Développer une extension Typo3

Maxime Fauquemberg

Typo3 est connu comme un puissant outil de gestion de contenu mais c'est aussi un environnement de développement complet et puissant – illustration avec les bases de la création d'un plugin. Découvrez toutes les pistes documentaires pour avancer dans le développement de ses propres extensions Typo3 !

linux@software.com.pl

En intervenant sur des projets Typo3 existant je découvre assez souvent des extensions qui n'ont purement et simplement aucune raison d'exister – commençons donc par les mauvaises raisons d'écrire une extension :

- *Gérer des menus* : il existe tout ce qu'il faut en Typo Script pour créer des menus – du plus simple au plus complexe. Dans le pire des cas vous aurez peut être une facétie graphique à traiter via une userfunc (dans un contexte professionnel ceci doit m'arriver une fois par an !) Donc avant de tenter d'écrire un plugin gérant un menu : aller lire ou relire la TSREF.
- *Réinventer la roue* : ceci peut paraître étonnant, mais sur de nombreux projets Typo3 j'ai découvert avec effroi des modules gestion de templates, de cache ou d'accès aux données ... Tout ceci existe dans Typo3 ! Dans la plupart des cas ces expériences malheureuses étaient dues à une méconnaissance du produit. Songez donc avant de vous lancer dans un développement spécifique à vous documenter sur les fonctionnalités du produit – il existe une abondante littérature sur le sujet et une offre de formation dispensée par des professionnel expérimentés. Autre cas de figure *l'éclair de génie* : vous avez découvert un moyen beaucoup plus intelligent de gérer une fonctionnalité native et vous vous préparez à le coder dans un plugin ... prenez donc une heure ou deux pour laisser votre ego se reposer – profitez en pour lire un peu de documentation (inside typo3 par exemple cité plus bas) ensuite soit vous découvrirez peut être que vous venez de découvrir une innovation importante à apporter au produit mais, et le plus souvent, vous vous rendrez compte que vous apprêtiez à commettre un grosse ânerie !

Cet article explique...

- Les bonnes et les mauvaises raisons d'écrire une extension.
- Comment utiliser l'extension kickstarter pour initier une extension.
- La communication entre typoscript et PHP.
- La gestion du cache sous typo3.

Ce qu'il faut savoir...

- Avoir de solides bases en PHP.
- Savoir utiliser le CMS typo3 : gestion de contenu, création de gabarits, installation et configuration d'extensions...
- Des notions de développement objet.

- *Privatisez une extension communautaire* : vous avez trouvé une extension communautaire qui correspond pour partie à vos besoins mais pour ajouter une fonctionnalité ou modifier une fonction existante vous avez décidé *d'emprunter* le code de cette extension et de l'adapter dans votre propre plugin. Même si ce cas de figure peut se présenter – en principe vous devriez pouvoir vous en sortir avec les outils de personnalisation internes au plugin – ou en étendant la classe d'origine, nous verrons un peu plus bas comment faire ceci.

Quelles sont les bonnes raisons alors ? Déjà de ne pas être dans un des trois cas de figure cités ci-dessus ! Et ensuite d'avoir une vue précise du besoin auquel on veut répondre et la manière dont on veut le traiter – ce qui n'est en fait pas très révolutionnaire en matière de développement.

L'extension exemple : annonces immobilières

L'exemple que je choisis généralement pour présenter les bases du développement de plugin Typo3 est celui de la gestion d'annonce immobilière pour le site d'une agence par exemple. L'exemple vaut ce qu'il vaut, et il existe sans doute des extensions communautaires qui répondraient au besoin, mais il permet en fait d'illustrer la plupart des concepts à connaître pour développer sous Typo3. Le principe est simple, il faut que l'agent immobilier puisse saisir des offres de vente ou de location, et que ces offres puissent décrire des maisons, appartements ou fonds de commerce et comprennent

Typo3 dispose d'un outil interne pour initier le développement d'extensions : le *kickstarter*. Depuis le module *extension manager* – installez si nécessaire l'extension kickstarter – vous disposez ensuite dans le menu déroulant d'une entrée `make new extension`. Nous allons d'abord donner un nom à notre extension et la décrire brièvement (voir Figure 2).

Extension key : saisir une chaîne sans espace ni caractères spéciaux – ce sera la clef de nommage de notre extension. La partie *general info* est destinée à fournir une description de ce que fait votre extension – ceci sera utile si à terme vous décidez de publier cette extension. Pressez `update` après avoir complété les informations. Nous allons avoir besoin d'une table en base de données pour stocker nos annonces – pour l'ajouter il suffit de presser sur le signe + à droite de *new database tables*. On donne un nom à notre table – qui est automatiquement préfixé par le nom de notre extension (voir Figure 3).

Arrivé à ce stade nous allons définir une structure de données simple pour qualifier nos annonces – il nous faudrait les champs suivants, pour tous les enregistrements :

- type de bien : maison, appartement ou fonds de commerce,
- ville,
- prix,
- description du bien avec enrichissement html possible,
- photos avec zoom en popup,
- code postal,
- superficie.

et uniquement pour les maisons :

- jardin : oui / non,
- nombre de niveau.

pour les appartements :

- étage,
- concierge : oui / non.

et pour finir les fonds de commerce :

- chiffre d'affaire annuel,
- type de commerce autorisé : liste d'activité.

En dessous du formulaire de description nous pouvons saisir le premier champ *type de bien* - *Field name* sera le nom de la colonne en base et *Field Title* l'étiquette affichée dans le back office. Pour ce qui est du type de bien nous allons choisir le *Field Type selector box* et le configurer avec notre type de bien comme sur Figure 4.

Dès que nous ajoutons un champ, un emplacement vide apparaît en dessous – nous allons donc procéder ainsi pour chacun des champs voulus.

Les différents types de données dans le *kickstarter* sont les suivants :

- `string input` : un champ de saisie simple sans contrôle particulier,
- `string input` advanced : le même que le précédent avec la possibilité d'ajouter des contrôles via JavaScript dans le back office,
- `Textarea` : un champ texte multiligne sans mise en forme particulière,
- `Textarea with rte` : une zone de saisie en HTML avec le composant htmlarea,
- `checkbox` : une case à cocher seule – permet par exemple de traiter un champ booléen,
- `checkbox 4/10 in a row` : une série de case à cocher – attention dans la mesure ou typo3 stocke la valeur saisie dans une seule colonne de la base de données, la saisie est considérée comme une valeur binaire. Par exemple quatre cases cochées seront évaluées comme 1111 et stockées en base sous la forme décimale 15 – lors de la restitution il faudra bien sur opérer la conversion dans l'autre sens. Ce type de champs n'est bien sûr pas adapté à une recherche SQL sur l'une des valeurs.
- `Link` : un lien interne ou externe.
- `Date / date and time` : format de date stocké en base sous forme de *timestamp*.
- `Integer` : un nombre entier,

Listing 1. Le contenu du dossier d'une extension

```
987846 -rw-r--r-- 1 www-data www-data   78 2009-02-08 18:29 ChangeLog
987725 drwxr-xr-x 2 www-data www-data 4096 2009-02-08 18:02 doc
987861 -rw-r--r-- 1 www-data www-data 1464 2009-02-08 18:29 ext_emconf.php
987848 -rw-r--r-- 1 www-data www-data  124 2009-02-08 18:29 ext_icon.gif
987849 -rw-r--r-- 1 www-data www-data  362 2009-02-08 18:29
       ext_localconf.php
987851 -rw-r--r-- 1 www-data www-data 1120 2009-02-08 18:29 ext_tables.php
987852 -rw-r--r-- 1 www-data www-data  841 2009-02-08 18:29 ext_tables.sql
987853 -rw-r--r-- 1 www-data www-data  135 2009-02-08 18:29
       icon_tx_immo_annonces.gif
987854 -rw-r--r-- 1 www-data www-data 1711 2009-02-08 18:30 locallang_db.xml
987844 drwxr-xr-x 3 www-data www-data 4096 2009-02-08 18:02 pi1
987847 -rw-r--r-- 1 www-data www-data   82 2009-02-08 18:29 README.txt
987855 -rw-r--r-- 1 www-data www-data 4990 2009-02-08 18:29 tca.php
```

- `selector box` : une liste déroulante avec des données statiques,
- `radio boutons` : des boutons radio (choix unique),
- `database relation` : permet de créer une relation entre deux tables de la base de données. Typo3 gère deux types de relations :
  - la relation simple : la table source stocke les identifiants des enregistrements en relation dans une liste d'éléments séparés par des virgules,
  - mm relation : utilise une table de correspondance qui contient les identifiants de chaque ligne en relation.
- `Files` : permet de télécharger des fichiers comme nos images par exemple.

Listing 2. Le TCA ou table configuration Array contient la définition de l'interface

```
...
"typebien" => Array (
                "exclude" => 1,
                        "label" => "LLL:EXT:immo/locallang_db.xml:
                                    tx_immo_annonces.typebien",
                        "config" => Array (
                                "type" => "select",
                                "items" => Array (
                        Array("LLL:EXT:immo/locallang_db.xml:
                                tx_immo_annonces.typebien.I.0", "0"),
        Array("LLL:EXT:immo/locallang_db.xml:
                tx_immo_annonces.typebien.I.1", "1"),
                                Array("LLL:EXT:immo/locallang_
                                      db.xml:tx_immo_annonces.typebien.
                                      I.2", "2"),
                                ),
                                "size" => 1,
                                "maxitems" => 1,
                        )
                ),
...
```

Listing 3. Définition par défaut de la gestion des types dans le TCA

```
"types" => Array (
                "0" => Array("showitem" => "hidden;;1;;1-1-1, typebien,
ville, prix, description;;;richtext[paste|bold|italic|underline|formatbl
ock|class|left|center|right|orderedlist|unorderedlist|outdent|indent|lin
k|image]:rte_transform[mode=ts], photos, codepostal, superficie, jardin,
nombredeniveau, etage, concierge, ca_annuel, tcommauth")
        ),
```

Listing 4. Le TCA modifié pour afficher trois types différents

```
"types" => Array (
        "0" => Array("showitem" => "hidden;;1;;1-1-1, typebien, ville,
prix, description;;;richtext[paste|bold|italic|underline|formatblock|
class|left|center|right|orderedlist|unorderedlist|outdent|indent|link|
image]:rte_transform[mode=ts], photos, codepostal, superficie, jardin,
nombredeniveau"),
        "1" => Array("showitem" => "hidden;;1;;1-1-1, typebien, ville,
prix, description;;;richtext[paste|bold|italic|underline|formatblock|clas
s|left|center|right|orderedlist|unorderedlist|outdent|indent|link|image]:
rte_transform[mode=ts], photos, codepostal, superficie, etage, concierge"),
        "2" => Array("showitem" => "hidden;;1;;1-1-1, typebien, ville,
prix, description;;;richtext[paste|bold|italic|underline|formatblock|class|
left|center|right|orderedlist|unorderedlist|outdent|indent|link|image]:rte_
transform[mode=ts], photos, codepostal, superficie,  ca_annuel, tcommauth"),
        ),
```

Une fois tous les champs créés nous devons préciser comment va se comporter notre table – en haut du formulaire de création de table nous indiquons – cf illustration – que le champ contrôlant qui rechargera l'interface est le champ *type de bien* (voir Figure 5).

Nous verrons un peu plus loin comment affiner notre structure de données notamment à travers la personnalisation du *table configuration array ou TCA* - Pour l'heure nous allons ajouter un plugin d'affichage des données avant de considérer l'édition des fichiers que le kickstarter aura généré.

Le plugin d'affichage

Tout comme pour ajouter une table, pour ajouter un plugin d'affichage il suffit de presser le signe + à droite de *front end plugins* - on donne un titre à notre plugin – c'est ce titre que verra le contributeur lors de l'insertion du contenu dans la page. Le bas de la page est une longue liste à choix unique permettant de définir quelle sera la position de notre composant dans le backend de Typo3 – le choix par défaut – dans la liste des plugins – étant le plus adapté nous reste-

Listing 5. La fonction principale de notre plugin

```
function main($content,$conf)
{
        $this->conf=$conf;

debug($conf,"variable conf");
        $varget = t3lib_div::
          _GET($this->prefixId);
        $template = $this
        ->cObj->fileResource($this
        ->conf['template']);
        if($uid) {
        $content = $this->detail_
          annonce ($uid,$template);
        }else {
        $content = $this->liste_
        annonce ($template);
                }
        return $this->pi_wrapInBase
                Class($content);
}
```

Listing 6. La fonction permettant de lister les annonces

```
function liste_annonce ($template) {
        $templiste = $this->cObj->getSubpart($template,"###LIST###");
        $res = $GLOBALS["TYPO3_DB"]->exec_SELECTquery('*',
              'tx_immo_annonces','pid = '.$this->cObj->data
              ["pages"].$this->cObj->enableFields(
                                    "tx_immo_annonces"),'');
                while ($row = $GLOBALS["TYPO3_DB"]->sql_fetch_assoc($res))
                {
                $confLien=array();
                $confLien['additionalParams']=
                                    "&tx_immo_pi1[uid]=".$row["uid"];
                $confLien['parameter']=$this->cObj->data["uid"];
                $confLien['useCacheHash']=1;
                $confLien['ATagParams']=' class="act_categorie"';
                $row["URLdetail"]=$this->cObj->typoLink
                                          ("detail", $confLien);
                $html .= $this->cObj->substituteMarkerArray
                                          ($templiste,$row,"###|###");
        }
        return $html;
        }
```

rons sur l'option par défaut. Par défaut la case à cocher *USER cObjects are cached. Make it a non-cached USER_INT instead* n'est pas active et c'est très bien ainsi ! La sortie HTML de notre plugin ira dans le cache de typo3 ce qui en terme de performance est le choix idéal.

Pour voir ce que le kicktstarter a généré pour nous – nous appuyons sur le bouton *view result* puis *write* - l'écran permet de lancer l'installation de l'extension et la création de la table dans la base de données. Nous devrions désormais pouvoir créer des enregistrements de type *annonce immobilier*. Pour s'en assurer nous ajoutons dans notre arborescence deux pages de type sysfolder nommées par exemple *vente* et *location* (voir Figure 6) . Ce sont ces dossiers qui vont contenir nos annonces.

Un *new* (bouton + ou menu contextuel sur le dossier) et l'interface devrait nous proposer de créer un enregistrement de type annonce.

En testant l'écran de saisie nous allons à ce stade avoir un soucis lorsque l'on change la valeur du menu type de bien l'écran se vide complètement : c'est normal nous n'avons indiqué nulle part quelles colonnes devaient être affichées en fonction du type choisi. Il est temps de s'intéresser aux fichiers qu'a généré pour nous le kicktstarter – notre extension est stockée dans le dossier *typo3conf/ext/clef-extension* – ce qui correspond chez moi à */home/dev/agenceimmo/www/typo3conf/ext/immo/* qui contient les fichiers comme sur Listing 1.

Les fichiers qui concernent la table et son interface de saisie sont *ext_tables.php* – *ext_tables.sql* qui contient l'instruction SQL de création de la table et *tca.php* qui contient la définition de l'interface. C'est à ce dernier que nous allons nous intéresser. Le TCA est un ensemble de tableaux imbriqués qui contient la définition de chacune des colonnes. On trouve par exemple pour le menu déroulant *type de bien* la configuration comme sur Listing 2.

Cette syntaxe de configuration est décrite dans un chapitre de *typo3 core API* (lien ci-dessous) – pour ce qui nous intéresse – la gestion des types – nous allons nous pencher sur les dernières lignes du fichier. Elles contiennent la relation logique entre le type de bien et les colonnes à afficher. Sur le Listing 3 on voit les champs à afficher pour la valeur par défaut de type de bien soit *maison*.

Nous allons personnaliser ce bloc en indiquant pour chaque type les colonnes que nous voulons afficher (voir Listing 4).

Une fois cette modification opérée et les caches de configuration de Typo3 vidés notre interface devrait réagir aux changements de types de biens. On peut le vérifier en saisissant

Figure 1. Le plugin d'affichage d'annonces en partie publique

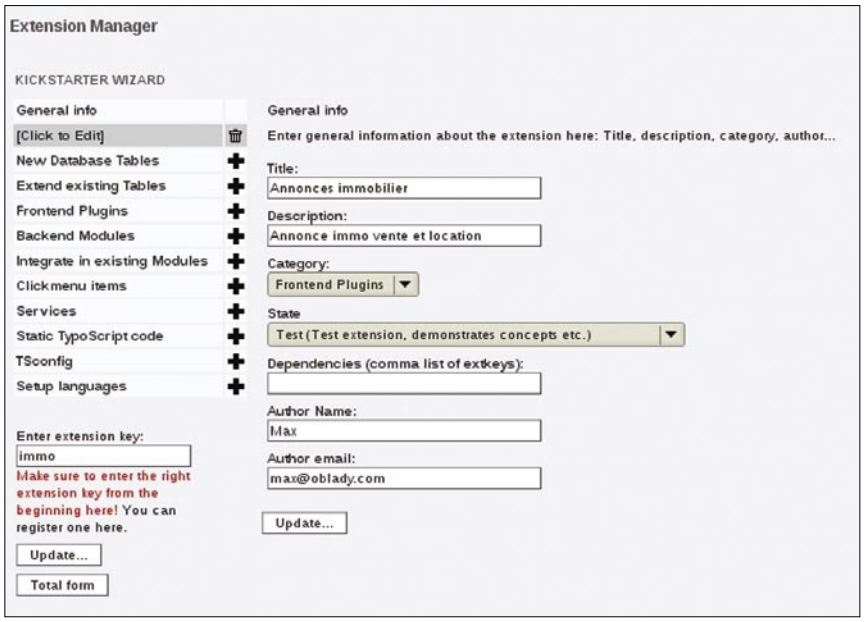

Figure 2. Le kickstarter module de création d'extensions Typo3

quelques annonces de test qui nous seront utiles à l'étape suivante : écrire le code PHP du plugin !

Le PHP affichant les données

Dans le dossier de notre extension : un sous dossier *pi1* – comme *plugin 1* – contient la classe constituant notre plugin. Pour l'instant ce fichier ne contient pas grand chose, on notera cependant que notre plugin est une classe fille de *tslib_pibase* - Cette classe dont hérite tous les plugins permet en fait de disposer de toutes les méthodes facilitant le rendu front office. Comme l'indique le code fourni à titre d'exemple le point d'entrée dans la classe est la méthode `main` et le code HTMLde sortie doit être envoyé avec l'instruction `return`. Notre plugin *annonces immobilières* affichera deux écrans, le premier la liste des annonces sera affiché par défaut et si l'utilisateur clique sur une annonce il affichera le second écran : le détail d'une annonce. Côté backend, l'administrateur du site choisi ce qu'il veut afficher en insérant le plugin comme un contenu de page et en choisissant le dossier système contenant les annonces (voir Figure 7).

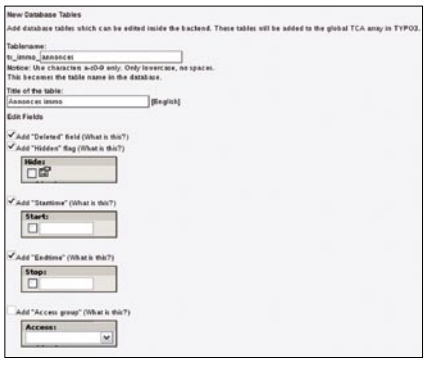

Figure 3. Création d'une table avec le kicktstarter

Avant de commencer le développement à proprement parler nous allons commencer par installer un outil bien utile l'extension *cc_debug*. Cette extension permet de disposer d'un debug mis en forme dans une fenêtre à part – le système de debug de typo3 se basant sur un masque d'ip – ceci afin de pouvoir mettre une trace sans en faire profiter tout le monde – si vous ne travaillez pas directement sur la machine vous devrez adapter la ligne suivant dans votre localconf avec votre masque d'adresse comme ci -dessous :

```
$TYPO3_CONF_VARS['SYS']['devIPmask']
= '127.0.0.1,::1,192.168.1.*';
```

Après cette modification apportée et un vidage des caches – vous pouvez ajouter dans le main de la classe une ligne telle que celle ci-dessous : `debug($conf,"variable conf");`.

En rafraîchissant la page en front office vous devriez obtenir une popup avec le debug comme le montre Figure 8.

Attardons nous ici sur la variable `conf` que nous affichons ici – elle ne contient pas grand chose actuellement – elle est destinée à recueillir la config typoscript qui va nous permettre de passer les informations de contexte à notre site sans le passer en dur dans le code PHP du plugin. Nous allons par exemple insérer dans le setup typoscript de notre template cette chaîne indiquant le chemin vers le gabarit HTML de notre plugin :

```
plugin.tx_immo_pi1.template =
fileadmin/immo/template.html
```

La valeur apparaît désormais dans notre debug du tableau de *conf* en *front office*. Nous allons maintenant structurer notre classe en plusieurs méthodes simples.

Dans le `main` nous testons la présence d'une variable `GET` auquel cas il faut afficher le détail d'une annonce et son absence on affiche la liste des annonces (voir Listing 5). À noter la méthode `t3lib_div::_GET` qui utilise la librairie *t3lib_div* (cf doc de l'API lien ci-dessous) qui permet de récupérer le tableau des variables `get`. La méthode suivante permet d'afficher la liste des annonces avec un lien vers le détail (voir Listing 6). Dans votre gabarit vous aurez une structure HTML classique ou les deux sections liste et détail doivent être entourées de deux marqueurs `###LIST###` et `###DETAIL###`. À l'intérieur de ces sections les éléments dynamiques comme le champ ville seront représentés par un marqueur comme `###ville###` . Ce fonctionnement est semblable à la plupart des moteurs de template.

À noter le recours à une autre classe de l'API Typo3 : `$GLOBALS["TYPO3_DB"]` qui regroupe toute les fonctions d'accès aux données. D'autre part le code fait référence à cObj ou tslibContent qui est l'objet commun à tout les éléments de contenu Typo3. En particulier ici la méthode enableFields qui renvoie la clause SQL `where` correspondant à la table passée en paramètre. Nous reste désormais – petit exercice – à coder la méthode `detail_annonce` qui diffère en fait assez peu de la méthode précédente, liste. Pour se faire vous devez :

- modifier la ligne `exec_SELECTquery` pour lui passer l'uid de l'annonce sur laquelle l'utilisateur a cliqué,
- pointer sur la section *detail*» du template.

En conclusion

Ce rapide tutoriel ne vous donne bien évidemment pas toutes les clefs du développement typo3 – mais vous avez cependant en main

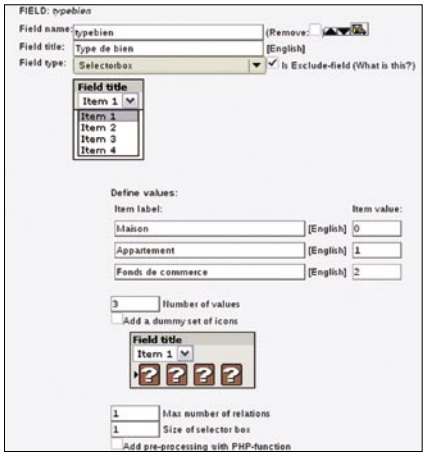

Figure 4. Configuration d'une liste d'options avec le kicktstarter

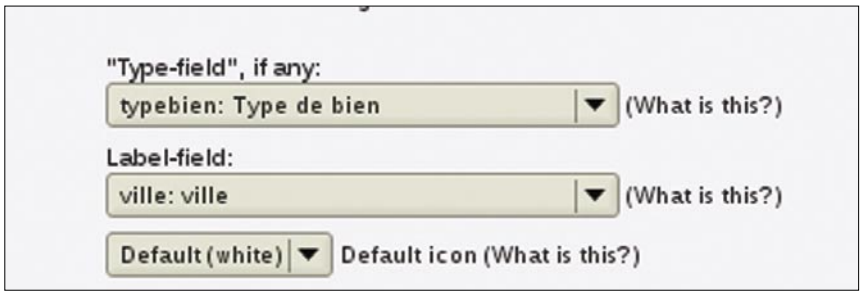

Figure 5. Ajout de la gestion des types pour notre table annonce

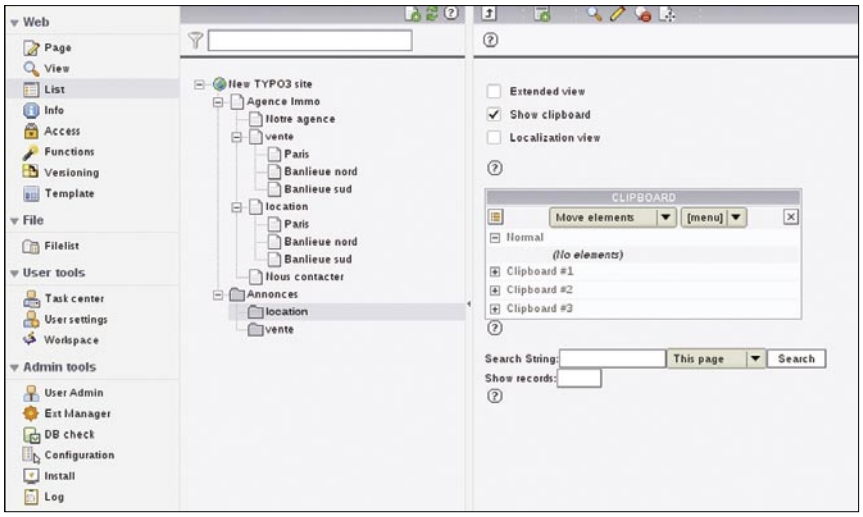

Figure 6. Le dossier système qui va contenir les annonces

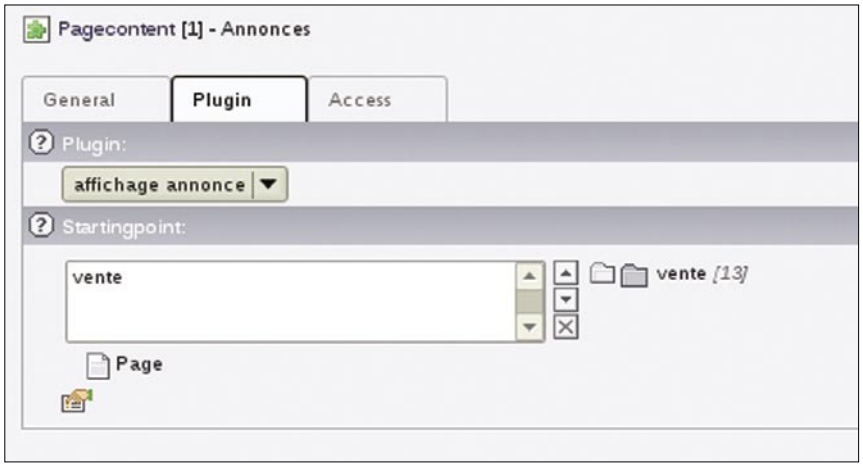

Figure 7. Insertion du plugin annonces dans une page

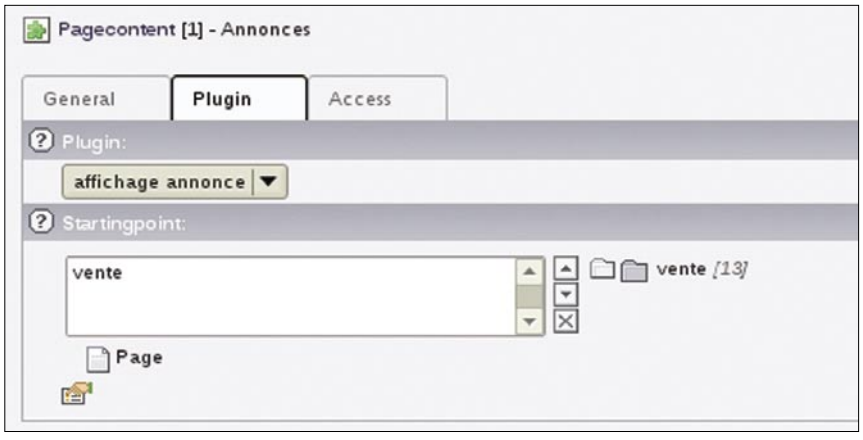

Figure 8. L'extension cc_debug à l'oeuvre

À propos de l'auteur

L'auteur, Maxime Fauquemberg, est le responsable technique et gérant d'Oblady *www.oblady.com*.

De formation littéraire il s'est après une brève expérience dans la presse rapidement consacré aux nouveaux médias en collaborant d'abord à des projets de cdrom éducatifs. Puis pendant plusieurs années il a dirigé des projets web pour différentes sociétés de service. Il a fondé en 2002 Oblady société experte en solution de gestion de contenu open source et pionnière dans l'utilisation de Typo3 en France. Oblady vient notamment de signer la refonte du site *http://www.lesinrocks.com* sous Typo3.

toutes les pistes documentaires pour avancer. Désormais vous n'avez plus qu'à vous plonger dans la documentation de l'API et utiliser les méthodes qui y sont offertes. N'hésitez pas à éplucher la documentation de l'API avant de vous lancer dans l'écriture de vos propres fonctions ce qui vous fera gagner à terme un temps précieux.

NB – Note linguistique : la plupart des références, menus ou termes techniques sont indiqués ici en anglais – il est bien sûr possible d'utiliser typo3 dans votre langue – et de configurer par l'utilisateur le back office dans sa langue préférée. Je conseille cependant aux webmasters / développeurs typo3 de travailler prioritairement dans la langue de Shakespeare ceci simplifie grandement l'accès aux documentations techniques et la compréhension des tables MySQL manipulées.

Sur Internet

- Vous retrouverez une présentation vidéo du kicktstarter et du développement de plugins sous typo3 : *http://typo3.org/documentation/videos/*,
- Bien comprendre le mécanisme de gestion de cache de typo3 : *http://typo3.org/development/articles/the-mysteries-of-chash/*,
- la documentation du TCA : *http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.2.0/view/4/1/*,
- API typo3 : *http://typo3.org/fileadmin/typo3api-4.0.0/*.