Persistence cache configuration

Introduction

Tech Note

Current implementation uses a caching library called Stash, via Stash-bundle. If this changes, then the configuration format will most likely change as well. Stash supports the following cache handlers: **FileSystem**, **Memcache**, **APC**, **Sqlite**, **XCache** and **BlackHole**.

Cache service

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

Configuration

During Setup wizard and manually using `ezpublish:configure` console command a default configuration is generated currently **using FileSystem**, using `%kernel.cache_dir%/stash` to store cache files. It falls back to *BlackHole*, a non caching cache handler.

The configuration is placed in ezpublish/config/ezpublish.yml, and looks like:

Default ezpublish.yml

```
stash:
    caches:
        default:
            handlers:
                - FileSystem
            inMemory: false
            registerDoctrineAdapter: false
```

The default settings used during setup wizard as found in `ezpublish/config/ezpublish_setup.yml`:

ezpublish_setup.yml

```
stash:
    caches:
        default:
            handlers:
                - BlackHole
            inMemory: true
            registerDoctrineAdapter: false
```

This setting works across all installs and just caches objects within the same request thanks to the `inMemory: true` setting.

If you want to change to another handler, see below for what kind of settings you have available.

Note for multiple sites installation
As cache configuration is global to the application, it's currently not possible to have different cache configuration by siteaccess. As such, when having multiple websites **using different content repositories**, you must only use the `BlackHole` handler with `inMemory` caching:

```
stash:
    caches:
        default:
            handlers: [BlackHole]
            inMemory: true
            registerDoctrineAdapter: false
```

This will result to the same persistence cache strategy as in eZ Publish 4.x.

Note for "inMemory" cache with long running scripts
Use of `inMemory` caching with BlackHole or any other cache handler should not be used for long running scripts as it will over time return stale data (inMemory cache is not shared across requests/processes, so invalidation does not happen)!

Stash handlers configuration

General settings

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

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

FileSystem

This cache handler is using 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 `ezpublish/cache/<env>/stash`
`dirSplit`	Number of times the cache key should be split up to avoid having to 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 200.
`keyHashFunction`	Algorithm used for creating paths, default `md5`

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 key that are sanitized really quickly.

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

Specifying key hash function

```
stash:
    caches:
        default:
            handlers:
                - 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.

APC

This cache handler is using shard memory using APC's user cache feature, as this is per server, it **does not support multi server (cluster) setups**, unless you use PHP-FPM on a dedicated server (this way, user cache can be shared among all the workers in one pool).

Limitation
As APC user cache is not shared between processes, it is not possible to clear the user cache from CLI, even if you set `apc.enable_cli` to On.
Hence publishing content from a command line script won't let you properly clear SPI 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 set to 500 (8.3 minutes)
`namespace`	A namespace to prefix cache keys with to avoid key conflicts with other eZ Publish sites on same eZ Publish installation, default is `null`.

Memcache

This cache handler 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 Memcached extensions. **However** only Memcache is officially supported on Redhat while Memcache<u>d</u> is on Debian.

If you have both extensions installed, **Stash will automatically choose Memcached**.

`servers`	Array of Memcached servers, with host/IP, port and weight **`server`**: Host or IP of your Memcached server **`port`**: Port where 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 Publish sites on same eZ Publish installation (default is an empty string). Must be the same on all server with the same installation. See Memcached option.
`compression`	default true. See Memcached option.
`libketama_compatible`	default false. See Memcached option
`buffer_writes`	default false. See Memcached option
`binary_protocol`	default false. See Memcached option
`no_block`	default false. See Memcached option
`tcp_nodelay`	default false. See Memcached option
`connection_timeout`	default 1000. See Memcached option
`retry_timeout`	default 0. See Memcached option
`send_timeout`	default 0. See Memcached option
`recv_timeout`	default 0. See Memcached option
`poll_timeout`	default 1000. See Memcached option
`cache_lookups`	default false. See Memcached option
`server_failure_limit`	default 0. See Memcached option

For in-depth information on the settings, see: http://php.net/Memcached

When using Memcache handler, **it's strongly recommended to also use `inMemory` cache** (see example below).
This will help reducing network traffic between your webserver and your Memcached server.

Example with Memcache

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