**Documentation En ligne**

1. Starting
2. Plugin installation
3. Structure idéal
   1. resources
      1. config
      2. translations
      3. view
4. Container et services
   1. Définition (container et service)

Le container de service est un object PHP qui vous aide à gérer et à centraliser vos class et object. Voir <http://symfony.com/doc/current/components/dependency_injection.html>

Un service n’est rien d’autre qu’un object php. Ici la liste des services

* 1. Comment utilisé le container et ses services
  2. Ajouter ses propres services (avec un fichier de config en YAML, xml ou php)
     1. Config file (Yaml, php or xml)

Un certains nombre de services (comme le service Mailer) existe déjà pour vous. Mais vous pouvez tout aussi bien créer vos propres service au sein de votre plugin ou de votre thème. Pour ce faire vous devez dans un premier temps créer vos class (avec ou sans namespance). Dans un second temps vous devez utilisez l’object classLoader (class du paquage composer) pour charger vos class dans la pile d’autoload PHP. A la toute fin, vous devefinissez vos class en tant que service dans un fichier de configuration (en Yaml, php ou xml). Il vous suffit de charger ces fichiers de configuration à l’aide du service loader\_factory.

* + 1. Enregistrer les class avec ClassLoader (obligatoire)
    2. Definir les services et les parametres
       1. Options disponibles
       2. YAML
       3. PHP
       4. XML
    3. Utiliser son service

1. ClassLoader
   1. Definition : Class loader is use to autoload your class with namespace or without namespace.
2. LoaderFactory
   1. Defintion : Ce service est à utilisé seulement pour charger les fichiers de configuration qui contiennent les services (class, shortcode etc) et les parameters.
3. Controller
   1. Définition

Tous les controller ont le container. Vous ne pouvez pas instancier vous même un controller, c’est le container de service qui s’en charge.

* 1. Create your own controller who extend Base Controller

1. ShortcodeController
   1. Definition : a shortcode is a service (a ShortcodeController class) with wp.shortcode tag and is load automatically by container.
   2. Create your shortcode class
      1. Create your ShortcodeClass who extend BaseShortcodeController and create shortcodeAction function.
      2. Define your shortcodeController as a service. Le shortcode est automatiquement instancié (sauf si il est marqué comme synthetic) dès que vous chargez votre fichier de configuration. Le nom du shortcode correspond au nom du service.
      3. Warning : you must load config file. See 4.C
2. Translation. See http://symfony.com/doc/current/components/translation.html
   1. Définition : translation avaible i yaml, xliff and php
   2. The translator
3. Widget
   1. Definition
   2. Create your widget class who extend \WP\_Widget
   3. Define your widget as service, taged wp.widget and with arguments in order
4. Validation
   1. Definition. Avaibles validation loader is Yaml and Xml
   2. Get validator (for use xml and yaml config file) or defaultValidator (annotation or metadata) with VaidatorFactory
   3. The validator http://symfony.com/doc/current/validation.html
      1. Validation object Annotation
      2. Validation by Yaml
      3. Validation by Xml
      4. Validation by Php (MetaData). You must create a static method ‘loadValidatorMetadata’ to validate your object.
   4. Define your validator as service
5. Form
   1. Definition
   2. Create formBuilder with FormFactory : with defaultValidator (annotation or metadata) or your validator (default validator + Yaml or xml). In parameters must containt validator who validate your form. This validator can be a validator service.
   3. Define your formfactory as service
6. Templating
   1. Definition : there are two templating implementation : TwigEngine and PhpEngine
   2. How get templating and register your templating
      1. Register template with twig
      2. Register template with PHP
   3. Templating with Twig (conseillé)
   4. Templating with Php
7. Mailer
8. RepositoryManager