Intro for eZ Publish 4.x/3.x developers

This document is not meant to be a comprehensive guide to either Symfony 2 or eZ Publish 5, but to give an overview of how common concepts and tasks used in older versions translate into the new one.

Glossary

New names of existing concepts:

4.x	5.x
(Content) Class	Content Type
(Content) Class Group	Content Type Group
(Content) Class Attribute	Field Definition
(Content) Object	Content (Meta info in: Content Info)
(Content Object) Version	Version Info
(Content Object) Attribute	Field
(Content Object) Attribute content	Field Value
Datatype	Field Type
Node	Location

The reason for renaming some of the concepts in eZ Publish 5.x was to use terms which might be more familiar to users coming from other CMS and avoid confusion with terms used in OOP programming.

New concepts:

> You can also refer to Symfony glossary page as many of the following concepts are the same or taken from Symfony.

5.x	Description
Controller	An OOP version of what was called "**module**" in 4.x, **should extend** `eZ\Bundle\EzPublishCoreBundle\Controller`. A good start is to read Symfony documentation on this topic.
Action	The method on the Controller, similarly to what was referred to as "**view**" in 4.x, contains business logic bootstrap (business logic should be done in services).
View	The **template** that displays the result of the "action", can be either .twig (default) or .php. Note that **Twig is highly recommended** as it has eZ Publish specific helpers.
Service	A Service is a generic term for any PHP object that performs a specific task and thus contains business logic. Services are instantiated and maintained a the *service container*.
Service Container	A Service Container, also known as a *Dependency Injection Container*, is a special object that manages the instantiation of services inside an application. Instead of creating services directly, the developer trains the service container (via configuration) on how to create the services. See Symfony documentation on the service container.
Render	A function that allows you to *Embed* other controller calls, as in *Hierarchically* render other controllers and embed it in your result. This effectively removes the need for having business logic in template using **fetch** functions, session management and so on.
Bundle	An "**extension**" in Symfony Framework and hence the extensions of eZ Publish 5.x kernel.

Console	`ezpublish/console` is the starting point of all console commands in Symfony2 (referred to as `app/console` in Symfony2 documentation) and eZ Publish 5.x kernel, use `php ezpublish/console -h` from the root of your install using command line to get started.
web/	The public folder of your site, two console commands exists to symlink/hard-copy *Bundles* and *Legacy* assets respectively. Front controllers (index.php/index_dev.php) can be modified to fit your needs.
ezpublish/	The application folder (aka `app/` in the Symfony documentation). It contains the main Kernel class where you can declare the bundles you want to use. You will also find `cache/`, `logs/` and main `config/` folders. It can also contain application-level resource files, such as global templates (`ezpublish/Resources/`).
src/	This is where your application code stands (from your own bundles)
vendor/	In this directory stand all your dependencies (3rd party bundles/libs, eZ Publish codebase, Symfony codebase...)

Existing concepts FAQ

Settings

- See http://confluence.ez.no/display/EZP52/Configuration
- **New-style settings:**
  - In a single yml file: **ezpublish.yml** for eZ Publish, **config.yml** for other bundles (it is possible to import other files from here as well).
    Note that it's possible to specialize settings for specific environments (e.g. `ezpublish_dev.yml`, `config_dev.yml`)
  - 3 scopes (**global** == **override**; extension scope is gone)
  - Order of precedence between extensions (bundles) is set by bundle loading order.
  - Runtime siteaccess configuration is resolved through the ConfigResolver service.
- **Some notes about generic (non-eZ) settings for Symfony**
  - Symfony supports 3 formats for config files: **yml**, **xml**, **php** (eZ recommends **yml**)
  - A single configuration file exists: **config.yml** (at least, in the final, compiled version, many files exist on disk)
  - A configuration file can include another one via the imports statement (e.g. `config_dev.yml` would import `config.yml`)
  - When creating a bundle, it is not sufficient to add a config file to it, the developer needs to specify that it is to be loaded (either via an import statement from app/config/config.yml or via a ServiceContainer extension in php).
    Note that if you use the generator bundle command, all needed code will be automatically generated for you.

Siteaccesses

- See http://confluence.ez.no/display/EZP52/Siteaccess+Matching
- Mostly unchanged
- Better way of not duplicating settings: "groups" of siteaccesses
- Siteaccess settings *specializes* siteaccess group settings.
- New way to define siteaccess: via environment variable (super-fast)

  Q: Apart from host url and port, are any other matchers available (host+url, …)

  A: working on an improved fleximatcher witch will be part of 5.1, this includes host+uri combined matching

- Possible now to define custom siteaccess matchers.

Urls

- More flexible routing system.
- View parameters are still present.
- Configuration done in yml file.
- Making params optional is done by giving them a default value in the config file

Q: How is the siteaccess part of urls managed (standard Sf code has no concept for this)?

A: The eZP stack uses an enhanced router. This is transparent to developers.

Q: Support for linking across siteaccess?

A: Not yet possibly in 5.0.

Designs

Q: How is the concept of designs + fallback supported ?

A: Planned for 5.future. Might reuse an Sf bundle from Liip which allows *themes* (theme == design).

- **Side question:** Can a tpl in one bundle override a tpl from another one?
- **A:** Yes, but via bundle inheritance a bundle can have only 1 child – clearly not powerful enough compared to eZP4, but there seems to be progress in Symfony about improving this.

Templates (twig)

- See Twig documentation and templating basics in Symfony.
- For extending Twig, refer to Twig developer documentation and Symfony cookbook.

i18n

Refer to the translation section in Symfony documentation.

Loading css, js

Best practice is to use **Assetic**.

Caching

Content cache is fully Http cache.

Q: How to clear caches for eZ Publish 5? Is there a single action to clear caches for both stacks?

A: Content (Http) cache is purged at publish time or from admin interface. You can use the `cache:clear` command to clear all Symfony cache. No unified command yet.

4-in-5

Q: Will it always be possible to run the Legacy Stack as a standalone app?

A: Yes.

Q: Is there any rewrite rule needed to access var/storage/images of old stack from within the vhost of the new stack?

A: No, this is achieved by *symlinking* to `web/` the `design/`, `extension/`, `share/` and `var/` directories of the eZ Publish legacy stack (done via `ezpublish:legacy:assets_install` script).

The official name for the eZ Publish 4.x stack included within eZ Publish 5.x is not eZ Publish 4.x but eZ Publish Legacy (+"Stack" in case of eZ Publish+extensions).

Extensions

Q: Implemented as bundles?

A: Yes.

Q: Declared as services with a special tag or naming convention? (see e.g. How twig registers stuff).

A: Reuse as much as possible from Sf, do not enforce standards unless needed.

Q: Will need naming convention?

Q: Installed via composer?

A: Yes.

Q: What part of a website to put in `ezpublish/`, what part to put in `src/`? (equivalent of eZ “all in extension” dev philosophy)

A: `ezpublish/` is for global configuration/templates, `src/` is for site-specific developments.

- `ezpublish_legacy/` is at root level because it is considered as an application in itself.
- All current extensions will be renamed with LS suffix (for Legacy Stack) to distinguish them from new implementations.

Q: Best practices for naming php classes (filenames and directories)?

A: Follow PSR-0, with namespaces.
See http://symfony.com/doc/2.0/cookbook/bundles/best_practices.html

Debugging

- Debug toolbar + profiler === DebugOutput
- More info: http://symfony.com/doc/current/book/internals.html
- Sending log messages:

```
$logger = $this->get('logger');

$logger->info('We just got the logger');
$logger->err('An error occurred');
```

Testing

Refer to Symfony documentation on testing and PHPUnit.