



JoomlaDayTM

FRANCE - 9 et 10 mai 2015

NICE

Twitter Hashtag
#jd15fr

organisé par

AFUJ
Association Francophone
des Utilisateurs de Joomla!®



Développement avancé avec le Framework Joomla



Par Marc Studer et Jérôme Glatigny

Sommaire

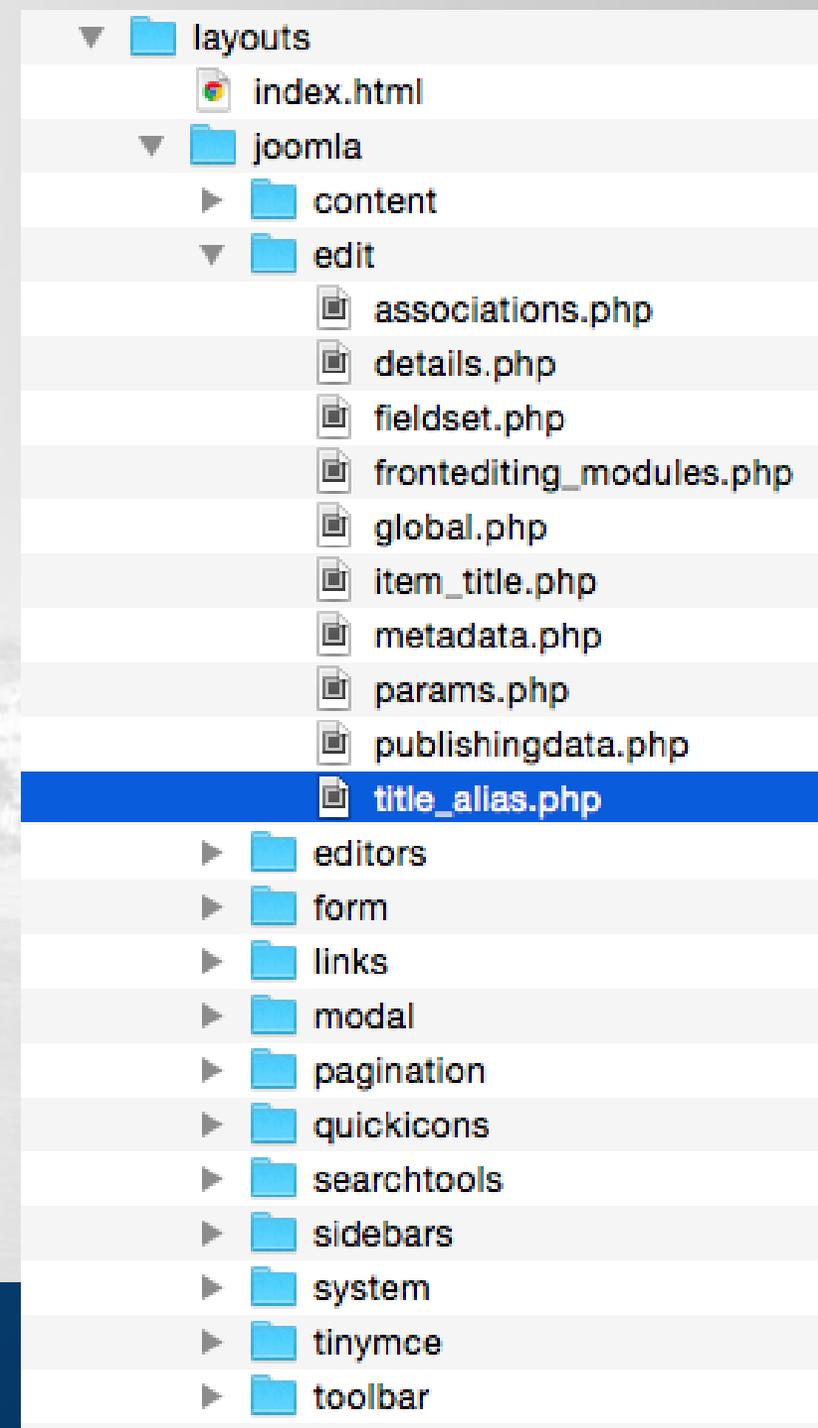
1. Le Framework de Joomla
2. Objets avancés de l'API Joomla
 1. Les JLayout
 2. La balise <media>
 3. Packages
 4. Paramétrage XML
3. Développement d'une extension "flexible"
4. Vos attentes / actions communautaires

Le Framework de Joomla

- Le Framework de Joomla 3 offre aux développeurs de nouvelles API pour simplifier, fiabiliser et optimiser la conception d'extension.
- **Etude de cas** : architecture de développements d'une extension.
L'intérêt de miser sur les évènements et groupes de plugins

Les JLayout

- Nouveau dossier dans Joomla : /layouts
- Utilisation des **JLayout** dans les TMPL des vues admin.



Les JLayout

```
<?php
```

```
    echo JLayoutHelper::render(  
        'joomla.edit.title_alias', $this  
    );
```

```
?>
```



The screenshot shows the Joomla! administration interface for editing an article. At the top, a dark blue header contains a pencil icon and the text "Gestion des articles : Modifier un article". Below this is a row of six buttons: "Enregistrer" (green), "Enregistrer & Fermer" (green checkmark), "Enregistrer & Nouveau" (green plus), "Enregistrer une copie" (green document), "Versions" (grey folder), and "Fermer" (red X). The main content area has two input fields: "Titre *" with the value "A propos de ce site" and "Alias" with the value "a-propos". Below these fields is a horizontal menu with tabs: "Contenu" (selected), "Publication", "Images et liens", "Paramètres", "Paramètres de création/modification", and "Droits".

Les JLayout

- joomla.edit.title_alias

```
$form = $displayData->getForm();  
  
$title = $form->getField('title') ? 'title' : ($form->getField('name') ? 'name' : '');  
  
>  
<div class="form-inline form-inline-header">  
    <?php  
        echo $title ? $form->renderField($title) : "";  
        echo $form->renderField('alias');  
    ?>  
</div>
```

- Attention : l'utilisation de ces Layouts pour afficher des blocs de données « pré-codés », nécessite d'utiliser un modèle de données « **normalisé Joomla** » !

La balise <media>

Eradication totale des accès HTTP sur les dossiers des extensions !!!

- Les objets clients (images, CSS, JS) doivent être stockés dans le dossier **/media** du site Joomla
- l'implémentation de la balise XML 'media' pour un code "safe"
- A utiliser pour toutes les extensions :
 - Composants
 - Modules
 - Plugins

La balise <media>

- Attributs :
 - **folder** : dossier source
 - **destination** : le nom de l'extension (avec prefixe)

```
<media folder= « media » destination= « com_joomladay »>  
  <folder>js</folder>  
  <folder>css</folder>  
  <folder>images</folder>  
</media>
```

- Creation d'un dossier **/media/com_joomladay/js**

Le Package Library

- Création d'une extension de **type « Library »**
- Introduction de la réutilisation du code
- Installation et mise à jour de bibliothèques pour tous vos développements
- Appel via la commande **jimport()**

Le Package Library

- Descripteur de déploiement XML
 - **Type** : library
 - **Libraryname** : dossier
 - **Folder** : sous-dossier de la bibliothèque

/libraries/missing/smart/

```
<?xml version="1.0" encoding="utf-8"?>
<extension type="library" method="upgrade" version="3.0">
  <name>Missing Tools!</name>
  <libraryname>missing</libraryname>
  <author>Joseph LeBlanc</author>
  <version>1.0</version>

  <files>
    <folder>smart</folder>
  </files>
</extension>
```

Le Package Library

- Classe de la library
 - **Fichier** : drops.php
 - **Nom de la classe** :

libraryName+folder+fichier

/libraries/missing/smart/drops.php

```
<?php
defined( '_JEXEC' ) or die;

class MissingSmartDrops
{
    // ...
    // ...
    // ...
}
```

Le Package Library

- Implémentation
 - **import**: chemin du package
 - Instanciation de la classe
 - ...

```
<?php
defined( '_JEXEC' ) or die;
jimport('missing.smart.drops');

$colors = array('Red', 'Yellow', 'Blue');
$color_drop = new MissingSmartDrops('color', $colors);
echo $color_drop;
```

Les Packages de Packages

- Composé d'un zip de zips
- Tout installer en une seule étape !

- **Extensions**

```
<file type="component" id="com_helloworld">com_helloworld.zip</file>
```

- **Plugins**

```
<file type="plugin" id="helloworld" group="system">plg_sys_helloworld.zip</file>
```

- **Modules**

```
<file type="module" id="helloworld" client="site">mod_helloworld.zip</file>
```

- **Librairies**

```
<file type="library" id="helloworld">lib_helloworld.zip</file>
```

- **Templates**

```
<file type="template" id="helloworld" client="site">tpl_helloworld.zip</file>
```

Paramétrage – l'attribut XML ShowOn

« Show on » permet de donner une condition sur l'affichage du champs. Il faut indiquer le nom du champs dans la même section et la valeur souhaitée.

```
<fieldset
  name="ftp"
  label="CONFIG_FTP_SETTINGS_LABEL">
  <field
    name="ftp_enable"
    type="radio"
    class="btn-group btn-group-yesno"
    default="0"
    label="COM_CONFIG_FIELD_FTP_ENABLE_LABEL"
    description="COM_CONFIG_FIELD_FTP_ENABLE_DESC"
    filter="integer">
    <option value="1">JYES</option>
    <option value="0">JNO</option>
  </field>

  <field
    name="ftp_host"
    type="text"
    label="COM_CONFIG_FIELD_FTP_HOST_LABEL"
    description="COM_CONFIG_FIELD_FTP_HOST_DESC"
    filter="string"
    showon="ftp_enable:1"
    size="14" />
</fieldset>
```

The image displays two screenshots of the Joomla! FTP settings interface. The left screenshot shows the 'FTP Settings' section with 'Enable FTP' set to 'No' and 'Enable Proxy' set to 'No'. The right screenshot shows the same settings, but 'Enable FTP' is now set to 'Yes', and the 'FTP Host' field is visible and empty. The 'Proxy Settings' section is partially visible at the bottom of the right screenshot.



Paramétrage – Le type de champ Repeatable

- Nouveau type de champ (> J3.2)
- Permet de renseigner des **valeurs multi-valuées** de **groupes de champs**
- Enregistrement des valeurs au format JSON

Nombre maximal de liens

5

Niveau de similitude

Au moins un

Ordre des résultats

Nombre de tags correspond...

Choix des templates



Sélectionner



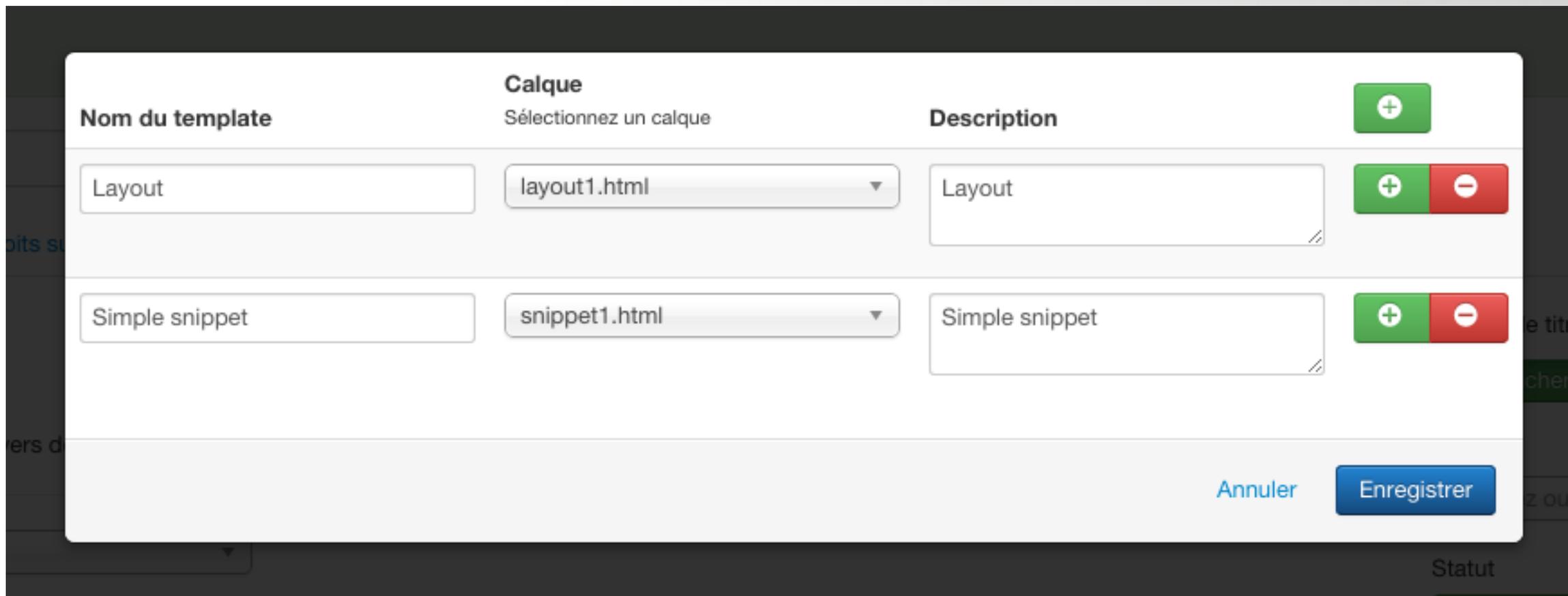
Paramétrage – Le type de champ Repeatable

```
<field name="list_templates" type="Repeatable" label="Choix des templates" icon="list" select="Sélectionner"
    default="{ 'template':['Layout','Simple snippet'],'location':['layout1.html','snippet1.html'],'description':
['HTMLLayout','Simple HTML snippet']}">
    <fields name="params">
        <fieldset name="list_templates_modal" hidden="true" repeat="true">
            <field name="template"
                label="Nom du template" size="30" type="text" />
            <field name="location"
                label="Calque" description="Sélectionnez un calque" size="30" type="filelist"
                directory="media/editors/tinymce/templates"
                exclude="index.html" hide_default="true" hide_none="true" />
            <field name="description du template"
                label="Description" size="40" type="textarea" />
        </fieldset>
    </fields>
</field>
```



Paramétrage – Le type de champ Repeatable

- Ajout et suppression de groupes de champs en lightbox



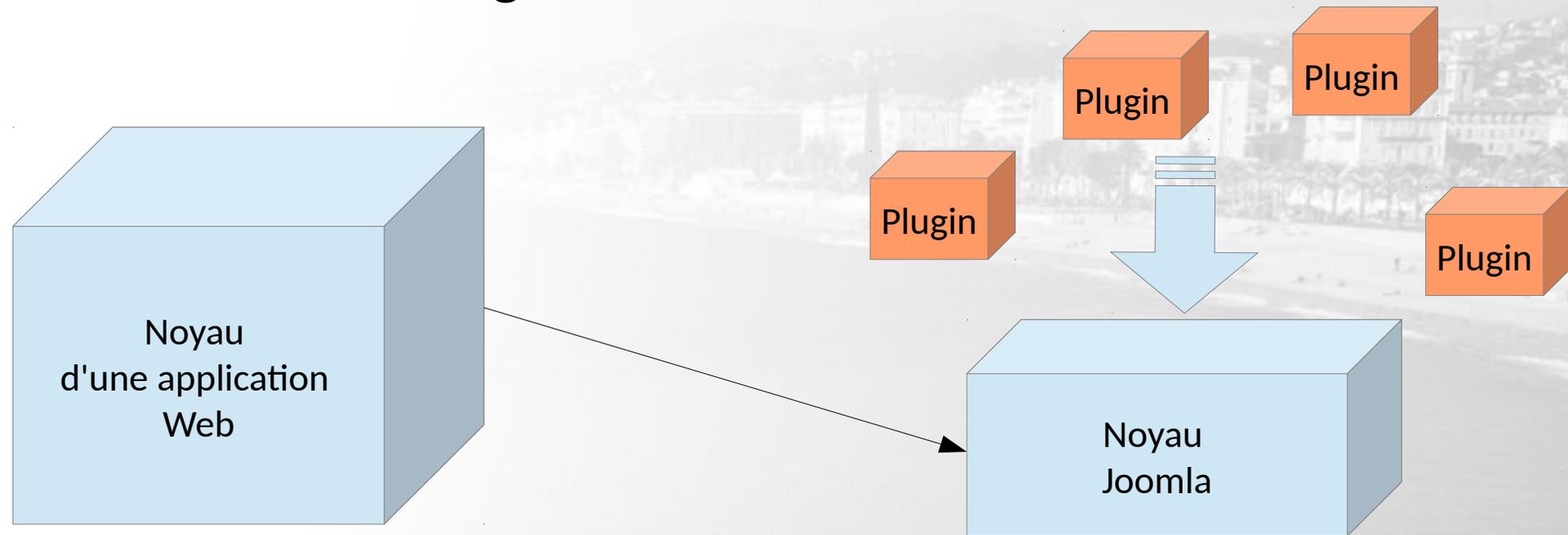
The screenshot shows a lightbox interface for configuring repeatable fields. It features a table with three columns: 'Nom du template', 'Calque', and 'Description'. The 'Calque' column has a dropdown menu with the instruction 'Sélectionnez un calque'. The 'Description' column has a text area and a green '+' button. The first row shows 'Layout' as the template name, 'layout1.html' as the layer, and 'Layout' as the description. The second row shows 'Simple snippet' as the template name, 'snippet1.html' as the layer, and 'Simple snippet' as the description. At the bottom right, there are 'Annuler' and 'Enregistrer' buttons.

Nom du template	Calque Sélectionnez un calque	Description	
Layout	layout1.html	Layout	+
Simple snippet	snippet1.html	Simple snippet	+ -

Annuler Enregistrer

Développement d'une extension "flexible"

- méthodologie de développement d'une extension "flexible"
- Répartition des traitements entre le composant (Noyau applicatif) et des familles de Plugins liées à des évènements



Développement d'une extension "flexible"

- Penser un noyau simple
- Identifier/Séparer les fonctions par processus
 - Les implémenter dans des plugins
 - Les connecter à des évènements
- **Objectif** : Flexibilité du code source du Composant
 - l'ajout de nouvelles fonctionnalité sans provoquer de régression sur le noyau du Composant,
 - la participation de la communauté dans l'amélioration et l'enrichissement des fonctionnalités du Composant.

Développement d'une extension "flexible"

- Les évènements
- Quand ajouter un évènement ?
 - Thème fonctionnel
 - Moment (After/Before)
 - Actions (Save, Delete ...)
- Trigger (déclencheur)
 - Evènement
 - Paramètres (contexte, ...)

```
JPluginHelper::importPlugin('jday');  
...  
$dispatcher = JDispatcher::getInstance();  
$results = $dispatcher->trigger(  
    'onSpeakerBeforeSave',  
    array('com_jday.speaker', &$data, $this->params)  
);
```

Vos attentes / actions communautaires

- 2015-2016 : Réorganisation de la communauté des développeurs Francophones ...
 - Outils ...
 - Projets collaboratifs ...
 - Communication ...
 - Livrables ...
 - ... etc