cjw NETWORK
Developers united in eZ Publish

cjw

Learnings from Real eZ Publish 5 Projects

Donat Fritschy - Webmanufaktur
Ekkehard Dörre - Coolscreen

cjw NETWORK
Developers united in eZ Publish

Presenters

- **Donat** is owner of Webmanufaktur, a full service web agency in Switzerland. He works as projects manager, software architect and developer and likes thinking outside of the box. In the last year he has been involved in major eZ 5 projects.

- **Ekke** is a consultant with deep knowledge in eZ Publish 4 and 5, eZ Find / Apache Solr and with a faible for coming cutting edge web technologies. He is one of the organizers of the PHP Unconference since seven years.

- Members of CJW Network

Learnings from Real eZ Publish 5 Projects

- Last year we have presented our Cookbok...

- ... today you get a Ratatouille ;-)

- Let's share the experiences you and we made in the last year!
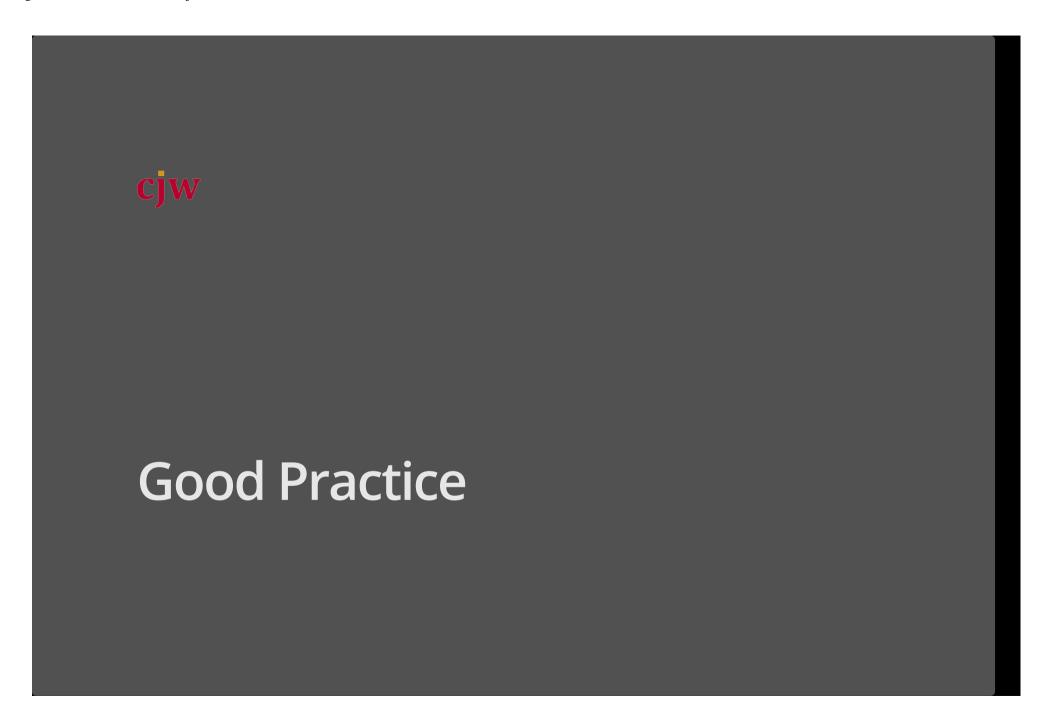
cjw #ezsummer

4/53

01.09.14 10:34

Agenda

Things we would like to discuss:

· Good Practice

· ez View Cache vs. HTTP Cache

· Debugging

· Pitfalls

· MultiSite Setup

cjw #ezsummer

01.09.14 10:34

cjw

Good Practice

Team up with a Symfony Crack

- To be honest: as eZ 4 developers, we are complete novices in eZ 5

- It's easier for a Smyfony Crack to learn eZ than other way round

- Symfony community is hungry for a CMS, so watch out for new competition

cjw #ezsummer

01.09.14 10:34

Think in MVC

- A radical different thinking

- eZ 4 mangeled all together in the TPL -> the view implemented the logic (fetch)

- Symfony enforces a clean separation, from routing to the controller to the rendering of the view

cjw **#ezsummer**

8/53

Think in Bundles

- Split your application in different bundles (site specific, functional, ...)
- Reuse your code: create and maintain a Base Bundle with general functions

Creating bundles is easy, don't work in the DemoBundle ;-)

```bash
$ php ezpublish/console generate:bundle
```

Organize your config files

- The standard eZ installation is a mess...

- ... and the DemoBundle only slowly becoming a source of good practice

Our approach

- keep config in `ezpublish/config` as general as possible

- it should merely decribe your environment, not the application

- move all site/function specific settings to the bundle

cjw #ezsummer   10/53

Keep ezpublish.yml small

`ezpublish/config/ezpublish.yml`

```yml
imports:
    - {resource: "@CjwExampleBundle/Resources/config/ezpublish.yml"}
```

Keep these sections:

· doctrine (Database)

· ezpublish (Siteaccess Matching, Siteaccesses, Languages, Caching)

· stash

Can even be shorter - get inspiration from https://github.com/lolautruche/metalfrance

Note: prepending configuration does not work well with parameters

cjw #ezsummer   11/53

Config Files in Bundle

We keep them in a separate directory and name them as in good old eZ...

```
ExampleBundle
└ Resources
  └ config
    └ ezpublish
        image.yml
        override.yml
      ezpublish.yml
      parameters.yml
      routing.yml
      services.yml
```

cjw #ezsummer   12/53

Controllers

After several tries, we ended up with...

· Basically one controller per full view

· Separate controllers for navigation etc.

· Retrieve all needed data (location, children, ...)

· Prepare the data for easy processing in the templates

· Consider caching (TTL, X-Location Header)

cjw #ezsummer

13/53

Ways to Fetch Content

- SearchService::findContent()

- SearchService::findLocations()

- LocationService::loadLocationChildren()

- Legacy fetch functions

cjw #ezsummer   14/53

SearchService::findContent()

The only `SearchService` function you will find in `DemoBundle`

- returns full `content` objects with ALL attributes in ALL languages

- does not work ~~well~~ with multiple locations

- scales very badly

- no `asObjects` flag as in eZ 4

- fetching a content tree with 116 locations took 30 seconds

- most of the time is spent in manipulating the SQL result in PHP

- Another test: 24 hits, PHP array 44'880 rows with 39 elements each, highly redundant

http://share.ez.no/blogs/donat-fritschy/searchservice-performance

cjw #ezsummer

15/53

SearchService::findLocations()

Available from 2014.05 / 5.3 only

Roughly equivalent to good old `fetch( 'content', 'list' )`

- returns `location` objects with `contentInfo` only
- fetching a content tree with 116 locations took < 1 second

LocationService::loadLocationChildren()

«Think of `LocationService::loadLocationChildren()` as primarily intended for administration interface.»

«If what it offers suits you for the frontend as well, great, but otherwise you will have to use SearchService. In other words, this method will not get filtering capabilities.»

(From a discussion with Petar)

http://www.netgenlabs.com/Blog/Fetching-content-in-eZ-Publish-5-using-Search-service

Legacy Fetch Functions

```php
use eZFunctionHandler;
...
$mySearchResults = $this->getLegacyKernel()->runCallback(
    function () use ( $searchPhrase, $sort, $contentTypeIdenfiers )
    {
        // eZFunctionHandler::execute is the equivalent for a legacy template fetch function
        // The following is the same than fetch( 'ezfind', 'search', hash(...) )
        return eZFunctionHandler::execute(
            'ezfind',
            'search',
            array(
                'query'     => $searchPhrase,
                'sort_by'   => $sort,
                'class_id'  => $contentTypeIdenfiers
            )
        );
    }
);
```

cjw #ezsummer

18/53

TWIG Templates

- A generalized full and line view template for the easy stuff
- And again, basically one template per full view

How to hande children (sub items)?

- render directly in the template - the data is usally already there
- use `{{ render( controller( 'ez_content:viewLocation', {'locationId': child.id, 'viewType': 'line'} )) }}`

When to use ESI?

- nice concept, but quite a big overhead
- better suited for larger chunks

How to organize Templates?

The Symfony way...

```
views
└ Customer
    CustomerDetail.html.twig
└ Product
```

The classic eZ way...

```
views
└ full
    customer.html.twig
└ line
```

Two approaches, both valid. Follow your taste.

cjw #ezsummer   20/53

cjw

eZ View Cache vs. HTTP Caching

eZ View Caching

When the pagelayout is rendered, the `{$module_result.content}` part will be replaced with the actual output. If view caching is enabled, the entire result of the module will be cached. This means that the contents of the "module_result" variable will be put into a cache file (...)

When a new version (...) of an object is published, the system will automatically clear the view cache for the following items:

· All published nodes of the object

· The parent nodes

· Related nodes (keywords, object relations)

https://doc.ez.no/eZ-Publish/Technical-manual/4.x/Features/View-caching

cjw #ezsummer

01.09.14 10:34

HTTP Expiration and Validation

The HTTP specification defines two caching models:

· With the **expiration model**, you simply specify how long a response should be considered "fresh" by including a Cache-Control and/or an Expires header. Caches that understand expiration will not make the same request until the cached version reaches its expiration time and becomes "stale";

· When pages are really dynamic (i.e. their representation changes often), the **validation model** is often necessary. With this model, the cache stores the response, but asks the server on each request whether or not the cached response is still valid. The application uses a unique response identifier (the Etag header) and/or a timestamp (the Last-Modified header) to check if the page has changed since being cached.

http://symfony.com/doc/current/book/http_cache.html

In Short (and much simplified...)

eZ View Cache caches **content** and **content fragements**

- Standard TTL is 2 hours
- Is purged on content modifications (with smart cache clearing rules)

Symfony's HTTP Cache caches **requests**

- eZ uses Expiration model by default
- Standard TTL is 60 seconds (86400 for tree menu!)
- Symfony Cache is purged from backend, **but only for ONE location**

Let's dive in a bit deeper...

cjw #ezsummer

24/53

ez 4 Cache Directives

Code from `ezpublish_legacy/kernel/private/classes/ezpkernelweb.php`

```php
// send header information
foreach (
    eZHTTPHeader::headerOverrideArray( $this->uri ) +
    array(
        'Expires' => 'Mon, 26 Jul 1997 05:00:00 GMT',
        'Last-Modified' => gmdate( 'D, d M Y H:i:s' ) . ' GMT',
        'Cache-Control' => 'no-cache, must-revalidate',
        'Pragma' => 'no-cache',
        ...
    ) as $key => $value
)
{
    header( $key . ': ' . $value );
}
```

This guarantees that every request is handled by eZ

eZ 5 Cache Directives

Code from `vendor/ezsystems/demobundle/EzSystems/DemoBundle/Controller/DemoController.php`

```php
                                                                      PHP
// Setting HTTP cache for the response to be public and with a TTL of 1 day.
$response = new Response;
$response->setPublic();
$response->setSharedMaxAge( 86400 );
// Menu will expire when top location cache expires.
$response->headers->set( 'X-Location-Id', $rootLocationId );
// Menu might vary depending on user permissions, so make the cache vary on the user hash.
$response->setVary( 'X-User-Hash' );
```

This effectively sets the Response free, out of the reach of eZ

cjw #ezsummer   26/53

Emulating eZ 4 Cache behaviour in eZ 5

This patch to `index.php` disables client and proxy caching without sacrifying the benefits of the Symfony HTTP cache. Use at own risk!

```php
$response = $kernel->handle( $request );                                      PHP

// Emulate eZ 4 cache control
$response->headers->set( 'Cache-Control', 'no-cache, must-revalidate' );

$response->send();
$kernel->terminate( $request, $response );
```

cjw #ezsummer   27/53

Cache Recommendations

- Read the specifications

- Use Shared Caches with caution

- Set TTL values low enough

- Cave: `setTtl()` vs. `setClientTtl`

- For higher traffic sites, use Varnish

- http://tools.ietf.org/html/rfc2616#page-74

- https://www.mnot.net/cache_docs/

cjw #ezsummer

cjw

Debugging

Coping with blank screens

Blank screen, "503 Service not available"

- PHP errors (Syntax error, Memory, Outdated Autoloads, …)

- Configuration errors (DB connection, …)

- Switch to DEV mode for better debugging

- Check the log files

```
Apache/PHP Log
ezpublish/logs/<env>.log
ezpublish_legacy/var/log/*
ezpublish_legacy/var/<siteaccess>/log/*
```

- Check write permissions on log files!

cjw #ezsummer   30/53

TwigBundle:Exception:error500.html.twig

· NEVER a Twig error!

· Caused by response 500 "Internal Server Error" and missing error template

· Checks as before

Twig Exception: Invalid variation "<variation>"

Caused by problems when accessing images

- Check if the file exists
- Check permissions on `ezpublish_legacy/var/<siteaccess>/storage`
- Check log files
- Clear cache

Class 'ezxFormToken' not found

- Usually found with fresh installations involving legacy extensions

- Regenerate Autoloads

```bash
$ cd ezpublish_legacy
$ php bin/php/ezpgenerateautoloads.php -e -p
```

cjw

Pitfalls

Avoid the traps...

Memory limit exceeded in DEV mode

- DEV mode takes a lot of memory

- Stash Logging is the worst

- Disable Stash Logging in ezpublish.yml

```yml
stash:
    logging: false
    caches:
        default:
            handlers:
                - FileSystem
            inMemory: true
            registerDoctrineAdapter: false
```

414 Request-URI Too Long

When doing subrequests, particularly ESI or Hinclude ones, current SiteAccess is transmitted in a serialized form, with its matcher. With a large number of configured SiteAccesses using Map\Host or Map\URI matcher (around 40, which is not uncommon for a multi-national, multi-lingual site) the URL can exceed the size of 8192 Bytes which most servers accept. As a result, the ESI/Hinclude call fails.

- Fixed in Version 5.3.3 (2014.07)

- https://jira.ez.no/browse/EZP-23168

- https://github.com/ezsystems/ezpublish-kernel/pull/949

cjw #ezsummer

Multi-Site/Multi-Repository Setup

Why a Multi-Site/Multi-Repository Setup?

· At CJW Network we have developed a multi-site/multi-repository setup for eZ Publish 4 several years ago

· This allows us to host many individual sites on a single eZ Publish installation

Advantages:

· Central site administration (site activation, cronjobs, …)

· Easy deployment (update site extension with Subversion)

· Highly reduced maintenance costs (security patches, upgrades)

· Highly efficient use of hardware resources

Disadvantages:

· Some Kernel patches needed

Multi-Site/Multi-Repository Setup in eZ 5

First Approach (proven in production)

- Use different `ezpublish` directories to host the different sites

Second approach (under development)

- Use `CJW MultiSiteBundle`

TODO EKKE

Multi-Site-Setup (old) Directory structure

```
ezpublish                   <-- not used
ezpublish_legacy
└extension
 └site_customer             <-- each customer has its own extension and database
 └site_customertwo
└var
 └site_customer             <-- each customer has its own var directory
 └site_customertwo
site_customer               <-- each customer has its own Symfony app
site_customertwo
src
└CjwNetwork
 └SiteCustomerBundle        <-- each customer has its own bundle
 └SiteCustomertwoBundle
```

cjw #ezsummer

41/53

Multi-Site-Setup (old) Detail ezpublish_legacy

```
ezpublish_legacy
∟extension
 ∟site_customer          <-- each customer has its own extension
  ∟classes
  ∟design
  ∟modules
  ∟settings
   ∟site.ini
    ∟[DatabaseSettings]  <-- each customer has its own database
     ∟Database=database_site_customer
 ∟site_customertwo
  ∟[...]
  ∟settings
   ∟site.ini
    ∟[DatabaseSettings]
     ∟Database=database_site_customertwo
∟var
 ∟site_customer          <-- each customer has its own var directory
 ∟site_customertwo
```

cjw #ezsummer   42/53

Multi-Site-Setup (old) Directory structure

```
site_customer
└autoload.php
└bootstrap.php.cache
└cache
└check.php
└config     <-- all yml files like ezpublish folder, (to improve)
 └config.yml
 └ezpublish.yml
 └ parameters.yml
└console
└logs
└phpunit.xml.dist
└Resources
└sessions
└SiteCjwbaseCache.php
└SiteCjwbaseKernel.php
└SymfonyRequirements.php
```

cjw #ezsummer   43/53

END TODO EKKE

Introducing CJW MultiSiteBundle

Although the first approach works fine, it has several drawbacks:

- Application code scattered at different places (site directory, bundle, legacy extension), hard to maintain in VCS, hard to deploy

- Redundancy in config files

- No global settings

- No central site activation/administration

- Goal: keep everything in one place!

CJW MultiSiteBundle Features

- Boots kernel and environment based on domain name mappings

- Handles local, staging and live domain names

- Allows for global activation of bundles

- Allows for global settings

- Provides a common console for all sites

- Caches domain name mappings

- Moves cache and log files away from the ezpublish folder

- more to come ...

cjw #ezsummer   46/53

cjwpublish Directory

The `cjwpublish` application directory sits next to the `ezpublish` directory.

```
cjwpublish
└ config
    cjwpublish.yml                  <-- defines active bundles
    config.yml                      <-- allows for global settings
  CjwPublishKernel.php              <-- inherits from CjwMultiSiteKernel.php
  CjwPublishCache.php               <-- inherits from CjwMultiSiteCache.php
  console
```

cjw #ezsummer   47/53

Symfony's app directory is back

Site Bundle Directory Layout

```
src
∟ Cjw
  ∟ SiteExampleBundle
    ∟ app
      ∟ config
          cjwpublish.yml          <-- contains domain mappings
          config.yml
          ezpublish.yml
          ...
        CjwSiteExampleKernel.php   <-- inherits from CjwPublishKernel.php
        CjwSiteExampleCache.php    <-- inherits from CjwPublishCache.php
    ∟ Controller
      ...
```

cjw #ezsummer   48/53

Caveats

Adjustments needed in `config.yml` to reflect different relative location of kernel

```yml
assetic:
    ...
    read_from:      %kernel.root_dir%/../../../../web
    write_to:       %kernel.root_dir%/../../../../web
    ...
ez_publish_legacy:
    ...
    root_dir: %kernel.root_dir%/../../../../ezpublish_legacy

parameters:
    ezpublish.kernel.root_dir: %kernel.root_dir%/../../../../vendor/ezsystems/ezpublish-kernel
```

More problems of this kind expected!

Project Status

- Currently in private Beta, not yet released

- Ideas and Feedback welcome

- Public Beta in October

- info@cjw-network.com

- https://github.com/cjw-network/MultiSiteBundle

cjw #ezsummer

01.09.14 10:34

Ressources

Slides as PDF

- see `src/Cjw/SummerCampBundle/Resources/doc`
- https://github.com/cjw-network/SummerCampBundle/Resources/doc

Slides (Source)

- https://github.com/dfritschy/cjw-summercamp-slides

CJW MultiSiteBundle

- https://github.com/cjw-network/MultiSiteBundle
- info@cjw-network.com

cjw

http://vote.netgenlabs.com/

Please Vote!

<Thank You!>

Ekkehard Dörre
http://share.ez.no/community/profile/7431
@ekkeD
http://www.coolscreen.de

Donat Fritschy
https://github.com/dfritschy
http://share.ez.no/community/profile/10451
@webmanufaktur
http://www.webmanufaktur.ch

cjw