Persistence cache configuration

Introduction

Tech Note

Current implementation uses a caching library called Stash (via StashBundle). Stash supports the following cache backends: **FileSystem, Memcache**, **APC, Sqlite**, **Redis** and BlackHole.

Use of Memcached (or experimentally using Redis) is a requirement for use in Clustering setup. For an overview of this feature, see Clustering.

> When eZ Platform changes to another PSR-6 based cache system in the future, then configuration documented below will change.

Cache service

The cache system is exposed as a "cache" service, and can be reused by any other service as described on the Using Cache service page.

Configuration

By default, configuration is currently using **FileSystem**, with `%kernel.cache_dir%/stash` to store cache files.

The configuration is placed in `app/config/config.yml` and looks like this:

Default config.yml

```
stash:
    caches:
        default:
            drivers:
                - FileSystem
            inMemory: true
            registerDoctrineAdapter: false
```

> **Note for "inMemory" cache with long running scripts**
> Use `inMemory` with caution, and avoid it completely for long running scripts for the following reasons:
>
> - It does not have any limits, so can result in the application running out of PHP memory.
> - Its cache pool is by design a PHP variable and is not shared across requests/processes/servers, so data becomes stale if any other concurrent activity happens towards the repository.

Multi repository setup

In `ezplatform.yml` you can specify which cache pool you want to use on a siteaccess or sitegroup level. The following example shows use in a sitegroup:

ezplatform.yml site group setting

```
ezpublish:
    system:
        # "site_group" refers to the group configured in site access
        site_group:
            # "default" refers to the cache pool, the one configured on stash.caches
above
            cache_pool_name: "default"
```

One cache pool for each repository
If your installation has several repositories *(databases)*, make sure every group of sites using different repositories also uses a different cache pool.

Stash cache backend configuration

General settings

To check which cache settings are available for your installation, run the following command in your terminal:

```
php app/console config:dump-reference stash
```

FileSystem

This cache backend uses the local filesystem, by default the Symfony cache folder. As this is per server, it **does not support multi server (cluster) setups**!

We strongly discourage you from storing cache files on NFS, as it defeats the purpose of the cache: speed.

Available settings

`path`	The path where the cache is placed; default is `%kernel.cache_dir%/stash`, effectively `app/cache/<env>/stash`
`dirSplit`	Number of times the cache key should be split up to avoid having too many files in each folder; default is 2.
`filePermissions`	The permissions of the cache file; default is 0660.
`dirPermissions`	The permission of the cache file directories (see dirSplit); default is 0770.
`memKeyLimit`	Limit on how many key to path entries are kept in memory during execution at a time to avoid having to recalculate the path on key lookups; default is 200.
`keyHashFunction`	Algorithm used for creating paths; default is md5. Use crc32 on Windows to avoid path length issues.

Issues with Microsoft Windows

If you are using a Windows OS, you may encounter an issue regarding **long paths for cache directory name**. The paths are long because Stash uses md5 to generate unique keys that are sanitized really quickly.

Solution is to **change the hash algorithm** used by Stash.

Specifying key hash function

```
stash:
    caches:
        default:
            drivers:
                - FileSystem
            inMemory: true
            registerDoctrineAdapter: false
            FileSystem:
                keyHashFunction: 'crc32'
```

This configuration is only recommended for Windows users.

Note: You can also define the **path** where you want the cache files to be generated to be able to get even shorter system path for cache files.

FileSystem cache backend troubleshooting

By default, Stash Filesystem cache backend stores cache to a sub-folder named after the environment (`i.e. app/cache/dev`, `app/cache/prod`). This can lead to the following issue: if different environments are used for operations, persistence cache (manipulating content, mostly) will be affected and cache can become inconsistent.

To prevent this, there are 2 solutions:

1. Manual

Always use the same environment, for web, command line, cronjobs etc.

2 . Share stash cache across environments

Either by using another Stash cache backend, or by setting Stash to use a shared cache folder that does not depend on the environment.

In `ezplatform.yml`:

```
stash:
    caches:
        default:
            FileSystem:
                path: "%kernel.root_dir%/cache/common"
```

This will store stash cache to `app/cache/common.`

APC & APCu

This cache backend is using shard memory with APC's user cache feature. As this is per server, it **does not support multi server (cluster) setups**.

Not supported because of following limitation
As APC(u) user cache is not shared between processes, it is not possible to clear the user cache from CLI, even if you set `apc.enabl`

e_cli to On. That is why publishing content from a command line script won't let you properly clear SPI Persistence cache.

Please also note that the default value for apc.shm_size is 128MB. However, 256MB is recommended for APC to work properly. For more details please refer to the APC configuration manual.

Available settings

ttl	The time to live of the cache in seconds; default is 500 (8.3 minutes)
namespace	A namespace to prefix cache keys with to avoid key conflicts with other eZ Platform sites on same eZ Platform installation; default is null.

Memcache(d)

This cache backend is using Memcached, a distributed caching solution. This is the only supported cache solution for multi server (cluster) setups!

> **Note**
> Stash supports both the php-memcache and php-memcached extensions. **However** only php-memcached is officially supported as php-memcache is missing many features and is less stable.

servers	Array of Memcached servers, with host/IP, port and weight **server**: Host or IP of your Memcached server **port**: Port that Memcached is listening to (defaults to 11211) **weight**: Weight of the server, when using several Memcached servers
prefix_key	A namespace to prefix cache keys with to avoid key conflicts with other eZ Platform sites on same eZ Platform installation (default is an empty string). Must be the same on all servers with the same installation. See Memcached prefix_key option *****
compression	default true. See Memcached compression option *****
libketama_compatible	default false. See Memcached libketama_compatible option *****
buffer_writes	default false. See Memcached buffer_writes option *****
binary_protocol	default false. See Memcached binary_protocol option*****
no_block	default false. See Memcached no_block option *****
tcp_nodelay	default false. See Memcached tcp_nodelay option *****
connection_timeout	default 1000. See Memcached connection_timeout option *****
retry_timeout	default 0. See Memcached retry_timeout option *****
send_timeout	default 0. See Memcached send_timeout option *****
recv_timeout	default 0. See Memcached recv_timeout option *****
poll_timeout	default 1000. See Memcached poll_timeout option *****
cache_lookups	default false. See Memcached cache_lookups option *****
server_failure_limit	default 0. See PHP Memcached documentation *****
socket_send_size	See Memcached socket_send_size option *
socket_recv_size	See Memcached socket_recv_size option *
serializer	See Memcached serializer option *
hash	See Memcached hash option *
distribution	Specifies the method of distributing item keys to the servers. See Memcached distribution option *

***** All settings except `servers` are only available with memcached PHP extension. For more information on these settings and which version of php-memcached they are available in, see: http://php.net/Memcached

When using Memcache cache backend, you *may* use inMemory to reduce network traffic as long as you are aware of its limitations mentioned above.

Example with Memcache

```
stash:
    caches:
        default:
            drivers: [ Memcache ]
            inMemory: true
            registerDoctrineAdapter: false
            Memcache:
                prefix_key: ezdemo_
                retry_timeout: 1
                servers:
                    -
                        server: 127.0.0.1
                        port: 11211
```

Connection errors issue
If memcached does display connection errors when using the default (ascii) protocol, then switching to binary protocol *(in the stash configuration and memcached daemon)* should resolve the issue.