GDidées

Typo3 - Support de cours

Nom du document : Cours_Typo3.doc

Copyright 2000-2008, David Goïtré, gdidees.free.fr

Ce document est publié sous licence libre

Le contenu de ce document fait référence à TYPO3

Table des matières

Plan de la formation

1. Installation du CMS Typo 3	Page 5
a) nécessités techniques	Page 5
b) installation de Typo3	Page 5
c) installation de la bibliothèque graphique ImageMagick	Page 6
d) configuration de typo 3	Page 6
2. Présentation de l'architecture Typo3	Page 6
a) application PHP – MYSQL	Page 6
b) structure des répertoires	Page 7
c) centralisation des sources	Page 7
d) les différents répertoires	Page 8
e) résumé	Page 8
3. Présentation de l'arborescence	Page 9
a) organisation de l'arborescence	Page 9
b) l'arborescence vue en base	Page 9
c) le type « sysfolder »	Page 9
d) le mode liste	Page 10
e) le mode liste : Fonctions d'éditions avancées	Page 11
4. Prise en main BackOffice	Page 11
a) créer des pages et des contenus avec le manuel du rédacteur	Page 11
b) application d'un gabarit par défaut	Page 11
c) ajout de contenu	Page 11

5. I	Intégrer son propre gabarit (Méthode classique)	Page 12
	a) intégrer un gabarit	Page 12
	b) créer des menus	Page 15
	c) créer un menu spécial (fil d'ariane, directory)	Page 16
	d) TP1 : insérer un menu de niveau 2	Page 17
	e) TP2 : créer un menu à deux niveaux	Page 17
6. I	Intégrer son gabarit sous TemplaVoila	Page 18
	a) mise en place d'un gabarit de page	Page 18
	b) créer des FCE (Flexible Content Element)	Page 21
	c) créer des FCE alternatifs	Page 22
	d) générer des FCE (Flexible Content Element)	Page 22
	e) TP1 : créer et mapper un template	Page 22
7. (Gestion des droits sous Typo3	Page 22
	a) simplifier l'interface	Page 22
	b) méthodologie	Page 23
	c) création d'un groupe	Page 23
	d) droits d'accès aux modules	Page 24
	e) droits sur les tables	Page 24
	f) type de pages autorisées	Page 25
	g) les « exclude fields »	Page 25
	h) masque de création de page	Page 28
8. \	Workspace Workflow	Page 28
	a) le live workspace	Page 28
	b) le draft workspace	Page 28
	c) custom workspace	Page 29
	d) droits d'accès	Page 29
	e) swap vs publish	Page 30

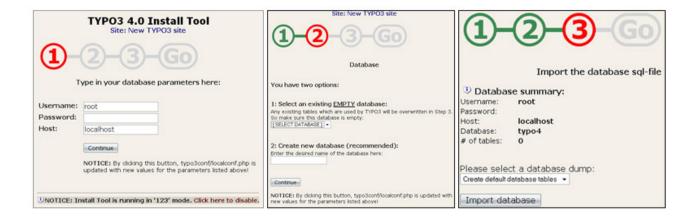
9.	Ajouter des extensions	Page 31
	a) installation/Configurer des extensions	Page 31
	b) inserer une extension via Typoscript (Insertion d'une boite login)	. Page 32
10	Développement d'extensions	Page 33
	a) composition d'une extension	Page 33
	b) l'extension Kickstarter	Page 34
	c) utilisation du Kickstarter : Infos générales	Page 35
	d) utilisation du Kickstarter : création d'une nouvelle table	. Page 35
	e) utilisation du Kickstarter : création d'une extension	. Page 36
	f) utilisation du Kickstarter : génération des fichiers	. Page 36
	g) travaux pratiques	Page 37
1	1. RealURL	Page 38
	a) TypoScript / Config	Page 38
	b) Le fichier .htaccess	Page 38
	c) Le fichier localconf.php	Page 39
	d) UrlTool	Page 39
	e) Configuration via URL Tool	Page 40
	f) UrlTool – preVars / postVarSets	Page 41
	g) Exemple de preVars	Page 41
	h) Exemple de fixedPostVars	Page 42
	i) Documentations	Page 42
1	2. Annexes	Page 43
	a) Les extension essentiels	Page 43
	b) Liens utiles	Page 43
	c) Problèmes liés à typo3	Page 44
	d) Personnaliser le RTE	Page 44

a) Pour faire fonctionner typo3 il faut :

- un serveur Web (application qui répond aux requêtes d'un navigateur). Pour typo3 c'est Apache qui est utilisé (version 1.3 ou supérieure).
- un serveur de base de données pour stocker les informations. (MYSQL pour typo3). Typo3 peut fonstionner avec diverses bases de données (mySQL, postgreSQL, Oracle,...). Cependant, le fonctionnement est optimal avec MySQL, et l'utilisation d'un autre serveur de bases de données peut être source de problèmes (syntaxe des requetes sql lors de l'installation d'xtensions, par ex).
- un interpréteur php puisque le code php n'est pas directement interprété par le navigateur.
- la bibliothèque graphiques qui permettra à typo3 de gérer tout ce qui est manipulation et transformation d'image. (Imagemagick).

Liens: http://typo3.sunsite.dk/software/windows/imagemagick-win-4.2.9.zip
http://typo3.org/download/packages/

b) L'installation manuelle étape par étape :



- a) Première étape (les paramètres dépendent de la base utilisée)
- username : le login de la base mysql, par defaut root
- Password : le mot de passé de la base mysql, par defaut rien
- Host : le nom du serveur, par defaut localhost
- b) Deuxième étape
- soit selection d'une base existante
- soit creation de la base
- c) Troisième étape
- résumé des paramètres
- importation de la base

c) Installation d'image magick

On dézippe le package télécharger dans un répertoire (si possible sans espace). Ensuite dans le back end on va dans « Install» (par défaut le mot de passe est joh316).

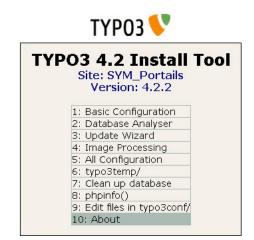
Sous linux, il est très probable qu'Image Magick soit disponible sous la forme de paquets. Il est dans ce cas plus simple de l'installer.

Si ImageMagick n'est pas trouvé, dans le menu 1 « Basic configuration » descendez quasiment tout en bas jusqu'à la section « Search for ImageMagick » et saisissez le chemin du répertoire dans lequel vous avez installé ImageMagick, cochez la case « Check LZW capabilities » puis enfin « Send ».

d) Configuration de Typo 3

La configuration de typo 3 se passé dans cette fenêtre :

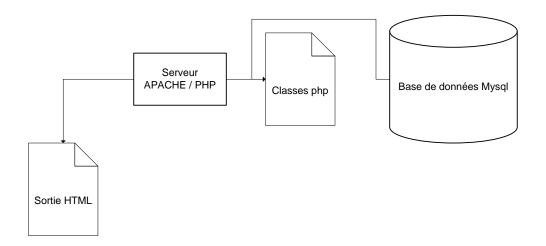
- 1. Basic Configuration : permet de verifier si tout est ok
- 2. Database Analyser : permet de comparer, importer, effacer des tables ou créer un Admin User
- 3. Update Wizard: permet de metre à jour Typo 3
- 4. Image Processing : permet de tester les function d'Image Magick
- 5. All Configuration : permet de paramètrer typo 3
- 6. Typo3temp/: permet de vider le repertoire
- 7. Clean satabase : permet de vider le cache des images
- 8. Phpinfo(): permet de verifier la configuration du serveur apache



Attention: pour des raisons de sécurité, ne pas oublié de supprimer le répertoire « install » de typo 3.

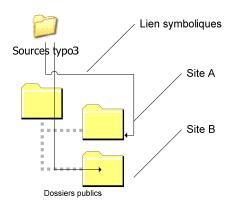
L'architecture de Typo 3

a) Typo3 est une application php mySql « classique »



b) Typo3 est distribué en deux «pakages »

- Les sources : contient le moteur typo3 à proprement parler les sources sont à priori installées hors du répertoire de publication web. Les sources peuvent être partagées par plusieurs sites web.
- Le « dummy » package : est une boîte vide destinée à recevoir toutes les données spécifiques à un site. Le contenu du dummy package est situé dans le répertoire public d'Apache.



Pour créer des liens ymboliques sous Windows, utilise le logiciel "**linkmagic**" http://www.rekenwonder.com/linkmagic.htm

Il y a plusieurs types d'installation :

Sous linux:

- On peut créer un repertoire pour chaque site. Dans chacun de ces sites, on décompresse le les sources ET le dummy package. On a donc une copie des sources pour chaque site.
- On peut séparer les sources et les sites. Les sources sont dans un repertoires, les sites dans un autre. Chaque site a des liens symboliques vers les sources. Avantage, on peut migrer tous les sites d'une version de typo a une autre en une seule fois ; toute modification au niveau des sources est repercutée sur tous les sites automatiquement.

Sous windows:

- Les liens symboliques ne sont possibles que sous windows 2000 et +, a condition que le disque soit partitionné en NTFS5 ou+. Il faut alors les outils complementaires de microsoft pour effectuer des jonctions ou bien linkmagic. On peut alors faire comme sous linux et separer les sources des sites.
- Sous windows anterieur a windows 2000, on ne peut effectuer de liens, et on est obligé de créer une copie des sources pour chaque site.

c) Typo 3: Global / local

Le mode local prime sur le mode global. Exemple : une extension mise dans les deux modes, on peut modifier celle du local sans pour autant modifier celle du global.

- L'architecture de typo3 est conçue pour pouvoir accueillir plusieurs sites sur un même serveur en facilitant les tâches de maintenance et de distribution.
- Les sources sont centralisées pour faciliter les mises à jour, si utilisation de liens symboliques
- Les extensions peuvent être globales (communes à tout les sites) ou locales (accessibles uniquement à un site).
- On peut surcharger une extension localement.
- Les fichiers propres au site sont concentrés pour faciliter les sauvegardes.

d) Typo 3 : Les répertoires

- typo3_src : contient package src = lien symbolique vers le dossier des sources
- t3lib : contient les librairies de Typo3 = lien symbolique vers typo3_src/t3lib
- typo3 : contient le back office de typo3 = lien symbolique vers typo3_src/typo3
- typo3/ext : contient les extensions globales de Typo3
- index.php : copie le fichier des sources dans le package dummy ou faire un lien symbolique vers typo3_src/index.php
- Fileadmin : stockage des données (images, templates, ressources) propres à une instance typo3
- typo3conf : la config locale de Typo3
- typo3conf/ext : les extensions locales de typo3
- typo3temp : cache de typo3
- uploads : sources des images retaillées

e) Typo 3: Architecture résumé

- le code est entièrement orienté objet (utilisation des classes)
- tout passe par « index.php »
- les sources de typo3 sont centralisées. Cf (Typo3 est distribué en deux «pakages »)
- typo3 est conçu pour être utilisé prioritairement sur des plateformes linux / unix, il utilise donc abondamment les ressources du système et se calque souvent sur la logique linux.
- typo3 dispose d'une architecture permettant de gérer le multisite de plusieurs manières

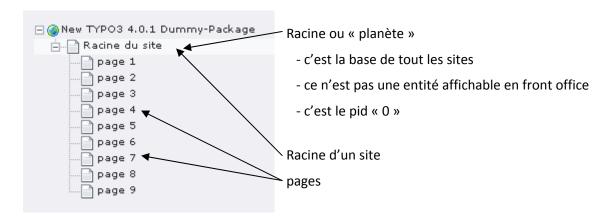
Présentation de l'architecture de typo 3

a) Organisation de l'arborescence

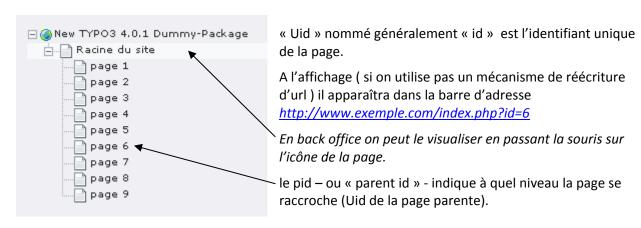
La racine Typo3 symbolisée par le pictogramme planète contient tout les éléments transverses qui peuvent être utilisés dans toutes l'arborescence, on y trouvera par exemple :

Les utilisateurs et groupes back office

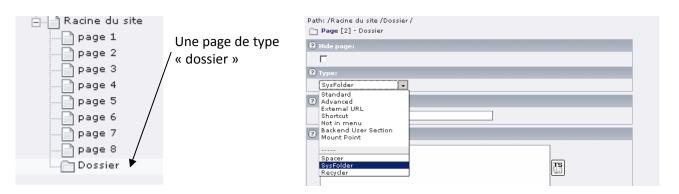
Les données de localisation : monnaies, noms de pays ..



b) Organisation de l'arborescence



c) Typo3 dispose d'un type de page particulier le « sysfolder » ou dossier.



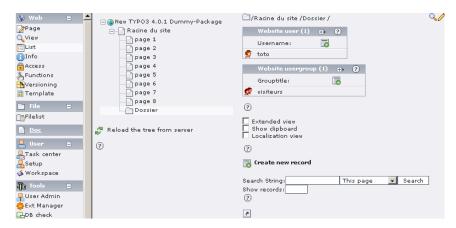
Les sysfolders ont plusieurs fonctions :

- bloquer le listage en partie publique par les menus.
- indiquer à l'utilisateur du back office que des éléments non affichables directement en partie publique sont susceptibles d'être stockés ici. L'utilisateur va est donc invité à passer en mode « liste »

d) Le mode liste

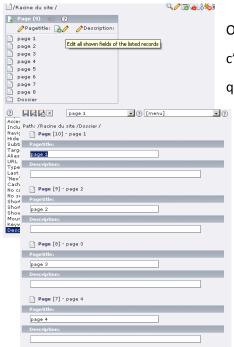
Il est utilisé dans les sysfolder mais peut aussi permettre de manipuler les pages « classiques », et aussi :

- de visualiser les éléments qui ne sont pas des contenus de page
- d'avoir une vue proche de la représentation en base de données
- de procéder à des manipulations avancées (édition multiples, copiercoller complexes, classement)
- d'exporter des enregistrements en csv



Ici le « dossier » est visualisé en mode « liste » on voit qu'il contient deux enregistrement des tables « utilisateurs sites » et « groupe d'utilisateurs sites » ces deux enregistrements ont donc un « pid » égal à l'id de la page « dossier »

e) Le mode liste / édition des pages



On peut aussi passer en mode liste sur une page classique, c'est alors toutes les pages filles et les contenus de la page qui apparaissent dans la liste.

Ceci peut être intéressant par exemple pour renommer massivement une grappe de pages. (exemple ci-contre)

Prise en main BackOffice

a) Créer des pages et des contenus

(Toutes ces manipulations sont décrites dans « le manuel du rédacteur »)

- créer des pages
- créer des contenus dans les pages
- déplacer une page
- copier coller une page
- déplacer un contenu

b) Utilisation du mode liste

- lister une seule table
- classer les enregistrements de la table
- copier coller en utilisant les différents « presse papiers »

c) Appliquer un gabarit par défaut

- sélectionner un gabarit existant et l'appliquer à la racine du site

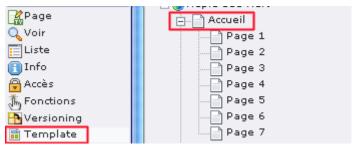
Intégrer son propre gabarit (Méthode classique)

a) Intégrer un gabarit

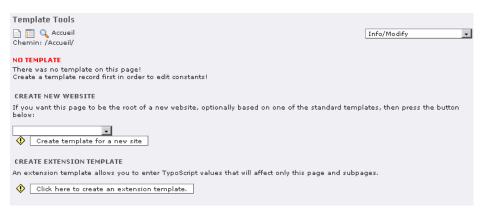
Normalement un "template" est un FICHIER HTML qui décrit une composition (layout) et certaines des zones (areas) qui devraient être substituées par un autre contenu (content) déterminé.

Un "template TypoScript" est en fait un enregistrement (record) dans une table, nommée "sys_template", de la base de données de Typo3. Le fait d'enregistrer un lien entre une Page quelconque et ce "template record" va définir la manière dont la page va s'afficher en appliquant le Template auquel on l'a associée.

Pour afficher le template typoscript, cliquer « Template » dans la colonne gauche et se placer sur la page racine du site.



A la racine du site, si aucun template n'existe il faut le créer, ici cliquer le bouton : « Create template for a new site »





Title: titre du template (informatif)

SiteTitle : Apparaît comme titre de la fenêtre du navigateur, suivi du titre de la page courante

Constant: contient les "Constantes" que vous déclarez et qui sont ensuite utilisées à fin de substitution dans le champ "setup"

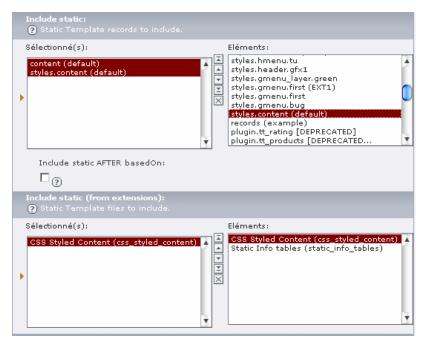
Setup : Ce champ contient le code de configuration de TypoScript



page.10.value = HELLO WORLD!

Pour afficher le code du setup, dans le combo en haut à droite sélectionner « Info / Modify » et cliquer sur le crayon : « Setup ».

Ce qu'on voit au départ est un exemple par défaut qui affiche le texte HELLO WORLD! dans la page.



Pour pouvoir insérer des contenus dans les pages il faut inclure une partie du code pré-programmée en typoscript : les static templates. Ce code permet de restituer les éléments tels que texte, texte+image par exemple, et de les assigner à 4 zones prévues dans les pages : LEFT, RIGHT, BORDER et NORMAL ; ce qui permet d'utiliser jusqu'à 4 colonnes dans les pages du site. (left, center, right, border)

Cliquer sur « Edit whole template record » et sélectionner :

- content (default)
- styles.content (default)
- CSS Styled Content

En TypoScript on distingue des « Objets » et des « Propriétés». Les objets sont typiquement des objets de contenu (exemple : TEXT, IMAGE...) qui peuvent hériter de certaines Propriétés. Par exemple un objet IMAGE aura comme propriétés :

- file (chemin vers le fichier source)
- altText (attribut alt de l'image)
- etc.

Pour raccorder un template HTML afin de restituer les pages selon une charte prédéfinie, il faut déclarer l'objet « PAGE » en typoscript, exemple :

```
page = PAGE
       page.typeNum = 0
       page.stylesheet= fileadmin/css/style.css
       page.10 = TEMPLATE
       page.10 {
       template = FILE
       template.file = fileadmin/templates/template.html
       workOnSubpart = DOCUMENT_BODY
       subparts.CONTENU < styles.content.get
       }
NB: le code compris entre les accolades pourrait s'écrire de cette façon :
       page.10.template = FILE
       page.10.template.file = fileadmin/templates/template.html
       page.10.workOnSubpart = DOCUMENT_BODY
       page.10.subparts.CONTENU < styles.content.get
Le fichier HTML du template ressemble à ceci :
       <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
       <html>
        <head>
        <title>Mon titre</title>
        </head>
        <body>
        ###DOCUMENT BODY###
               ###CONTENU###
               ###CONTENU###
        ###DOCUMENT_BODY###
        </body>
```

Typo va intervenir entre les balises ###DOCUMENT_BODY### et remplacer les balises ###CONTENU### par le contenu de la colonne NORMAL.

</html>

L'objet config

Typo3 gère la partie « header » de la page, c'est pourquoi dans l'exemple il ne remplace les contenus qu'à partir des balises
body> (qu'il gère aussi).

L'objet CONFIG permet d'affiner les propriétés du header de la page, il est indispensable pour afficher les pages correctement en mode classique ou templavoila, exemple :

```
config {
language=fr
htmlTag_langKey=fr
sys_language_uid=0
doctype = xhtml_trans
xmlprologue=none
xhtml_cleaning=all
}
```

Dans cet exemple on détermine : le langage général du site, l'attribut lang de la balise HTML, le doctype de la page. La propriété « xhtml_cleaning » nettoie les erreurs éventuelles liées au doctype XHTML.

b) Créer des menus

Création et paramétrage de l'objet typoscript : exemple simple (menu à 1 niveau) :

```
lib.menu = HMENU
      lib.menu.wrap=|
      lib.menu.1 = TMENU
      lib.menu.1.NO {
      allWrap=|
      }
      lib.menu.1.ACT = 1
      lib.menu.1.ACT.ATagParams = class="active"
      HMENU = menu hiérarchique : cet objet est indispensable pour tous les types de menus
      TMENU = text menu : renvoie des balises du type : <a href= "URL">Titre page</a>
Le menu sera rendu comme ceci:
      <a href="URL">Titre page 1</a>
             <a href="URL">Titre page 2</a>
             <a href="URL" class="active">Titre page 3</a>
```

Branchement du menu pour le rendu HTML

Une fois l'objet créé il faut l'injecter dans la page. Dans l'écran n°10 nous avions ce code, on y ajoute le menu :

```
page = PAGE

page.typeNum = 0

page.stylesheet= fileadmin/css/style.css

page.10 = TEMPLATE

page.10 {

template = FILE

template.file = fileadmin/templates/template.html

workOnSubpart = DOCUMENT_BODY

subparts.CONTENU < styles.content.get

subparts.MENU < lib.menu
}

et dans le template HTML il suffit d'ajouter les marqueurs suivants :

###MENU###

###MENU###
```

à l'endoit où on veut le faire apparaı̂tre.

c) Créer un menu spécial

- Création d'un fil d'ariane
lib.breadcrumb= HMENU
lib.breadcrumb.special = rootline
lib.breadcrumb.includeNotInMenu=1
lib.breadcrumb.special.range = 1|-1
lib.breadcrumb.1 = TMENU
lib.breadcrumb.1.stdWrap.htmlSpecialChars=1
lib.breadcrumb.1.noBlur=1
lib.breadcrumb.1.NO {
allWrap = | /|*| | /|*| |
ATagTitle.field = title
}

- Création d'un menu de type directory

```
lib.menu.special = directory
lib.menu.special.value = 16 (uid de la page ou dosiier don't on veut lister les pages enfants)
lib.menu.1 = TMENU
lib.menu.1.noBlur=1
lib.menu.1 {

NO.allWrap = <|i class="first">|</|i>| |*| <|i>||*| <|i class="last">|</|i>|
}
```

d) TP1: insérer un menu à deux niveaux

Dans le gabarit précédent, ajouter un menu représentant les pages de niveau 2 de l'arborescence.

Indication : La propriété entryLevel de l'objet HMENU permet de spécifier le niveau d'arborescence ciblé.

e) TP2 : Créer un menu à deux niveaux imbriqués

Dans le gabarit précédent, remplacez le menu de niveau 2 par un menu présentant les niveaux 2 et 3 de manière imbriquée.

Indication : L'objet menu crée dans les exercices précédent est en réalité un tableau. En chargeant au fur et à mesure dans le même tableau on peut générer un menu à plusieurs niveaux imbriqués.

Corrigé:

```
lib.menu = HMENU
lib.menu.entryLevel = 1
lib.menu.1 = TMENU
lib.menu.1 {
noBlur=1
lib.menu.1.NO {
wrapItemAndSub =<dl>|</dl>
lib.menu.1.ACT =1
lib.menu.1.ACT {
wrapItemAndSub =<dl>|</dl>
lib.menu.2=TMENU
lib.menu.2 {
noBlur=1
wrapItemAndSub=<dt>|</dt>
lib.menu.2.NO {
wrapItemAndSub=<dt>|</dt>
}
```

Intégrer son propre gabarit (Méthode templavoila)

Templavoila permet de créer des structures de pages plus flexibles que celles utilisant la notion classique de "colonnes" et facilite le mapping de contenu grâce à son interface graphique.

Avant d'utiliser Templavoila comme mécanisme d'intégration, il faut que Typoscript soit paramétré pour son utilisation.

Il faut aussi créer un autre élément essentiel de notre site le "Storage Folder". Il s'agit d'une page définie comme "SysFolder". Celui-ci est nécessaire afin de disposer d'un point central de stockage des éléments relatifs à l'ensemble du site et pas seulement à une seule page.

Pour cela le contenu de l'objet « PAGE » doit être lié à la fonction de traitement de TemplaVoila

```
page = PAGE
page.typeNum = 0
page.10 = USER
page.10.userFunc = tx_templavoila_pi1->main_page
```

a) Mise en place d'un gabarit de page

Default PAGE object:

Pour créer un gabarit de page avec Templavoila sélectionnez « Filelist » dans la colonne de gauche Apparaît alors la liste des fichiers disponibles (les fichiers qui composent le gabarit HTML doit

Apparaît alors la liste des fichiers disponibles (les fichiers qui composent le gabarit HTML doi obligatoirement se trouver dans le dossier fileadmin/templates).



Cliquer sur l'icône associée au fichier que vous souhaitez utiliser dans Templavoila. Dans le menu contextuel qui s'ouvre, sélectionnez « TemplaVoila »

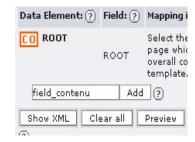


L'interface qui suit va permettre de réaliser le mapping de page et de contenu à l'instar de ce qui était réalisé avant en TypoScript.



La création d'une nouvelle zone de contenu au sein de notre gabarit s'effectue en trois temps :

- On crée l'élément de contenu
 - on donne un nom à la zone
 - on clique sur « Add



- On définit les propriétés de la zone

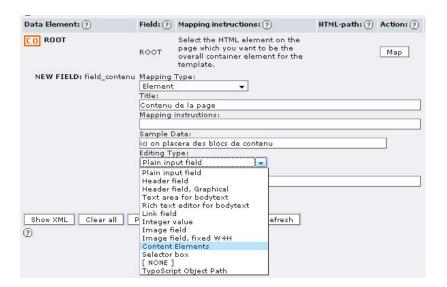
Mapping Type: définit le type de mapping

- « Element » pour un élément de contenu standard
- « Attribut » pour un attribut d'une balise (ex : src de l'image)

Title: Titre de la zone

Sample Data : texte de substitution lors des prévisualisations

Editing Type: type d'élément « Content Elements pour un élément de contenu classique »



- Plain Unput field : champ de texte
- Header field : champ d'entête
- Header field Graphical : champ d'entête graphique
- Text area for body text : champ de texte sans mise en forme RTE
- Rich text editor for bodytext : champ de texte avec mise en forme RTE
- Link field : champ de lien URL
- Content Elements : élément de contenu de page
- Selector box : liste déroulante
- TypoScript Object Path: objet typo défini dans le setup par la suite (pour menu, login, etc...)

- On mappe la zone

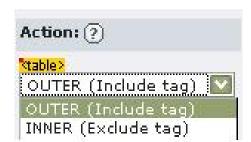
Chaque élément crée doit être mappé, c'est à dire que l'on définit la zone dans le gabarit HTML ou l'élément de contenu apparaîtra. Pour cela sélectionner le bouton « Map » associé à l'élément à mapper. Attention le root level doit toujours être mapper sur le body et avoir le type d'action INNER

- en mode INNER, la balise mappée est conservée, seul son contenu est modifié.
- en mode OUTER la balise mappée ET son contenu sont remplacés.
- le mode RANGE permet de mapper une balise et les balises qui la suivent.

On peut mapper un élément, mais aussi un simple attribut d'une balise.

Dans la page qui s'ouvre, sélectionnez le tag HTML dans lequel l'élément de contenu doit être inclus.

Une fois l'élément mappé, sélectionnez le type d'action **OUTER** ou **INNER**

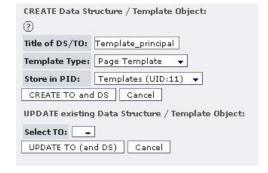




- Une fois tous les éléments mappés, sauvegardez grâce au bouton « Save As ».

Dans la fenêtre qui s'ouvre :

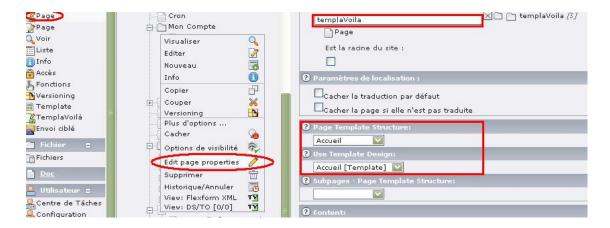
- choisir un nom pour votre template
- définir le Template Type comme template de page « Page Template ».
- sélectionner le dossier sysfolder qui stockera le gabarit.



Arrivé à ce stade nous disposons d'une structure de page qui définit des zones de contenus et leur mapping au sein d'une page HTML statique.

Le type content Element permet de définir des templates de FCE (une autre particularité de templaVoila) Nous pouvons désormais affecter à une page le gabarit que nous venons de créer. Pour cela :

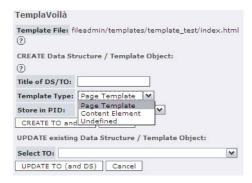
- créer le template à partir du module « **Template** » sur la page « **racine** », puis cliquer sur « Click here to edit whole template record »
- il faut éditer page « racine » est l'associer au sysfolder (dossier de stockage du gabarit).
- affecter les structures DS et TO sur la page « racine » (éditer->options)



b) Créer des FCE (Flexible Content Element)

Les FCE sont des contenus de page flexibles. On peut s'en servir pour faire des modèles de pages. Pour créer ces FCE, il faut :

- mapper sur le gabarit principal la partie qui doit être un contenu flexique avec un « Editing Type » sélectionner sur un « Content Element »
- créer une copie HTML du gabarit de base, supprimer les éléments inutiles et mapper la partie utile pour le FCE ou mapper la partie utile directement sur la gabarit existant.
- sauvegarder notre template FCE avec un « **title of DS/TO** » différent du gabarit et un « **Template types** » sélectionner sur « Content Element »



Une fois notre FCE créer, cliquer sur le module TemplaVoila et vérifié si tout et correct. Ensuite il ne reste plus qu'a générer notre FCE.

c) Créer des FCE alternatifs

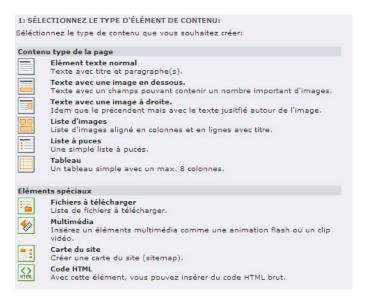
Un FCE alternatif et une copie du FCE original mais avec un TO différent.

- sélectionner l'onglet Flexible CE dans templavoila
- cliquer sur « Create new Template Object » dans la partie TO
- donner un titre au nouveau TO puis enregistrer
- mapper et sauvegarder le nouveau TO

d) Générer des FCE (Flexible Content Element)

A la façon des gestion de contenu des pages en mode classique, :

- cliquez sur l'icône Page TV
- éditer la page ou l'on veut ajouter du contenu
- cliquer sur « ajouter un nouvel élément de contenu »
- sélectionner dans la liste le contenu voulu



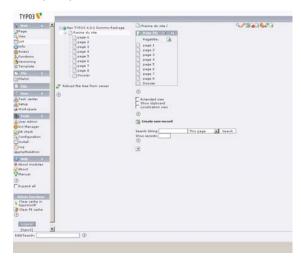
e) TP1 : Créer et mapper un template

(Toutes ces manipulations sont décrites dans « le manuel du rédacteur »)

- a l'aide du module « list » appliquer la TV au fichier HTML
- a l'aide du module TV créer les mapping de page
- sauvegarder le template (DS/TO)
- modifier le css dans typoscript
- affecter le template aux pages souhaitées (onglet extended)
- créer le typoscript de chaque mapping

a) Simplifier l'interface

Lorsque l'on se connecte pour la première fois au back office de typo3 - généralement comme « admin » on voit quelque chose comme ceci :



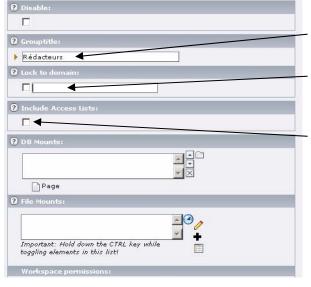
Ce qui peut laisser Perplexe les contributeurs... on va donc configurer les droits de telle sorte qu'ils ne voient que ce qui leur est strictement nécessaire.

b) Méthodologie

- pour faciliter la maintenance du système on va centraliser la gestion des droits à travers des groupes et placer ensuite les utilisateurs dans ces groupes.
- un utilisateur pouvant appartenir à plusieurs groupes, on veillera à distribuer les différents droits en autant de groupes que nécessaire.

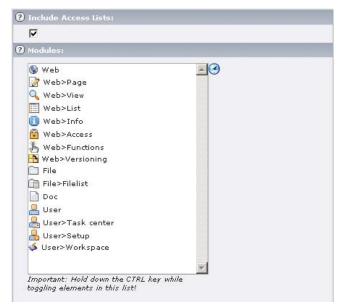
c) Création d'un groupe

- les groupes d'utilisateurs et les utilisateurs back office, comme tous les éléments transverses à l'arborescence sont stockés à la racine de typo3. On se placera donc à la racine pour créer un nouvel enregistrement de type « backend user group »



- on donne tout d'abord un nom à notre groupe, par exemple « rédacteurs ».
- si notre instance dispose de plusieurs noms de domaine, on peut limiter l'accès à un seul de ces domaines.
- pour pouvoir configurer finement les droits de notre groupe d'utilisateurs on coche la case
 « Include Access Lists » ce qui a pour effet de recharger l'interface avec de nouveaux champs.

d) Droits d'accès aux modules

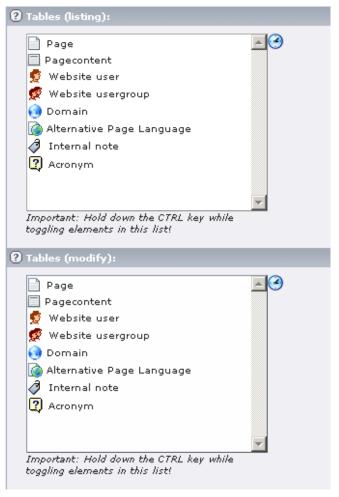


Nous pouvons désormais donner accès aux modules à notre groupe rédacteurs.

En pressant la touche « ctrl » on sélectionne les modules autorisés pour ce groupe.

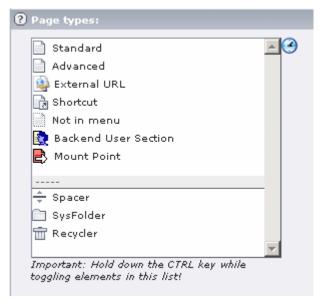
Attention certaines entités sont des conteneurs : « web » par exemple regroupe toutes les fonctionnalités liées à la gestion des pages. Donc si on veut que les utilisateurs aient accès à web>page il faut autoriser l'accès au conteneur « web »

e) Droits sur les tables



- pour chacune des tables existantes
 on peut indiquer si le groupe dispose de droits
 en lecture et / ou en écriture.
- si les droits d'écriture sont présent il est inutile d'accorder des droits en lecture.
- si une extension ajoute une ou plusieurs tables au système ces tables viendront s'ajouter automatiquement à la liste de gestion des droits.
- pour notre exemple de groupe « rédacteur » on accordera par exemple les droits sur les tables pages et contenus.
- dans le cas d'un site multilingue on accorderait aussi les droits en lecture au groupe sur
 « alternative page language »

f) Type de pages autorisées



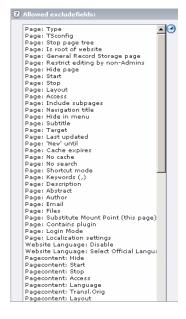
- on a vu dans le « manuel du rédacteur que dans Typo3 l'on pouvait créer différents types de pages, ici nous allons autoriser notre utilisateur à créer des pages des types sélectionnés.
- bien sûr si l'utilisateur ne dispose pas des droits suffisants pour créer la page le choix ne lui sera pas offert dans l'interface.

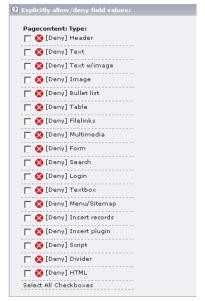
g) Les « exclude fields »

- le mécanisme interne de description des tables en base de données permet d'indiquer un champ de la table comme « exclude fields » ce qui signifie que l'on va pouvoir gérer des droits au niveau de ce champ. (Dans les faits quasiment tout les champsle permettent).
- si le champ n'est pas autorisé il disparaîtra simplement de l'interface.
- de la même manière on sélectionne les champs autorisés en prenant garde de laisser le doigt sur la touche « ctrl ».

Le type de contenu (texte, texte image etc ...) est stocké dans le champ Ctype de la table content il n'est donc pas directement possible de le contrôler via les ACL. On peut en revanche autoriser ou interdire différentes valeurs.

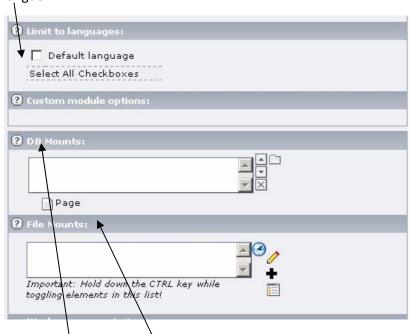
Chaque ajout de type de contenu via l'installation de plugin fera apparaître l'élément ici.





Dans le cas d'un site multilingue on va pouvoir ici configurer le groupe pour l'autoriser à intervenir sur une ou plusieurs langues.

Les langues listées ici sont celles configurées pour le site. Dans la capture d'écran on est dans le cas d'un site mono-langue.

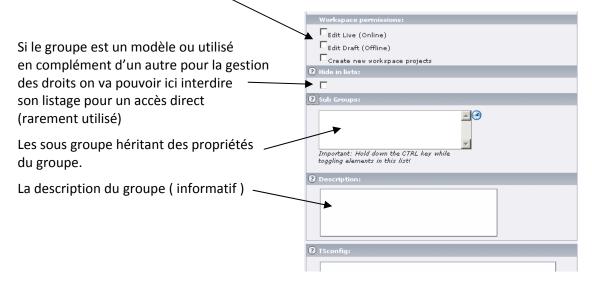


Point de montage ou dbmount, on pointe a ici sur le haut de la grappe de page autorisée pour le groupe. On peut ainsi limiter les éditeurs à une section du site. A noter aussi l'analogie avec les système d'exploitation de type unix.

Sur le même principe que le dbmount le filemount indique quelle(s) partie(s) du système de fichiers est autorisée au groupe.

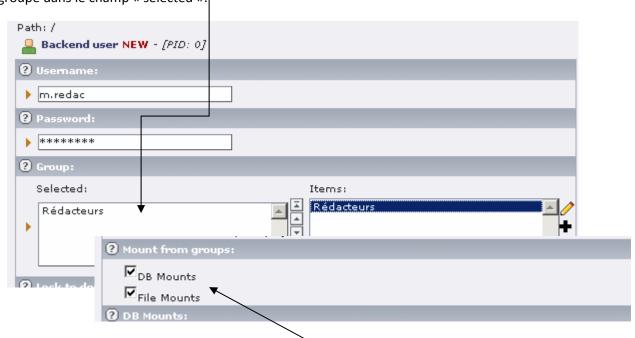
Le « filemount » est une entité à part entière qu'il faut enregistrer indépendamment (signe + à côté du browser).

Les espaces de travail ou workspace sont le centre de la gestion du workflow dans typo3 on va pouvoir ici autoriser ou non l'accès à un espace de travail pour le groupe. On peut autoriser ici aussi la création de nouveaux espaces de travail. <



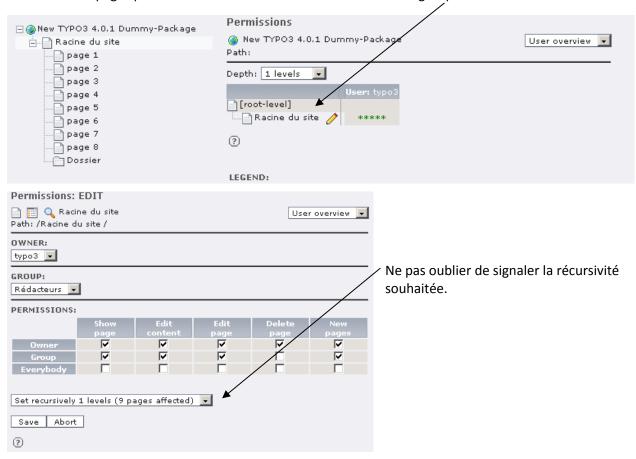
La TSconfig permet de configurer finement le contexte des utilisateurs, on va par exemple dans le cas d'un site utilisant une seule colonne masquer les colonnes non pertinentes. (cf ref ts config)

Il nous reste bien sûr à placer les utilisateurs concernés dans le groupe « rédacteurs » en faisant coulisser le groupe dans le champ « selected ».



Par défaut le user hérite des points de montage du groupe.

Encore une fois on retrouve dans Typo3 l'analogie au système unix, on va en effet devoir positionner les droits sur les pages pour donner les droits d'accès à notre nouveau groupe.



Il ne reste plus qu'à vérifier notre configuration en prenant l'identité de l'utilisateur cible (su = switch user) A noter qu'il existe deux modes de su dans Typo3. - switch back : lors de la déconnexion on revient au premier user - change to : lors de la déconnexion la session est fermée. <u></u>Setup Modules TSconfig 🍊 Workspace TSconfig HL Tools Update: 💂 User Admin Ext Manager RESHIT <mark>-</mark>DB check Usemames: Configuration ① / 2 2 m ≈ ≈ 🕌 typo3 ▼ Install

h) Masque de création de page

Log

Maintenant notre user « rédacteur » est créé, il dispose maintenant des droits suffisants pour créer des pages et des contenus. Quels seront les droits affectés à ces nouveaux éléments ? la règle assez simple :

- le propriétaire de la page est le user qui l'a créé
- le groupe de la page est le premier dans la liste des groupes auxquels appartient le user.
- le masque des droits est hérité des pages supérieures.

Il importe donc de ne pas créer de conflit entre les users intervenant sur une section.

Workspace Workflow

a) le live workspace

L'espace de travail « live » est en prise directe avec la version publique : tous les changements sont immédiatement pris en compte.

b) le draft workspace

Le Draft workspace ou « brouillon » permet de passer dans un « bac à sable » où :

- aucun changement n'impacte la version en ligne
- le versioning est implicite, chaque modification crée automatiquement une nouvelle version
- l'utilisateur peut visualiser directement ses modifications dans la mise en forme publique, telles qu'elles apparaîtront après acceptation éventuelle.
- l'interface offre un suivi des modifications

Uniquement les entités supportant le versioning sont éditables ce qui exclut les tables dont le tca n'autorise pas la gestion des versions et les fichiers (template html, images ...)

c) Custom workspace

Les espaces de travail « utilisateurs » sont des espaces partageant le comportement du workspace « Draft » mais pour lesquels on peut définir :

- un ou des propriétaires
- un ou des valideurs
- un ou plusieurs membres
- de(s) points de montage fichiers (filemount)

Les custom workspace sont conçus pour permettre la configuration d'un workflow simple (rédacteur - valideur-publicateur)

De plus certaines des contraintes de l'espace « draft» sont levées :

- on peut autoriser l'édition « live » des éléments en base ne supportant pas le versionning.
- on peut autoriser la manipulation du sytème de fichier (ie via l'accès au(x) filemount)

d) Droits d'accès

1- accès

- live : les utilisateurs doivent être explicitement autorisés (via la config user ou groupe) à acceder à l'espace
- draft : les utilisateurs doivent être explicitement autorisés (via la config user ou groupe) à acceder à l'espace
- custom : L'accès est autorisé directement dans l'enregistrement workspace où sont définis
 - les groupes ou users « éditeurs »
 - les groupes ou users « valideurs »
 - les groupes ou users « publicateurs »
 - les propriétaires

2- édition

- live: version en ligne
- draft : version brouillon exclusivement des éléments supportant le « versioning »
- custom : version brouillon des éléments supportant le « versioning », édition en live des éléments ne supportant le versioning si autorisation.

3- DB mounts

- live : celui défini dans la config user ou groupe.
- draft : celui défini dans la config user ou groupe.
- custom: peut être défini dans le workspace auquel cas prend le pas sur celui défini dans la config user ou groupe mais ne peut être défini qu'a l'intérieur de la grappe autorisée à l'utilisateur.

Si aucun dbmount n'est défini dans le workspace c'est celui du profil qui est utilisé.

4- accès au système de fichiers

live : directdraft : interdit

- custom : peut être optionnellement autorisé

5- publication - retrait planifiés

- live : N/A - draft : N/A

- custom : peut être défini via une tâche cron

6- validation

live : N/Adraft : N/A

- custom : **les membres** peuvent passer les contenus de l'état « édition » vers « validation » lls ne peuvent éditer les contenus qui sont dans l'état « édition » ou « refusé »

Les valideurs peuvent passer les les éléments de l'état « edition » à « validation », et vers « publication » ou les rejeter.

Les propriétaires peuvent accéder à tous les états d'un élément.

On peut configurer des alertes par mail lors du passage entre différents états.

7- publication

- live: aucune limitation
- draft : si le user a aussi accès au « live »
- custom : les propriétaires peuvent publier même s'ils n'ont pas accès à la version live. Les membres et valideurs ne peuvent pas publier même s'ils ont accès à la version live.

8- paramètrage

- live : aucun - draft : aucun

- custom : les propriétaires peuvent ajouter/éditer membres et valideurs et éditer le paramétrage du workspace.

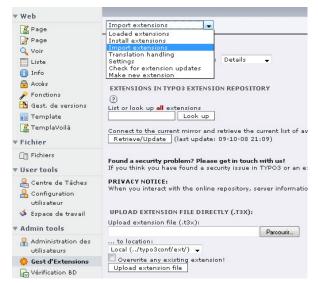
e) Swap Vs Publish

Il existe deux façons pour un élément de passer de l'état online à l'état offline :

- la publication « publish » : l'élément on-line passe à l'état archivé, et le contenu offline le remplace.
- la bascule ou « swap » : les deux éléments sont inversés « online-offline », ce qui sous-entend qu'ils pourront l'être de nouveau. Il s'agit d'éléments événementiels récurrents (par exemple une message de « bonne année » ou une promotion proposée régulièrement)

a) installation/Configurer des extensions

- cliquer sur le module « extension manager »
- dans la partie de droite , sélectionner dans la liste déroulante « import extension »



а

Loaded extensions : permet de voir la liste des extensions présentes dans typo 3.

Install extensions : permet de voir la liste des extensions installé dans typo3

Import extensions : permet de chercher une extension sur le net ou sur le disque dur et de l'importer.

Translation handing : permet d'installer une autre langue.

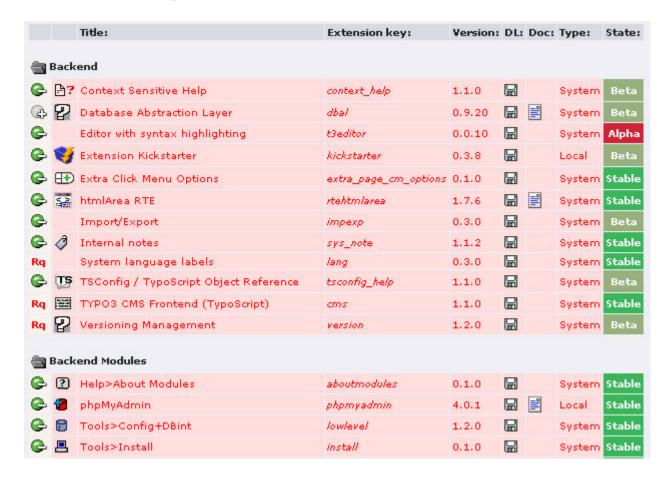
Settings: permet de choisir le lien web pour les extensions.

- cocher la case dans setting et cliquer sur « retrieve/Update » pour lister toutes les extensions
- saisir le nom de l'extension et cliquer sur « look up »
- cliquer sur le picto rouge à gauche de l'extension choisie



- cliquer sur parcourir et chercher une extension sur le disque dur
- cliquer sur « upload extension file »

Dans la liste ci-dessous, on peut voir les nouvelles extensions installées



b) inserer une extension via Typoscript (Insertion de la boite login)

le code ci-dessous affiche la boite login dans un objet menu et paramètre ses propriétés

```
lib.menu < plugin.tx_felogin_pi1
Plugin.tx_felogin_pi1 {
   pid list=25
   var 2 =toto
}</pre>
```

a) composition d'une extension

Dans Typo3 il y a trois notions à ne pas confondre :

- extension : une extension est une brique fonctionnelle dans Typo3. Elle concerne généralement une fonctionnalité précise du site (ex: une gestion de news). Une extension peut contenir plusieurs extensions et/ou modules.
- plugin : un plugin est une sous partie d'une extension. Elle répond à une fonctionnalité particulière d'une extension. (ex : afficher une archive des news parues). Un plugin génère du contenu pour affichage Front Office.
- module : un module est une sous partie d'un plugin. Elle répond à un besoin fonctionnel particulier côté BackOffice. (ex : créer/cataloguer une news). Les modules sont disponibles dans le menu gauche du BO Typo3.

Les fichiers composant une extension Typo3

Un certain nombre de fichiers sont nécessaires au bon fonctionnement d'une extension Typo3 et s'agencent de la façon suivante :

les fichiers communs à toutes les extensions se trouvent à la racine de l'extension. Il s'agit principalement des fichiers :

```
ext_tables.phpext_tables.sqltca.php
```

les fichiers propres à chaque plugin se trouvent dans un dossier particulier. Ce dossier porte le nom pi+un numéro incrémental (ex : pi1, pi2)

les fichiers propres à chaque module se trouvent dans un dossier particulier. Ce dossier porte le nom mod+un numéro incrémental (ex: mod1, mod2, etc..)

Le fichier TCA, commun à toutes les extensions et modules, est le fichier qui établit la relation entre les champs de la base de données et sa représentation dans le BackOffice Typo3.

Exemple:

Le fichier ext_tables.sql contient les instructions de création des différents tables utilisées par notre plugin. Ce fichier est exécuté lors de l'installation de notre plugin.

```
Exemple:
```

```
CREATE TABLE `oblady_exemple` (
   `uid` int(11) NOT NULL auto_increment,
   `pid` int(11) NOT NULL default '0',
   `tstamp` int(11) NOT NULL default '0',
   `crdate` int(11) NOT NULL default '0',
   `cruser_id` int(11) NOT NULL default '0',
   `deleted` int(1) NOT NULL default '0',
   `results` text NOT NULL,
   PRIMARY KEY (`uid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO INCREMENT=1;
```

Le fichier ext_tables.php contient quand à lui les informations génériques sur les tables comme :

- titre de la table en BackOffice
- champ utilisé pour le tri des enregistrements

Exemple:

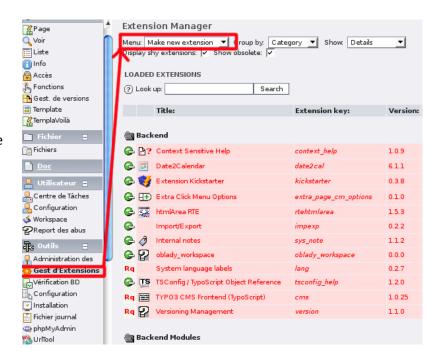
```
$TCA["tx_Oblady_questionnaire"] = Array (
    "ctrl" => Array (
    "title" => "LLL:EXT:Oblady_questionnaire/locallang_db.php:tx_Oblady_questionnaire",
    "label" => "question",
    "tstamp" => "tstamp"
    .....
```

b) l'extension Kickstarter

Le kickstarter est une extension Typo3 qui permet de générer tous les fichiers composant un plugin et ses extensions de manière dynamique et automatique.

Le plugin kickstarter n'est pas inclus au noyau Typo3 et doit donc préalablement être installé depuis le repository Typo3.

Une fois installé, l'extension kickstarter est disponible dans la liste déroulante du gestionnaire d'extension de Typo3

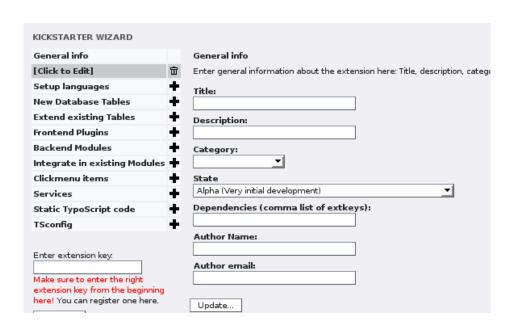


c) Utilisation du Kickstarter : Infos générales

La partie « General Info » permet de définir les informations propres au plugin.

Nous y spécifierons :

- le titre
- la description
- la catégorie

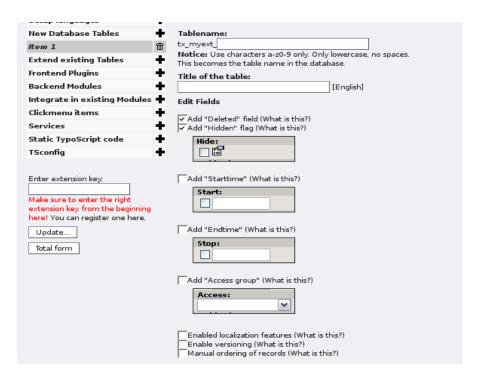


d) Utilisation du Kickstarter : Création d'une nouvelle table

Dans la partie New Database table, nous accèdons à une interface nous permettant de créer tous les champs de notre base de données et de définir la façon dont ils seront saisis en BackOffice.

Nous y spécifierons :

- le titre de notre extension (titre qui s'affiche dans le BackOffice Typo3).
- la façon dont les utilisateurs auront accès à cette extension (depuis la liste des extensions, etc..)

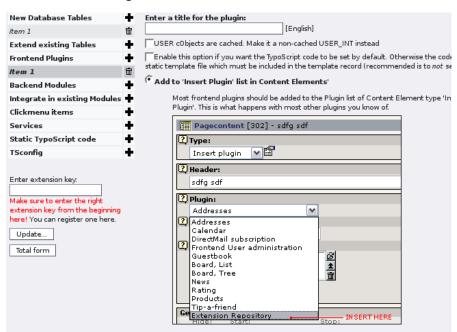


e) Utilisation du Kickstarter : Création d'une extension

Une fois la base de données, nous pouvons créer une extension si nécessaire. Pour cela, il suffit de sélectionner "Frontend Plugins » dans le menu de gauche.

Nous y spécifierons :

- le titre de notre extension (titre qui s'affiche dans le BackOffice Typo3).
- la façon dont les utilisateurs auront accès à cette extension (depuis la liste des extensions, etc..)



f) Utilisation du Kickstarter : Génération des fichiers

Une fois les différents éléments créés (plugin, tables, extensions, modules,...) nous pouvons lancer la génération des fichiers.

Pour cela nous devons spécifier une clef pour notre extension. Cette clef servira d'intitulé pour les noms des classes, fichiers et dossier de notre extension. Cette clef doit donc être unique.

Cette clef se saisit dans le champ « Enter extension key » dans le menu de gauche. Cliquez ensuite sur « Update » puis sur « View Result ».

g) Exercices pratiques:

TP1: Utilisez le kickstarter pour créer une extension permettant de gérer une bibliothèque.

La base contiendra 3 tables:

- auteur
- livre
- catégorie

et une extension. Visualisez ensuite le rendu dans le BackOffice.

TP2: En utilisant l'API Typo3, écrivez le code PHP dans votre extension permettant d'afficher la liste des livres de la bibliothèque.

Indication:

- la classe t3lib_db de l'API contient toutes les fonctions nécessaires à la récupération des informations.
- la définition de l'API typo3 est disponible à cette adresse http://www.api-typo.com/

TP3: En utilisant l'API Typo3, dissociez le code PHP du code HTML afin d'avoir un fichier montemplate.html contenant tout le code HTML et aucun élément HTML dans le PHP.

Indication:

- la classe tslib_cObj contient les fonctions nécessaires à la dissociation du HTML et du PHP
- la classe tslib_cObj est instanciée par la classe tslib_pibase. De ce fait toutes les fonctions de la classe tslib_cObj sont utilisables depuis

\$this->cObj->nomdelafonction.

TP4: Transformez le code précédent pour avoir une fonction qui génère l'affichage liste et codez ensuite l'affichage d'une fiche « livre ».

Indication:

- la classe tslib_pibase contient les fonctions permettant de générer les liens vers des pages Typo3.
- en préfixant par le nom de la classe, une variable passée en GET ou en POST (ex: &nomdemaclasse_pi1[nomdemavariable]), on peut récupérer la variable grâce à l'objet \$this->piVars['nomdemavariable'].

RealURL est une extension qui permet la réécriture d'URL via le composant mod.rewrite d'Apache.

Au lieu d'url du type « www.domain.com/index.php?id=12 » par exemple on aura :

« www.domain.com/titre-de-la-page/titre-de-la-page-en-dessous/ » et ainsi de suite.

Cette extension est configurable pour afficher de la même manière des paramètres spécifiques à des extensions, ou des paramètres tels que la langue du site, etc.

a) TypoScript / Config

```
Dans la page racine du site, il est nécessaire de modifier l'objet config (dans la partie setup) :

config {

simulateStaticDocuments = 0

baseURL = http://www.domaine.com/

tx_realurl_enable = 1

linkVars = L (si le site est multilingue)
```

Attention! dans beaucoup d'exemples trouvés sur Internet on trouve ceci:

config.baseURL=1

Ceci correspond à d'anciennes versions de RealURL et n'est plus utilisé.

uniqueLinkVars = 1 (seulement si linkVars est présent)

b) Le fichier htaccess

Une fois le TypoScript setup modifié, les liens www.domain.com/index.php?id={id de la page} deviennent : www.domain.com/{titre de la page}/etc.

Pour que le serveur pointe sur la bonne page il est nécessaire d'utiliser un fichier .htaccess qui renseignera Apache pour pointer sur la bonne page.

Voici la configuration à insérer dans le fichier .htaccess :

```
RewriteEngine On

RewriteRule ^typo3$ - [L]

RewriteRule ^typo3/.*$ - [L]

RewriteCond %{REQUEST_FILENAME} !-f

RewriteCond %{REQUEST_FILENAME} !-d
```

RewriteRule .* index.php

c) Le fichier localconf.php

Il existe plusieurs méthodes de réécriture d'URL dans RealURL, la plus courante étant typiquement : tx realurl advanced.php.

Pour information les autres méthodes permettent la réécriture comme ceci, par exemple :

au lieu de www.domain.com/index.php?id={id page}

on a: www.domain.com/{id page}/...

Pour choisir la méthode de configuration il est nécessaire de modifier manuellement le fichier localconf.php situé dans le dossier : dossier racine/typo3conf.

On trouve sur le web des exemples de configuration typiques, notamment dans la documentation officielle de RealURL (http://typo3.org/documentation/document-library/extension-manuals/realurl/current/) ou sur Typo3Wiki (http://wiki.typo3.org/index.php/EXT/realurl/manual).

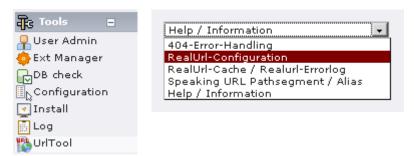
Il est toutefois possible de se simplifier la tâche en utilisant l'extension « URL Tool » () qui permet de configurer directement RealURL via le backend de Typo3.

d) UrlTool

L'extension URL Tool une fois installée offre un nouveau module backend dans Typo3.

Une fois l'outil sélectionné, on a un menu qui permet notamment :

- la gestion d'erreur 404
- affichage des stats (erreurs, URL)
- la possibilité de vider les caches de RealURL
- e t surtout la configuration de RealURL



e) Configuration via URL Tool

URL Tool est un extension livrée avec un fichier de configuration pour RealURL standard, qui utilise par défaut le mode tx realurl advanced.

Pour que l'extension modifie le fichier localconf.php, il suffit de sélectionner « Real URL configuration » dans le menu, puis :

Sous le menu de droite, cliquer sur « Load Default Configuration »

A gauche, cocher la case « Use this Real Url configuration »

En bas du module, cliquer le bouton « Save ».



Le fichier de config proposé par URL Tool est modifiable via le backend, sinon on peut le trouver dans le dossier : racine/typo3conf/urltoolconf_realurl.php

Tout d'abord, paramétrer deux valeurs:

- dans la partie « pagePath », la valeur « rootpage_id » doit être renseignée ; il faut entrer ici l'id de la page racine du site pour que RealURL fonctionne correctement.

```
'pagePath' => array (
    'type' => 'user',
    'userFunc' => 'EXT:realurl/class.tx_realurl_advanced.php:&tx_realurl_advanced->main',
    'spaceCharacter' => '-',
    'languageGetVar' => 'L',
    'expireDays' => '7',
    'rootpage_id' => '1',
),
```

Si on veut ajouter le suffixe « .html » derrière le chemin des URL générées, laisser le fichier de conf tel quel, sinon modifier cette ligne, dans la partie « fileName » :

'defaultToHTMLsuffixOnPrev' => true, -> ajoute .html

'defaultToHTMLsuffixOnPrev' => false, -> pas de suffixe

Pour d'autres paramètres de ces modules, se référer à la documentation officielle, ou sinon au wiki.

f) UrlTool - preVars / postVarSets

La méthode "advanced" de RealURL restitue les Url comme ceci :

[TYPO3_SITE_URL] [preVars] [pagePath] [fixedPostVars] [postVarSets] [fileName]

Toutes ces options, mise à part TYPO3_SITE_URL sont configurables via Url Tool;

ce sont les paramètres de réécriture des URL séparées en modules logiques.

Typiquement les *preVars* et les *postVarSets* sont utilisées pour réécrire les variables *GET* générées par des paramètres :

- soit directement liés à la navigation comme la langue du site, une page en no cache...
- soit générées par les extensions

Dans le fichier de configuration par défaut d'URL Tool des exemples sont fournis pour ces deux types de réécriture.

g) Exemples de preVars

```
'preVars' => array (

'0' => array (

'GETvar' => 'no_cache',

'valueMap' => array (

'nc' => '1',

),

'noMatch' => 'bypass'
),
```

Cette première partie traduit la variable &no_cache=1 en /nc/ en détectant la variable "no_cache" et en assignant un tableau de valeurs de correspondance.

```
'1' => array (
    'GETvar' => 'L',
    'valueMap' => array (
        'fr' => '0',
        'en' => '1',
    ),
    'noMatch' => 'bypass',
    ),
}
```

Ici sur le même principe on va détecter la variable de langue du site (L) ; ici la valeur par défaut est le français, et dans le site on a jouté une langue anglais qui à pris l'id 1 (table sys_language).

Au lieu de : www.domain.com/news?L=1&no cache=1

on aura: www.domaine.com/nc/en/news/

h) Exemple de fixedPostVars

Dans l'exemple fourni par URL Tool on retrouve le même mode de fonctionnement que les preVars : récupération de variable et traduction via tableau de correspondance.

Mais ici il y a plus intéressant : dans cet exemple il est possible de choisir dans une table de la base de données un champ spécifique en récupérant un autre champ (typiquement l'uid) dans l'URL de départ.

```
'article' => array (
            '0' => array (
              'GETvar' => 'tx_ttnews[tt_news]',
              'lookUpTable' => array (
                 'table' => 'tt news',
                 'id_field' => 'uid',
                 'alias field' => 'title',
                'addWhereClause' => ' AND NOT deleted',
                'useUniqueCache' => '1',
                'useUniqueCache conf' => array (
                   'strtolower' => '1',
                   'spaceCharacter' => '-',
                 ),
              ),
            ),
 ),
```

Ici on reçoit une variable de l'extension tt_news : tx_newsboy[tt_news]

On va chercher dans la table tt_news le champ "title" dont le paramètre reçu {uid de la news} correspond au champ uid : ce sont des paramètres de requête SQL (attention aux autres paramètres, à gérer au cas par cas).

Au lieu de : www.domaine.com/actualites?tx_newsboy[tt_news]={uid de la news}

On aura: www.domaine.com/actualites/titre-de-la-news

i) Documentations

Ce document ne détaille qu'une aide à la mise en route ainsi que des pistes et quelques exemples destinés à la compréhension de RealURL.

Cette extension comme on l'a vu dans les exemples permet de générer des URL "intelligentes" pour peu qu'elle soit correctement paramétrée.

La doc officielle se trouve ici :

http://typo3.org/documentation/document-library/extension-manuals/realurl/1.1.4/

Je conseillerai toutefois celle du Wiki Typo3, qui est améliorée et assez compréhensible :

http://wiki.typo3.org/index.php/EXT/realurl/manual

a) Les extensions essentiels

- phpmyadmin : permet de consulter la base au sein de typo3
- t3editor : permet d'avoir la couleur syntaxique dans typoscript
- kickstarter : permet de créer et gérer une extension
- rtehtmlarea : permet de paramétrer l'éditeur RTE
- templavoila : permet de créer et gérer un site plus facilement
- ccdebug : permet de débugger le script php d'un plugin
- ttnews : permet d'avoir des news sur son site
- direct_mail : gére les courriels
- indexed search : permet de mettre un moteur de recherche
- loginbox macmade : ajoute une boite de login
- Meta tags, extended : permet de gérer les métatags (pour exemple voir http://www.typo3journal.info/articles/typo3-metatags.html)
- KB_TV_Content Slide: permet de créer un footer avec menu sitemap changer dans le code XML du DS (module templavoila) la ligne suivante, a chaque modification du fichier HTML du template:

```
« 10.source.current=1 »
par
10.source.postUserFunc = tx_kbtvcontslide_pi1->main
10.source.postUserFunc.field = field_openings
```

b) Liens utiles

- http://www.oswd.org/ : galerie de templates typo3
- http://www.osskins.com/main/category/typo3/: galerie de templates typo3
- http://www.typo3journal.info/: toute l'actualité du CMS TYPO3
- http://www.api-typo.com/index.html : documentation des API de typo3
- http://wiki.typo3.org/index.php/: wiki de typo3
- http://forum.typo3.fr/ : forum de typo3
- http://www.rggooglemap.com/: ressources pour typo3

c) Problèmes liés a typo3 : désactiver les warning

Typo3 a un bug de warning assez désagréable. Il est possible de se retrouver avec un warning géant entraînant l'impossibilité d'utiliser le module « **Template** ». Pour remédier à ce problème il suffit de suivre de modifier la ligne « **error_reporting (E_ALL ^ E_NOTICE);** » par « **error_reporting (0);** » dans les trois fichiers suivants :

```
- fichier 1 : « typo3_src\typo3\init.php »
- fichier 2 : « typo3_src\index.php »
- fichier 3 : « \index.php »
```

d) Personnaliser le RTE

Le RTE de typo3 inclus automatiquement les balises ****, , etc... Il est possible de les supprimer en copiant le code ci-dessous dans le setup du template.

```
    enlève toutes les balises du RTE
    lib.parseFunc_RTE.nonTypoTagStdWrap.encapsLines.removeWrapping = 1
    enlève la balise P du RTE
    lib.parseFunc_RTE.nonTypoTagStdWrap.encapsLines >
```

Le RTE de typo3 n'intègre pas tous les boutons utiles au texte. Copier le code ci-dessous dans le tsConfig de la page Root :

```
- ajoute les boutons de couleurs dans RTE
RTE.default {
    disableColorPicker = 0
    disableSelectColor = 0
}
```