Core - Bug # 63810

Story # 63815 (New): Reduce communication between server and client

Status:	Accepted	**Priority:**	Must have
Author:	Michiel Roos	**Category:**	
Created:	2014-12-12	**Assigned To:**	Mathias Schreiber
Updated:	2014-12-16	**Due date:**	
TYPO3 Version:	6.2		
PHP Version:	5.4		
Complexity:			
Is Regression:	No		
Sprint Focus:			
Subject:	pages SYS_LASTCHANGED does not update when page cache is cleared		

Description

When the frontend cache is cleared, the SYS_LASTCHANGED value is not updated. This means that the 'Last Modified' header sent out by TYPO3 will stay unchanged for all the pages. All visitors holding a cached page in their browser cache will keep this page forever. They will check back with the server after the page 'expires' from their cache, but will never fetch a new version because the server says the page has not been modified.

Related issues:

related to Core - Bug # 63798: SYS_LASTCHANGED does not updated when TCEMAIN....	**New**	**2014-12-12**
related to Grid Elements - Feature # 56868: Update SYS_LASTCHANGED	**On Hold**	**2014-03-13**

History

#1 - 2014-12-12 12:22 - Markus Klein

Can you debug that?
Some entry points:

- \TYPO3\CMS\Frontend\Controller\TypoScriptFrontendController::setSysLastChanged
- \TYPO3\CMS\Frontend\Controller\TypoScriptFrontendController::fetch_the_id (very bottom)
- \TYPO3\CMS\Frontend\ContentObject\ContentObjectRenderer::lastChanged

#2 - 2014-12-12 12:33 - Michiel Roos

Mitigation code:

ext_localconf.php:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_tcemain.php']['clearCachePostProc'][] =
   'MyComp\\SomeModule\\DataHandler\\ProcessClearCacheQueue->clearCacheCmd';
```

Classes/DataHandler/ProcessClearCacheQueue.php

```
<?php
namespace MyComp\SomeModule\DataHandler;

/**
 * Class ProcessClearCacheQueue
```

```
 *
 * @package MyComp\SomeModule\DataHandler
 */
class ProcessClearCacheQueue {

    /**
     * Update SYS_LASTCHANGED of pages when clearCacheCmd is called
     *
     * This hook may be called from two locations:
     * 1). \TYPO3\CMS\Core\DataHandling\DataHandler::processClearCacheQueue()
     *     In this case the parameters contain:
     *     array(
     *         'table' => $table,
     *         'uid' => $uid,
     *         'uid_page' => $pageUid,
     *         'TSConfig' => $TSConfig
     *     );
     *     We can just update the SYS_LASTCHANGED for the page with $pageUid.
     *
     * 2). \TYPO3\CMS\Core\DataHandling\DataHandler::clear_cacheCmd()
     *     array(
     *         'cacheCmd' => strtolower($cacheCmd)
     *     );
     *     In this case we need to figure out the actual cacheCmd and clear the
     *     affected pages.
     *
     * @param array $parameters
     * @param \TYPO3\CMS\Core\DataHandling\DataHandler $parent
     *
     * @return void
     */
    public function clearCacheCmd($parameters, $parent) {
        /** @var \TYPO3\CMS\Core\Database\DatabaseConnection $databaseConnection */
        $databaseConnection = $GLOBALS['TYPO3_DB'];
        $now = time();
        if (is_array($parameters) AND isset($parameters['uid_page'])) {
            $databaseConnection->exec_UPDATEquery('pages', 'uid=' . (int)$parameters['uid_page'], array('SYS_LASTCHANGED' => $now));
        } else {

            // Clear cache for either ALL pages or ALL tables!
            switch ($parameters['cacheCmd']) {
                case 'pages':
                case 'all':
                case 'temp_cached':
                case 'system':
                    $databaseConnection->exec_UPDATEquery('pages', '1=1', array('SYS_LASTCHANGED' => $now));
                    break;
                default:
            }

            // Clear cache for a page ID!
            if (\TYPO3\CMS\Core\Utility\MathUtility::canBeInterpretedAsInteger($parameters['cacheCmd'])) {
                $databaseConnection->exec_UPDATEquery('pages', 'uid=' . (int)$parameters['cacheCmd'], array('SYS_LASTCHANGED' =>
```

```
$now));
            }

            // flush cache by tag
            if (strpos($parameters['cacheCmd'], 'cachetag:') === 0) {
                $pageIds = array();
                $cacheTags = explode(',', $parameters['cacheCmd']);
                foreach($cacheTags as $tag) {
                    if (strpos($tag, 'pageId_') === 0) {
                        $pageIds[] = (int)strrchr($tag, '_');
                    }
                }
                if (count($pageIds)) {
                    $databaseConnection->exec_UPDATEquery('pages', 'uid IN(' . implode(',', $pageIds) . ')', array('SYS_LASTCHANGED' =>
$now));
                }
            }
        }
    }
}
```

#3 - 2014-12-12 12:56 - Patrick Broens

I can affirm this behaviour. Strange that nobody ever did notify this.

#4 - 2014-12-12 13:47 - Markus Klein

Keep in mind that $parameters['cacheCmd'] is only a single keyword. So no need to explode on comma.

#5 - 2014-12-12 14:02 - Patrick Broens

Some clarification is needed:
The issue is for some topics:

- Cached lists which are taking records from another place (like news and a storage folder for the items)
- Changing TypoScript, templates and such, which is not a change in the actual page.

How browser caching is working:

- Browser gets page A served
- Page A has an "expires" time and a "Last-Modified" header
- Browser caches the page as long as expires still active
- If expires has run out, the browser checks again at server with the previous Last-Modified
- Server tells that Last-Modified has not changed, so browser will again serve from cache

When changing something which is not directly related to the page, like TypoScript or a template, SYS_LASTCHANGED will not change

We, as developers or integrators, are so used by reloading a page or clearing the browser cache, a visitor will not do this. Reloading a page will indeed refresh the page by getting a new copy from the server, but visitors are looking at a local cached version.

#6 - 2014-12-12 15:36 - Michiel Roos

Here is the fixed 'flush by tag' code:

```
// flush cache by tag
if (strpos($parameters['cacheCmd'], 'cachetag:pageid_') === 0) {
    $pageId = (int)substr(strrchr($parameters['cacheCmd'], '_'), 1);
    $databaseConnection->exec_UPDATEquery('pages', 'uid = ' . $pageId, array('SYS_LASTCHANGED' => $now));
}
```

#7 - 2014-12-12 15:37 - Alexander Opitz

- Parent task set to #63815

#8 - 2014-12-15 09:25 - Patrick Broens

Did some investigation in this topic today. Seems we have a hidden feature which must be used by content elements which run into problems like the one described. Look at \TYPO3\CMS\Frontend\ContentObject\ContentObjectRenderer::lastChanged(). With this method it is possible to set a timestamp higher than the pages SYS_LASTCHANGED from the content element itself. When the page is being rendered this value will be written to the pages DB entry in \TYPO3\CMS\Frontend\Controller\TypoScriptFrontendController::setSysLastChanged().

So it is up to the content element if the "Last Modified" header needs to be updated, which is exactly what we want.

This solves the problem partly. No need to set TCEMAIN.clearCacheCmd(). Still, when, for instance, a new template is added, we still need the change in SYS_LASTCHANGED when the cache is cleared for page, all or on tag.

Using above method needs to be documented for extension builders

#9 - 2014-12-15 09:29 - Mathias Schreiber

- Status changed from New to Accepted
- Assigned To set to Mathias Schreiber

#10 - 2014-12-15 17:14 - Patrick Broens

There are some other points where SYS_LASTCHANGED should be taken into account:
Not taken into account:

- starttime and endtime of pages and content
- Menu's (Pages/Content should be checked for tstamp)
- Fluid (File changes)
- FILE (File changes)
- FILES (File changes)
- FLUIDTEMPLATE (File changes)
- IMAGE (File changes)
- IMG_RESOURCE (File changes)
- IMGTEXT (File changes)
- MEDIA (File changes)
- MULTIMEDIA (File changes)
- QTOBJECT (File changes)
- SVG (File changes)

- SWFOBJECT (File changes)
- TEMPLATE (File changes)
- All file changes for external files like JS, CSS, HTML, TypoScript
- Changes up in rootline, like TypoScript changes

Taken into account:

- RECORDS
- CONTENT

#11 - 2014-12-16 11:39 - Jo Hasenau

We discussed that issue in our team today and the idea came up to solve this issue with a different approach, although I have to admit that I am not completeley convinced yet.

The idea: Actually SYS_LASTCHANGED should not be used to mark the "last modified" header for the stuff TYPO3 delivers at all, since it marks active changes by the users and is used for frontend output like "this page has been modified 12/13/14".

This issues is not about this particular timestamp though, since the "last modified" header should contain the time of the last creation of the cache file anyway.
So while the SYS_LASTCHANGED field might still need to be updated by more actions, its value should be irrelevant for the header until the cache will be created the next time.
And even if the cache would be created another time without changes to SYS_LASTCHANGED, it should still deliver another value for "last modified".

This way SYS_LASTCHANGED itself would not have to be updated when changing JS, CSS, HTML or TypoScript, but it could still be updated after actions of "real" editors.

#12 - 2014-12-16 11:47 - Patrick Broens

- File Browser caching.pdf added

SYS_LASTCHANGED was introduced by Kasper in 2003 or before, exactly with the purpose to track all page related changes. It was not really intended to be displayed in the frontend. Probably that is why he made the field name uppercase ;-)

I do not agree that the Last-Modified header should contain the last creation date of the the cache file. That is what's the "Date" header is already doing. Last-Modified is about content, not about caching.

I've just written a document describing the whole problem. I'll attach it here as well.

#13 - 2014-12-16 11:52 - Jo Hasenau

Actually I agree with you, Patrick, since this was my point in the discussion here as well.

#14 - 2014-12-16 15:31 - Helmut Hummel

Patrick Broens wrote:

> *I do not agree that the Last-Modified header should contain the last creation date of the the cache file. That is what's the "Date" header is already doing. Last-Modified is about content, not about caching.*

While I agree that the Last-Modified header **should** be close to the modification date of the entity, I disagree that it **must** be.
See the specification: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html which clearly supports this view.

For now, we have two easy options:

1. Remove the Last-Modified header completely (it is optional)
2. Set it to the cache creation date to avoid caching issues mentioned above

For the future we **may** implement a solution which **must** be independent from cache clearing, which calculated SYS_LAST_CHANGED of a page correctly. I consider this a "nightmare" task with our current code base and (missing) API, as too many things can happen in the system which changes the state of a "page" entity (starting but not ending with extensions or even the core directly modifying the database), that we can never completely detect all state changes.

#15 - 2014-12-16 16:00 - Patrick Broens

Hi Helmut,

If setting it to the cache entry date, it will never be called by the browser, because it first lets the cache expire and then it will look if there is a modification. So it is useless that way.

Files

Browser caching.pdf	71.5 kB	2014-12-16	Patrick Broens