Fabrizio Branca (blog.html)

TYPO3 Individual cObject Caching

Von
Fabrizio Branca (kontakt.html)
am 20.02.2012 in
TYPO3 (blog/typo3.html) und Development (development.html)

Update (2012-02-27)

- My submitted patch (https://review.typo3.org/#change,9245) made into the core (thanks Tolleiv (https://twitter.com/tolleiv), for taking care...) and will be even backported to TYPO3 4.5 and 4.6.
- Also check out the detailed documentation on forge (http://forge.typo3.org/issues/34307#note-1) on this feature.
- In order to make TYPO3's cache (tag) handling even more convinient, and more easily to access for developers and integrators, I've added some more patches (still pending):
  - Add clearing cache by tag and by identifier to t3lib_tcemain->clear_cacheCmd() (http://forge.typo3.org/issues/34352)
  - Add cache_tags to pages table (http://forge.typo3.org/issues/34363)
  - Add hook to tslib_fe->get_cache_timeout() (http://forge.typo3.org/issues/34346)
  - Add hook to stdWrap_cacheStore (http://forge.typo3.org/issues/34343)
- Also checkout the comments of this blog post for some further information on this feature...

Original Blog Post...

TYPO3's way of caching content is very powerful and complex. Roughly summed up it works like this:

During page rendering TYPO3 detects which parts are dynamic parts (USER_INT and COA_INT cObjects) and renders everything but them, while adding placeholders to the places where the dynamic parts will be inserted later. The content with the paceholder will be stored in the page cache and before delivering the content to the client the dynamic elements will be renderend and inserted into their corresponding positions.

Cacheable cObjects (by default that is all cObjects including USER cObjects) will be directly rendered and stored with the page content into the cache. This results in two conclusions:

1. Cacheable content is always page specific. Pages cannot share cached content.
2. Cacheable content has no individual lifetime but is bound to the lifetime of the page where it is placed.

That means content is not cached at all (*_INT) or bound to a specific page and its lifetime.

This behaviour is very similar to Varnish's ESI handling and Magento Enterprise's FullPageCache.

Individual caching

But actually Magento offers another way how to handle caching: Every Block (that can be roughly compared to a cObject in TYPO3 - also check my blog post "Magento for TYPO3 developers (magento-for-typo3-

developers.html)") has its own cache key, lifetime and tags:

```
abstract class Mage_Core_Block_Abstract extends Varien_Object {
    public function getCacheKey() { [...] }
    public function getCacheTags() { [...] }
    public function getCacheLifetime() { [...] }
}
```

So it's up to every block to decide if it is cached, how long and what key to use. The developer is reponsible to create a cache key that reflects the different cache versions (e.g. taking the store or the language into account).

Compared to TYPO3's way of caching this approach allows some clever possibilities for caching and thus for increasing performance on your website.

If cObjects could have their own cache key instead of being bound to the page you could reuse them on all pages. That means you could reuse those cObjects for example for menu generation or any other element that might be expensive to regenerate on every page.

Solution

Inspired by Magento's block caching and Krystian Szymukowicz's **EXT:coa_go (http://forge.typo3.org/projects/show/extension-coago)** (which is not compatible to the latest TYPO3 versions anymore and has been stopped being developed since two years now) I created the **TYPO3 extension cobjcache (http://forge.typo3.org/projects/show/extension-cobjcache)** that brings this block element caching to stdWrap (and thus to cObjects - even if you might nest some content elements to make use of this feature).

Using the stdWrap hook (tslib_content_stdWrapHook) my extension adds new configuration options to the stdWrap toolset that allow controlling this fine-grained caching behaviour, store renderend content in case of a cache miss into the cache and deliver it from cache instead of rerendering it in case of a cache hit.

```
5 = TEXT
5 {
        cache.key = mycurrenttimestamp
        cache.tags = tag_a,tag_b,tag_c
        cache.lifetime = 3600
        data = date : U
        strftime = %H:%M:%S
}
```

In this example the current time will be cached with the key "mycurrenttimstamp". While the "key", "tags" and "lifetime" configuration support stdWrap themselves in this example the key is fixed and does not take the current page id into account. That means the cObj will be cached and reused on all pages. If you add this to your typoscript and click around the different pages, all those pages should show the same timestamp.

The module uses TYPO3's caching framework and stores all data into the cache_hash tables.

Get the extension from **forge (http://forge.typo3.org/projects/show/extension-cobjcache)** (EXT:cobjcache). Any feedback is welcome.