Service Container

Definition

A service container, (aka **DIC**, *Dependency Injection Container*) is a special object that greatly facilitates dependencies resolution in your application and sits on Dependency Injection design pattern. Basically, this design pattern proposes to inject all needed objects and configuration into your business logic objects (aka **services**). It avoids the massive use of singletons, global variables or complicated factories and thus makes your code much more readable and testable. It avoids magic.

The main issue with dependency injection is how to resolve your dependencies for your services. This is where the service container takes place. The role of a service container is to build and maintain your services and their dependencies. Basically, each time you need a service, you may ask for it to the service container which will either build it with the configuration you provided, or give you the existing instance if it is already available.

In eZ Publish 5

eZ Publish 5 uses Symfony service container .

It is very powerful and highly configurable. We encourage you to read its dedicated documentation as it will help you understand how eZ Publish 5 services are made :

- Introduction and basic usage
- Full documentation of the Dependency Injection Component
- Cookbook
- Base service tags

Service tags

Service tags in Symfony DIC are a useful way to dedicate services to a specific purpose. They are usually used for extension points.

For instance, if you want to register a Twig extension to add custom filters, you will need to create the PHP class and declare it as a service in the DIC configuration with the *twig.extension* tag (see Symfony cookbook entry for a full example).

eZ Publish 5 exposes several features this way (see the list of core service tags). This is for example the case for Field Types.

You will find all service tags exposed by Symfony in its reference documentation.

Related pages

- Service tags in eZ Publish 5