Easy Authcache documentation

Contents

Description

Easy Authcache is module that makes authcache implementation easy,faster and makes it comportable with drupal community modules. Easy authcache module provide api that allow to implement authcache module functionality fast and easy. We have several pugins for easy authcache. Each plugin integrates easy authcache with some drupalsubstance .

This plugins are:

1. **Easy authcache block** – implements getting drupal blocks with authcache ajax
2. **Easy authcache comments** – implements getting drupal comments with authcache ajax
3. **Easy authcache plus 1** – implements getting drupal plus 1 link with authcache ajax
4. **Easy authcache views** – implements getting drupal views with authcache ajax

Advantages of module differs it from Authcache:

1. It makes unnessary for developer to write client side callbacks for authcache module (js functions).
2. Makes server side callback as hook (drupal style).
3. Includes all modules.
4. Handle js files and settings, needed for retrived with ajax html, on final page.
5. Handle url context dependent code.
6. Provide api to handle theme functions, you need, with authcache.

Disadvantages of module differs it from Authcache:

1. Makes bootstrap FULL on authcache ajax callback. That can make slowly then authcache manual implementation.
2. Can lose some js Drupal.settings after authcache ajax result.
3. Higher, then for manual authcache implementation can be, load on client side (js code).

Installation

To install Easy authcache

1. Install required modules (common way - Authcache)
2. Add Easy authcache module drupal contrib modules dir (usual /sites/all/modules).
3. Copy ***authcache_custom.php*** file from ***easy_authcache/_authcache/authcache_custom.php*** to ***authcache/ajax/*** and change (***authcache/ajax/authcache.php*****:str 42**) to include ***authcache_custom.php***, if needed.
4. Enable Easy authcache and it's plugins you need in drupal admin part url ***/admin/build/modules/list***
5. Config Easy authcache and/or it's plugns (some of them needs Easy Authcache UI module enabled to config).

Easy authcache and it's plugins configs

Easy Authcache have and some of it's plugins provide configs in admin part.

Easy authcache

Url: /admin/settings/performance/authcache/easyauthcache

Option	Default value	Desciption
Use single ajax request	**Checked**	**Checked – easy authcache sends single ajax request to get all dynamic regions. Unchecked – easy authcache will send one ajax request per easy authcache plugin On practise single request is faster then multiple (because each ajax request makes on server side bootstrap FULL), but for debug tasks multiple request is usefull.**
Max ajax age	**3600**	**Settes time in seconds broswer ajax request cache TTL**

Easy authcache block

Each block have it's own config.

For example URL: /admin/build/block/configure/user/1

Option	Default value	Desciption
Download by ajax	**unChecked**	**Checked – block will be getted dynamical with ajax**
Just run by ajax	**unChecked**	**Checked – html of the block will be cached as static, but block will be runned with ajax request. Usefull for statistic counted on block run**

Easy Authcache views

Url: /admin/settings/performance/authcache/easyauthcache

Select with checkboxes view-display that will be getted with ajax dynamicaly.

Posible problems, risky cases and things you need to know.

Do not forget about this risky cases

1. Limit of url length
2. Event races on ajax – js files/settings sync code
3. Third party js files with wrong behaviours / or with out them can cause troubles wit h js
    a. Function attached several times to same dom element
    b. Unattached js to html getted wit h ajax

4. Third party code that checked arguments in url with $_REUQEST array (wrong url context)
5. Wrong type of easy authcache result can cause bugs with replacing by js

Debug advices

It use usefull to debug

1. Enable authcache debug ***/admin/settings/performance/authcache***
2. Comment strings (***authcache/js/authcache.js:str 167***) and (***authcache/js/authcache.js:str 169***). This will allow you to see js errors you get on authcache client callback
3. Disable single easyauthcache request ***/admin/settings/performance/authcache/easyauthcache*** allow you to debug what easy authcache plugin breaks js.
4. Using Firebug.
5. Adding var_dump() and die() in to easy_authcache plugins hook_autcache_cached implementation.

How Easy authcache works?

Easy Authcache provide api that alow to override callback for any theme function ***easy_authcache_override_theme(&$theme_registry, $theme_name, $callback)*** (**easy_authcache/easy_authcache.module:str 75**). Developer of plugin should handle 2 cases. In case page is none caching he have to run default theme callback (for themes based on functions only, more detailed in Theme override) ***easy_authcache_run_default_theme_function(&$vars, $theme_name)*** (**easy_authcache/easy_authcache.module:str 84**). If page will be added into cache he have to output mock html with unique id and add params for ajax call back into js Drupal.settings struct ***easy_authcache_apply_dynamic_theme($module, &$vars, $selector, $params, $save_html = FALSE)*** (**easy_authcache/easy_authcache.module:str 10**). Then on client side, easy authcache will send all this params with authcache ajax (**easy_authcache/js/easy_authcache.js:str 18-33**). Server side handle to callback module we need with parameters that was passed to Drupal.settings by this module (**easy_authcache/easy_authcache.module:str 103**). Function that will used as callback is ***hook_authcache_cached($data)*** (example: **easy_authcache/modules/easy_authcache_block.module:str 83**). Module's callback returns struct, contains selector, result and result type (**easy_authcache/modules/easy_authcache_block.module:str 182**). Selector determinate what dom object to replace with result. Type of result determinates if result is html or plain text. Easy authcache attach drupal js array to module's callback result(**easy_authcache/easy_authcache.module:str 109**). And on client side, easy authcache merge getted js settings with Drupal.settings we already have on page (**easy_authcache/js/easy_authcache.js:str 39-41**), upload new js files, if needed, (**easy_authcache/js/easy_authcache.js:str 43-50**) , then replace dom object with result and attach drupal behaviours, if needed (**easy_authcache/js/easy_authcache.js:str 53-67**).

Integration with authcache

Easy authcache provides ***authcache_custom.php*** file with required functions. You have to copy it from ***easy_authcache/_authcache/authcache_custom.php*** to ***authcache/ajax/*** and change (***authcache/ajax/authcache.php*:str 42**) to include ***authcache_custom.php***, if needed. On applying dynamic theme ***easy_authcache_apply_dynamic_theme($module, &$vars, $selector, $params, $save_html = FALSE)*** (**easy_authcache/easy_authcache.module:str 10**), easy_authcache adds ***easy_authcache.js*** file and add ***$params*** to Durpal.settings.easy_authcache config. On client side js will send authcache ajax request (**easy_authcache/js/easy_authcache.js:str 18-33**), after result returning

authcache will call ***_authcache_easy_authcache(data)*** (**easy_authcache/js/easy_authcache.js:str 36**) as authcache client side call back. Server side authcache callback defined in (**easy_authcache/callback/easy_authcache.ajax.inc:str 2**). That file included in (***authcache/ajax/authcach_custom.php*:str 25**).

IMPORTANT: (***authcache/ajax/authcach_custom.php*:str 24**) make bootstrap FULL. After this all modules functions are avaliable.

Theme override

To make some region loaded by authcache ajax you need to override it's theme function callback. For this your plugin should implement ***hook_theme_registry_alter ();***

IMPORTANT: You can find out that implementing of hook_preprocess_{theme name} will gives you same effect. But this is not true. Do not use hook_preprocess_{theme name} because this way do not guarantees that your code will be run the last one and does not add path to mock template (**easy_authcache/theme/block-cache.tpl.php**).

Example of theme overriding you can see here (**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 11**).

```
function easy_authcache_comments_theme_registry_alter(&$theme_registry) {
  easy_authcache_override_theme($theme_registry, 'comment_wrapper',
'easy_authcache_comments_dynamic_load');
}
```

As you can see 'comment_wrapper' is name of theme and 'easy_authcache_comments_dynamic_load' is name of theme callback that will be run instead of theme. Drupal have 2 types of themes: template base theme and function based theme. Easy authcache have to override them in different way and you, as developer, have to handle custom callbacks in different way.

Template based theme

For template based themes callback should looks this way: (**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 15**)

```
function easy_authcache_comments_dynamic_load(&$vars) {
  global $is_page_authcache;
  if ($is_page_authcache) {
     $node = $vars['node'];
     $selector = easy_authcache_get_selector('comments-'. $node->nid);
     $params = array('nid' => $node->nid);
     easy_authcache_apply_dynamic_theme(EASY_AUTHCACHE_COMMENTS_ID, &$vars,
$selector, $params);
  }
}
```

If page are going to be cached you should

1. Retrieve params you will need on server side ajax callback (in our case it is node 'nid' to get comments for) (**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 20**)

2. Prepare unique selector for dom object that will be replaced with result of ajax callback (**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 19**).
3. Apply dynamic theme with function ***easy_authcache_apply_dynamic_theme()***;(**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 21**)

IMPORTANT: To get selector use ***easy_authcache_get_selector($selector)***, it will makes correct work with inserting selector, you have passed, as id in to mock dom object and correct using this selector in js file. But remember that $selector shoud be unique for page, and this is developer responsibility.

When you call ***easy_authcache_apply_dynamic_theme($module = EASY_AUTHCACHE_COMMENTS_ID, &$vars, $selector, $params);***
(**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 21**) $vars['template'] will be overridden to (**easy_authcache/theme/block-cache.tpl.php**) that will return mock html with ***id = $selector***. ***$module, $selector, $params*** will be passed to ***Drupal.settings.easy_authcache.***

IMPORTANT: ***$module*** determinates what module will handle ajax request for this dynamic theme. For now we allways use the same module that overrides theme. Server side ajax callback will be determinated as ***{$module} _authcache_cached()***;

If page are not going to be cached you should

Do nothing. Because you have not apply dynamic theme $vars contains default template and will run as usual.

Function based theme

For function based themes callback should looks this way:
(**easy_authcache/modules/easy_authcache_comments_count/easy_authcache_comments_count.module:str 19**)

```
function easy_authcache_comments_count_dynamic_load(&$vars) {
 global $is_page_authcache;
 if ($is_page_authcache) {
    $nid = $vars['nid'];
    $type = $vars['type'];
    $selector = easy_authcache_get_selector('comments-count-'. $nid .'-'. $type);
    $params = array( 'nid' => $nid, 'type' => $type, );
    easy_authcache_apply_dynamic_theme(EASY_AUTHCACHE_COMMENTS_COUNT_ID,
&$vars, $selector, $params);
 } else {
   easy_authcache_run_default_theme_function($vars,
EASY_AUTHCACHE_COMMENTS_COUNT_THEME);   }
 }
```

If page are going to be cached you should:

For developer this steps do not differs from template based theme steps - If page are going to be cached you should.

But on easy authcache level it's overriding differs.Easy authcache makes, overriden function based theme, as template base theme. It use template (**easy_authcache/theme/block-cache-function.tpl.php**). And store function in 'authcache_function' field in theme registry. This is the reason why hadling of non cached page changed.

If page are not going to be cached you should

Developer have to run ***easy_authcache_run_default_theme_function($vars, $theme);*** this function will run default theme function, stored in 'authcache_function', and make it's result rendered in template (**easy_authcache/theme/block-cache-function.tpl.php**). Example you can see (**easy_authcache/modules/easy_authcache_comments_count/easy_authcache_comments_count.module:str 32**). Remember that second parameter should be name of overridden theme function, required to get 'authcache_function' from theme registry.

Server side ajax callback (hook)

Easy authcache plugin module should also implement hook_authcache_cached($data) that will be callback for ajax request for theme functions ovenrrided with plugin module name passed as $module param to functi ***easy_authcache_apply_dynamic_theme($module, &$vars, $selector, $params);***

You can see example of hook_autcache_cached implementation here (**easy_authcache/modules/easy_authcache_comments/easy_authcache_comments.module:str 25**)

```
function easy_authcache_comments_authcache_cached($data) {
  $return = array();
  foreach($data as $widget) {
    $node = node_load($widget->params->nid);
    $output = comment_render($node);
    $return[] = array(
      'selector' => $widget->selector,
      'html' => $output,
      'type' => 'div',
    );
  }
  return $return;
}
```

$data is array of all regions related to that module. In example it is several comments lists for different nodes. Each $widget is structure with fields 'selector' and 'params'. Result of this hook implementation is array of items that presented as array with keys 'selector', 'html' and 'type'. Selector determinates what dom object to replace. For now it is allway getted from widget->selector. html is value that will be shown to final user. type can be 'text' or some html tag that will cover inserted html, for example 'div'. The differs of this 2 types is that for 'text' just insert as plain text, for all others we insert into dom tree as html and cover it with tag that setted as type and apply Drupal.behaviours to it's content. (We need cover because of Behaviours that applies only to content of context, that makes us to create cover html).

Url context dependent code

There are a lot of url context code in drupal. This is code that call arg() function and use it's result in logic. Because authcache ajax request sends to 'index.php' easy authcache have mechanism to solve this problem. easy_authcache.js add current url as param to ajax request (**easy_authcache/js/easy_authcache.js:str 2-6**). On server side

(***authcache/ajax/authcach_custom.php*:str 6-22**) we replace url passed with ajax as into GET and $SERVER['REQUEST_URI']. This way makes all code running after work correct with arg() function and etc.

WARNING: Easy Authcache do not rewrite REQUEST array. If there is any code that checking REQUEST array to get some GET params it will fail. Same problem can be if function arg() will changed and start checling s

---

Ajax working with JS

Sometimes developers call ***drupal_add_js()*** from theme functions. I even read that this is pattern allowed to override js files and/or settings. Realy it can cause problem with ajax, because ajax result do not have mechanism to include that files or settings to the page ajax called from. Easy authcache have this mechanism.

JS files loading

Easy authcache add function ***easy_authcache_store_js_files(&$vars)*** as preprocess function on theme 'page' (**easy_authcache/easy_authcache.module:str 135**). ***easy_authcache_store_js_files(&$vars)*** store all js files added to current page in to Drupal.settings. easy_authcache_js array. This idea is usful when we have enabled aggregated js files to know on client side what js files realy was aggregated. Easy authcache ajax callback adds js files & settings to the response (**easy_authcache/easy_authcache.module:str 109**). On client side js have checked if there are some js files getted with ajax that does not exists in Drupal.settings. easy_authcache_js. Unexsited files uploaded with js and there names added into Drupal.settings. easy_authcache_js. (**easy_authcache/js/easy_authcache.js:str 43-50**).

IMPORTANT: If new js files (getted with js after ajax response) have behaviours they will be applied to whole dom document (**easy_authcache/js/easy_authcache.js:str 81-105**).

Js settings getting

Easy authcache merge Drupal.settings that we have on page with Drupal.settings we getted with ajax. (**easy_authcache/js/easy_authcache.js:str 39-41**).

WARNING: Theoreticaly there can be cases when some values in Drupal.settings from page would be overrided with unncessasery Drupal.settings getted with ajax.

WARNING: There is not semaphore to sync gettings js files and settings. So this point is pontetial can have event races bugs.

Example of implementation on one of modules

```
<?php

define('EASY_AUTHCACHE_COMMENTS_ID', 'easy_authcache_comments');


/**
 * Implement hook_theme_registry_alter
 *
 * @param array $theme_registry
 */
function easy_authcache_comments_theme_registry_alter(&$theme_registry) {
  easy_authcache_override_theme($theme_registry, 'comment_wrapper',
'easy_authcache_comments_dynamic_load');
}

/**
 * Call back for theme fucntions
 * @global bool $is_page_authcache if page is going to be cached
 * @param array $vars variabels passing to the theme function
 */
function easy_authcache_comments_dynamic_load(&$vars) {
  global $is_page_authcache;
  if ($is_page_authcache) {
     $node = $vars['node'];
     // Create unique selector
     $selector = easy_authcache_get_selector('comments-'. $node->nid);
     // Prepare parameters passed to ajax callback function
     $params = array('nid' => $node->nid);
     // Apply dynamic theme. Override template path. Add nessasery js data and js
easy_authcache.js
     easy_authcache_apply_dynamic_theme(EASY_AUTHCACHE_COMMENTS_ID, &$vars,
$selector, $params);
  }
}

/**
 * Easy authcache ajax callback
 * @param array $data array of comments we need, array(selector, params);
 * @return array results
 */
function easy_authcache_comments_authcache_cached($data) {
  $return = array();
  // for each dynamic comments region
  foreach($data as $widget) {
    // Load node to get comments to
    $node = node_load($widget->params->nid);
```

```
            // Render comments
            $output = comment_render($node);
            // Adding element to result array
            $return[] = array(
              // selector to dom mock object
              'selector' => $widget->selector,
              // html that will be replaced with
              'html' => $output,
              // Type html = insert as dom html struct, apply behaviours
              'type' => 'html',
            );
          }
          return $return;

}
```