eZ Publish 5 Architecture - Introduction & Overview

eZ Publish Technical Architecture

With the 5.2 release eZ Publish is making an important leap forwards in terms of technology.
This document will explain why we are renewing our technology platform, and to some degree explain what this evolution in architecture means to eZ Publish developers and users, and last but not least how eZ Systems is affected by these changes.

Why this change in technology

The background for the changes is to meet the challenges ahead:

- Customer and User Experience Management requirements: Beyond simple web content management, enable the building of any digital user experience
- Performance and scalability: Deliver on the ever increasing need for performance and scalability
- The big data situation: Embrace big data situation as an opportunity to build new digital services and not view it as a hurdle
- Fulfill the multichannel vision: Enable content to live in any screen, any app, any device

The overall goal is to reach Digital Service Excellence, ie, we would like our partners to be able to offer the customers exactly what they want, within acceptable timeframes and price tags on the eZ Publish platform.

What are the changes?

eZ Systems is now introducing the following:

- An extended and sustainable Public API. This will speed up developments on the platform and improve their quality and maintainability both for eZ core developers and extension developers, which in turn will lead to more efficient implementation projects
- An improved REST API with read and write functionality and support all the core content management functionality. This way, eZ Publish will better integrate with any programming framework, for instance into mobile applications or any other web application. Better meaning faster and simpler. in order to fulfill the multichannel vision and execute on the User Experience demands and needs of customers and users of the eZ Publish platform

- Introducing Symfony as the PHP framework under the platform to make developers lives easier and to make developments on eZ Publish accessible for more developers to further support innovation in eZ Publish
- Introducing TWIG as the template engine to simplify working with templates in eZ Publish. And as a standard template engine this will also make eZ Publish templating accessible to more developers
- Introducing a new storage system for scalability, performance and maintainability reasons

And as a result of all these changes, a major last item is

- Introducing a backward compatibility between the new architecture resulting from the above and the legacy architecture as known in eZ Publish 4.x

What will be gained?

Symfony2

With Symfony as the web framework eZ Publish will be more accessible. Thanks to the framework, it will be used for instance to extend eZ Publish applications with new features, potentially not based at all on the content repository (for example, business specific application logic) using a standard PHP framework and a very clean application design with minimum interlocking with eZ Publish itself.

Any developer knowing Symfony will then be able to easily develop extensions for eZ Publish.

This will cater for:

- a better quality of the code
- a lower entry point to do developments in eZ Publish
- more innovation faster since more developers will use a standard PHP framework
- Symfony has an open-source community that will help developers
- Symfony is commercially backed up, so they are in it for the long run and Sensio Lab (maker of Symfony) will naturally extend the eZ offering to the framework level if need be.

TWIG

Initially, eZ Systems has developed its own templating engine. eZ has also worked and redeveloped a second one at some point as part of the eZ Component project. When designing eZ Publish 5, we knew the old template engine was not good enough anymore and needed to be replaced. We knew the eZ Components template engine we contributed and originated was an option, but we decided to go for another one: TWIG. Reasons are multiple: quality of course (even if the eZ components one was also very high quality), integration and cohesion to use it with the Symfony stack and finally again the strength and size of the community using it today. Symfony has developed TWIG as a standard template engine, and eZ Systems is using it to further the following:

- to update the legacy template engine
- to simplify template coding in eZ Publish
- to have faster page rendering
- to benefit of more features which were not in the original engine
- to make extension developments easier to speed up innovation in eZ Publish

This is a move that will influence every stakeholder in the eZ Publish development. As developers will implement faster, this also mean project will significantly improve in total cost of ownership as well as in time to market.

New storage systems

In order to meet performance and scalability requirements, eZ introduces new storage systems with the version 5 serie. In 5.2, this storage system lives beside the legacy storage system and data model, but will use the new API to access the data. Also in 5.2 version, this new storage engine only support Mysql relational database, nevertheless it is designed to allow the development of drivers for other storage engines through the Persistence SPI (service provider interface) and in the future will include drivers for NoSQL and Document based storage engine. The ultimate goal is to open for custom storage developments.

Rest API

In order to meet all multi-channel requirements we are developing a Rest API that cover all core feature of content management so we can integrate with any application on any channel in any programming language. The gain for users will be for anyone integrating eZ Publish with other applications, not only the development will be significantly improved but more importantly, the value of an API also lies in the maintainability and sustainability it offers. the new rest api is designed to stay and will remain identical in all future 5.x version. this means that development done on top of the api will seamlessly support eZ Publish version upgrades.

Php API

The PHP API, also called Public API, is the development glue and will a create shield between internal and external developments on eZ Publish. This will cater for an easier maintainability of code and speed up the performance of eZ Publish. PHP developers will experience a better extensibility which in turn will enable them to create extensions to eZ Publish faster and easier.

The Public API is key to development speed, shorter projects and better quality. Important to be noted: the php api is the foundation for the rest api and the second is naturally relying on the 1st.

Compatibility with the legacy architecture

When we introduce changes of this magnitude, eZ Systems as an international software house must also consider the reality of the installed customer base. Every installation must be able to take care of the old and create on the new architecture. The reason for change is of course to be able to meet new requirements and the need to enable progressive changes.

Understanding the architecture in more detail

The target architecture

The first important thing to understand about the new architecture is to explain it standalone, without considering the old legacy architecture. The following diagram shows a simplistic view of this new architecture.

The new architecture is layered and uses clearly defined API's between the layers.

- The business logic is defined in a new kernel. This business logic is exposed to applications via an API (the Public API). Developers rely on this to develop websites and web applications using Symfony to organize the way they develop the user interface layer.
- User interfaces are developed using the TWIG template engine but directly querying the public API.
- Integration of eZ Publish in other applications are done using the REST API, which itself relies also on the Public API.
- finally development of extensions of eZ Publish is done using the Symfony framework when it comes to the structure of the code, and once again relying on the Public API when it comes to accessing content management functions

To a lower level, the new architecture also totally redefined the way the system store data. while this is not finalized in version 5.2 (where the new storage system is only shipped with mysql support), the architecture, when finalized will rely on a storage api that will be used to develop drivers to any kind of storage subsystem.

A motto for this new architecture is to heavily use api's that will be maintained on the long term to ease upgrades and provide lossless couplings between each part of the architecture, improving the migration capabilities of the system at the same time.

The "real" version 5 architecture

The chapter above is only explaining the new architecture but, as mentioned below, version 5 also offers a way to run the legacy eZ Publish stack, in order to simplify upgrade and switch to version 5. This result in the end in a more sophisticated architecture that is illustrated in the diagram below.

The main difference is, the cohabitation between the new architecture explained in the previous chapter (on the right) and the previous architecture (on the left).

If we look at the old architecture, we can see that it is more monolithic: no defined public php api, a business logic implemented in the kernel but very dependent of the storage system and the underlying data model, an existing rest api but limited to read access to the content repository.

This whole legacy architecture is in its whole included with version 5, and can be used as is.

this means that, for people having developed 4.x websites and that are reluctant to invest time in migrating or even learning the new architecture components, they can use version 5 exactly as they were using version 4. Even the controller (access to the application through the web server) can totally bypass the new architecture (in that case the Symfony framework controller) and directly call the legacy eZ Publish controller and the legacy template engine.

On its side, the new architecture has been implemented, and eZ will implement new features and applications on top of it subsequently. So, as part of 5.2, the new architecture is in place, but does not provide yet the full application scope.

What is more interesting to understand is how these two integrate:

First on the presentation side, the new eZ Publish 5 controller makes it possible to serve pages and functions that are either resulting from the new template engine or the legacy template engine. This is a first level of dual compatibility that will help developers in a smooth transition from one architecture to the other, starting with legacy templates and progressively replacing them with templates for the new system, TWIG.

Second, on the api side, the Public API has been designed to work against the business logic and to be used either on top of the legacy storage or on top of the new storage system. This means that, by implementing the new architecture and embracing the php public api, developers enable an easy transition from the old data model to the new one. An extension developed on top of the Public API will equally work on an old content repository or on a brand new one based on the new architecture.

These two ways to implement a compatibility between the past architecture and the new one offers a wide range of possibilities and a smooth transition path.

Summary on the ways to use eZ Publish 5.2

Using eZ Publish 5.2 in full legacy mode

This way is the less disruptive. In this way, eZ Publish 5.2 totally behave as if it was an eZ Publish 4.7, or we should say 4.8. This is ideal for users who have large existing applications with large amount of data and who are not willing to invest in learning and migrating them immediately.

In this way, even the siteaccess and vhost configuration bypass the legacy stack, and developers will see almost no differences.

Using ez publish 5.2 through the legacy stack but relying on the new controller and new template system as well as the new kernel.

This way offers a transition and allows to combine old template and new templates in the same application. In this case, the users will rely on the administration interface of eZ Publish as well as on the ez tool bar for front-end editing, through the legacy templates, but the front end will be

either based on legacy or new twig based templates.

In this model, the two kernels can be used and the system can this way benefit from the Public API and the new REST API built on top.

Using the brand new architecture only

This case is the one that will deliver very strong improvements in scalability and performance, in this case the whole new architecture is used and there is no way to reuse components from the legacy architecture.

This means that:

- the administration interface is not available in 5.2
- existing templates and site won’t run without having been migrated
- the old storage system is not used any more

While this might sound restricting for the time being, it is clearly the foundation of the future of eZ Publish. In the context of eZ Publish 5, it can be useful for new projects relying only on the concept of "content as a service" the platform is a high performance and scalability content repository with very advanced services but provide no editorial user interface. for traditional content management.