Smart HttpCache clearing

EZP >= 2015.01

Smart Http cache clearing refers to the ability to clear cache for locations/content that can be in relation with the content being currently cleared.

When published, any content item usually has at least one location, figured by its URL. Therefore Http cache being bound to URLs, if content `A` is updated (new version is published), we want Http cache for all its locations to be cleared, so the content itself can appear up-to-date everywhere it is supposed to be displayed. Sometimes, clearing cache for the content's locations is not sufficient. You can for instance have an excerpt of it displayed in a list from the parent location, or from within a relation. In this case, cache for the parent location and/or the relation need to be cleared as well (at least if an ESI is not used).

The mechanism

Smart Http cache clearing is an event based mechanism. Whenever a content item needs its cache to be cleared, the cache purger service sends an `ezpublish.cache_clear.content` event (also identified by `eZ\Publish\Core\MVC\Symfony\MVCEvents::CACHE_CLEAR_CONTENT` constant) and passes a `eZ\Publish\Core\MVC\Symfony\Event\ContentCacheClearEvent` event object. This object contains the ContentInfo object we need to clear the cache for. Every listener for this event can add location objects to the *cache clear list*.

Once the event dispatched, the purger passes collected location objects to the purge client, which will effectively send the cache `BAN` request.

> **Note**: The event is dispatched with a dedicated event dispatcher, `ezpublish.http_cache.event_dispatcher`.

Default behavior

By default, following locations will be added to the cache clear list:

- All locations assigned to content (`AssignedLocationsListener`)
- Parent location of all content's locations (`ParentLocationsListener`)
- Locations for content's relations, including reverse relations (`RelatedLocationsListener`)

Implementing a custom listener

By design, smart Http cache clearing is extensible. One can easily implement an event listener/subscriber to the `ezpublish.cache_clear.content` event and add locations to the cache clear list.

Example

Here's a very simple custom listener example, adding an arbitrary location to the list.

> **Important**: Cache clear listener services **must** be tagged as `ezpublish.http_cache.event_subscriber` or `ezpublish.http_cache.event_listener`.

```php
namespace Acme\AcmeTestBundle\EventListener;

use eZ\Publish\API\Repository\LocationService;
use eZ\Publish\Core\MVC\Symfony\Event\ContentCacheClearEvent;
use eZ\Publish\Core\MVC\Symfony\MVCEvents;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;

class ArbitraryLocationsListener implements EventSubscriberInterface
{
    /**
     * @var LocationService
     */
    private $locationService;

    public function __construct( LocationService $locationService )
    {
        $this->locationService = $locationService;
    }

    public static function getSubscribedEvents()
    {
        return [MVCEvents::CACHE_CLEAR_CONTENT => ['onContentCacheClear', 100]];
    }

    public function onContentCacheClear( ContentCacheClearEvent $event )
    {
        // $contentInfo is the ContentInfo object for the content being cleared.
        // You can extract information from it (e.g. ContentType from its
contentTypeId), using appropriate Repository services.
        $contentInfo = $event->getContentInfo();

        // Adding arbitrary locations to the cache clear list.
        $event->addLocationToClear( $this->locationService->loadLocation( 123 ) );
        $event->addLocationToClear( $this->locationService->loadLocation( 456 ) );
    }
}
```

```yaml
parameters:
    acme.cache_clear.arbitrary_locations_listener.class:
Acme\AcmeTestBundle\EventListener\ArbitraryLocationsListener

services:
    acme.cache_clear.arbitrary_locations_listener:
        class: %acme.cache_clear.arbitrary_locations_listener.class%
        arguments: [@ezpublish.api.service.location]
        tags:
            - { name: ezpublish.http_cache.event_subscriber }
```