

Context aware HTTP cache in eZ Publish 5.2-5.3

This feature is available from eZ Publish **5.2** / **2013.07**

Workflow

1. Reverse proxy receives the HTTP request (without the user hash).
2. Reverse proxy does a sub-request (emulated in the case of *HttpCache*).

Sub-request **must** have the following headers:

- X-HTTP-Override: AUTHENTICATE
 - Accept: application/vnd.ez.UserHash+text
 - Original cookie (mainly to keep trace of the sessionId)
3. eZ Publish returns an HTTP response containing the user hash in X-User-Hash header.
 4. Reverse proxy adds the X-User-Hash header to the original request.

Note on performance

User hash is **not** generated for each AUTHENTICATE request. It is cached using the Cookie header string as key.

Hence each user has its own hash, generated once per session.

Hash generation being based by default on roles and limitations, **a user can share the same hash with another one** if their profile are similar. This is precisely what offers the possibility to *share HTTP cache* between several logged-in users.

Tip

You can customize user hash generation. Read [How to customize UserHash generation](#) in the developer cookbook to learn more about this.

Varnish

Described behavior comes out of the box with Symfony reverse proxy, but it's of course possible to use Varnish to achieve the same.

This can be done thanks to [Varnish Curl vmod](#).

```

import curl;

sub vcl_recv {
    # Do a standard lookup on assets
    # Note that file extension list below is not extensive, so consider completing it
    to fit your needs.
    if (req.request == "GET" && req.url ~
"\.(css|js|gif|jpe?g|bmp|png|tiff?|ico|img|tga|wmf|svg|swf|ico|mp3|mp4|m4a|ogg|mov|avi
|wmv|zip|gz|pdf|ttf|eot|woff)$") {
        return (lookup);
    }

    if (req.request == "GET") {
        # Pre-authenticate request to get shared cache, even when authenticated
        if (req.http.Cookie !~ "eZSESSID" ) {
            # User don't have session cookie => Set a hardcoded anonymous hash
            set req.http.X-User-Hash = "38015b703d82206ebc01d17a39c727e5";
        } else {
            # User is authenticated => fetch user hash
            curl.header_add("X-HTTP-Override: AUTHENTICATE");
            curl.header_add("Accept: application/vnd.ez.UserHash+text");
            curl.header_add("Cookie: " + req.http.Cookie);
            # Customize with real backend host
            # E.g. curl.get("http://www.metalfrance.net");
            curl.get("http://<host_of_your_backend>");
            if (curl.status() == 200) {
                set req.http.X-User-Hash = curl.header("X-User-Hash");
            }
        }
    }

    # If it passes all these tests, do a lookup anyway;
    return (lookup);
}

```

Securing hash generation request

By default, hash generation requests are granted for localhost (127.0.0.1, ::1, fe80::1).

If you want to enlarge the scope (e.g. if your Varnish server is not running on the same machine), you can override `canGenerateUserHash()` protected method in your main kernel class (mostly `EzPublishKernel`).