Using Cache service

Using the internal cache service allows you to use an interface and to not have to care whether the system has been configured to place the cache in Memcached or on File system. And as eZ Platform requires that instances use a cluster-aware cache in Cluster setup, you can safely assume your cache is shared *(and invalidated)* across all web servers.

> **Interface will change in the future**
> Current implementation uses a caching library called Stash, via StashBundle. We plan to move to a PSR-6 compatible cache service capable of supporting cache Tagging and multi get/set in the future, when that happens the interface of the cache service will change!

> **Use unique vendor prefix for Cache key!**
> When reusing the cache service within your own code, it is very important to not conflict with the cache keys used by others. That is why the example of usage below starts with a unique "myApp" key. For the namespace of your own cache, you must do the same! So never clear cache using the cache service without your key specified, otherwise you'll clear all cache.

Get Cache service

Via Dependency injection

In your Symfony services configuration you can simply define that you require the "cache" service in your configuration like so:

yml configuration

```
myApp.myService:
        class: %myApp.myService.class%
        arguments:
            - @ezpublish.cache_pool
```

The "cache" service is an instance of the following class: `Tedivm\StashBundle\Service\CacheService`

Via Symfony Container

Like any other service, it is also possible to get the "cache" service via container like so:

Getting the cache service in PHP

```
/** @var $cacheService \Tedivm\StashBundle\Service\CacheService */
$cacheService = $container->get('ezpublish.cache_pool');
```

Using the cache service

Example usage of the cache service:

Example

```
$cacheItem = $cacheService->getItem('myApp', 'object', $id);
    $myObject = $cacheItem->get();
    if ($cacheItem->isMiss()) {
        $myObject = $container->get('my_app.backend_service')->loadObject($id)
        $cacheItem->set($myObject);
    }
    return $myObject;
```

For more info on usage, take a look at Stash's documentation.

Clearing Persistence cache

Persistence cache uses a separate Cache Pool decorator which by design prefixes cache keys with "ez_spi". Clearing persistence cache can thus be done in the following way:

getting the cache service in php

```
/** @var $cacheService \eZ\Publish\Core\Persistence\Cache\CacheServiceDecorator */
$cacheService = $container->get('ezpublish.cache_pool.spi.cache.decorator');

// To clear all cache
$cacheService->clear();

// To clear a specific cache item (check source code in
eZ\Publish\Core\Persistence\Cache\*Handlers for further info)
$cacheService->clear('content', 'info', $contentId);

// Stash cache is hierarchical, so you can clear all content/info cache like so:
$cacheService->clear('content', 'info');
```