Concept Caching

Your Rating: Results: 115 rates

Implemented in 3.6

Cache module module employs a web cache to store server responses and Advanced Cache (EE) provides a collection of advanced cache strategies.

Official Documentation Available

This topic is now covered in Cache module.

Official Documentation Available

This topic is now covered in Advanced Cache module (EE).

Introduction

Summary of findings and ideas for new caching implementation.

⚠ Improvements to caching *will be an enterprise only* feature.

EHCache

We should use ehcache and drop the simple cache implementation. We should also look at the SimplePageCachingFilter.

Flush Strategy

Currently all pages are flushed no matter if there is a change in a certain page or not. The problem is that we don't know which pages references what content (menu, news lists, ...). At least we should extend the interfaces in a way that one can flush based on paths, levels, ... This would at least allow implementations for multisite instances

We can't go father without having proper referencing (dependency graph). We could also think about a learning mechanism which builds the dependency graph dynamically.

Expiration Headers

Both methods should be provided: fix time, fix duration

Apache Integration

Support mod cache so that most of the traffic can be handled by apache. To do so we must handle the "If-Modified-Since:" header and "304 Not Modified" response. We should also set a appropriate expiration date.

see: http://httpd.apache.org/docs/2.2/caching.html

This is mainly a matter of documentation

Increase cached page hits

The system can die if every request recaches the (same) pages. At least we must ensure that the two requests enter the caching mechanism for the same page.

Solution A: Threshold

The idea is to recache the most frequent accessed pages before flushing the content. The level in percent should be configurable.

Solution B: Decouple caching

The request is served with the outdated cached content while the content is recached in the background. A priority queue should be used.

Decoupled Caching

If we use decoupled caching we have to ether mock a request / response or to create a internal request to localhost.

Mocking by using Cactus

As cactus creates the objects by doing real request (via a so called proxy) this is not really a better option than doing a real request. Cactus main feature is really testing based on real request execution

Mocking by using Mockrunner

- com.mockrunner.servlet.ServletTestModule
- we can add a filter which will be executed (the magnolia filter instance)

Using real request

We could also use the actual request by using the following strategy. The request will recache only if:

- content is invalid
- no caching for that uri is in progress
- maximal number of caching request is not exceeded

Otherwise it serves the cached content.

Paragraphs

Ehcache

The paragraph renderer could cache based on a certain flag (on the content node). An other option is to provide a cache attribute on mgnl:includeTemplate. We should also have a look at SimplePageFragmentCachingFilter

OSCache

Has its own tag library.