Ebook

Création de templates Joomla! 3.x

EXTRAIT GRATUIT

Initiation HTML et CSS

Documentation technique

Tutoriel de création du template 1870j3

Tutoriel de création du template 1870j3bs

Remerciements

Je tiens à remercier tous ceux qui m'ont soutenu et encouragé dans ce projet, ceux qui m'ont aidé, qui ont répondu à mes questions et qui m'ont permis au fil du temps de rassembler toutes ces connaissances.

Je remercie tout particulièrement ma femme Ariane pour sa patience et sa gentillesse pendant les longues heures que j'ai passées pour pouvoir vous proposer ce document.

Cédric KEIFLIN

A propos de moi

J'ai commencé à travailler avec Joomla! 1.0 au début 2008 pour créer mon premier site. J'ai vite compris que le template représente un travail énorme et non négligeable car c'est lui qui contribue à l'aspect visuel et à la beauté du site. Un peu perdu au début, je me suis mis à créer mes propres templates.

Aujourd'hui je me suis spécialisé dans le développement d'extensions Joomla! et j'essaie toujours de fournir des documentations et des tutoriels à la communauté francophone de Joomla!.

Mes sites web

http://www.joomlack.fr
Le site Joomlack avec téléchargement des extensions que j'ai développées, ainsi que tous les documents et tutoriels que j'ai écrits.

http://tutoriels-joomla.joomlack.fr
Site dédié à Joomla! avec des tutoriels pour créer des menus, templates, et apprendre à développer avec Joomla!.

http://www.template-creator.com
Site dédié à mon composant Template Creator CK qui vous permet de créer vos propres templates Joomla de A à Z grâce à une interface ergonomique.

A qui est destiné cet ouvrage

A tous ceux qui veulent apprendre à créer leur propre template avec Joomla!. Novice ou confirmé, il vous suffira d'avoir des connaissances HTML et CSS suffisantes pour pouvoir profiter pleinement de l'ouvrage.

Pour démarrer

Téléchargez et installez Joomla! pour mettre en place votre environnement de test. Assurez-vous d'avoir les notions de composant, module et plugin, ainsi que les positions de mdoules.

TABLE DES MATIERES

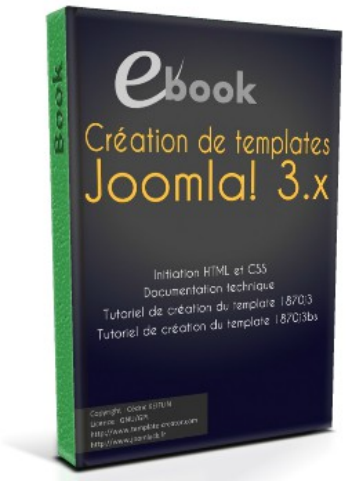
BIBLIOGRAPHIE

http://docs.joomla.org
http://www.w3.org/QA/2002/04/valid-dtd-list.html
http://www.alsacreations.com/
CSS Avancées vers HTML5 et CSS3 de Raphaël Goetter, Edition Eyrolles
http://forum.joomla.org

Le forum Joomla où les échanges avec les autres membres m'ont permis de récolter de nombreuses informations intéressantes et enrichissantes.

Ebook Création de template Joomla! 3

Ce tutoriel est un extrait gratuit du livre de Création de template Joomla! 3.x que vous pouvez retrouver sur http://www.joomlack.fr



Retrouvez l'intégralité du livre Création de template Joomla! 3.x sur http://www.joomlack.fr

320 pages pour vous apprendre à créer vos templates Joomla 3.x qui comprend:
- une documentation technique
- 1 tutoriel pour créer un template classique
- 1 tutoriel pour créer un template avec Bootstrap
- 1 tutoriel vidéo pour créer un template avec Template Creator CK
- 3 templates pour Joomla! 3 corresondant aux 3 tutoriels
- le gabarit à installer dans Template Creator CK

Template Creator CK

Il existe des outils pour créer vos propres templates. Voici une présentation rapide de Template Creator CK qui est un composant Joomla! que j'ai développé et qui vous permet de créer vos propres templates.

Les atouts de Template Creator sont :

- un gain de temps pour la création de vos templates
- un nombre de positions de modules illimité
- un code simple que vous pouvez ensuite modifier si besoin
- un template léger et sans javascript

Ouvrez Template Creator dans votre site Joomla!, donnez un nom à votre template, choisissez une structure de base que vous pouvez ensuite modifier, ajoutez vos styles sur les blocs et modules, puis exporter le pour récupérer le fichier ZIP à installer dans votre site.

Template Creator est compatible Joomla! 2.5 et joomla! 3.x.

Je vous invite à découvrir Template Creator CK sur http://www.template-creator.com

SECTION 3

(extrait du eBook Création de template Joomla! 3.x)

I. TUTORIEL DU TEMPLATE "1870J3"

La pratique à travers un tutoriel.

1. Cahier des charges

Voyons ce qu'on peut faire de beau avec notre template. On va essayer de mettre quelques contraintes pour s'amuser un peu !

1. Module override pour titre bicouleur des modules
2. 4 modules alignés à largeur variable (largeur différente pour chaque module)
3. Menu déroulant horizontal
4. Largeur de colonne qui s'adapte en fonction de la colonne droite et mode édition
5. Paramètre de couleur de fond de la page dans les options du template
6. Un footer qui prend toute la largeur du site
7. Comportement utilisant le responsive design pour les appareils mobiles

Bon voilà deux trois petites choses à faire.

2. Pour commencer

Il va nous falloir un peu d'équipement pour créer le template :

- un ordinateur personnel équipé d'une souris + clavier (et d'un écran accessoirement)
- de l'électricité pour faire tourner l'ordi
- d'un bout de papier pour prendre des notes avec le crayon qui va bien
- à boire (privilégiez la boisson sans alcool si vous voulez tenir jusqu'au bout...)
- un siège confortable ...

Bon allez, assez rigolé, passons aux choses sérieuses !

Pour développer le template il faudra un environnement de test adapté car vous l'imaginez, créer un template sur un site distant en uploadant tous les fichiers modifiés en FTP à chaque fois, c'est pas l'idéal.

2.1) L'éditeur

Commençons par installer un éditeur de texte amélioré. Vous avez le choix, il en existe plusieurs dont Notepad++ qui est distribué gratuitement sous licence GPL.

http://notepad-plus.sourceforge.net/fr/site.htm

Notepad++ nous offre la coloration syntaxique, l'autocompletion et surtout la possibilité de travailler en UTF-8 No-BOM. C'est ce langage que nous allons utiliser pour coder nos fichiers avec Joomla!.

Vous pouvez aussi opter pour un IDE comme Eclipse ou Netbeans si vous êtes plus familier avec ces programmes. Personnellement j'aime bien Notepad++ pour créer les templates car il est léger et qu'on n'a pas trop besoin de déboguer ni d'utiliser l'API de Joomla!.

2.2) Le serveur local

Pour interpréter le code PHP et nous générer nos pages web avec Joomla! il nous faut un serveur web. Nous allons opter pour le très répandu duo Apache / MySQL qui s'installe

très facilement avec des packages comme Wamp, Xamp, Lamp (pour Linux), etc...

http://www.wampserver.com/

http://www.apachefriends.org/fr/xampp.html

Personnellement j'utilise Xampp ou Wamp, ça dépend des humeurs ...

Si vous avez besoin d'aide pour cette installation, pensez à faire un tour sur le forum joomla.fr.

2.3) Firefox et ses extensions

Pour développer le template on va utiliser Firefox parce qu'il dispose d'extensions très utiles. On va donc opter pour Webdeveloper et Firebug qui seront d'une aide indispensable pour identifier les éléments HTML et les CSS associés.

http://www.mozilla-europe.org/fr/firefox/

https://addons.mozilla.org/en-US/firefox/addon/60

https://addons.mozilla.org/fr/firefox/addon/firebug/

Depuix que j'utilise Firebug je ne peux plus m'en passer, il deviendra vite votre compagnon lors de débogage CSS, alors installez le !

2.4) IEtester

Ce petit logiciel est intéressant pour notre projet car c'est lui qui va nous permettre de tester le site, et donc le template, sous toutes les versions d'Internet Explorer (6, 7, 8).

http://www.my-debugbar.com/wiki/IETester/HomePage

N'oubliez pas que vous pouvez aussi utiliser le raccourci F12 dans IE9 pour ouvrir la console de développement d'Internet Explorer et switcher le mode de rendu sur les versions allant de 7 à 9.

2.5) Les logiciels graphiques

Étant un adepte du logiciel libre, n'attendez pas de moi que je vante les mérites de Photoshop. Je vais plutôt vous suggérer de vous pencher sur l'excellent Gimp pour tout ce qui est de la construction de graphiques point par point, notamment pour l'élaboration des maquettes de templates. En ce qui concerne la création vectoriel, pour les boutons spéciaux, les logo ou icônes, c'est vers Inkscape que mon choix se tourne.

Ces deux logiciels sont déjà très élaborés et vous permettront de faire de grandes

choses. Si vous en doutez, c'est peut-être tout simplement parce que vous ne savez pas vous en servir ? …

http://www.gimp.org/

http://www.inkscape.org

2.6) Les bases du langage

Soyons de suite honnêtes, vous n'allez pas apprendre à coder le HTML ni le CSS dans ce tuto, mais juste à exploiter vos connaissances pour construire le template. Si votre niveau n'est pas suffisant (si vous vous posez la question c'est qu'il ne l'est visiblement pas... lol), allez donc faire un petit tour sur des tutoriels très instructifs :

http://www.siteduzero.com/

http://www.w3schools.com/

Sur les deux sites vous pourrez apprendre à maitriser les balises HTML, savoir la différence entre une balise à affichage de type bloc ou en ligne, mais également à aiguiser vos sens du CSS pour positionner, dimensionner et manipuler les éléments à votre guise.

Ces connaissances profondes ne sont pas nécessaires pour suivre le tutoriel, mais elles le seront pour vous permettre d'adapter à vos besoins tout ce que nous allons voir.

2.7) Installation de joomla!

Vous êtes surement déjà passé par cette étape et je ne vais pas vous la détailler. Si vous vous demander encore comment faire, jetez un oeil au document "joomla pour les nuls" qui est vraiment très complet pour aborder les fonctionnalités du logiciel.

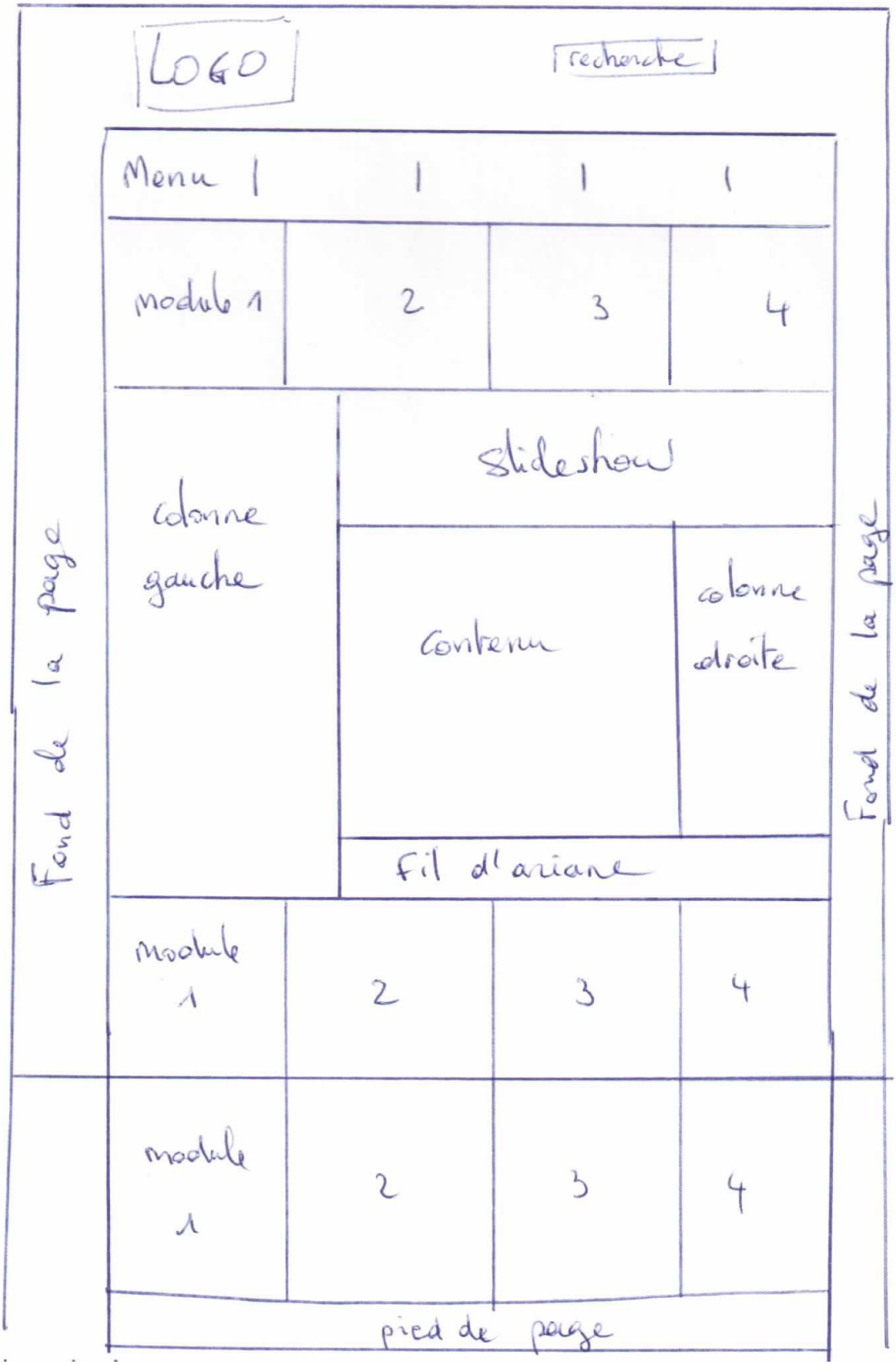
Sinon utilisez le forum.joomla.fr pour trouver une aide amicale qui pourra surement vous guider lors de cette étape.

3. Commençons...

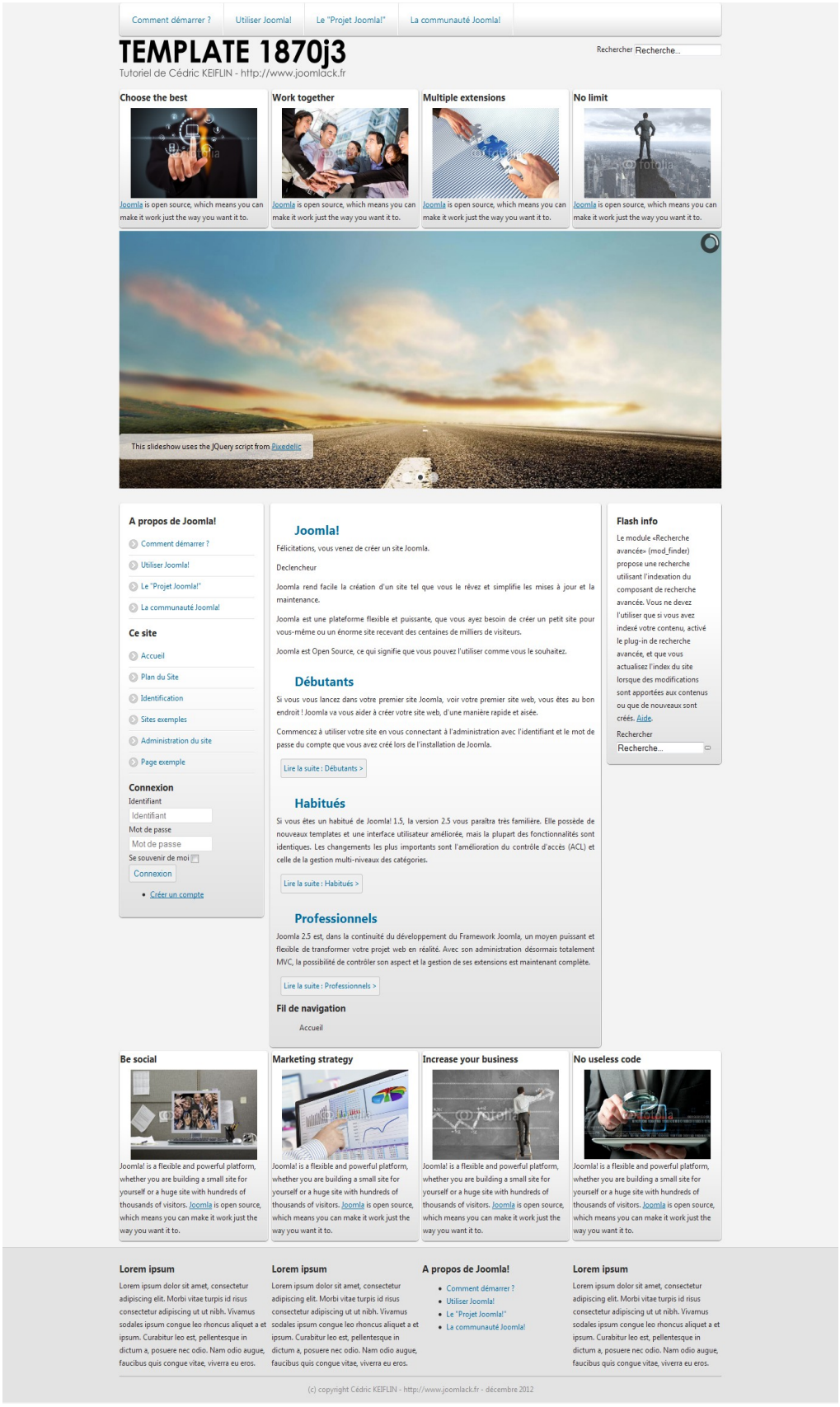
Ha Ha ! Ca y est, on commence... On va devoir s'équiper d'un peu de papier et d'un crayon. Nous allons créer un template avec les éléments dans l'ordre suivant :

- une bannière avec un logo à gauche et une zone de recherche à droite
- un menu horizontal
- une rangée de 4 modules
- un slideshow
- une zone centrale à 3 colonnes
- une rangée de 4 modules
- une rangée de 4 modules dans le pied de page (footer)
- un module simple pour mettre le copyright ou un truc du genre

Voici l'esquisse sortie tout droit de mon imagination ...


LOGO
recherche
Menu
module 1
2
3
4
slideshow
colonne gauche
contenu
colonne droite
fil d'ariane
module 1
2
3
4
module 1
2
3
4
pied de page
Fond de la page
Fond de la page

Voilà mon beau croquis ! Mais si, il est beau ! :)
C'est un dessin très simple mais au moins on voit tout de suite l'arrangement du template et où on va mettre les blocs. Ne négligez pas cette étape, moi ça m'a pris 3 minutes pour le faire.
Voyons à quoi devrait ressembler le template final :


Comment démarrer ?
Utiliser Joomla!
Le "Projet Joomla!"
La communauté Joomla!
TEMPLATE 1870j3
Tutoriel de Cédric KEIFLIN - http://www.joomlack.fr
Rechercher
Recherche...
Choose the best
Work together
Multiple extensions
No limit
Joomla is open source, which means you can make it work just the way you want it to.
This slideshow uses the JQuery script from Pixedelic
A propos de Joomla!
Comment démarrer ?
Utiliser Joomla!
Le "Projet Joomla!"
La communauté Joomla!
Ce site
Accueil
Plan du Site
Identification
Sites exemples
Administration du site
Page exemple
Connexion
Identifiant
Mot de passe
Se souvenir de moi
Connexion
Créer un compte
Joomla!
Félicitations, vous venez de créer un site Joomla.
Declencheur
Joomla rend facile la création d'un site tel que vous le rêvez et simplifie les mises à jour et la maintenance.
Joomla est une plateforme flexible et puissante, que vous ayez besoin de créer un petit site pour vous-même ou un énorme site recevant des centaines de milliers de visiteurs.
Joomla est Open Source, ce qui signifie que vous pouvez l'utiliser comme vous le souhaitez.
Débutants
Si vous vous lancez dans votre premier site Joomla, voir votre premier site web, vous êtes au bon endroit ! Joomla va vous aider à créer votre site web, d'une manière rapide et aisée.
Commencez à utiliser votre site en vous connectant à l'administration avec l'identifiant et le mot de passe du compte que vous avez créé lors de l'installation de Joomla.
Lire la suite : Débutants >
Habitués
Si vous êtes un habitué de Joomla! 1.5, la version 2.5 vous paraîtra très familière. Elle possède de nouveaux templates et une interface utilisateur améliorée, mais la plupart des fonctionnalités sont identiques. Les changements les plus importants sont l'amélioration du contrôle d'accès (ACL) et celle de la gestion multi-niveaux des catégories.
Lire la suite : Habitués >
Professionnels
Joomla 2.5 est, dans la continuité du développement du Framework Joomla, un moyen puissant et flexible de transformer votre projet web en réalité. Avec son administration désormais totalement MVC, la possibilité de contrôler son aspect et la gestion de ses extensions est maintenant complète.
Lire la suite : Professionnels >
Fil de navigation
Accueil
Flash info
Le module «Recherche avancée» (mod_finder) propose une recherche utilisant l'indexation du composant de recherche avancée. Vous ne devez l'utiliser que si vous avez indexé votre contenu, activé le plug-in de recherche avancée, et que vous actualisez l'index du site lorsque des modifications sont apportées aux contenus ou que de nouveaux sont créés. Aide.
Rechercher
Recherche...
Be social
Marketing strategy
Increase your business
No useless code
Joomla! is a flexible and powerful platform, whether you are building a small site for yourself or a huge site with hundreds of thousands of visitors. Joomla is open source, which means you can make it work just the way you want it to.
Lorem ipsum
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi vitae turpis id risus consectetur adipiscing ut ut nibh. Vivamus sodales ipsum congue leo rhoncus aliquet a et ipsum. Curabitur leo est, pellentesque in dictum a, posuere nec odio. Nam odio augue, faucibus quis congue vitae, viverra eu eros.
A propos de Joomla!
Comment démarrer ?
Utiliser Joomla!
Le "Projet Joomla!"
La communauté Joomla!
(c) copyright Cédric KEIFLIN - http://www.joomlack.fr - décembre 2012

4. Créer la structure pour commencer

Attention, tout au long du tutoriel il y a du code à copier et coller. Vous pouvez l'utiliser directement de ce fichier sauf pour les lignes de code qui sont trop longues et qui sont affichées sur 2 voire plusieurs lignes. Il faut alors supprimer le saut à la ligne et le remplacer par un espace (ou rien) dans vos fichiers. Sinon le code ne fonctionnera pas.

J'espère que vous êtes prêt, car c'est maintenant qu'on attaque le boulot. On va créer la structure de base de notre template qui nous servira tout au long du projet.
Le template que nous allons créer porter le nom : "**1870j3**".

Les lignes de code qu'il faudra mettre dans le template seront précédées du mot **CODE**. Les autres lignes de code servent à présenter les explications mais ne sont pas à intégrer telles quel dans le template.

Nous travaillerons dans le répertoire du site directement sans avoir à installer le template auparavant (évident puisqu'il n'existe pas encore). Nous ferons le package ZIP à la fin pour le tester et l'installer.

Commençons par démarrer le serveur web (xampp, wamp ou autre). Allons dans le dossier où sont stockés les templates, on y crée le dossier de notre template :
[site]/templates/1870j3

Ensuite on crée les fichiers vierges :

- index.html
- index.php
- templateDetails.xml

On crée un répertoire "css" dans celui du template dans lequel on crée 2 fichiers :

- index.html
- template.css

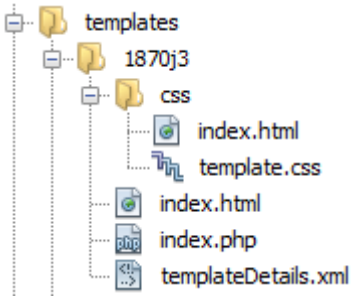


Illustration 1: Structure de base du template

On va remplir les données au fur et à mesure de la création.

5. index.html

Il suffit de mettre la ligne de code suivante qui nous servira dans tous les fichiers index.html :

CODE du fichier INDEX.HTML

```
<html><body bgcolor="#FFFFFF"></body></html>
```

6. index.php

Les premières lignes qui restreignent l'accès au code :

CODE du fichier INDEX.PHP

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

Ensuite on déclare le Doctype HTML5

CODE du fichier INDEX.PHP

```
<!DOCTYPE html>
```

On ouvre la page HTML et son en-tête. On prend soin de bien déclarer la langue ainsi que la direction de lecture pour les pays qui lisent de droite à gauche.

CODE du fichier INDEX.PHP

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="<?php echo $this-
>language; ?>" lang="<?php echo $this->language; ?>" dir="<?php echo $this-
>direction; ?>">
<head>
```

On demande à Joomla! d'inclure les balises meta, title, et tous les trucs nécessaires qui sont paramétrés dans le site

CODE du fichier INDEX.PHP

```
<jdoc:include type="head" />
```

On appelle les feuilles de style système pour inclure certaines fonctionnalités par défaut comme les champs invalide encadrés en rouge dans le formulaire de contact.

CODE du fichier INDEX.PHP

```
<link rel="stylesheet" href="<?php echo $this->baseurl ?
>/templates/system/css/system.css" type="text/css" />
    <link rel="stylesheet" href="<?php echo $this->baseurl ?
>/templates/system/css/general.css" type="text/css" />
```

On appelle ensuite la feuille de styles "template.css" qu'on a créée précédemment

CODE du fichier INDEX.PHP

```
<link rel="stylesheet" href="<?php echo $this->baseurl ?>/templates/<?php echo
$this->template ?>/css/template.css" type="text/css" />
```

$this->baseurl : renvoit l'adresse de base du site
$this->template : renvoit le nom du template (défini dans le fichier XML)

Allez on va fermer l'en-tête de la page et ouvrir le corps qui contiendra nos blocs.

CODE du fichier INDEX.PHP

```
</head>
<body>
```

La prochaine étape avant de voir chaque bloc en détail c'est de créer un conteneur principal. Ici on crée un bloc avec une classe "wrapper"

CODE du fichier INDEX.PHP

```
<div class="wrapper">
```

Alors pourquoi une classe ici et pas un ID ? Tout simplement parce que dans mon design (si si rappelez vous ma belle esquisse au stylo) je veux avoir des modules dans le pied de page qui ont un fond différent et qui prend toute la largeur de la page. J'aurai donc besoin de créer un deuxième conteneur pour la partie basse de la page et je pourrai partager les propriétés dans cette même classe CSS.

On va étudier le code HTML de chaque bloc séparément.

6.1) Menu horizontal

CODE du fichier INDEX.PHP

```
<?php if ($this->countModules('position-1')): ?>
<div id="nav" class="clearfix rounded white">
    <jdoc:include type="modules" name="position-1" style="none" />
</div>
<?php endif; ?>
```

C'est pas bien compliqué, on vérifie juste qu'au moins un module est chargé dans la position "position-1" (endroit classique où on charge le topmenu).
Notons tout de même la classe "clearfix" ajoutée au bloc du menu. Elle permet de s'assurer que le conteneur du menu s'affiche correctement et conserve ses proportions même si il contient des éléments flottants. Pour plus de détails sur cette partie il faut se reporter à la documentation technique du livre dans le chapitre d'initiation HTML et CSS.

Pour info, le contenu de la classe "clearfix" est le suivant :

```
.clearfix:after {
	content: " ";
	display: block;
	height: 0;
	clear: both;
	visibility: hidden;
	font-size: 0;
}

.clearfix {
	zoom: 1;
}
```

Les classes "rounded" et "white" seront définies en CSS pour créer des coins arrondis et un fond blanc dégradé, tout ça pour styler tous les modules de la même manière.

6.2) Bannière - logo et module

CODE du fichier INDEX.PHP

```
<div id="header" class="clearfix">
    <a id="logo" href="<?php echo $this->baseurl; ?>">
          <img src="<?php echo JURI::root() ?>templates/<?php echo $this-
>template ?>/images/logo.png" alt="<?php echo JFactory::getApplication()-
>getCfg('sitename') ?>" />
    </a>
    <div id="headermodule">
    <jdoc:include type="modules" name="position-0" style="none" />
    </div>
</div>
```

Ici on définit le logo avec une image sur laquelle on applique un lien qui redirige vers la page accueil. L'image est nommé "logo.png" et est stockée dans le dossier "images" du template

```
<?php echo JURI::root() ?>templates/<?php echo $this->template ?
>/images/logo.png
```

Finalement on ajoute un bloc "headermodule" dans lequel on charge une position de modules "position-0". C'est donc dans cette position qu'on va publier un module de recherche.

Pour les détails des appels JDOC, consultez le chapitre II.2

6.3) Rangée de 4 modules

Alors là, on attaque une partie déjà plus corsée. Ici on ne vérifie pas juste la présence d'un module, mais la présence de chacun des modules. En fonction du nombre de modules publiés on va créer une classe spécifique avec une variable PHP. Grâce à cette classe on pourra gérer les différentes situations :

- si aucun module n'est publié → on n'affiche rien
- si un seul module est publié → on affiche la rangée et le module prend 100% de la largeur
- si deux modules sont publiés → on affiche la rangée et chacun prend 50% de la place.
- si trois modules sont publiés → on affiche la rangée et chacun prend 33% de la place.
- si quatre modules sont publiés → on affiche la rangée et chacun prend 25% de la place.

D'abord on ajoute un code PHP dans l'en-tête de la page, c'est à dire juste avant la balise "</head>"

CODE du fichier INDEX.PHP - En-tête

```
<?php
$nbmodulesrow1 = (bool)$this->countModules('position-8') + (bool)$this-
>countModules('position-9') + (bool)$this->countModules('position-10') +
(bool)$this->countModules('position-11');
?>
```

(bool)$this->countModules('position-8') : est-ce qu'il y a au moins un module de chargé dans la position "position-8" ?

Exemples de situations :

Etat des modules	***Valeur de la variable $nbmodulesrow1***
Position-8 : aucun module publié Position-9 : aucun module publié Position-10 : aucun module publié Position-11 : aucun module publié	0
Position-8 : au moins un module publié Position-9 : aucun module publié Position-10 : au moins un module publié	2

Position-11 : aucun module publié	
Position-8 : au moins un module publié Position-9 : au moins un module publié Position-10 : au moins un module publié Position-11 : au moins un module publié	4

Dans le corps du template on ajoute ce code pour charger 4 modules

CODE du fichier INDEX.PHP

```
<?php if ($nbmodulesrow1): ?>
<div id="row1modules" class="clearfix <?php echo 'n'.$nbmodulesrow1 ?>">
       <?php if ($this->countModules('position-8')) : ?>
       <div class="row1module">
            <div class="inner rounded white">
                <jdoc:include type="modules" name="position-8"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-9')) : ?>
       <div class="row1module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-9" style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-10')) : ?>
       <div class="row1module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-10"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-11')) : ?>
       <div class="row1module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-11"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
</div>
<?php endif; ?>
```

Si la valeur de "$nbmodulesrow1" est supérieure à 0 (au moins 1 module à afficher), on crée le conteneur qu'on pourrait définir comme la ligne qui va contenir les 4 modules alignés. On lui donne une classe basée sur le nombre de modules chargés grâce à notre variable :

```
class="clearfix <?php echo 'n'.$nbmodulesrow1 ?>">
```

la classe peut prendre la valeur "n0" à "n4" ce qui nous permettra de gérer les largeurs en CSS.

Ensuite on crée chaque module en vérifiant bien qu'il y a quelque chose à afficher grâce à la condition PHP "countModules" :

```
<?php if ($this->countModules('position-8')) : ?>
```

On y insère un bloc qui a la classe "inner". C'est ce qu'on appelle la méthode des divs imbriquées (voir chapitre correspondant dans la documentation technique). De cette manière on peut appliquer la largeur du module au bloc qui a la classe "row1module"et les marges ainsi que tous les autres styles au bloc qui a la classe "inner". On évite les soucis de rendu sur les navigateurs.

On affiche chaque module avec le style "xhtml" qui permet d'avoir un module simple et un titre de module dans une balise H3.

On verra un peu plus tard dans la partie CSS comment gérer les différentes largeurs et possibilités, mais pour vous donner un aperçu de ce qu'on va pouvoir faire, voici un exemple de définition des largeurs :

```
#row1modules.n1 .row1module {
    width: 100%;
}

#row1modules.n2 .row1module {
    width: 50%;
}


#row1modules.n3 .row1module {
    width: 33%;
}


#row1modules.n4 .row1module {
    width: 25%;
}
```

Ça va j'espère que ce n'est pas trop compliqué à comprendre :). Vous verrez quand on attaquera la partie CSS on ira un peu plus loin dans la personnalisation des largeurs...

Pour finir notons juste qu'on a ajouté la classe "clearfix" sur le conteneur, étant donné qu'il va englober des éléments flottants (les modules auxquels on appliquera un "float:left;").

6.4) Slideshow

CODE du fichier INDEX.PHP

```
<div id="slideshow">
    <jdoc:include type="modules" name="position-3" style="xhtml" />
</div>
```

Bon, si ça peut vous rassurer on ne va pas coder un slideshow de A à Z ! le mieux est de prévoir un simple module dans lequel on pourra chargé n'importe quelle extension de slideshow. Personnellement j'aime bien mon Slideshow CK :)

Slideshow CK est un module de slideshow responsive et compatible mobiles qui vous permet de gérer vos images directement depuis l'interface du module avec un système Drag'n'Drop.
http://www.joomlack.fr/extensions-joomla/slideshow-ck

6.5) Zone centrale à 3 colonnes

Si on regarde notre maquette graphique on voit que la partie centrale est constituée de 3 colonnes (gauche, contenu et droite). Au dessus du contenu on ajoute un deuxième emplacement de slideshow et en dessous on met un fil d'ariane.

La grosse difficulté ici c'est de gérer la largeur des colonnes lorsque l'on publie des modules ou pas dans les colonnes. On se retrouve avec 3 scénarios :

- pas de module à gauche → on affiche le contenu et la colonne de droite
- pas de module à droite, ou mode édition → on affiche la colonne de gauche et le contenu
- pas de module à gauche ni à droite, ou pas de module à gauche et mode édition → on n'affiche que le contenu

Là vous me dites mais c'est quoi ce "mode édition" ? Imaginez que vous êtes connecté en partie frontale du site et que vous voulez éditer un article (ou autre chose). Le formulaire apparaît et vous pouvez modifier les données. A ce moment-là c'est toujours plus sympa d'avoir un maximum d'espace pour rédiger les textes et utiliser les icônes, c'est pourquoi on décide de cacher la colonne de droite et utiliser cet espace.

Définissons une variable "$mainclass" qui nous servira à savoir dans quel scénario on est. On insère le code dans l'en-tête du document avant la balise "</head>".

CODE du fichier INDEX.PHP - En-tête

```
<?php
$mainclass = '';
if (!$this->countModules('position-7'))
{
    $mainclass .= " noleft";
}
if (!$this->countModules('position-6') || $app->input->getCmd('task', '') ==
'edit')
{
    $mainclass .= " noright";
}
$mainclass = trim($mainclass);
?>
```

On récupère l'info dans les variables de Joomla ! à savoir si on est en train d'éditer un truc, genre un article.

```
$app->input->getCmd('task', '') == 'edit'
```

Donc notre variable "$mainclass" aura les valeurs : (vide), "noleft", ou "noright". Utilisons-là dans le code de notre template.

CODE du fichier INDEX.PHP

```
<div id="main" class="clearfix <?php echo $mainclass ?>">
    <?php if ($this->countModules('position-7')): ?>
    <div id="left">
        <div class="inner rounded white">
          <jdoc:include type="modules" name="position-7" style="xhtml" />
        </div>
    </div>
    <?php endif; ?>
    <div id="center">
          <div class="inner rounded white">
            <jdoc:include type="modules" name="position-5" style="xhtml" />
            <jdoc:include type="message" />
            <jdoc:include type="component" />
            <jdoc:include type="modules" name="position-2" style="xhtml" />
          </div>
    </div>
    <?php if ($this->countModules('position-6')) : ?>
    <div id="right">
          <div class="inner rounded white">
             <jdoc:include type="modules" name="position-6" style="xhtml" />
          </div>
    </div>
    <?php endif; ?>
</div>
```

De la même manière qu'avant on utilise la classe "clearfix" pour régler le souci des éléments flottant, et les classes "rounded" et "white" pour donner l'allure générale aux blocs. On utilise aussi la méthode des divs imbriquées en ajoutant des div avec les classes "inner". Je ne détaille pas puisqu'on a déjà vu tout cela précédemment !

```
<jdoc:include type="modules" name="position-5" style="xhtml" />
<jdoc:include type="message" />
<jdoc:include type="component" />
<jdoc:include type="modules" name="position-2" style="xhtml" />
```

Ici nous faisons les choses simples, on affiche le module de slideshow 2, les messages systèmes, le contenu principal de la page et le fil d'ariane en "position-2". Tout ça à la suite, étant donné que ce sont des éléments de type bloc ils se placeront automatiquement les uns sous les autres. Ne nous compliquons pas la vie lorsque ce n'est pas nécessaire !

Voyons rapidement comment on pourra utiliser la variable "$mainclass" dans les CSS pour définir les largeurs des éléments selon nos 3 scénarios.

```
#left {
    width: 25%;
}

#right {
    width: 25%;
}

#center {
    width: 50%;
}

.noleft #center {
    width: 75%;
}

.noright #center {
    width: 75%;
}

.noright.noleft #center {
    width: 100%;
}
```

On uitlise tout simplement les classes CSS "noleft" et "noright" pour cibler les colonnes, c'est très simple (c'est toujours simple quand on sait comment faire lol).

6.6) Rangée de 4 modules sous le contenu

Bon, ne réinventons pas la roue à chaque épisode, on avait une rangée de 4 modules tout à l'heure. Utilisons le même code juste en faisant varier les identifiants et classes HTML là où c'est nécessaire, ainsi que les positions des modules.

CODE du fichier INDEX.PHP - En-tête

```
<?php
$nbmodulesrow2 = (bool)$this->countModules('position-12') + (bool)$this-
>countModules('position-13') + (bool)$this->countModules('position-14') +
(bool)$this->countModules('position-15');
?>
```

CODE du fichier INDEX.PHP

```
<?php if ($nbmodulesrow2): ?>
<div id="row2modules" class="clearfix <?php echo 'n'.$nbmodulesrow2 ?>">
       <?php if ($this->countModules('position-12')) : ?>
       <div class="row2module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-12"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-13')) : ?>
       <div class="row2module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-13"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-14')) : ?>
       <div class="row2module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-14"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-15')) : ?>
       <div class="row2module">
            <div class="inner rounded white">
               <jdoc:include type="modules" name="position-15"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
</div>
<?php endif; ?>
```

6.7) Rangée de 4 modules dans le pied de page

Punaise, c'est qui qui a mis tous ces modules dans ce template ! lol

Bon on recommence la même opération avec une rangée "row3" et des positions de module "position-16" à "position-19".

Stop ! Hé là, pas si vite ! Regardons notre maquette graphique de plus près, on a bien dit qu'on voulait faire une séparation avec ces modules pour avoir un fond de page différent. Ça veut dire qu'on doit fermer le conteneur actuel (wrapper) et en ouvrir un autre ! Voyons ça avec le code ...

CODE du fichier INDEX.PHP - En-tête

```
<?php
$nbmodulesrow3 = (bool)$this->countModules('position-16') + (bool)$this-
>countModules('position-17') + (bool)$this->countModules('position-18') +
(bool)$this->countModules('position-19');
?>
```

CODE du fichier INDEX.PHP

```
</div>
    <div id="body2">
          <div class="wrapper">
<?php if ($nbmodulesrow3): ?>
<div id="row3modules" class="clearfix <?php echo 'n'.$nbmodulesrow3 ?>">
       <?php if ($this->countModules('position-16')) : ?>
       <div class="row3module">
            <div class="inner">
               <jdoc:include type="modules" name="position-16"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-17')) : ?>
       <div class="row3module">
            <div class="inner">
               <jdoc:include type="modules" name="position-17"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-18')) : ?>
       <div class="row3module">
            <div class="inner">
               <jdoc:include type="modules" name="position-18"
style="xhtml" />
            </div>
       </div>
       <?php endif; ?>
       <?php if ($this->countModules('position-19')) : ?>
```

```
        <div class="row3module">
             <div class="inner">
                <jdoc:include type="modules" name="position-19"
style="xhtml" />
             </div>
        </div>
        <?php endif; ?>
</div>
<?php endif; ?>
```

Voilà, un peu d'explications ?

```
</div>
    <div id="body2">
          <div class="wrapper">
```

On commence par fermer le wrapper précédent avec , et on en ouvre un nouveau bloc qui servira de fond de page. On lui ajoute aussi un identifiant pour pouvoir l'attaquer directement dans les CSS et lui mettre notre fameux fond de couleur différente.

Ensuite on crée un nouveau conteneur avec la classe "wrapper" qu'on a déjà utilisée tout au début. C'est lui qui va contenir les éléments du bas.
Notez qu'on n'applique pas les classes "white" et "rounded" ici.

6.8) Module de pied de page

On touche à la fin, c'est le dernier module à ajouter.

CODE du fichier INDEX.PHP

```
<div id="footer">
      <jdoc:include type="modules" name="position-4" style="none" />
</div>
```

Je crois que là après tout ce qu'on vient de voir vous arrivez à comprendre tout seul qu'il s'agit d'un simple bloc dans lequel on affiche les modules en "position-4".

6.9) Fin du document

Je pense qu'il est temps de clore cette partie et passer à la suite.

CODE du fichier INDEX.PHP

```
            </div>
        </div>
        <jdoc:include type="modules" name="debug" style="none" />
    </body>
    </html>
```

Le premier sert à fermer le conteneur "wrapper" qu'on a ouvert précédemment, et le deuxième sert à fermer son parent "body2". Ensuite on ajoute une position de module "debug" qui permet d'afficher les messages système de Joomla ! lorsque l'on active le mode de débogage. Et on finit simplement par fermer le corps de la page et le document HTML.

Ouf... on y est arrivé ! :)

7. templateDetails.xml

On ouvre le fichier XML (celui qu'on avait créé vierge au début) et on attaque le code.

CODE du fichier TEMPLATEDETAILS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE install PUBLIC "-//Joomla! 2.5//DTD template 1.0//EN"
"http://www.joomla.org/xml/dtd/2.5/template-install.dtd">
<extension version="3.0" type="template" client="site" method="upgrade">
```

On déclare simplement que l'on se trouve dans un document XML avec le DocType qui va bien, et on ouvre la balise "extension" pour y placer nos lignes de codes Joomla !.

method="upgrade" → cette option permet de réinstaller le template par dessus une ancienne version. Très pratique pour les mises à jour ! Sans cette option Joomla ! Affiche une erreur et vous dit qu'il y a déjà du monde dans cet emplacement.

On remplit ensuite les champs qui renseignent les informations sur le créateur, la version , la licence, etc...

CODE du fichier TEMPLATEDETAILS.XML

```
    <name>1870j3</name>
    <version>1.0</version>
    <creationDate>november 2012</creationDate>
    <author>Cédric KEIFLIN</author>
    <authorEmail>ced1870@gmail.com</authorEmail>
    <copyright>Copyright (C) 2012 Cédric KEIFLIN All rights
reserved.</copyright>
    <description>TPL_1870J3_XML_DESCRIPTION</description>
```

Attention à encoder le fichier en UTF8 No-Bom pour l'utilisation des accents et caractères spéciaux. Cette remarque est valable pour tous les fichiers du template.

Le paramètre `<name>` doit comporter le nom du template qui doit être le même que le nom du répertoire. Il ne doit pas comporter de caractères spéciaux ni d'espaces, et éviter les majuscules.

On utilise la variable "TPL_1870J3_XML_DESCRIPTION" pour pouvoir traduire cette valeur dans plusieurs langues grâce aux fichiers de langue du template.

Ensuite il faut lister les fichiers qui seront copiés sur le site lors de l'installation du template. Si un fichier est manquant dans la liste il ne sera pas installé.
Étant donné que l'on ne connaît pas encore le nom ni l'architecture de tous les dossiers et fichiers que nous aurons à la fin, nous allons préparer le travail et finaliser la liste à

la fin de la création du template.

CODE du fichier TEMPLATEDETAILS.XML

```
<files>
	<filename>favicon.ico</filename>
	<filename>index.html</filename>
	<filename>index.php</filename>
	<filename>template_thumbnail.png</filename>
	<filename>templateDetails.xml</filename>
	<filename>template_preview.png</filename>
	<folder>images</folder>
	<folder>css</folder>
</files>
```

Créons maintenant les positions de notre template. Ces positions sont celles que nous utilisons dans le fichier "index.php" avec les appels JDOC. Elles seront listées dans la liste de sélection dans l'administration du site (gestion des modules).

CODE du fichier TEMPLATEDETAILS.XML

```
<positions>
	<position>position-0</position>
	<position>position-1</position>
	<position>position-2</position>
	<position>position-3</position>
	<position>position-4</position>
	<position>position-5</position>
	<position>position-6</position>
	<position>position-7</position>
	<position>position-8</position>
	<position>position-9</position>
	<position>position-10</position>
	<position>position-11</position>
	<position>position-12</position>
	<position>position-13</position>
	<position>position-14</position>
	<position>position-15</position>
	<position>position-16</position>
	<position>position-17</position>
	<position>position-18</position>
	<position>position-19</position>
</positions>
```

Comme dit plus haut pour que les positions soient dans la liste déroulante dans l'administration du site, il faut qu'elle soit écrites ici.

On va ajouter les fichiers de langue qui permettent d'ajouter les traductions des termes et les descriptions des positions.

CODE du fichier TEMPLATEDETAILS.XML

```
	<languages folder="language">
		<language tag="fr-FR">fr-FR/fr-FR.tpl_1870j3.ini</language>
		<language tag="fr-FR">fr-FR/fr-FR.tpl_1870j3.sys.ini</language>
	</languages>
```

Dans le cahier des charges on a vaguement parlé d'un paramètre pour piloter la couleur de fond de la page. Alors allons-y ! On utilisera un champ "color" qui est un colorpicker vous permettant de sélectionner la couleur sur une palette. Cool, non ?

CODE du fichier TEMPLATEDETAILS.XML

```
	<config>
		<fields name="params">
			<fieldset name="advanced">
				<field
					name="templateBackgroundColor"
					type="color"
					default="#F4F6F7"
					label="TPL_1870J3_BACKGROUND_COLOR_LABEL"
					description="TPL_1870J3_BACKGROUND_COLOR_DESC" />
			</fieldset>
		</fields>
	</config>
```

Et on finit le fichier :

CODE du fichier TEMPLATEDETAILS.XML

```
</extension>
```

Vous savez quoi ? Notre template est prêt à être utilisé ! Hé oui difficile à croire, mais c'est la vérité ! Installons donc notre nouveau template.

Prenons l'ensemble du dossier du template auquel on a bien sur donné le bon nom "1870j3". Copions ce dossier dans le dossier
[site]/templates/

Sous Joomla 1.5 ça suffisait pour installer un template, maintenant il s'installe comme n'importe quelle autre extension. Il faut donc passer par le gestionnaire d'extensions :
Extension >> Gestion des extensions >> Découvrir

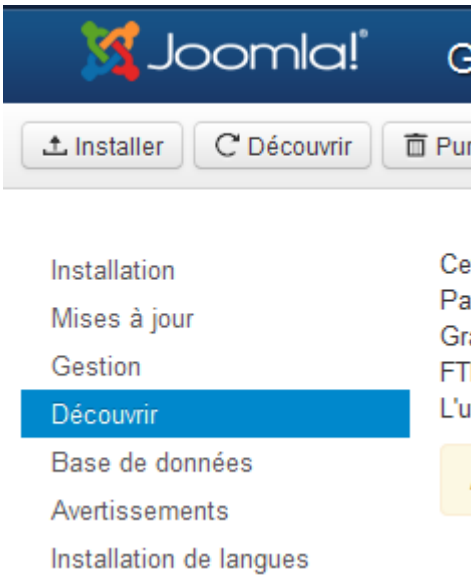
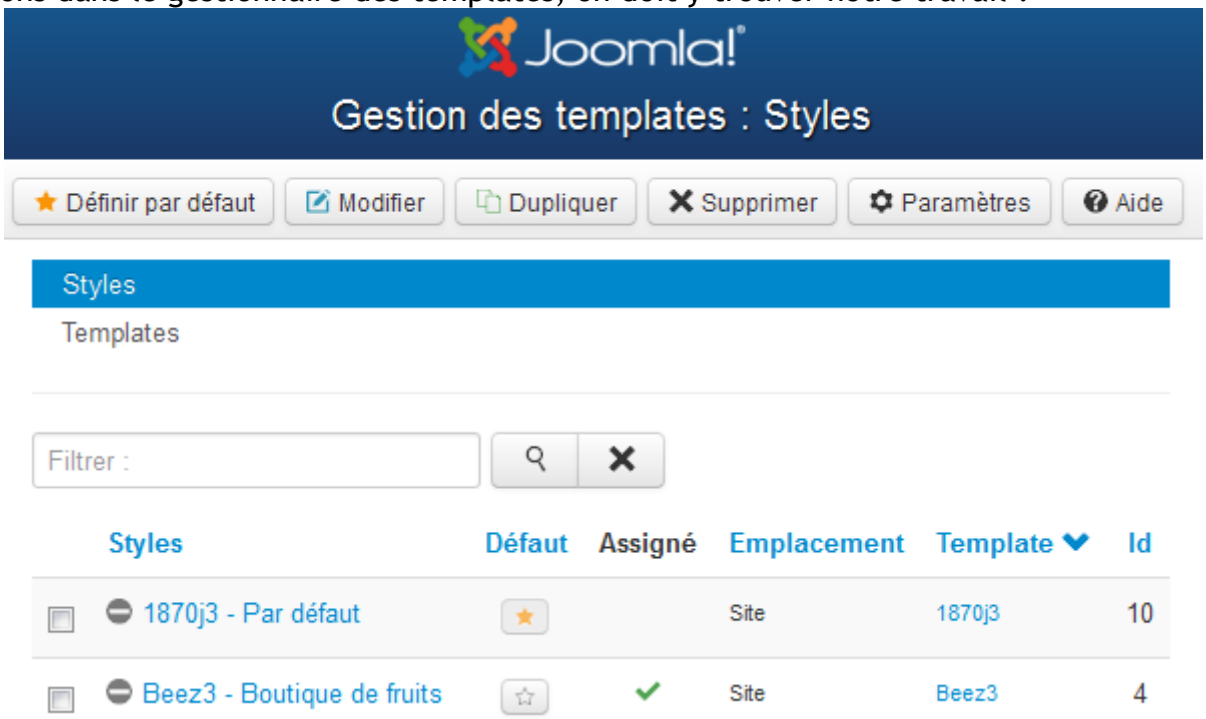


Illustration 2: Mode "Découvrir" dans la gestion des extensions

Joomla doit trouver notre template et nous proposer de l'installer. Pour cela on le sélectionne et on clique sur le bouton "Installer".
Allons dans le gestionnaire des templates, on doit y trouver notre travail :



On clique sur le bouton en forme d'étoile pour le désigner comme template par défaut.
Maintenant nous allons pouvoir travailler directement dedans pour y ajouter les styles et en faire un truc sympathique !

8. Option du template : couleur de fond

On a décidé d'après notre cahier des charges d'ajouter un paramètre dans l'administration du template qui permet de piloter la couleur du fond de la page.

Tout d'abord on édite le fichier XML pour y ajouter le champ qui sera visible dans l'administration du template. Pas la peine de le recopier, on l'a intégré précédemment dans notre fichier, je vous rappelle juste ici le code qui permet de montrer le champ avec le colorpicker.

```
<field
     name="templateBackgroundColor"
     type="color"
     default="#F4F6F7"
     label="TPL_1870J3_BACKGROUND_COLOR_LABEL"
     description="TPL_1870J3_BACKGROUND_COLOR_DESC" />
```

Maintenant il va falloir utiliser cette information dans le fichier "index.php". C'est très simple, on insère la valeur du paramètre dans une déclaration CSS qu'on ajoute dans l'en-tête du document juste avant la balise "</head>".

CODE du fichier INDEX.PHP - En-tête

```
<style type="text/css">
    body
    {
background-color: <?php echo $this->params->get('templateBackgroundColor');?>
    }
</style>
```

Voilà, rien de bien compliqué ! Par défaut on aura une couleur de fond #F4F6F7, et sinon ce sera la couleur définie dans les paramètres.

$this->params->get('templateBackgroundColor') → permet de charger la valeur stockée dans le paramètre qui porte le nom défini dans l'attribut "name" de celui-ci dans le fichier XML (ici "templateBackgroundColor").

Pour ceux qui ont lu les précédents livres que j'ai publiés, vous pouvez noter le changement de méthode. Auparavant j'injectais directement la valeur dans un attribut "style" ajouté en ligne sur le "body". Ici c'est une méthode plus générique qui permet de piloter n'importe quel élément de la page à condition d'utiliser les bons CSS.

label="TPL_1870J3_BACKGROUND_COLOR_LABEL" → On utilise toujours une chaîne de caractères spécifique au template que l'on traduira dans les fichiers de langue.

9. Mise en place des CSS

Bon j'imagine que beaucoup d'entre vous attendent ce moment... Attaquons les CSS ! Je vais essayer de vous guider au fur et à mesure pour faire le lien entre les CSS et les codes HTML que nous avons mis en place plus haut. Tous les CSS que nous allons créer se glisseront dans le fichier "template.css", sauf indication contraire. Commencez déjà par créer le fichier "template.css" dans un dossier "css" du template, et n'oubliez pas de mettre de suite un fichier "index.html" dans le dossier pour la sécurité du site.

Ce qu'il faut savoir avant toute chose, c'est que le langage CSS n'est pas simple, il n'est pas intuitif, il n'est pas facilement débogable. De tous les langages je crois que c'est sûrement le plus fourbe de tous ! Le seul moyen de l'apprivoiser est de tester, tester, et tester encore ! Alors courage lors de vos investigations.

9.1) Initialisation des styles

La première des choses est d'initialiser les styles des balises HTML. Vous savez sûrement que chaque navigateur a des valeurs prédéfinies pour l'ensemble des balises standards. Or ces valeurs varient d'un navigateur à l'autre ! La meilleure des choses est de s'assurer qu'on applique la même chose partout. Donc on commence par mettre les marges à zéro et on les redéfinit.

CODE du fichier template.css

```
html {
  height: 101%;
}

body {
  margin: 0;
  padding: 0;
}

* {
    padding: 0;
    margin: 0;
}

h1, h2, h3, h4, h5, h6, .contentheading, .componentheading {
  padding: 3px 0;
  margin: 0;
  line-height: 1.2;
  font-weight: bold;
  font-style: normal;
}

h1, .componentheading {
  font-size: 1.75em;
}
```

```
h2, .contentheading {
  font-size: 1.5em;
}

h3 {
  font-size: 1.25em;
}

h4 {
  font-size: 1em;
}

ul, ol {
  padding: .75em 0 .75em 0;
  margin: 0 0 0 35px;
}

ul.menu {
    margin: 0;
}

ul.menu li {
    list-style: none;
}

p {
  padding: 5px 0;
}

address {
  margin: .75em 0;
  font-style: normal;
}

a:focus {
    outline: none;
}

img {
  border: none;
}

em {
  font-style: italic;
}
strong {
  font-weight: bold;
}

form, fieldset {
  margin: 0;
  padding: 0;
  border: none;
}
input, button, select {
  vertical-align: middle;
```

```
}

.clr {
    clear : both;
}

.clearfix:after {
    content: " ";
    display: block;
    height: 0;
    clear: both;
    visibility: hidden;
    font-size: 0;
}

.clearfix {
    zoom: 1;
}
```

A la fin on définit la classe "clearfix" (rappelez-vous on l'a utilisée dans le code HTML du fichier "index.php"). Si vous n'avez pas bien compris à quoi ça sert, demandez à Google, c'est votre ami. Cette méthode est aussi clairement expliquée dans le livre de Raphaël Goetter et dans la section technique de ce livre.

Tout au long du tutoriel, dès que vous voyez un style CSS qui inclut un background avec une image, pensez à la chercher dans le template "1870j3" fourni avec le tutoriel pour la mettre dans le dossier "images" de votre template. (ou alors si vous êtes fainéant, copiez tout de suite toutes les images!)

9.2) Styles généraux

On va définir l'allure générale de la page : fond et couleur, police de caractères et taille. Ce seront les paramètres par défaut des éléments de la page.

CODE du fichier template.css

```
body {
    background: #f3f3f3;
    color: #2b2b2b;
    text-align: left;
    line-height: 20px;
    font-size: 12px;
    font-family: Segoe UI, sans-serif;
}
```

On va ensuite définir les styles par défaut des liens et des titres. les liens soulignés et les liens survolés non soulignés et chaque titre avec ses propres couleurs et styles. On définit aussi les styles des liens de titre H2 puisqu'ils sont utilisés pour les titres des articles qui peuvent être cliquable.

CODE du fichier template.css

```
a, a:visited {
    color: #036c9e;
}

a:hover {
    color: #000;
    text-decoration: underline;
}

h1, div.componentheading {
    color: #036c9e;
    text-align: left;
    letter-spacing: -1px;
    line-height: 25px;
    font-size: 22px;
}

h2, div.contentheading {
    color: #036c9e;
    text-align: left;
    font-size: 20px;
}

h2 a {
    text-decoration: none;
}

h2 a:hover, div.contentheading a:hover {
    color: #036c9e;
}
```

Une pincée de CSS pour donner la couleur des éléments standards utilisés et qui font partie du framework Bootstrap (liens, menus , boutons ..).

CODE du fichier template.css

```
.navbar-inner, .nav-list > .active > a, .nav-list > .active > a:hover,
.dropdown-menu li > a:hover, .dropdown-menu .active > a, .dropdown-menu
.active > a:hover,
.nav-pills > .active > a, .nav-pills > .active > a:hover,
.btn-primary {
    color: #036c9e;
}
```

Rappelez-vous que nous avons aussi défini des classes "rounded" et "white" dans notre code HTML. On les définit maintenant

classe "rounded" → ajoute des coins arrondis d'un rayon de 5px

classe "white" → ajoute un fond dégradé blanc #ffffff vers gris #e1e1e1, ainsi qu'une ombre de couleur #545454 légèrement décalée de 1px.

CODE du fichier template.css

```
.rounded {
          -moz-border-radius: 5px;
    -o-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
}

.white {
    background: #e1e1e1;
    background-image: url("white-gradient.svg");
    background-image: -o-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: -webkit-gradient(linear, left top, left
bottom,from(#ffffff),color-stop(40%,#ffffff), color-stop(100%, #e1e1e1));
    background-image: -moz-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    -pie-background: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    border: #ffffff 1px solid;
          box-shadow: #545454 0px 1px 1px 0px;
    -moz-box-shadow: #545454 0px 1px 1px 0px;
    -webkit-box-shadow: #545454 0px 1px 1px 0px;
          border: #eee 1px solid;
}


img {
    max-width: 100%;
    height: auto;
}
```

background: #e1e1e1; → malgré le dégradé on doit penser à définir une couleur de fond qui sera utilisé par les anciens navigateurs et tous ceux qui ne gèrent pas la propriété "gradient"

background-image: url("white-gradient.svg"); → pour ajouter la compatibilité du dégradé sur Internet Explorer 9 on charge un fichier SVG qui contient les mêmes valeurs de dégradé et qui sera lu sur IE9.

On place ce fichier SVG dans le dossier "css" du template avec le nom "white-gradient.svg".

CODE du fichier white-gradient.svg

```
<?xml version="1.0" ?>
    <svg xmlns="http://www.w3.org/2000/svg" preserveAspectRatio="none"
version="1.0" width="100%"
    height="100%"
    xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
    <defs>
    <linearGradient id="white"
    x1="0%" y1="0%"
    x2="0%" y2="100%"
    spreadMethod="pad">
    <stop offset="0%"   stop-color="#ffffff" stop-opacity="1"/>
    <stop offset="40%"   stop-color="#ffffff" stop-opacity="1"/>
    <stop offset="100%" stop-color="#e1e1e1" stop-opacity="1"/>
    </linearGradient>
    </defs>

    <rect width="100%" height="100%"
    style="fill:url(#white);" />
    </svg>
```

Reportez-vous à la documentation technique concernant l'utilisation de CSS3 pour de plus amples informations.

-pie-background: → propriété spécifique au script CSSPIE que l'on utilisera pour assurer la compatibilité des styles CSS3 avec les anciennes versions d'Internet Explorer. On verra plus tard comment ajouter cette fonctionnalité

On finit par ajouter un "max-width" sur les images, pensons au responsive, il ne faudrait pas que les images sortent de l'écran lorsque le visiteur surfe sur le site avec son mobile ou sa tablette. La propriété "height:auto" permet de conserver le ratio des images lorsqu'elles sont redimensionnées.

9.3) Conteneur principal

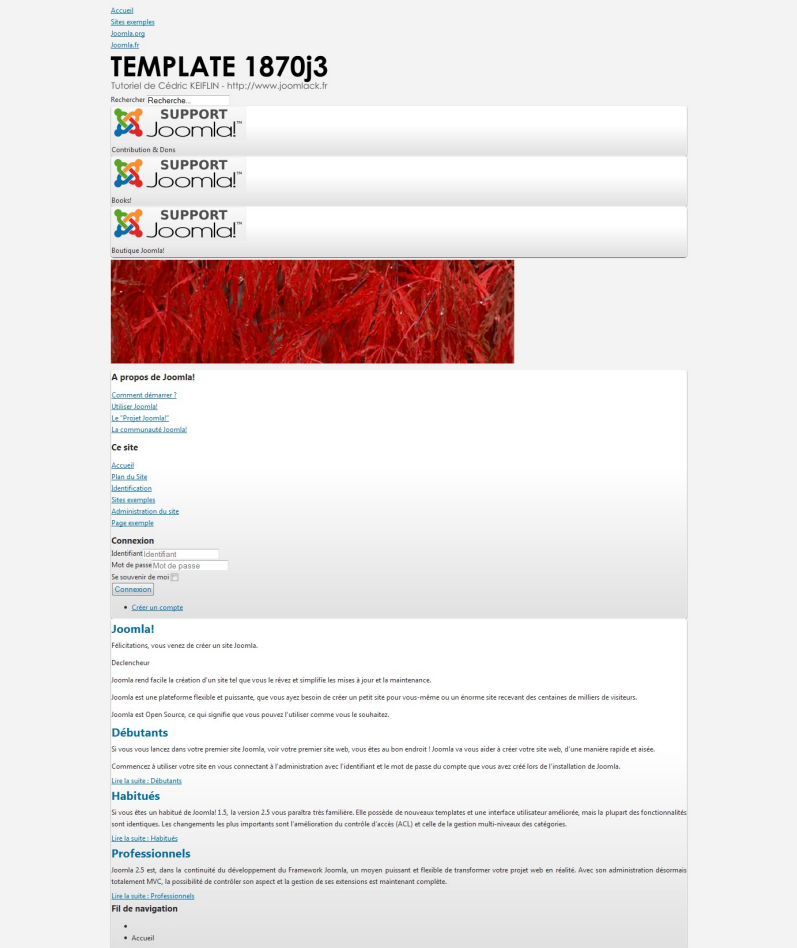
Ici on donne la largeur maximal que doit prendre le site : 1000px. Pourquoi une valeur maximale ? On pourrait juste donner une largeur, mais alors le site serait de largeur fixe et ne s'adapterait pas aux différentes résolutions. D'ailleurs si vous ne voulez pas de site fluide, changez juste la propriété "max-width" en "width".
Afin de centrer les conteneurs on ajoute une marge gauche et droite "auto".

CODE du fichier template.css

```
.wrapper {
    max-width: 1000px;
    margin: 0 auto;
}
```

Avec cette classe, tous les conteneurs (on en a deux si vous regardez à nouveau le code HTML) prendront ces valeurs.

Voici un petit aperçu du résultat qu'on obtient jusqu'ici (avec les données d'exemple préinstallées) :



Le site est centré, fait une largeur de 1000px, et les différents conteneurs ont un dégradé avec ombre et coins arrondis. C'est déjà pas mal !

9.4) Menu horizontal

Vous voyez le résultat ? Ah c'est motivant quand on voit tout de suite à quoi ça doit ressembler !

CODE du fichier template.css

```
#nav {
    margin: 0;
    padding: 0;
    min-height: 50px;
}

#nav ul.menu {
    margin: 0;
    padding: 0;
    zoom: 1;
}

#nav ul.menu:after {
    content : "";
    clear: both;
    display: block;
}

#nav ul.menu > li {
    margin: 0;
    padding: 0;
    list-style:none;
     border-right: 1px solid #fff;
     box-shadow: 1px 0 0 #e1e1e1;
    float: left;
}

#nav ul.menu > li > a, #nav ul.menu > li > span.separator {
     display:block;
    color: #036c9e;
    margin: 0px;
    padding: 15px;
    padding-right: 20px;
```

```
    padding-left: 20px;
    text-align: center;
    font-size: 14px;
     text-decoration: none;
}

#nav ul.menu > li:hover > a, #nav ul.menu > li:hover > span.separator,
#nav ul.menu > li.active > a, #nav ul.menu > li.active > span.separator {
    background: #036c9e;
    background-image: url("nav-gradient.svg");
    background-image: -o-linear-gradient(center top,#036c9e, #024e73 100%);
    background-image: -webkit-gradient(linear, left top, left
bottom,from(#036c9e), color-stop(100%, #024e73));
    background-image: -moz-linear-gradient(center top,#036c9e, #024e73 100%);
    background-image: linear-gradient(center top,#036c9e, #024e73 100%);
    -pie-background: linear-gradient(center top,#036c9e, #024e73 100%);
}

#nav ul.menu > li:hover > a,
#nav ul.menu > li.active > a {
    color: #fff;
}

#nav ul.menu li li a, #nav ul.menu li li span.separator {
    display:block;
    margin: 10px;
    padding: 5px;
    padding-right: 10px;
    padding-left: 10px;
    text-decoration: none;
}

#nav ul.menu li li:hover > a {
    background: #036c9e;
    background-image: url("nav-gradient.svg");
    background-image: -o-linear-gradient(center top,#036c9e, #024e73 100%);
    background-image: -webkit-gradient(linear, left top, left
bottom,from(#036c9e), color-stop(100%, #024e73));
    background-image: -moz-linear-gradient(center top,#036c9e, #024e73 100%);
    background-image: linear-gradient(center top,#036c9e, #024e73 100%);
    -pie-background: linear-gradient(center top,#036c9e, #024e73 100%);
    color: #fff;
}

#nav ul.menu li li.active > a {
    text-align: left;
    color: #000;
}

#nav ul.menu li ul, #nav ul.menu li:hover ul ul, #nav ul.menu li:hover ul ul
ul {
    position: absolute;
    left: -999em;
    z-index: 999;
    margin: 0;
    padding: 0;
    background: #ffffff;
```

```
    background-image: url("white-gradient.svg");
    background-image: -o-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: -webkit-gradient(linear, left top, left
bottom,from(#ffffff),color-stop(40%,#ffffff), color-stop(100%, #e1e1e1));
    background-image: -moz-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    -pie-background: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    border: #ffffff 1px solid;
    box-shadow: #545454 0px 1px 1px 0px;
    -moz-box-shadow: #545454 0px 1px 1px 0px;
    -webkit-box-shadow: #545454 0px 1px 1px 0px;
    border: #ccc 1px solid;
    width: 180px;
}


#nav ul.menu li:hover ul ul, #nav ul.menu li:hover li:hover ul ul, #nav
ul.menu li:hover li:hover li:hover ul ul {
    left: -999em;
}

#nav ul.menu li:hover > ul, #nav ul.menu li:hover ul li:hover > ul, #nav
ul.menu li:hover ul li:hover ul li:hover > ul, #nav ul.menu li:hover ul
li:hover ul li:hover ul li:hover > ul {
    left: auto;
}

#nav ul.menu li:hover ul li:hover ul {
    margin-top: -45px;
    margin-left: 170px;
}

#nav ul.menu li li {
    float: none;
}
```

Si on décortique un peu le code, voyons les points qui peuvent poser problème ou soulever des questions.

```
#nav {
    margin: 0;
    padding: 0;
    min-height: 50px;
}
```

min-height : 50px; → donne une hauteur mini au menu, sans pour autant le figer. Admettons que vous augmentiez les marges ou qu'il y ait une image assez grande sur un élément, le menu s'adaptera en hauteur.

```
#nav ul.menu {
    margin: 0;
    padding: 0;
    zoom: 1;
}
```

zoom : 1; → ajoute le hasLayout sur l'élement pour la compatibilité Internet Explorer. Mais pourquoi faire ? C'est pour la prise en compte du style ":after" que l'on va voir de suite (identique à la classe clearfix) :

```
#nav ul.menu:after {
    content: " ";
    display: block;
    height: 0;
    clear: both;
    visibility: hidden;
    font-size: 0;
}
```

:after → ajoute un élément de type block invisible à la fin du menu auquel on applique un "clear:both ;". Ça permet de stopper la flottaison des éléments <li> du menu et s'affranchir des problèmes que ça peut poser (voir chapitre de la documentation technique sur la flottaison des éléments).

```
#nav ul.menu > li {
    margin: 0;
    padding: 0;
    list-style:none;
     border-right: 1px solid #fff;
     box-shadow: 1px 0 0 #e1e1e1;
    float: left;
}
```

Ici on fait flotter les éléments sur la gauche avec un "float:left" et on annule les marges.

list-style:none ; → supprime les puces de la liste (les puces sont les petites ronds ou carrés qui se mettent automatiquement sur les menus)

box-shadow: 1px 0 0 #e1e1e1; → ajoute une ombre d'1 px, de couleur grise. Cette ombre va se placer à droite car on la déplace d'un seul pixel et elle a une largeur nulle. Largeur nulle ? Ben oui, si on ne décale pas l'ombre on ne la verrait pas. Ici ça permet de créer un deuxième trait de bordure à droite. Voyons ça en images.

Illustration 3: Bordure blanche 1px et ombre grise 1px

```
#nav ul.menu > li > a, #nav ul.menu > li > span.separator {
     display:block
...
```

Sur les liens on va ajouter la couleur, les marges, la taille du texte et tout. Rien de compliqué sauf la propriété "display:block" qui est très importante.

display:block ; → ici on demande à l'ancre qui est une balise de type en ligne, de se comporter comme un bloc. Il va donc prendre toute la largeur de son parent et ainsi occuper tout l'espace du bloc <li> qui le contient. Si on ne fait pas ça on aura des zones de survol qui seront sur le <li> mais pas sur le lien <a> et donc qui ne servent à rien.

On ne va pas tout détailler ligne par ligne, grosso modo, on ajoute des couleurs et tailles aux éléments. Un dégradé sera appliqué aux sous menus, de couleur blanc vers gris comme pour la classe "white". On appelle un fichier SVG pour garantir la compatibilité IE9 et le -pie-background pour la compatibilité avec IE8 et inférieur (reportez-vous au chapitre sur les dégradés CSS3 de la documentation technique).

li:hover → état survolé de l'élément, lorsque la souris passe dessus.

li.active → état actif de l'élément, c'est lorsque l'on clique sur le lien, on arrive sur la nouvelle page et à ce moment là le lien devient actif. C'est ce lien qui pointe vers la page (ou qui contient le lien actif dans ses sous-menus).

#nav ul.menu li li a → cible tous les liens des sous menus. Le fait d'avoir deux fois le terme "li" fait qu'on vise une architecture à au moins deux niveaux.

```
#nav ul.menu li ul, #nav ul.menu li:hover ul ul, #nav ul.menu li:hover ul ul
ul {
    position: absolute;
    left: -999em;
    z-index: 999;
    margin: 0;
    padding: 0;
    background: #ffffff;
    background-image: url("blocnav-gradient.svg");
    background-image: -o-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: -webkit-gradient(linear, left top, left
bottom,from(#ffffff),color-stop(40%,#ffffff), color-stop(100%, #e1e1e1));
    background-image: -moz-linear-gradient(center top,#ffffff,#ffffff 40%,
#e1e1e1 100%);
    background-image: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    -pie-background: linear-gradient(center top,#ffffff,#ffffff 40%, #e1e1e1
100%);
    border: #ffffff 1px solid;
    box-shadow: #545454 0px 1px 1px 0px;
    -moz-box-shadow: #545454 0px 1px 1px 0px;
    -webkit-box-shadow: #545454 0px 1px 1px 0px;
    border: #ccc 1px solid;
    width: 180px;
}
```

Ici on style le conteneur de sous-menus, le plus important est de le positionner. Par défaut on lui donne une "position:absolute" pour le sortir du flux et le positionner hors de l'écran à "left :-999em". Le "z-index:999" permet de s'assurer que le sous menu sera toujours au dessus des autres éléments. Pour finir on lui donne une largeur fixe de 180px.

```
#nav ul.menu li:hover > ul, #nav ul.menu li:hover ul li:hover > ul, #nav
ul.menu li:hover ul li:hover ul li:hover > ul, #nav ul.menu li:hover ul
li:hover ul li:hover ul li:hover > ul {
    left: auto;
}
```

Ici on décide d'afficher le sous-menu (oui celui qu'on a envoyé à -999em tout à l'heure) lorsque la souris survole l'élément parent : li:hover > ul. A ce moment là on lui remet une position "left" à "auto" pour qu'il vienne gentiment se placer sous son parent.

```
#nav ul.menu li:hover ul li:hover ul {
    margin-top: -45px;
    margin-left: 170px;
}
```

Avec cette structure on cible quoi maintenant ? Hé bien chaque niveau est défini par un couple "ul li". Comme là on attaque le troisième "ul", c'est le troisième niveau de sous-menus qu'on vise. On lui donne juste les marges qui vont bien puisqu'on ne veut pas qu'il

se place sous son élément parent (contrairement au niveau2), mais à droite ! On le remonte de 45px et on le bouge à droite de 170px (c'est pas pour rien qu'on a donné une largeur fixe de 180px aux sous-menus tout à l'heure, c'est aussi pour connaître la valeur à donner pour placer correctement ces éléments).

On crée un deuxième fichier SVG pour créer le dégradé au survol du menu à mettre dans le dossier "css" du template.

CODE du fichier nav-gradient.svg

```
<?xml version="1.0" ?>
    <svg xmlns="http://www.w3.org/2000/svg" preserveAspectRatio="none"
version="1.0" width="100%"
    height="100%"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <defs>
    <linearGradient id="nav"
    x1="0%" y1="0%"
    x2="0%" y2="100%"
    spreadMethod="pad">
    <stop offset="0%"   stop-color="#036c9e" stop-opacity="1"/>
    <stop offset="100%" stop-color="#024e73" stop-opacity="1"/>
    </linearGradient>
    </defs>

    <rect width="100%" height="100%"
    style="fill:url(#nav);" />
    </svg>
```

9.5) Bannière logo et module de recherche

Sans faire dans la difficulté on va afficher le logo à gauche et le module de recherche à droite.

CODE du fichier template.css

```
#logo {
    float: left;
    margin: 5px 0;
    display: block;
    max-width: 100%;
}

#headermodule {
    float: right;
    margin: 10px 0;
}
```

Ici on a des éléments flottants ('float:left" et "float:right"), pour les aligner respectivement à gauche et à droite.

Étant donné que la balise HTML du "logo" est un lien, on lui ajoute un "display:block" pour pouvoir le positionner à gauche et lui appliquer les marges.

9.6) Rangée de 4 modules

Normalement on devrait s'amuser un peu plus ici. Pensez à la gestion des scénarios, en fonction des modules publiés les largeurs des blocs varient.

On en a déjà parlé plus haut lorsque l'on a mis en place le code HTML .On avait alors vu que le code qui permet de gérer les situations est basé sur la classe qui varie de "n1" à "n4" (1 à 4 modules à afficher).

Le plus simple pour nous serait de donner une largeur identique à tous les modules, mais dans notre cahier des charges on avait bien spécifié qu'on voulait des modules pouvant prendre chacun une valeur différente.

Si vous voulez faire simple, vous pouvez mettre le code suivant :

```
#row1modules .row1module {
    float: left;
}

#row1modules.n1 .row1module {
    width: 100%;
}


#row1modules.n2 .row1module {
    width: 50%;
}


#row1modules.n3 .row1module {
    width: 33%;
}
```

```
#row1modules.n4 .row1module {
    width: 25%;
}
```

Mais ce n'est pas ce code là que nous voulons ! Ici pas moyen de définir des tailles différentes pour chaque module. Voici le code un peu plus complet à utiliser.

CODE du fichier template.css

```
#row1modules .row1module {
    float: left;
}

/* 1 module */
#row1modules.n1 > .row1module {
    width: 100%;
}

#row1modules.n1 > .row1module > div.inner {
    margin: 5px 0;
}
/* 2 modules, premier module */
#row1modules.n2 > .row1module {
    width: 50%;
}

#row1modules.n2 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 2 modules, deuxieme module */
#row1modules.n2 > .row1module + div {
    width: 50%;
}

#row1modules.n2 > .row1module + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 3 modules, premier module */
#row1modules.n3 > .row1module {
    width: 33.3%;
}

#row1modules.n3 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 3 modules, deuxieme module */
#row1modules.n3 > .row1module + div {
    width: 33.3%;
}

#row1modules.n3 > .row1module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 3 modules, troisieme module */
#row1modules.n3 > .row1module + div + div {
    width: 33.3%;
}
```

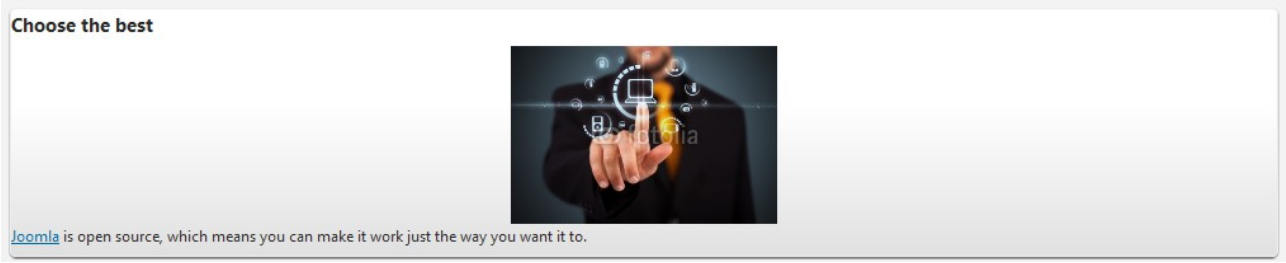
```
#row1modules.n3 > .row1module + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 4 modules, premier module */
#row1modules.n4 > .row1module {
    width: 25%;
}

#row1modules.n4 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 4 modules, deuxieme module */
#row1modules.n4 > .row1module + div {
    width: 25%;
}

#row1modules.n4 > .row1module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, troisieme module */
#row1modules.n4 > .row1module + div + div {
    width: 25%;
}

#row1modules.n4 > .row1module + div + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, quatrieme module */
#row1modules.n4 > .row1module + div + div + div {
    width: 25%;
}

#row1modules.n4 > .row1module + div + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

Les explications s'imposent ! A moins que vous ayez tout compris... mais j'en doute un peu. Alors voyons ça dans le détail.

```
#row1modules .row1module {
    float: left;
}
```

Là on commence par faire flotter les blocs pour les aligner les uns à côté des autres.

a) Scénario 1 : un seul module




```
/* 1 module */
#row1modules.n1 > .row1module {
    width: 100%;
}

#row1modules.n1 > .row1module > div.inner {
    margin: 5px 0;
}
```

On définit une largeur de 100% pour le module qui est tout seul à occuper tout l'espace. On lui ajoute des marges de 5px en haut et en bas, et 0px à gauche et à droite.

b) Scénario 2 : deux modules


```
/* 2 modules, premier module */
#row1modules.n2 > .row1module {
    width: 40%;
}

#row1modules.n2 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 2 modules, deuxieme module */
#row1modules.n2 > .row1module + div {
    width: 60%;
}
```

```
#row1modules.n2 > .row1module + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

On utilise le sélecteur CSS d'enfant ">" pour sélectionner le premier module :

```
#row1modules.n2 > .row1module {
```

On lui donne ainsi la largeur de 40 %. Pour le second module on utilise le sélecteur d'enfant avec le sélecteur d'adjacence. Ainsi on cible le module qui suit directement le premier module, à savoir le deuxième module. On lui donne la largeur de 60 %.

#row1modules.n2 > .row1module → premier module

#row1modules.n2 > .row1module + div → deuxième module

Il ne faut pas oublier de donner les marges qui vont bien, c'est à dire une marge nulle à gauche pour le premier module et une marge nulle à droite pour le second module.

c) Scénario 3 : trois modules


```
/* 3 modules, premier module */
#row1modules.n3 > .row1module {
    width: 40%;
}

#row1modules.n3 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 3 modules, deuxieme module */
#row1modules.n3 > .row1module + div {
    width: 30%;
}

#row1modules.n3 > .row1module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 3 modules, troisieme module */
#row1modules.n3 > .row1module + div + div {
    width: 30%;
```

```
}

#row1modules.n3 > .row1module + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

On utilise exactement la même méthode que pour le scénario à 2 modules. C'est à dire le sélecteur d'enfant et le sélecteur d'adjacence pour cibler le premier module, le 2e et le 3 séparément.

#row1modules.n3 > .row1module → premier module

#row1modules.n3 > .row1module + div → deuxième module

#row1modules.n3 > .row1module + div + div → troisième module

Encore une fois pour les marges on les mets à zéro tout à gauche et tout à droite, et entre les modules on y ajoute une marge de 3px sur chaque module, soit un espace de 6px entre deux modules.

d) Scénario 4 : quatre modules


```
/* 4 modules, premier module */
#row1modules.n4 > .row1module {
    width: 30%;
}

#row1modules.n4 > .row1module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 4 modules, deuxieme module */
#row1modules.n4 > .row1module + div {
    width: 20%;
}

#row1modules.n4 > .row1module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
```

```
/* 4 modules, troisieme module */
#row1modules.n4 > .row1module + div + div {
    width: 27%;
}

#row1modules.n4 > .row1module + div + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, quatrieme module */
#row1modules.n4 > .row1module + div + div + div {
    width: 23%;
}

#row1modules.n4 > .row1module + div + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

On commence à avoir compris le principe, grâce aux sélecteurs CSS. Ici nous donnons les valeurs de 30 %, 20 %, 27 % et 23 % aux modules respectivement de la gauche vers la droite. Notons que la somme des pourcentages est égale à 100 %, ça me paraît logique !

#row1modules.n4 > .row1module → premier module

#row1modules.n4 > .row1module + div → deuxième module

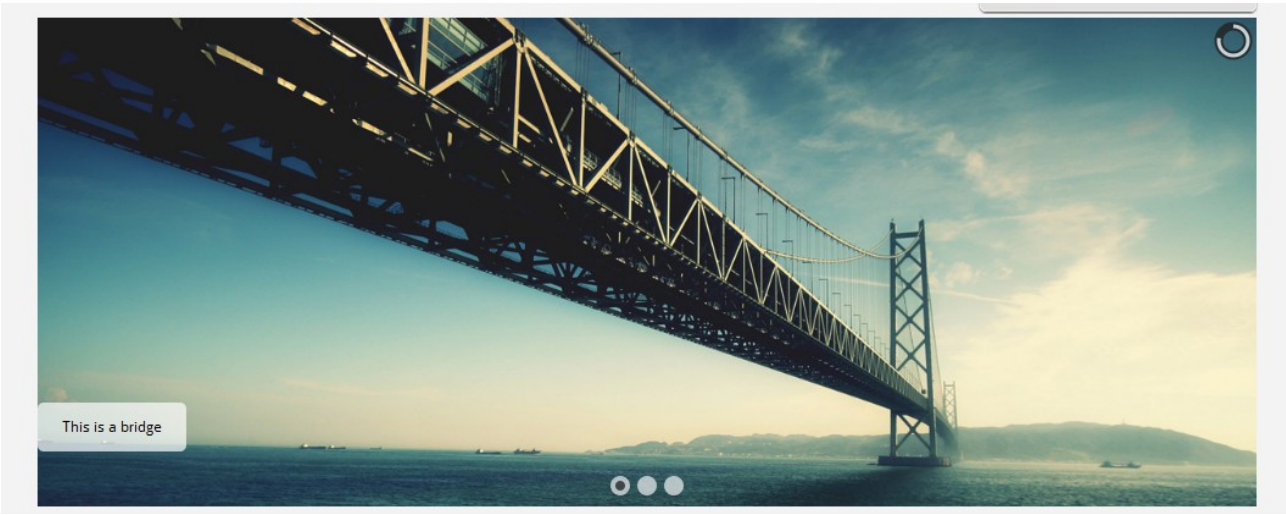
#row1modules.n4 > .row1module + div + div → troisième module

#row1modules.n4 > .row1module + div + div + div → quatrième module

La marge tout à gauche et tout à droite est toujours mise à zéro pour bien caler la rangée sur l'alignement du reste du template.

Nous n'avons pas encore expliqué pourquoi on n'ajoute pas la largeur et les marges aux mêmes éléments, mais si vous avez bien tout suivi jusqu'à maintenant vous n'êtes pas sans ignoré la méthode des divs imbriquées ! C'est nécessaire pour garantir l'affichage de nos éléments sans faire de calcul savant. Imaginez devoir calculer la largeur de chaque bloc en déduisant de la largeur voulue la valeur des marges internes, des marges externes et des bordures... Heu, non merci ! Je crois que les blocs ne seraient jamais alignés correctement ! lol

9.7) Slideshow



Alors là vous risquez d'être déçu ! Il n'y a rien à faire ! La seule opération nécessaire pour afficher le slideshow est de télécharger et installer un module dans votre site. Vous pouvez utiliser mon module Slideshow CK qui est responsive et s'adaptera très bien à toutes les largeurs d'écran.

Ici j'ai utilisé le thème "default_3" pour le slideshow. Vous pouvez trouver les thèmes sur joomlack.fr : http://www.joomlack.fr/slideshowck/themes-slideshow-ck
ainsi que sa documentation pour vous aider dans le paramétrage :
http://www.joomlack.fr/component/dms/view_document/58-documentation-slideshow-ck

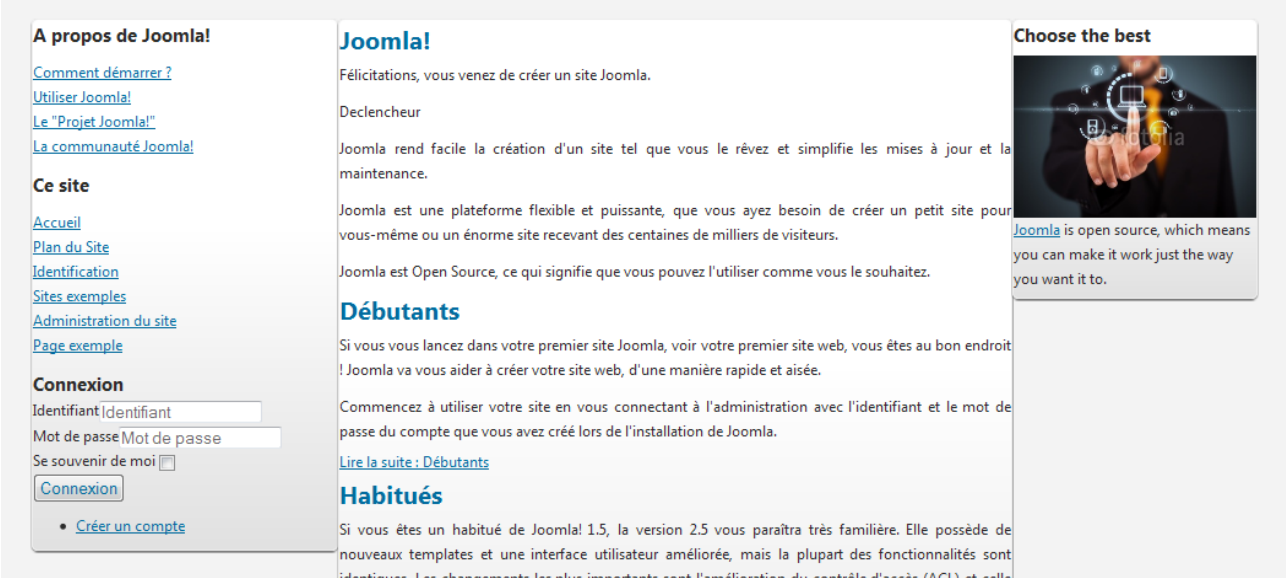
9.8) Contenu principal à 3 colonnes

On a fait une bonne pause avec le slideshow, c'était juste pour mieux attaquer les styles qui vont piloter nos colonnes. D'après notre maquette on a 3 colonnes, gauche, contenu et droite.

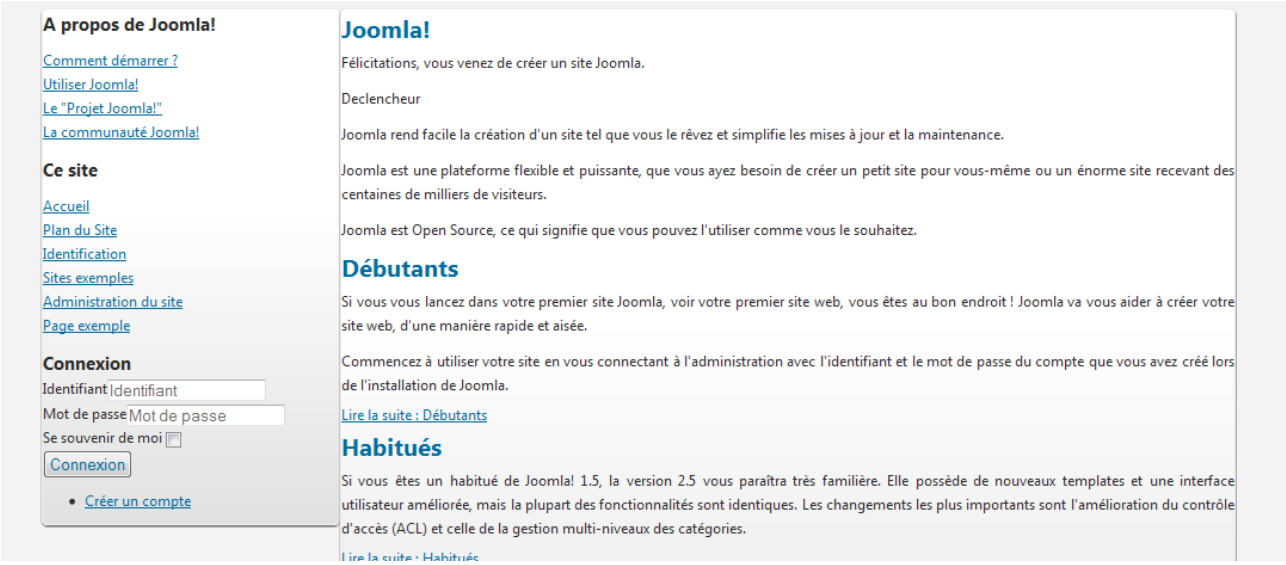
Quels sont nos critères à respecter ?

- donner une largeur spécifique à chaque colonne
- aligner les colonnes horizontalement
- conserver l'aspect fluide du template
- faire en sorte que les largeurs s'adaptent en fonction des scénarios

Aperçu à 3 colonnes



Aperçu à 2 colonnes



Ne vous en faites pas sur les aperçus aucun style n'est appliqué à part la gestion des colonnes. C'est donc normal que ce soit moche et que les colonnes soient collées !

Voyons comment gérer nos colonnes.

CODE du fichier template.css

```
#left, #right, #center {
```

```
    float: left;
}

#left {
    width: 25%;
}

#right {
    width: 20%;
}

#center {
    width: 55%;
}

.noleft #center {
    width: 80%;
}

.noright #center {
    width: 75%;
}

.noright.noleft #center {
    width: 100%;
}
```

D'abord il faut aligner les colonnes avec un "float:left", c'est d'ailleurs pour cela que vous avons ajouté la classe "clearfix" sur le conteneur "main". C'est ce même conteneur auquel on a appliqué la classe pilotée par la variable "$mainclass" qui peut prendre les valeurs suivantes :

- noleft → pas de colonne gauche
- noright → pas de colonne droite
- noleft noright → aucune colonne gauche ni droite, juste le contenu

Ce sont ces classes que l'on retrouve dans notre code CSS. Elles sont simplement utilisées pour reproduire les scénarios.

Classe	Largeur de la colonne de gauche	Largeur du contenu	Largeur de la colonne de droite
(aucune)	25 %	55 %	20 %
noleft	0	80 %	20 %
noright	25 %	75 %	0
noleft noright	0	100 %	0

Vous pouvez tester en publiant des modules dans les positions "position-7" pour la colonne de gauche et "position-6" pour la colonne de droite. Si vous jouez avec l'état de publication des modules vous devez voir apparaître ou disparaître les colonnes et l'ensemble du contenu doit s'adapter.

Maintenant que cette partie est réglée, il va falloir s'attaquer au design et donner un peu d'allure à tout ça.

Commençons par espacer les colonnes pour aérer notre design. Attention à la méthode des divs imbriquées, on vient d'appliquer la largeur aux conteneur "left" et "right", il faut donc appliquer les marges sur les conteneurs intérieurs "div.inner".

CODE du fichier template.css

```
#left > div.inner {
    margin-right: 10px;
    padding: 15px;
}

#right > div.inner {
    margin-left: 10px;
    padding: 15px;
}
```

Voilà, on écarte les colonnes avec les margin-left et margin-right, et grâce au padding on fait en sorte que le texte ne soit pas collé non plus aux bords de la colonne.

a) Personnalisation du menu vertical

Amusons-nous à styler un menu. Ici c'est le module de menu standard de Joomla qu'on utilise et qui charge le code suivant dans la page (exemple de code par défaut):

```
<ul class="nav menu">
    <li class="item-437">
    <a href="/joomla30fr/index.php/comment-demarrer" >Comment démarrer ?</a>
    </li>
    <li class="item-280">
    <a href="/joomla30fr/index.php/utiliser-joomla" >Utiliser Joomla!</a>
    </li>
</ul>
```

CODE du fichier template.css

```
#left ul.menu li a, #left ul.menu li span.separator,
#right ul.menu li a, #right ul.menu li span.separator {
    display: block;
    background:  url(../images/tick.png) left center no-repeat;
    border-bottom: 1px solid #e3e3e3;
    box-shadow: 0 1px 0 #fff;
    padding-top: 5px;
    padding-bottom: 7px;
    padding-left: 20px;
    text-decoration: none;
}
```

Le code est très simple mais mérite quelques explications tout de même.

display:block → transforme le lien en balise de type bloc pour qu'elle prenne toute la largeur de son conteneur et pour qu'on puisse lui appliquer des dimensions et des marges. Sans cette propriété la zone cliquable du lien se limiterait au texte qu'il contient.

Illustration 4: à gauche un lien sans display:block, à droite un lien avec display:block

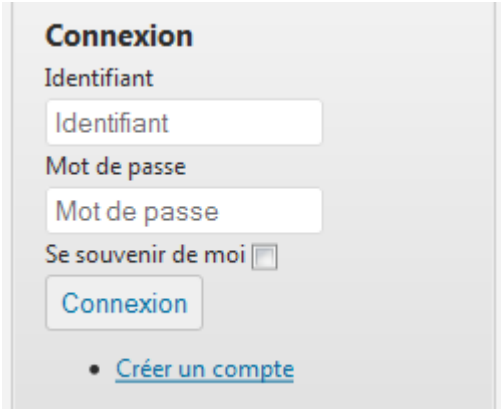
padding-left:20px → décale le texte du lien de 20px vers la droite, laissant ainsi la place libre pour l'image de fond que l'on a ajoutée avec la propriété "background". Les autres padding servent à donner un peu d'espace aux liens.

text-decoration : none → fait en sorte de supprimer le soulignement des liens, ça rend notre menu un peu plus beau !

Le style de menu définit ici sera appliqué dans les deux colonnes, celle de gauche "left" et celle de droite "right".

N'oubliez pas de mettre l'image "tick.png" dans le dossier "images" du template, sinon vous ne verrez qu'un espace vide. Vous pouvez récupérer l'image dans le template fourni avec ce tutoriel sur http://www.joomlack.fr

b) Personnalisation du module de connexion



CODE du fichier template.css

```
#form-login-username label, #form-login-password label {
    display: block;
}

#form-login-username input, #form-login-password input {
    padding: 3px;
    border: 1px solid #ddd;
    -moz-border-radius: 3px;
    -o-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
}

#form-login-username input:focus, #form-login-password input:focus {
    border: 1px solid #036c9e;
}

#form-login-submit button {
    background: #efefef;
    border: 1px solid #c3c3c3;
    padding: 4px;
    -moz-border-radius: 3px;
    -o-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    cursor: pointer;
}

#form-login-submit button:hover {
    background: #ccc;
}
```

#form-login-username → bloc qui contient les champs pour l'identifiant

#form-login-password → bloc qui contient les champs pour le mot de passe

#form-login-submit → bloc qui contient le bouton de connexion

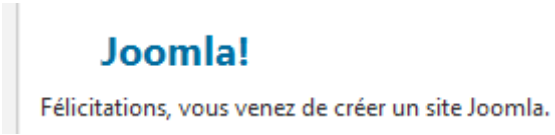
#form-login-remember → bloc qui contient l'option "se souvenir de moi"

Ici on utilise les conteneurs pour donner des marges internes (padding), une bordure et des coins arrondis aux champs de texte, là où on renseigne l'identifiant et le mot de passe.
On utilise la pseudo-classe ":focus" pour styler ces mêmes champs lorsque l'on clique dessus, on y ajoute une bordure bleu foncé pour que l'utilisateur sache qu'il est dans la zone éditable et qu'il peut maintenant écrire ses données.

On finit par styler le bouton de connexion avec un peu de couleur et des coins arrondis, et surtout on lui applique un style dans l'état ":hover" pour que l'utilisateur sache lorsque la souris survole le bouton pour qu'il puisse le cliquer.

cursor:pointer → transforme le curseur de la souris en une main qui indique généralement qu'une action de clic est possible à cet endroit. Ça permet d'aiguiller un peu l'utilisateur lors de sa navigation sur le site

c) Personnalisation du titre des articles



Le code HTML par défaut qui affiche le titre d'un article est :

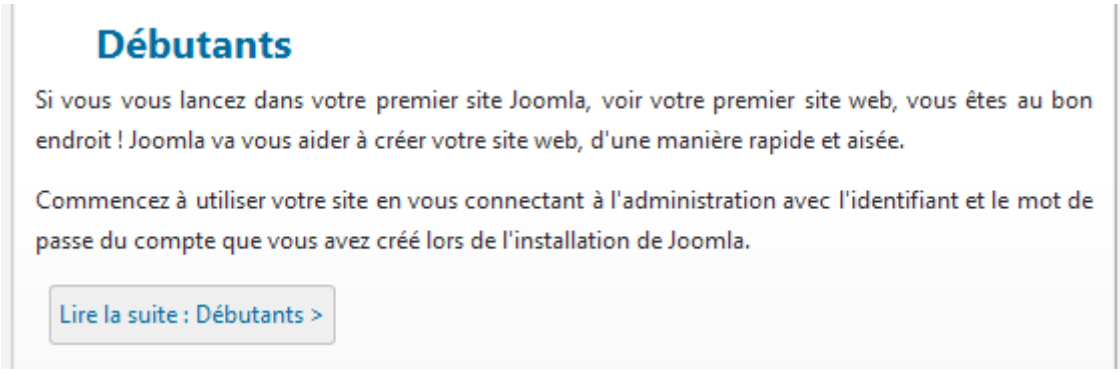
```
<h2 class="item-title">
    <a href="/..."> Joomla!</a>
</h2>
```

On va donner un peu d'espace à notre titre pour bien marquer la séparation avec les paragraphes et le reste du contenu. Un petit peu de marges feront l'affaire. Bien sur si vous voulez donner une couleur, taille, police différente vous pouvez laisser libre court à votre imagination !

CODE du fichier template.css

```
h2.item-title, h2.item-title > a {
    padding-left: 15px;
    margin-top: 15px;
}
```

d) Personnalisation du lien lire la suite



Ici on s'attaque au lien "lire la suite" qui permet de lire l'intégralité d'un article lorsque l'on se trouve sur une page de type blog. Voici à quoi ressemble par défaut le lien :

On est d'accord de dire que c'est relativement moche et ça ne met pas l'article en valeur. Pour arriver à notre résultat on utilise les mêmes styles que pour le bouton du module de connexion que l'on a stylé précédemment. Mais en plus on lui ajoute cette petite flèche à droite ">" qui donne le sens du lien à savoir qu'il y a une suite. J'aime bien cette petite indication, je trouve que c'est plus parlant qu'un texte, on voit de suite de quoi il en retourne.

Pour ajouter la flèche on va utiliser la propriété ":after" qui permet d'ajouter du contenu après un élément HTML.

CODE du fichier template.css

```
div.item a.btn:after {
    content: " >";
}
```

Ensuite on donne les css pour l'allure générale du bouton.

CODE du fichier template.css

```
div.item a.btn {
    text-decoration: none;
    background: #efefef;
    border: 1px solid #c3c3c3;
    padding: 4px;
    -moz-border-radius: 3px;
    -o-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    cursor: pointer;
    margin: 7px;
    display:inline-block;
}
```

display : inline-block → fait en sorte que l'élément se comporte comme un bloc tout en restant aligné comme un élément en ligne. L'avantage c'est qu'on peut alors lui ajouter des marges (7px). Sans cette propriété les marges n'aurait aucun effet !

e) Personnalisation du fil de navigation (fil d'ariane)

Ici la structure HTML classique du fil d'ariane se compose ainsi :

```
<ul class="breadcrumb">
    <li>
        <a class="pathway" href="/joomla30fr/">Accueil</a>
        <span class="divider">/</span>
    </li>
</ul>
```

L'idée est très simple, on fait en sorte d'aligner les liens sur la même ligne et on les espace en ajoutant un peu de marge sur les séparateurs (avec la classe "divider").

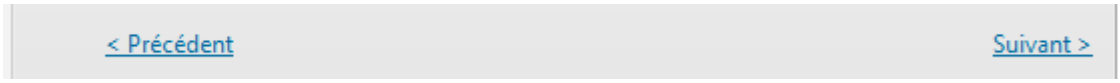
CODE du fichier template.css

```
ul.breadcrumb li {
    display: inline;
    list-style: none;
}

ul.breadcrumb li a {
    text-decoration: none;
}

ul.breadcrumb .divider {
    margin: 3px;
}
```

display: inline → affiche les éléments en ligne, ça permet tout simplement de les aligner

f) Personnalisation de la navigation entre articles

Alors maintenant qu'on a fait un beau bouton "lire la suite", il faut cliquer dessus ! Et là on arrive sur un article. S'il appartient à une catégorie contenant plusieurs articles la navigation s'ajoute en-dessous.



Le lien qui permet d'aller à la page précédente est fait du code suivant :

```
<li class="previous">
    <a rel="prev" href="/joomla30fr/index.php/sample-sites">Précédent ></a>
</li>
```

Le lien qui permet d'aller à la page suivante est fait du code suivant :

```
<li class="next">
    <a rel="next" href="/joomla30fr/index.php/sample-sites">Suivant ></a>
</li>
```

CODE du fichier template.css

```
div.item-page li {
    display: inline;
    list-style: none;
}

li.next{
    float: right;
}
```

On aligne à nouveau les liens avec un "display:inline", mais on renvoit aussi le lien "suivant" à droite avec un "float:right". Ici je n'ai pas donner de forme de bouton comme pour les autres car je trouve qu'il ne faut pas abuser des bonnes choses et que les petits liens vont bien. Maintenant si vous le désirez vous pouvez aisément copier et insérer des styles sur ces liens pour les rendre un peu plus séduisants !

g) Personnalisation des informations de l'article

Tout en haut de la page on a des informations sur l'article (auteur, date, etc) ainsi que deux boutons d'action pour imprimer et envoyer par email.

Les boutons "imprimer" et "email" sont définis par le code suivant :

```
<ul class="dropdown-menu actions">
    <li class="print-icon">
          <a>Imprimer</a>
    </li>
    <li class="email-icon">
          <a>Email</a>
    </li>
</ul>
```

On va devoir les aligner à droite et leur donner une couleur de texte et une couleur de fond. N'oubliez pas que les couleurs de textes doivent être appliquées sur les balises "<a>" car on a déjà défini une couleur pour les liens dans les CSS et si vous appliquez la couleur sur l'élément parent "<li>" cela n'aura aucun effet, puisque la couleur des liens sera prépondérante. Grâce au "border-radius" on donne aussi un peu d'arrondi aux angles.

CODE du fichier template.css

```
ul.actions li {
    float: right;
    padding: 2px 5px;
    background: #e3e3e3;
    -moz-border-radius: 3px;
    -o-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    margin: 3px;
}

ul.actions li a {
    color: #fff;
    text-decoration: none;
}
```

float:right → aligne les éléments à droite, peu importe si on les sort du flux puisqu'on n'utilise pas le conteneur "ul.actions" pour lui définir des styles

Voyons maintenant de quoi est fait l'en-tête définie par le titre de page ainsi que le texte "Ecrit par …".

```
<div class="page-header">
    <h2>
    <a href="..."> Professionnels</a>
    </h2>
    Écrit par Joomla
</div>
```

Tout ça est englobé dans une div ayant la classe "page-header", c'est donc cette classe qui va nous servir à mettre une petite bordure. La bordure marque la séparation entre le haut de l'article et son contenu.

CODE du fichier template.css

```
.page-header {
    border-bottom: 1px solid #e3e3e3;
}
```

Passons ensuite aux informations sur l'article à savoir le nom de l'auteur, la date, etc...
Voici le code HTML de la page :

```
<div class="article-info muted">
    <dl class="article-info">
          <dt class="article-info-term">Détails</dt>
          <dd>
                <div class="category-name">
                      Catégorie :
                      <a href="/...">Joomla!</a>
                </div>
          </dd>
          <dd>
                <div class="published">
                      Publication : 9 janvier 2012
                </div>
          </dd>
          <dd>
                <div class="hits">
                      Affichages : 39
                </div>
          </dd>
    </dl>
</div>
```

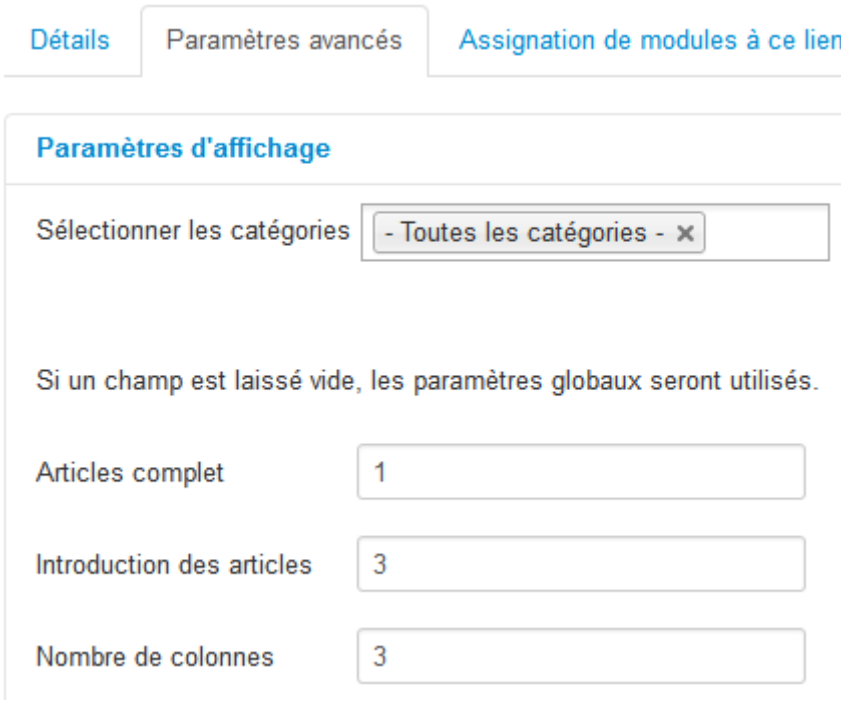
La seule classe qui nous intéresse ici c'est "article-info". On donne une couleur pâle aux textes pour bien montrer que ce ne sont que des informations et que ça ne fait pas partie du corps de l'article, et on diminue un peu la taille de police.

CODE du fichier template.css

```
.article-info {
    color: #c3c3c3;
    font-size: 10px;
}
```

h) Articles en colonnes en mode blog

Un point très bête, le mode Blog (comme par exemple la page d'accueil). Admettons que dans la configuration de votre site vous avez décidé d'afficher des articles en colonnes sur la page d'accueil, vous allez le paramétrer dans le lien de menu :



Hé bien vous pouvez paramétrer comme vous voulez, si votre template ne tient pas compte de cet affichage ça ne marchera pas !

On ajouter quelques css dans notre fichier afin de gérer jusqu'à 3 colonnes d'articles.

CODE du fichier template.css

```
.cols-1
{
        display: block;
        float: none !important;
        margin: 0 !important;
}

.cols-2 .column-1
{
        width:46%;
        float:left;
}

.cols-2 .column-2
{
```

```
        width:46%;
        float:right;
        margin:0
}

.cols-3 .column-1
{
        float:left;
        width:29%;
        padding:0px 5px;
        margin-right:4%

}
.cols-3 .column-2
{
        float:left;
        width:29%;
        margin-left:0;
        padding:0px 5px
}
.cols-3 .column-3
{
        float:right;
        width:29%;
        padding:0px 5px
}


.items-row
{
        overflow:hidden;
        margin-bottom:10px !important;
}

.column-1,
.column-2,
.column-3
{
        padding:10px 5px
}

.column-2
{
        width:55%;
        margin-left:40%;
}
```

```
.column-3
{
        width:30%
}
```

Voilà je crois qu'on en a fini avec la personnalisation du contenu principal. Passons à la prochaine étape, à savoir la rangée de modules du bas.

9.9) Rangée de modules sous l'article

Ici pas de grande parole ni d'explications exceptionnelles, on va juste copier-coller le code que l'on a utilisé pour la première rangée de modules. On penses bien sur à remplacer tous les termes "row1" par "row2" et le tour est joué !

N'oubliez pas que vous pouvez personnaliser les largeurs de chaque module si vous le voulez.

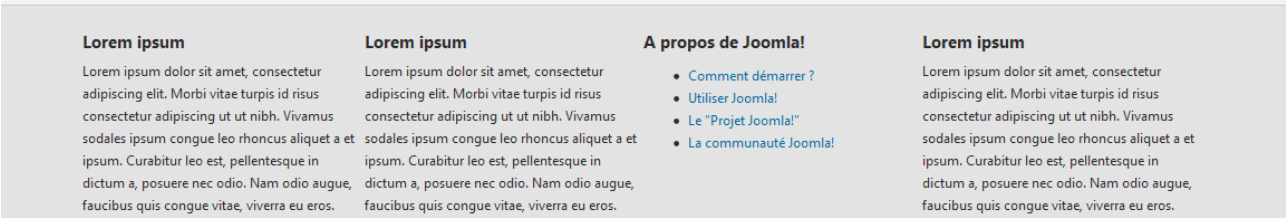
CODE du fichier template.css

```
#row2modules .row2module {
    float: left;
}

/* 1 module */
#row2modules.n1 > .row2module {
    width: 100%;
}

#row2modules.n1 > .row2module > div.inner {
    margin: 5px 0;
}
/* 2 modules, premier module */
#row2modules.n2 > .row2module {
    width: 50%;
}
```

```
#row2modules.n2 > .row2module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 2 modules, deuxieme module */
#row2modules.n2 > .row2module + div {
    width: 50%;
}

#row2modules.n2 > .row2module + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 3 modules, premier module */
#row2modules.n3 > .row2module {
    width: 33.33%;
}

#row2modules.n3 > .row2module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 3 modules, deuxieme module */
#row2modules.n3 > .row2module + div {
    width: 33.33%;
}

#row2modules.n3 > .row2module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 3 modules, troisieme module */
#row2modules.n3 > .row2module + div + div {
    width: 33.33%;
}

#row2modules.n3 > .row2module + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 4 modules, premier module */
#row2modules.n4 > .row2module {
    width: 25%;
}

#row2modules.n4 > .row2module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 4 modules, deuxieme module */
#row2modules.n4 > .row2module + div {
    width: 25%;
}

#row2modules.n4 > .row2module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, troisieme module */
#row2modules.n4 > .row2module + div + div {
    width: 25%;
}
```

```
#row2modules.n4 > .row2module + div + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, quatrieme module */
#row2modules.n4 > .row2module + div + div + div {
    width: 25%;
}

#row2modules.n4 > .row2module + div + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

9.10) Modules en bas de page



Ici aussi on va copier-coller le code pour aligner les modules, mais avant cela on va utiliser notre bloc "body2" pour créer une séparation avec le reste de la page.
Lorsque l'on a créé notre fichier "index.php" on a inséré ce bloc qui est directement enfant du "body" (corps de la page). C'est donc un élément de type bloc qui prendra toute la largeur de son parent, à savoir la page entière. Il suffit de lui donner un peu de couleur et des marges hautes pour décoller un peu le contenu. On lui ajoute aussi une petite bordure pour bien créer la séparation avec le reste de la page.

CODE du fichier template.css

```
#body2 {
    background: #e3e3e3;
    border-top: 1px solid #ccc;
    margin-top: 5px;
    padding-top: 15px;
}
```

Voilà, c'est maintenant qu'on recopie à nouveau les styles pour la rangée de modules, mais avec le terme "row3" cette fois-ci.

CODE du fichier template.css

```
#row3modules .row3module {
    float: left;
}

/* 1 module */
#row3modules.n1 > .row3module {
```

```
    width: 100%;
}

#row3modules.n1 > .row3module > div.inner {
    margin: 5px 0;
}
/* 2 modules, premier module */
#row3modules.n2 > .row3module {
    width: 50%;
}

#row3modules.n2 > .row3module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 2 modules, deuxieme module */
#row3modules.n2 > .row3module + div {
    width: 50%;
}

#row3modules.n2 > .row3module + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 3 modules, premier module */
#row3modules.n3 > .row3module {
    width: 33.33%;
}

#row3modules.n3 > .row3module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 3 modules, deuxieme module */
#row3modules.n3 > .row3module + div {
    width: 33.33%;
}

#row3modules.n3 > .row3module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 3 modules, troisieme module */
#row3modules.n3 > .row3module + div + div {
    width: 33.33%;
}

#row3modules.n3 > .row3module + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
/* 4 modules, premier module */
#row3modules.n4 > .row3module {
    width: 25%;
}

#row3modules.n4 > .row3module > div.inner {
    margin: 5px 3px 5px 0;
}
/* 4 modules, deuxieme module */
#row3modules.n4 > .row3module + div {
    width: 25%;
```

```
}

#row3modules.n4 > .row3module + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, troisieme module */
#row3modules.n4 > .row3module + div + div {
    width: 25%;
}

#row3modules.n4 > .row3module + div + div > div.inner {
    margin: 5px 3px 5px 3px;
}
/* 4 modules, quatrieme module */
#row3modules.n4 > .row3module + div + div + div {
    width: 25%;
}

#row3modules.n4 > .row3module + div + div + div > div.inner {
    margin: 5px 0px 5px 3px;
}
```

On va faire un petit exercice de styles, en ajoutant une pincée de CSS pour redonner une allure normale au menu (si toutefois on insère un module de menu dans une des positions de la rangée).

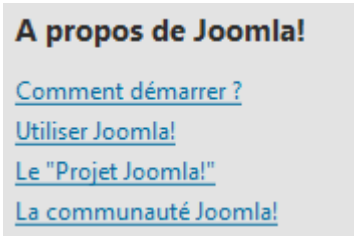


Illustration 5: Affichage du module de menu avant

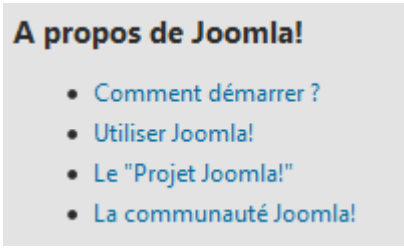


Illustration 6: Affichage du module de menu après

Là il faut rétablir les puces pour la liste, redonner le décalage gauche au menu et on finit par enlever le soulignement qui ne sert à rien puisqu'on sait par défaut qu'on peut cliquer sur des liens de menu.

CODE du fichier template.css

```
#row3modules ul {
    margin-left: 40px;
}

#row3modules ul li {
    list-style: disc;
}

#row3modules ul li a {
    text-decoration: none;
}
```

Voilà, je crois qu'il ne nous reste plus qu'à attaquer le pied de page.

9.11) Pied de page

On va finir en douceur, je crois que les neurones ont bien été utilisé jusqu'à présent. Ici on crée juste une petite bordure et on éclaircit le texte.

CODE du fichier template.css

```
#footer {
    border-top: 1px solid #aaa;
    padding: 5px;
    color: #888;
}
```

Voilà ! On a enfin fini la partie CSS ! C'est le plus gros morceau d'un template, après tout c'est le design du template qui fait tout son attrait.

10. Création des fichiers de langue

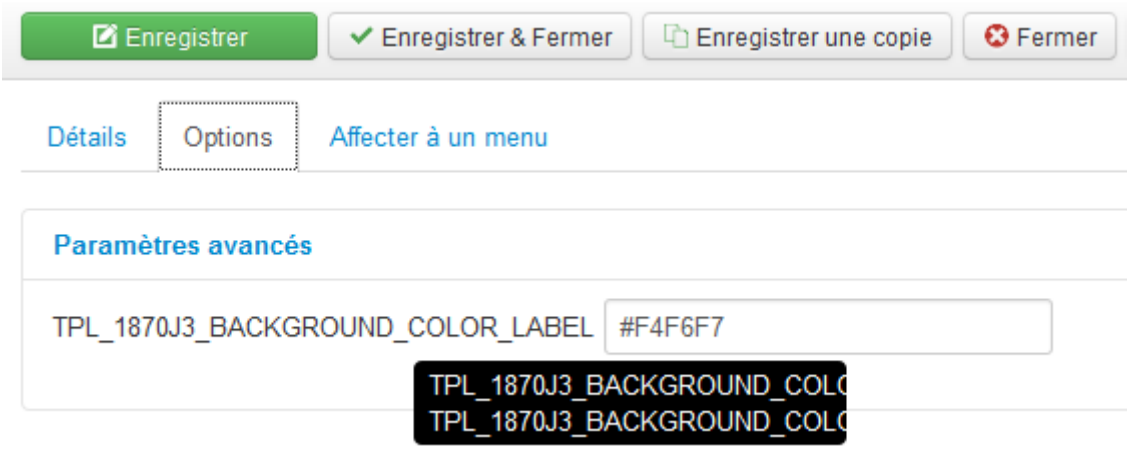
Voilà, on arrive vers la fin...
C'est le moment où on va se donner la peine de traduire les expressions que l'on a utilisées dans notre template. On commence par créer les deux fichiers suivants, encodés en UTF8 No-BOM :
templates/1870j3/languages/fr-FR/fr-FR.tpl_1870j3.ini
templates/1870j3/languages/fr-FR/fr-FR.tpl_1870j3.sys.ini

Dans un premier temps on va travailler dans le fichier
templates/1870j3/languages/fr-FR/fr-FR.tpl_1870j3.ini

La question est de savoir ce qu'on doit y mettre. Regardons notre fichier "templateDetails.xml", on y a déjà plusieurs termes :

```
<description>TPL_1870J3_XML_DESCRIPTION</description>

label="TPL_1870J3_BACKGROUND_COLOR_LABEL"

description="TPL_1870J3_BACKGROUND_COLOR_DESC"
```

Pour l'instant lorsque l'on édite le template (Gestion des templates >> 1870j3 >> Modifier) on voit que les champs sont affichés en brut :



Attention à bien utiliser l'encodage UTF8 No-BOM pour vos fichiers de langue, sinon gare aux problèmes et aux caractères bizarres !

Voici les quelques lignes à ajouter pour traduire nos éléments :

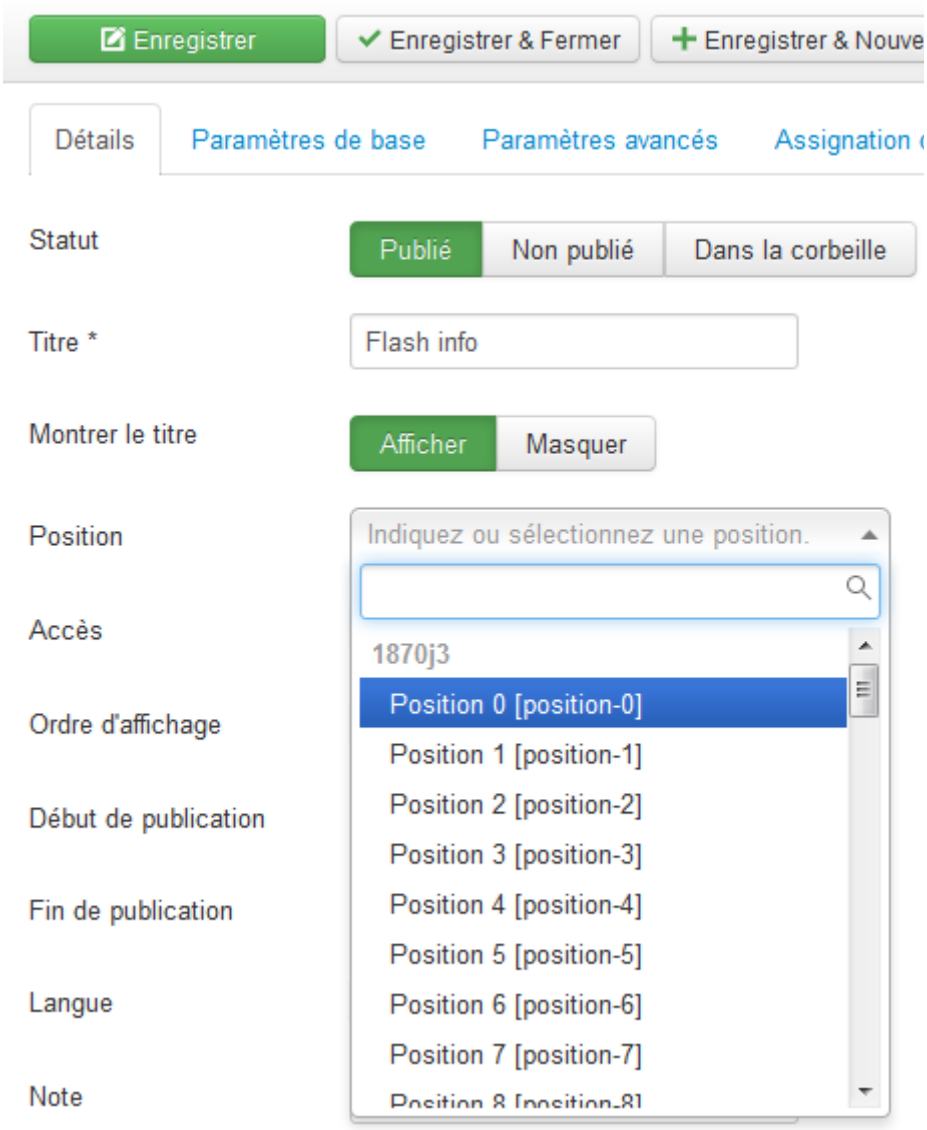
CODE du fichier fr-FR.tpl_1870j3.ini

```
; Template 1870j3
; du tutoriel de Cédric KEIFLIN disponible sur http://www;joomlack.fr

TPL_1870J3_XML_DESCRIPTION="Template 1870j3 du tutoriel de Cédric KEIFLIN
disponible sur http://www;joomlack.fr"
TPL_1870J3_BACKGROUND_COLOR_LABEL="Couleur de fond"
TPL_1870J3_BACKGROUND_COLOR_DESC="Choisir une couleur à appliquer au fond de
la page"
```

Et voilà qui est fait !

Maintenant on change de fichier et on édite le petit frère qui se trouve dans le même dossier de votre template.
templates/1870j3/languages/fr-FR/fr-FR.tpl_1870j3.sys.ini

Ce fichier stocke les traductions pour les termes utilisés lors de la phase d'installation du template, mais aussi pour la liste des positions de module que l'on peut voir dans la gestion des modules, en éditant un module :



Pour l'instant aucune information complémentaire n'est disponible pour chacune des positions qu'on a créé. On peut changer ça ! On va ajouter des petites descriptions pour permettre au webmaster de savoir où se trouve la position dans le template. Pensez-y, surtout si vous voulez redistribuer vos templates ou que vous n'êtes pas seul à travailler sur votre site. Et puis même pour vous, imaginez dans un mois ou deux, vous voulez ajouter un module et là … mince ! À quoi correspond déjà cette position ? lol

Pour réaliser cette opération on doit utiliser la chaîne de caractères sous la forme suivante :

```
TPL_[nom du template]_POSITION_[nom de la position]="Description ici"
```

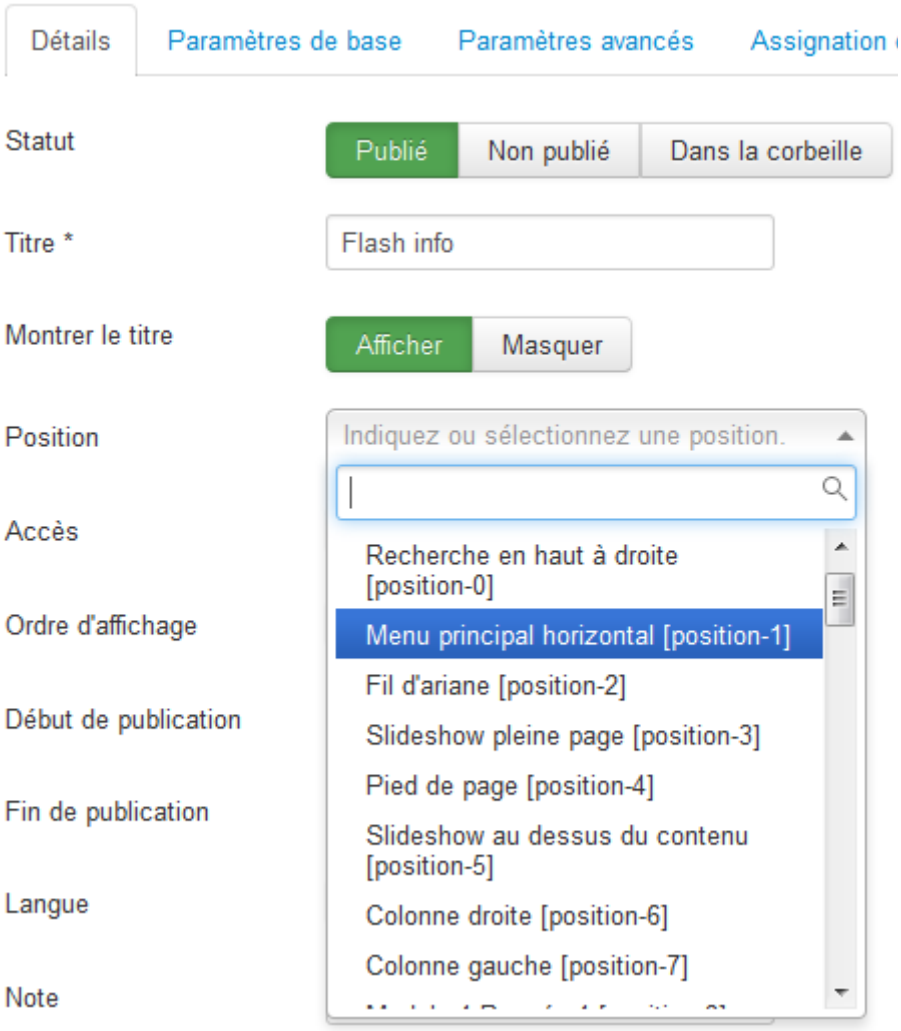
Exemple concret :

```
TPL_1870J3_POSITION_POSITION-0="Recherche en haut à droite"
```

On va donc remplir notre fichier.

CODE du fichier fr-FR.tpl_1870j3.sys.ini

```
; Template 1870j3
; du tutoriel de Cédric KEIFLIN disponible sur http://www;joomlack.fr

TPL_1870J3_XML_DESCRIPTION="Template 1870j3 du tutoriel de Cédric KEIFLIN
disponible sur http://www;joomlack.fr"
TPL_1870J3_POSITION_DEBUG="Debogage"
TPL_1870J3_POSITION_POSITION-0="Recherche en haut à droite"
TPL_1870J3_POSITION_POSITION-1="Menu principal horizontal"
TPL_1870J3_POSITION_POSITION-2="Fil d'ariane"
TPL_1870J3_POSITION_POSITION-3="Slideshow pleine page"
TPL_1870J3_POSITION_POSITION-4="Pied de page"
TPL_1870J3_POSITION_POSITION-5="Slideshow au dessus du contenu"
TPL_1870J3_POSITION_POSITION-6="Colonne droite"
TPL_1870J3_POSITION_POSITION-7="Colonne gauche"
TPL_1870J3_POSITION_POSITION-8="Module 1 Rangée 1"
TPL_1870J3_POSITION_POSITION-9="Module 2 Rangée 1"
TPL_1870J3_POSITION_POSITION-10="Module 3 Rangée 1"
TPL_1870J3_POSITION_POSITION-11="Module 4 Rangée 1"
TPL_1870J3_POSITION_POSITION-12="Module 1 Rangée 2"
TPL_1870J3_POSITION_POSITION-13="Module 2 Rangée 2"
TPL_1870J3_POSITION_POSITION-14="Module 3 Rangée 2"
TPL_1870J3_POSITION_POSITION-15="Module 4 Rangée 2"
TPL_1870J3_POSITION_POSITION-16="Module 1 Rangée 3"
TPL_1870J3_POSITION_POSITION-17="Module 2 Rangée 3"
TPL_1870J3_POSITION_POSITION-18="Module 3 Rangée 3"
TPL_1870J3_POSITION_POSITION-19="Module 4 Rangée 3"
```



C'est fini ! Enfin pour la partie traduction...

11. titre de module bicouleur

On passe à la suite avec le point du cahier des charges qui nous impose d'avoir un titre de module à deux couleurs. Pour réaliser cette opération on va utiliser la méthode d'override Module Chrome (reportez-vous au chapitre correspondant dans la documentation technique).

On commence par créer un dossier "html" dans le dossier du template dans lequel on ajoute deux fichiers

- index.html → fichier servant à la sécurisation du dossier, on y place une seule ligne de code :

```
<html><body bgcolor="#FFFFFF"></body></html>
```

- modules.php → fichier dans lequel on crée la fonction d'override de module

Voici le code qu'on insère dans le fichier "modules.php".

CODE du fichier modules.php

```
<?php
/**
 * tutoriel de Cédric Keiflin pour créer son template Joomla! 3.x
 * http://www.joomlack.fr
 */

defined('_JEXEC') or die;

function modChrome_perso($module, &$params, &$attribs)
{
    if (empty ($module->content)) return;
    $titres = explode(' ', $module->title);
    $titre = str_replace($titres[0], '<span class="titreperso">'.
$titres[0] .'</span>', $module->title);
    ?>
          <div class="moduletable<?php echo htmlspecialchars($params-
>get('moduleclass_sfx')); ?>">
          <?php if ((bool) $module->showtitle) : ?>
                <h3><?php echo $titre; ?></h3>
          <?php endif; ?>
                <?php echo $module->content; ?>
          </div>
          <?php
}
```

$titres = explode(' ', $module->title); → transforme la chaîne de

caractères en un tableau (un tableau est une variable PHP permettant de stocker des informations) contenant tous les mots séparés par un espace. Par exemple pour le titre "Ce site", le résultat sera un tableau contenant "Ce" et "site". Donc $tableau[0] renvoit "Ce" et $tableau[1] renvoit "site".

$titre = str_replace($titres[0], ''. $titres[0] .'', $module->title); → str_replace est une fonction PHP qui permet de remplacer des mots (ou chaines de caractères) par des autres. Ici dans notre exemple on remplace "Ce" par "Ce" dans le titre du module. On ajoute donc une balise "" avec une classe CSS pour l'identifier avec un style particulier.

Il ne nous reste plus qu'à trifouiller un peu les CSS et le tour est joué !

CODE du fichier template.css

```
.titreperso {
    color: #036c9e;
}
```

Et on regarde le résultat … Quoi ?! Rien ! Comment ça se fait ? Hé bien il ne suffit pas de déclarer le style, il faut aussi l'appliquer dans le template sinon on ne sait pas à quelle position de modules il faut l'utiliser. Sachant que pour chaque position de modules on peut utiliser un autre style.

Allons dans le fichier "index.php" du template, on va appliquer notre style aux rangées de modules du haut et du bas (positions "position-8" à "position-15"). Comme nous avons créé une fonction "modChrome_perso", c'est le style "perso" qu'on va appliquer dans les appels JDOC. A la place du style "xhtml" on met le style "perso" comme indiqué ci-dessous sur toutes les positions (8 à 15) :

MODIFICATION du fichier index.php

```
<jdoc:include type="modules" name="position-8" style="perso" />
```

Et voilà un aperçu du résultat :

12. Responsive design – adaptation pour mobiles

Alors là ça va être super facile ! Lol Commencez déjà par réduire la fenêtre de votre navigateur. Là vous vous apercevez que le contenu s'adapte en largeur, et c'est normal puisque tout au long de notre périple pour appliquer les CSS nous n'avons utilisé que des valeurs en pourcent (%) pour les largeurs des conteneurs. Et que le conteneur principal "wrapper" n'a qu'une largeur maximale "max-width", ce qui ne l'empêche pas de se réduire.

La question qui se pose à nous, c'est que faut-il faire pour que ce soit lisible par tous les écrans ? On s'aperçoit qu'en dessous d'une certaine résolution les images sont très petites et les lignes de texte vite tronquées. Il va donc falloir élargir les blocs en dessous d'une largeur définie.

On se fixe une largeur de 758px. Pour utiliser des CSS spécifiques à une certaine résolution on utilise la technique des mediaqueries. C'est quoi les mediaqueries ? Tout simplement une condition CSS, enfin on va se limiter à cette définition.

Exemple de mediaqueries pour détecter un affichage sur écran dont la résolution n'excède pas 758px :

```
@media screen and (max-width: 758px) { … }
```

Faisons le point sur ce que l'on doit modifier pour les petits écrans :

- le menu reste comme il est, les uns se calent les uns sous les autres automatiquement
- le logo et le module de recherche se calent aussi automatiquement, donc rien à faire
- les 3 rangées de modules doivent être modifiées pour que chaque module s'affiche sur la largeur de la page
- les colonnes de contenu (gauche, centre et droite) doivent s'afficher également sur toute la largeur de la page
- les articles en mode blog doivent s'aligner les uns sous les autres
- le slideshow est responsive, donc rien à faire de ce côté là

voilà, donc en gros il n'y a que les rangées de modules, les colonnes centrales, et les articles à modifier. On ajoute le code dans le fichier "template.css".

CODE du fichier template.css

```
@media screen and (max-width: 758px) {

.row1module, .row2module, .row3module, #left, #center, #right {
    float: none !important;
```

```
    width: 100% !important;
}

.row1module > div.inner, .row2module > div.inner, .row3module > div.inner,
#left > div.inner, #center > div.inner, #right > div.inner {
    margin: 5px 0px 5px 0px !important;
}


.items-row .item, .column, .flexiblemodule, .logobloc {
    width: auto !important;
    float: none;
    margin: 0 !important;
}

}
```

Voilà, je crois qu'avec ça on a remplit notre cahier des charges. Il ne reste plus qu'à finaliser le template.

13. Création du package final

Pour réaliser cette étape, il faut commencer par nettoyer tous les fichiers indésirables (images test, scripts indésirables, etc...) que vous avez pu utiliser pendant le développement. Ensuite on va éditer notre fichier XML pour vérifier que l'on a listé tous les fichiers et dossiers dont on a besoin dans le template. Voici ce que l'on doit avoir.

CODE du fichier TEMPLATEDETAILS.XML

```
<files>
		<filename>favicon.ico</filename>
		<filename>index.html</filename>
		<filename>index.php</filename>
		<filename>template_thumbnail.png</filename>
		<filename>templateDetails.xml</filename>
		<filename>template_preview.png</filename>
		<folder>images</folder>
		<folder>css</folder>
		<folder>html</folder>
</files>
```

Ce qu'il faut c'est refléter la structure du template, ici nous avons donc ajouté le dossier "html" à la liste.

Il faut encore que l'on crée nos images de prévisualisation qui seront utilisées dans l'administration de Joomla !. On crée l'image miniature "template_thumbnail.png" et l'aperçu de plus grande taille "template_preview.png" pour l'aperçu du template dans l'administration du site. Elle font typiquement 200x150px et 600x400px respectivement.

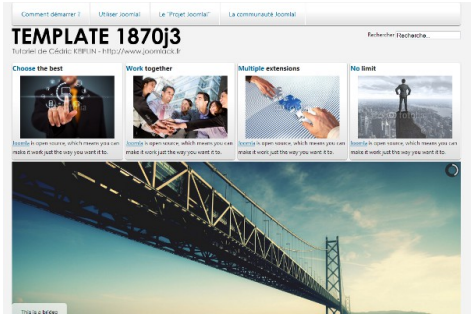


Illustration 7: Image "template_thumbnail.png"

Ensuite on s'occupe de l'icône "favicon.ico", celle qui s'affichera dans la barre d'adresse du navigateur. Pour convertir un PNG en icône on peut utiliser le site http://converticon.com/ qui fonctionne très bien.

Voilà, on met ces images dans le dossier racine du template.
Maintenant, prenez le répertoire du template "1870j3" et zippez-le avec votre outil de compression favori. Et voilà ! C'est fini ! On a notre package installable. Le template est terminé.

Pour le tester il vous suffit de prendre le fichier ZIP créé et d'essayer de l'installer dans votre site Joomla ! 3.

Notons qu'ici le design même s'il n'est pas très compliqué contient tout de même des dégradées, des ombres et des coins arrondis. Et pourtant nous n'avons que 2 images de chargées, le logo et les puces de menu. On charge donc 10Ko d'images et 16Ko de CSS, soit 26Ko pour le template ! Je vous mets au défi de trouver plus léger...

J'espère que avez réussi à suivre toutes les étapes du tutoriel et que vous avez appris ce qu'il vous fallait. Si vous êtes toujours motivé et que vous voulez apprendre à créer un template en utilisant Bootstrap, je vous invite à lire la suite avec le deuxième tutoriel "Créer son template Joomla! 3.x avec Bootstrap" disponible à la suite de ce livre.

En effet aurez remarqué que dans ce tutoriel nous ne tenons pas compte des améliorations faites dans le code de Joomla ! pour profiter du framework Bootstrap. En effet de nombreux effets, et certaines informations ne sont disponibles que si vous utilisez ce framework. Par exemple les tooltips sur le fil d'ariane, la pagination d'articles et bien d'autres.
Ce second tutoriel est disponible dans les prochaines pages du livre dont est extrait ce tutoriel.

Ce tutoriel est un extrait gratuit du livre de Création de template Joomla! 3.x que vous pouvez retrouver sur http://www.joomlack.fr

14. BONUS – Créer le template 1870j3 avec Template Creator CK

Nous avons vu comment créer le template 1870j3 de A à Z en partant de zéro. Maintenant la bonne nouvelle c'est que vous pouvez créer le même template en 40 minutes grâce à mon composant Template Creator CK.
Template Creator CK est un composant qui s'installe dans Joomla! 2.5 ou 3.x.

Le gabarit ainsi que le tutoriel vidéo sont réalisés avec la version 2 de Template Creator CK

Voir le tutoriel vidéo pour créer le template 1870j3 avec Template Creator CK :
https://www.youtube.com/watch?v=ScPvgnGmkDU

Le template créé avec Template Creator CK ainsi que le gabarit (format .tck) que vous pouvez importez dans le composant pour modifier et regénérer le template sont inclus dans ce tutoriel.

Retrouvez le composant Template Creator CK sur http://www.template-creator.com