

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apollina Saint Ouen |  | PAQUET Judicaël |  | Judicael.paquet@gmail.com  https://github.com/las93 |

Apollina PHP

[Le nouveau système de templating PHP]

Table des matières

1. [Présentation 2](#_Toc409472878)

[Installation](#_Toc409472879)

[Concept du routeur](#_Toc409472880)

# Présentation

Apollina est une nouvelle alternative de système de templating de qualité de développement open-source dans les environnements LAMP.

## Installation

L’installation du système de templating est relativement simple. Vous pouvez télécharger la version .zip sur le Github <https://github.com/las93> ou ajouter las93/apollina dans votre composer.json.

## Concept du routeur

Xxx

# Utilisation basique

Pour bien utiliser Attila, voici les premières étapes à respecter afin de proposer le meilleur des ORM à vos différents projets.

## XXX

XXX l’ORM

# Utilisation avancée

Voici la liste des utilisations avancées d’Apollina qui pourront beaucoup vous aider dans vos différents projets.

## Indiquer le niveau de rendu

**Disponibilité : Apollina 3.0.0-beta2**

Cette nouvelle version d’Apollina (par rapport à la version intégrée à Venus 1) vous permet d’indiquer un niveau de rendu. Cela permet par exemple de ne pas afficher le layout si vous voulez par exemple juste charger le contenu central en Ajax.

Voici comment le redéfinir dans votre contrôleur :

|  |
| --- |
| $this->view->setRenderLevel(\Apollina\Template::LEVEL\_NO\_RENDER); |

Nous avons aujourd’hui les trois niveaux présentés ci-dessous :

|  |
| --- |
| /\*\*  \* constant to define the render level  \* **@var** int  \*/  **const** LEVEL\_NO\_RENDER = 0;  **const** LEVEL\_VIEW = 1;  **const** LEVEL\_LAYOUT = 2; |

Voici comment l’utiliser par exemple dans Venus 2 ($this->layout représente une instance d’Apollina où l’on a indiqué qu’on était en mode layout) où on ne désire pas afficher le layout mais seulement les vues :

|  |
| --- |
| $this->layout  ->setRenderLevel(\Apollina\Template::LEVEL\_VIEW)  ->display(); |

Vous pouvez également désactiver les couches que vous désirez comme ceci (vous pouvez potentiellement en désactiver plusieurs dans le tableau) :

|  |
| --- |
| $this->layout  ->disableLevel(array(\Apollina\Template::LEVEL\_VIEW => 1))  ->display(); |

## Ajouter de la traduction I18n

**Disponibilité : Apollina 3.0.0-beta2**

Sur Apollina, on peut surcharger notre I18n pour par exemple avoir plusieurs fichiers de traductions qui fonctionnent en même temps.

Nous allons présenter la méthode utilisée dans Venus2 qui utilise Apollina afin de rajouter les traductions de chaque plugin (les traductions sont sur plusieurs fichiers) :

|  |
| --- |
| **foreach** (Config::*get*('Plugins')->list **as** $iKey => $sPlugin) {    **if** (file\_exists(\_\_DIR\_\_.DIRECTORY\_SEPARATOR.'..'.DIRECTORY\_SEPARATOR.'src'.DIRECTORY\_SEPARATOR.'plugins'.DIRECTORY\_SEPARATOR.$sPlugin.DIRECTORY\_SEPARATOR.'i18n'.DIRECTORY\_SEPARATOR.$this->getLanguage().$this->getIntermediaiteDirectory().$sPlugin.'.json')) {  $oJson = json\_decode(file\_get\_contents(\_\_DIR\_\_.DIRECTORY\_SEPARATOR.'..'.DIRECTORY\_SEPARATOR.'src'.DIRECTORY\_SEPARATOR.'plugins'.DIRECTORY\_SEPARATOR.$sPlugin.DIRECTORY\_SEPARATOR.'i18n'.DIRECTORY\_SEPARATOR.$this->getLanguage().$this->getIntermediaiteDirectory().$sPlugin.'.json'));    $fCallBack = **function**($sValue) **use** ($oJson)  {  **if** (**isset**($oJson->$sValue)) { **return** $oJson->$sValue; }  **else** { **return** $sValue; }  };  }    CoreI18n::*addCallback*($sPlugin, $fCallBack);  } |

On fait une fonction de callback que l’on rajoute pour chaque plugin qui permet d’aller chercher la bonne valeur dans un json. Cette méthode exige de bien maitriser les fonctions anonymes mais sera très pratique si vous ajoutez Apollina dans votre framework.