

The following checklist can be used to check the performance variables of a WebManager website. Performance involves more than GX WebManager and the caching module so other relevant settings are also taken into account.

3.1 Infrastructure

3.1.1 Application Server

- **Application Server (Tomcat or other) has sufficient memory.**

Recommendation: make sure the application server meets the system requirements for your version of WebManager and that the required amount of memory is available for the application server process.

How to check: this is beyond the scope of this document.

- **The number of other applications (such as other application servers) is as low as possible**

Recommendation: for optimal performance run only one application server per server

How to check: use a process list in UNIX/Windows and/or the netstat command to scan port usage

- **Check incoming requests on the application server**

Recommendation: On a well tuned website the page duration time should be well under 1 second for cached pages. The average duration for well tuned WebManager websites is around 100 milliseconds.

How to check: use `http://<backend-hostname>/web/admin/status` to watch all incoming page requests on the application server. If the load on the server is high check for:

1. Pages with a high duration. Solution: investigate the page and check the points mentioned under 'JSP/programming'.
2. Large number of requests from one IP address. Solution: block the IP address.
3. Crawlers/search engines/Robots (see the user agent string). Solution: add a `robots.txt` file or block the IP address.

3.1.2 Database Server

- **Make sure the transaction speed is fast enough**

Recommendation: Websites with many database interactions can suffer severely from slow database servers and/or connections. Normal WebManager queries should take no longer than 1-2 seconds (maximum) with an average of tenths (1/10) of seconds. Queries to non-WebManager content should also be optimized.

How to check:

- Monitor the log of the application server for database/query timeouts or JDBC errors
- Use the database manager in WebManager to run some test queries, for example:
- Use a database analysis tool to monitor the database speed and connection. This is beyond the scope of this document.

3.2 Application

3.2.1 Configuration

- **Make sure caching is enabled**
How to check: See chapter 2.1
- **Make sure there are no backend server names and server aliases listed under the frontend server names and server aliases**
How to check: Go to the `/web/setup` tool and for each web initiative check the fields 'frontend_server_alias' and 'frontend_hostname'. These should not contain backend hostnames.
- **Consider using 'dumped' static pages.**
Recommendation: When the normal caching mechanism fails or when one or more pages are for the larger part static you might consider dumping certain pages, such as the homepage.
How to check: This can be set up in
Configuration > Server configuration > [Dumped Content]

3.2.2 JSPs/programming

- **Make sure the URLs contain as little query string parameters as possible**
Recommendation: try to avoid using query string parameters to pass parameters to other pages or page parts. This will create more unique URLs and thus more pages have to be generated and not cached
How to check: use `http://<backend-hostname>/web/admin/status` to watch all incoming page requests.
- **Use Server Side Includes (SSIs) for media overviews and queries**
Recommendation: Try to use SSIs for all media collection and query elements. This is also the default setting in the GX WebManager Cleansite (the default set of JSPs that come with GX WebManager)
How to check: use `http://<backend-hostname>/<page-url>/ssidebug=true` to see all SSIs on a page. On pages that contain media collections you should see the SSI URLs.
- **Don't use too many Server Side Included (SSIs)**
Recommendation: More than 5 SSIs will more create more overhead and add more complexity to the caching mechanism
How to check: use `http://<backend-hostname>/<page-url>/ssidebug=true` to see all SSIs on a page. Also see chapter 1.4.
- **Avoid using hardcoded queries in JSPs**
Recommendation: hardcoded queries in JSPs don't profit from the caching/timestamp mechanism and should therefore be avoided. Instead, use the WebManager query element or database pages.

4 FAQ

Q: Where can I find the cache directory?

A: The cache directory is set in the `/web/setup` tool, under 'frontend_system_settings' in the field 'cache_directory'. The default setting is the `/cache` directory.

Q: What happens if the (application) server is restarted? Will the cache be emptied?

A: No. The cached objects are stored on disk and the timestamps are stored in the database.

Q: Is it OK to remove the files from the cache directory or can the cache get corrupt?

A: No. The proxy will detect if the file is present on disk. If not it will be regenerated and written to disk again. Warning: be very careful with removing all cached files on a live server because it could create a significant load to regenerate all the pages. It's better to use the soft timestamping mechanism if you want to clear the entire cache (see chapter 2.4).

Q: What happens if I clear all the timestamp labels?

A: No timestamps will be found for any object or page so the cached file will be returned automatically. This happens until a requested object is changed and saved again.