UTC

Moodle 2, votre premier thème graphique

Un tutorial pour créer un thème graphique pour moodle v2

Dominique Chambelant
03/03/2011

Moodle v2.0.x

Développement: Thèmes 2,0 créer votre premier thème

Ce document explique comment créer un thème pour Moodle 2.0. Il suppose que vous avez une certaine compréhension de la façon dont les thèmes travaillent au sein de Moodle ainsi qu'une bonne compréhension du langage HTML et CSS.

Mode concepteur de thème

En fonctionnement normal Moodle fait plusieurs choses au nom de la performance, l'une d'entre eux est de combiner l'ensemble des CSS dans un fichier, de le minimiser, le mettre en cache sur le serveur, puis l'envoyer au navigateur. Après la première demande la version en cache est envoyé pour améliorer considérablement les performances.

Qu'est-ce que cela signifie pour vous en tant que désigner de thème ? Lorsque vous apportez des modifications, elles ne seront pas immédiatement visibles. En fait, vous devrez dire à Moodle de reconstruire le cache. Ce n'est pas pratique pour la conception de thèmes de cours. Donc le mode concepteur de thème a été ajouté. Une fois activé, il dit à Moodle de ne pas combiner les CSS et d'utiliser le cache. Ceci a pour inconvénient d'allonger les temps de chargement de la page, cependant, vous pourrez voir vos modifications immédiatement à chaque chargement.

Le mode concepteur de thème peut être activé via Administration> Présentation> Thèmes> Réglages Thème .

Avertissement : Les versions d'Internet Explorer 6 et 7 ont de gros problèmes quand un site tente de lier à plus de 31 feuilles de style, dans ce cas, soit un nombre limité ou aucun styles sont appliqués. Parce que Moodle envoie jusqu'à tous les CSS, tout le temps avec le mode concepteur de thème allumé. Il ya une grande chance, vous obtiendrez très plus de 31 feuilles de style étant inclus. Ce sera, bien entendu, causer de graves problèmes pour Internet Explorer jusqu'à ce que le mode concepteur de thème soit éteint.

Mise en route

La première chose que vous devez faire est de créer les répertoires et les fichiers que vous allez utiliser. La première chose à créer le répertoire réel pour votre thème, ce devrait être le nom de votre thème, dans mon cas, c'est «excitement». Le répertoire doit être situé dans le répertoire de thèmes de Moodle,. / Moodle / theme / excitement / sera le répertoire que je crée.

Maintenant, dans ce répertoire, nous pouvons immédiatement créer plusieurs fichiers que nous savons que nous allons avoir besoin.

Ainsi, les fichiers que nous voulons créer sont les suivants:

config.php

Tous nos paramètres seront définis ici.

/ Style /

Ce répertoire contient l'ensemble de nos feuilles de style.

/ Style / excitement.css

Notre feuille de style CSS.

/ Pix /

Dans ce répertoire on va mettre une capture d'écran de notre thème ainsi que notre favicon et les images que nous utilisons dans CSS.

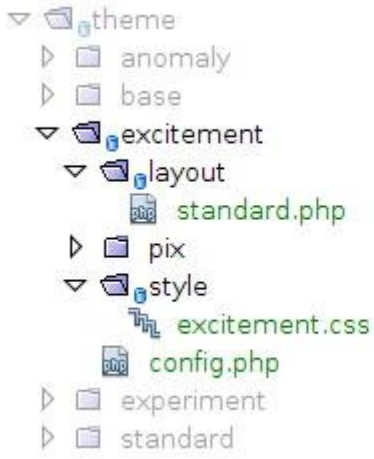
/ Layout /

Notre mise en page des fichiers se retrouveront dans ce répertoire.

/ Layout / standard.php

Ce sera notre fichier layout de base.

Donc, après cette étape de configuration, vous devriez avoir une structure similaire à ce qui est indiqué ci-dessous.



Configuration de notre thème

Ouvrir config.php dans votre éditeur favori et commencez par ajouter l'ouverture PHP balises <?php

Maintenant, nous devons ajouter les paramètres:

```
$THEME->name = 'excitement';
```

Très simplement, cela indique à Moodle le nom de votre thème.

Ensuite, les parents de ce thème.

```
$THEME->parents = array('base');
```

Ceci indique que mon nouveau thème «excitement» étend (hérite) le thème de base.

Un thème peut étendre un certain nombre de thèmes. Plutôt que de créer un thème entièrement nouveau et de copier tous les CSS, vous pouvez simplement créer un nouveau thème, de prolonger le thème que vous aimez et juste ajouter les modifications que vous souhaitez à votre thème.

A noter également le but du thème de base: il nous offre une structure de base et juste assez de CSS pour tout mettre en place.

Maintenant les feuilles de style :

```
$THEME->sheets = array('excitement');
```

La dernière chose que nous devons ajouter dans le fichier config.php notre thème est la définition de la mise en page pour notre thème:

```
$THEME->layouts = array(
    'base' => array(
        'file' => 'standard.php',
        'regions' => array(),
    ),
    'standard' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
    ),
    'course' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
```

```
        'defaultregion' => 'side-post'
    ),
    'coursecategory' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
    ),
    'incourse' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
    ),
    'frontpage' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
    ),
    'admin' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre'),
        'defaultregion' => 'side-pre',
    ),
    'mydashboard' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
        'options' => array('langmenu'=>true),
    ),
    'mypublic' => array(
        'file' => 'standard.php',
        'regions' => array('side-pre', 'side-post'),
        'defaultregion' => 'side-post',
    ),
    'login' => array(
        'file' => 'standard.php',
        'regions' => array(),
        'options' => array('langmenu'=>true),
    ),
    'popup' => array(
        'file' => 'standard.php',
        'regions' => array(),
        'options' => array('nofooter'=>true),
    ),
    'frametop' => array(
        'file' => 'standard.php',
```

```
        'regions' => array(),
        'options' => array('nofooter'=>true),
    ),
    'maintenance' => array(
        'file' => 'standard.php',
        'regions' => array(),
        'options' => array('nofooter'=>true, 'nonavbar'=>true),
    ),
    'print' => array(
        'file' => 'standard.php',
        'regions' => array(),
        'options' => array('nofooter'=>true, 'nonavbar'=>false),
    ),
);

/** List of javascript files that need to be included on each page */
$THEME->javascripts = array();
$THEME->javascripts_footer = array();
```

Ci-dessus les différents layouts pour notre thème. Il y a un layout pour chaque type général de page. Avec mon thème «excitement» , j'ai choisi d'utiliser ma propre mise en page. Sauf qu'il ya une raison particulière de le faire, normalement vous n'auriez pas besoin de créer vos propres mises en page, vous pouvez étendre le thème de base, et l'utilisation de ses layouts, ce qui signifie que vous avez seulement à écrire les CSS pour réaliser le look désiré.

Pour chaque mise en page ci-dessus, vous remarquerez les quatre points suivants sont fixés:

file :

C'est le nom du fichier layout que nous voulons utiliser, il doit toujours être situé dans le répertoire layouts des thèmes ci-dessus. Pour nous, c'est de standard.php puisque nous n'avons qu'un seul fichier layout.

regions :

Il s'agit de l'ensemble de bloc régions de notre thème. Chaque entrée ici peut être utilisés pour placer des blocs quand cette mise en page est utilisé.

defaultregion :

Si un layout a des régions, il devrait avoir une région par défaut. C'est là que les blocs viennent se mettre lorsque ils sont ajoutés en premier.

options :

Ce sont des paramètres spéciaux, tout ce qui se mettre dans le tableau d'options est disponible plus tard, quand nous écrivons notre fichier layout.

Il existe des paramètres supplémentaires qui peuvent être définis dans le fichier config.php - voir Thèmes 2.0 pour la liste complète.

Ecrire le fichier Layout

Le thème ‘excitement’ n'a qu'un seul fichier de format.
L'inconvénient est que le fichier doit rendre la mise en page pour l’ensemble de Moodle, ce qui signifie que j'ai besoin de faire usage de certaines options (définies dans le fichier config.php).
L'avantage est que je n'ai besoin que de conserver un fichier.
Autre que la maintenance, l'utilisation de multiples fichiers layout présente de nombreux avantages. Vous pouvez facilement modifier et personnaliser la disposition pour atteindre les objectifs de l'organisation utilisant le thème.
Donc commençons par écrire standard.php, le fichier layout pour mon thème ‘excitement’.

Le début du fichier layout

```
<?php
$hassidepre = $PAGE->blocks->region_has_content('side-pre', $OUTPUT);
$hassidepost = $PAGE->blocks->region_has_content('side-post', $OUTPUT);
echo $OUTPUT->doctype(); ?>
<html <?php echo $OUTPUT->htmlattributes() ?>>
<head>
    <title><?php echo $PAGE->title ?></title>
    <?php echo $OUTPUT->standard_head_html() ?>
</head>
```

Regardons le code qui va dans cette section:

```
<?php
echo $OUTPUT->doctype(); ?>
```

Ceci est très important et il est nécessaire d'être tout en haut de la page. Cet appel envoie au navigateur l’entête de type de document déterminé par les paramètres dans Moodle.

```
<html <?php echo $OUTPUT->htmlattributes() ?>>
```

Ici nous ouvrons la balise HTML et Moodle pour imprimer les attributs de la balise.

```
<title><?php echo $PAGE->title ?></title>
```

crée simplement la balise title avec son contenu.

```
<?php echo $OUTPUT->standard_head_html() ?>
```

Ici, nous demandons à Moodle d'envoyer au navigateur la balise HEAD et son contenu. Cela inclut les feuilles de style, des balises de script et ligne de code JavaScript.

L'entête de la page.

```
<body id="<?php echo $PAGE->bodyid; ?>" class="<?php echo $PAGE->bodyclasses; ?>">
<?php echo $OUTPUT->standard_top_of_body_html() ?>
<div id="page">
<?php if ($PAGE->heading || (empty($PAGE->layout_options['nonavbar']) && $PAGE-
>has_navbar())) { ?>
    <div id="page-header">
        <?php if ($PAGE->heading) { ?>
            <h1 class="headermain"><?php echo $PAGE->heading ?></h1>
            <div class="headermenu"><?php
                echo $OUTPUT->login_info();
                if (!empty($PAGE->layout_options['langmenu'])) {
                    echo $OUTPUT->lang_menu();
                }
                echo $PAGE->headingmenu
            ?></div>
        <?php } ?>
        <?php if (empty($PAGE->layout_options['nonavbar']) && $PAGE->has_navbar()) { ?>
            <div class="navbar clearfix">
                <div class="breadcrumb"><?php echo $OUTPUT->navbar(); ?></div>
                <div class="navbutton"> <?php echo $PAGE->button; ?></div>
            </div>
        <?php } ?>
    </div>
<?php } ?>
```

Donc, il ya un peu plus de choses ici évidemment.

```
<body id="<?php echo $PAGE->bodyid; ?>" class="<?php echo
$PAGE->bodyclasses; ?>">
```

Encore une fois, nous avons commencé à écrire la balise body ouverture et nous demandons alors à Moodle de nous donner l'ID de la balise body ainsi que les classes que nous devrions utiliser.

```
<?php echo $OUTPUT->standard_top_of_body_html() ?>
```

Cet appel très important, écrit quelques morceaux critiques de JavaScript dans la page. Il doit toujours être situé après la balise body dès que possible.

```
<?php if ($PAGE->heading || (empty($PAGE->layout_options['nonavbar']) && $PAGE->has_navbar())) { ?>
......
<?php } ?>
```

Ici nous vérifions si oui ou non nous avons besoin d'afficher l'en-tête de la page. Il y a trois contrôles a faire ici:

1. **$ Page-> rubrique** : On s'assure que la page dispose d'un en-tête.
2. **empty ($ page-> layout_options ['nonavbar'])** : Maintenant, cette vérification se penche sur les options de layout que nous avons mis dans notre fichier config.php. Il cherche à voir si l'option 'nonavbar' est vrai (true).
3. **$ Page-> has_navbar ()** : on vérifie si la page a une barre de navigation à afficher.

Si la page a un en-tête et si elle a une barre de navigation, nous affichons l'en-tête.

```
<?php if ($PAGE->heading) { ?>
    <h1 class="headermain"><?php echo $PAGE->heading ?></h1>
    .....
<?php } ?>
```

Cette ligne va vérifier s'il existe un en-tête. Dans ce cas, nous affichons l'en-tête de la page.

```
<div class="headermenu"><?php
    echo $OUTPUT->login_info();
    if (!empty($PAGE->layout_options['langmenu'])) {
        echo $OUTPUT->lang_menu();
    }
    echo $PAGE->headingmenu
?></div>
```

Ici, nous affichons le menu et le contenu que vous voyez en haut de la page (généralement à droite). Nous commençons par afficher les informations de connexion pour l'utilisateur actuel. Si l'utilisateur est connecté à cet endroit on affiche leur nom et un lien vers leur profil, sinon il y aura un lien pour se connecter.

Ensuite, en fonctions des options du layout, on affiche un menu de langue (Français, Anglais, ...). Si dans les options du layout dans le fichier config.php, il y est définit **langmenu => true** . Nous allons afficher le menu des langues, une liste déroulante qui permet à l'utilisateur de changer la langue qui utilisé par Moodle.
Enfin, la page a aussi un en-tête de menu

```
<?php if (empty($PAGE->layout_options['nonavbar']) && $PAGE->has_navbar()) { ?>
    <div class="navbar clearfix">
        <div class="breadcrumb"><?php echo $OUTPUT->navbar(); ?></div>
        <div class="navbutton"> <?php echo $PAGE->button; ?></div>
    </div>
<?php } ?>
```

La dernière partie de l'en-tête.
Ici, nous affichons la barre de navigation de la page, si elle existe.
En supposant qu'il ya un en-tête, puis il ya deux choses que nous avons besoin d'afficher. La première est la barre de navigation et le second est un bouton.
Dans les deux cas nous pouvons choisir de les envelopper dans un div. Bien que ce soit pour l'en-tête. Il y a beaucoup de PHP par rapport aux autres sections du fichier de configuration mais il ne change pas et peut être copié et collé entre les thèmes.

Le contenu de la page

Le thème défini deux blocs régions, plus le bloc contenu principal.
Parce que ce thème hérite du thème de base, le HTML est un peu intense. C'est parce que, c'est une mise en page avec des div flottante dont le contenu vient d'abord et ensuite nous obtenons les colonnes (même si une colonne sera à la gauche du contenu.) Ne vous inquiétez pas trop à ce sujet. Quand il s'agira d'écrire votre propre thème, vous pourrez faire ce que vous voudrez.

```
<div id="page-content">
    <div id="region-main-box">
        <div id="region-post-box">
            <div id="region-main-wrap">
                <div id="region-main">
                    <div class="region-content">
                        <?php echo core_renderer::MAIN_CONTENT_TOKEN ?>
                    </div>
                </div>
            </div>
            <?php if ($hassidepre) { ?>
                <div id="region-pre">
                    <div class="region-content">
```

```
                        <?php echo $OUTPUT->blocks_for_region('side-pre') ?>
                    </div>
                </div>
                <?php } ?>

                <?php if ($hassidepost) { ?>
                <div id="region-post">
                    <div class="region-content">
                        <?php echo $OUTPUT->blocks_for_region('side-post') ?>
                    </div>
                </div>
            <?php } ?>
        </div>
    </div>
</div>
```

En ce qui concerne cette section, le PHP est très facile. Il n'y a que trois lignes pour toute la section une pour obtenir le contenu principal et une pour chaque région bloc.

```
<?php echo core_renderer::MAIN_CONTENT_TOKEN ?>
```

Cette ligne affiche le contenu principal de la page.

```
<?php if ($hassidepre) { ?>
....
<?php } ?>
```

Ces lignes de code vérifie les variables que nous avons créé plus tôt, pour décider si nous devons afficher le bloc de région ‘side-pre’. Si vous essayez d'afficher un bloc de région qui n’existe pas ou qui n'a pas de contenu, Moodle va afficher un message d'erreur. Ces lignes sont donc très importantes.
Si vous avez un message d'erreur de type "unknown block region side-pre" ou "unknown block region side-post". Il suffit d'ajouter controler les variables ‘$hassidepre’ et ‘hassidepost’ avant de définir les div des blocs et tout ira bien.

```
<?php echo $OUTPUT->blocks_for_region('side-pre') ?>
```

Cette ligne reçoit tous les blocs et plus particulièrement le contenu pour le bloc région ‘side-pre’. Ce bloc région sera affiché à gauche du contenu.

```
<?php if ($hassidepost) { ?>
....
<?php } ?>
```

Encore une fois, nous procédons à la vérification de l'existence de bloc pour la région 'side-post'.

```
<?php echo $OUTPUT->blocks_for_region('side-post') ?>
```

Ici, nous obtenons le bloc région 'side-post' qui sera affiché à droite du contenu.

Le pied de page

Ici, nous voulons imprimer le pied de la page, tout le contenu requis par Moodle, puis fermez les dernières balises.

```
<?php if (empty($PAGE->layout_options['nofooter'])) { ?>
    <div id="page-footer" class="clearfix">
        <p class="helplink"><?php echo
page_doc_link(get_string('moodledocslink')) ?></p>
        <?php
        echo $OUTPUT->login_info();
        echo $OUTPUT->home_link();
        echo $OUTPUT->standard_footer_html();
        ?>
    </div>
    <?php } ?>
</div>
<?php echo $OUTPUT->standard_end_of_body_html() ?>
</body>
</html>
```

La section de code est responsable de l'affichage du pied de page pour la page.

```
<?php if (empty($PAGE->layout_options['nofooter'])) { ?>
    <div id="page-footer" class="clearfix">
        <p class="helplink"><?php echo
page_doc_link(get_string('moodledocslink')) ?></p>
        <?php
        echo $OUTPUT->login_info();
        echo $OUTPUT->home_link();
```

```
        echo $OUTPUT->standard_footer_html();
        ?>
    </div>
    <?php } ?>
```

La première chose à faire avant d'afficher le pied de page est de vérifier ce que nous allons afficher. Cela se fait avec les options pour la mise en page dans le fichier config.php. Si **nofooter => true,** nous ne voulons pas afficher le pied de page.
En supposant que nous voulons afficher un pied de page, nous créons un div pour hébergé le contenu. Il ya quatre choses a afficher dans un pied de page typique :

echo page_doc_link (get_string ('moodledocslink'))
Ce qui affichera un lien vers la page d'aide Moodle.org.

echo $ OUTPUT login_info-> ();
C'est la même chose que dans le haut de la page, on affiche les informations de connexion pour l'utilisateur actuel.

echo $ OUTPUT home_link-> ();
Ceci affiche un lien vers la page d'accueil de Moodle pour ce site.

echo $ OUTPUT standard_footer_html-> ();
Cet affichage HTML est déterminé par les paramètres du site. Des choses telles que des informations de performance et de débogage seront affichés par cette ligne si elles sont activées.

Et la dernière ligne de code de notre fichier de format est le suivant:

```
<?php echo $OUTPUT->standard_end_of_body_html(); ?>
```

C'est l'une des lignes de code les plus importantes dans le fichier layout. Il demande à Moodle d'envoyer tout le contenu requis dans la page, et il y aura probablement beaucoup même si la plupart de celui-ci ne sera pas visuelle.
Ce sera plutôt des choses comme les scripts en ligne et les fichiers JavaScript nécessaire d'aller au bas de la page. Si vous oubliez cette ligne, le JavaScript ne va pas fonctionner!
Nous avons maintenant écrit notre fichier layout, ci-dessous, le code source complet. N'oubliez pas que si vous voulez des exemples plus pratique regarder les fichiers situés dans le répertoire layout des autres thèmes.

```
<?php
$hassidepre = $PAGE->blocks->region_has_content('side-pre', $OUTPUT);
$hassidepost = $PAGE->blocks->region_has_content('side-post', $OUTPUT);
echo $OUTPUT->doctype() ?>
<html <?php echo $OUTPUT->htmlattributes() ?>>
```

```
<head>
    <title><?php echo $PAGE->title; ?></title>
    <link rel="shortcut icon" href="<?php echo $OUTPUT->pix_url('favicon', 'theme')?>"
/>
    <?php echo $OUTPUT->standard_head_html() ?>
</head>

<body id="<?php echo $PAGE->bodyid; ?>" class="<?php echo $PAGE->bodyclasses; ?>">
<?php echo $OUTPUT->standard_top_of_body_html() ?>
<div id="page">
<?php if ($PAGE->heading || (empty($PAGE->layout_options['nonavbar']) && $PAGE-
>has_navbar())) { ?>
    <div id="page-header">
        <?php if ($PAGE->heading) { ?>
        <h1 class="headermain"><?php echo $PAGE->heading ?></h1>
        <div class="headermenu"><?php
            echo $OUTPUT->login_info();
            if (!empty($PAGE->layout_options['langmenu'])) {
                echo $OUTPUT->lang_menu();
            }
            echo $PAGE->headingmenu
        ?></div><?php } ?>
        <?php if (empty($PAGE->layout_options['nonavbar']) && $PAGE->has_navbar()) { ?>
            <div class="navbar clearfix">
                <div class="breadcrumb"><?php echo $OUTPUT->navbar(); ?></div>
                <div class="navbutton"> <?php echo $PAGE->button; ?></div>
            </div>
        <?php } ?>
    </div>
<?php } ?>

    <div id="page-content">
        <div id="region-main-box">
            <div id="region-post-box">
                <div id="region-main-wrap">
                    <div id="region-main">
                        <div class="region-content">
                            <?php echo core_renderer::MAIN_CONTENT_TOKEN ?>
                        </div>
                    </div>
                </div>
                <?php if ($hassidepre) { ?>
                <div id="region-pre">
                    <div class="region-content">
                        <?php echo $OUTPUT->blocks_for_region('side-pre') ?>
```

```
                    </div>
                </div>
                <?php } ?>

                <?php if ($hassidepost) { ?>
                <div id="region-post">
                    <div class="region-content">
                        <?php echo $OUTPUT->blocks_for_region('side-post') ?>
                    </div>
                </div>
                <?php } ?>
            </div>
        </div>
    </div>

    <?php if (empty($PAGE->layout_options['nofooter'])) { ?>
    <div id="page-footer" class="clearfix">
        <p class="helplink"><?php echo
page_doc_link(get_string('moodledocslink')) ?></p>
        <?php
        echo $OUTPUT->login_info();
        echo $OUTPUT->home_link();
        echo $OUTPUT->standard_footer_html();
        ?>
    </div>
    <?php } ?>
</div>
<?php echo $OUTPUT->standard_end_of_body_html() ?>
</body>
</html>
```

Ajout d'un peu de CSS

Avec le fichier config.php et standard.php, le thème est maintenant utilisable et commence à ressembler à un vrai thème, mais si vous l'activez à l'aide du sélecteur de thème, vous remarquerez qu'il manque encore tout le style.
C'est bien sûr là où le CSS entre en jeu. Lors de l'écriture du code, les développeurs Moodle sont fortement encouragés à ne pas utiliser les styles en ligne n'importe où. C'est fantastique pour nous en tant que créateurs de thèmes, car il n'y a rien (ou du moins très peu) dans Moodle qui ne peuvent pas être stylé a l'aide de CSS.

CSS Moodle de base

Dans Moodle 2.0, tous les CSS pour l'ensemble de Moodle sont envoyés a chaque chargement de la page. Cela a été fait pour des raisons de performances. Moodle lit tous les CSS, il les combine en un seul fichier, l'allège en enlevant tout les espaces blancs, le met en cache, et ensuite l'envoie au navigateur.
Qu'est-ce que cela signifie pour vous en tant que Themer?
Vous aurez besoin d'écrire de bonnes règles CSS qui ne seront pas en conflit avec toute autre règle CSS dans Moodle.
Moodle est tellement grand et complexe, qu'il n'y a aucun moyen de s'assurer que les classes ne sont pas réutilisées. Toutefois ce que nous pouvons contrôler, c'est les classes et les id qui sont ajoutés à la balise "body" pour chaque page. Lors de l'écriture CSS, il est fortement encouragés de faire plein usage de ces classes 'BODY', en les utilisant les règles CSS que vous écrivez aurons moins de chance de provoquer des conflits.
Vous devriez également prendre le temps de regarder comment les thèmes Moodle utilise les CSS. Regardez leur utilisation des classes 'BODY' et la façon dont ils séparent le CSS dans des fichiers séparés sur la base de la région de Moodle auxquelles il s'applique.

Commencer à écrire excitement.css

```
a {text-decoration: none;}
.addcoursebutton .singlebutton {text-align: center;}

h1.headermain {color: #fff;}
h2.main {border-bottom: 3px solid #013D6A;color: #013D6A;text-align: center;}
h2.headingblock {font-size: 18pt;margin-top: 0;background-color: #013D6A;color:
#FFF;text-align: center;}
#page-header {background-color: #013D6A;}
#page-header .headermenu  {color: #FFF;}
#page-header .headermenu a {color: #FDFF2A;}

.navbar {padding-left: 1em;}
.breadcrumb li {color: #FFF;}
.breadcrumb li a {color: #FFF;}

.block {background-color: #013D6A;}
.block .header .title {color: #FFF;}
.block .header .title .block_action input {background-color: #FFF;}
.block .content {border: 1px solid #000;padding: 5px;background-color: #FFF;}
.block .content .block_tree p {font-size: 80%;}

.block_settings_navigation_tree .content .footer {text-align: center;}
.block_settings_navigation_tree .content .footer .adminsearchform {margin-left:
5%;width: 90%;font-size: 9pt;}
.block_settings_navigation_tree .content .footer .adminsearchform #adminsearchquery
{width: 95%;}
```

```
.block_calendar_month .content .calendar-controls a {color: #013D6A;font-weight: bold;}
.block_calendar_month .content .minicalendar td {border-color: #FFF;}
.block_calendar_month .content .minicalendar .day {color: #FFF;background-color:
#013D6A;}
.block_calendar_month .content .minicalendar .day a {color: #FFF000;}
.block_calendar_month .content .minicalendar .weekdays th {border-width: 0;font-weight:
bold;color: #013D6A;}
.block_calendar_month .content .minicalendar .weekdays abbr {border-width: 0;text-
decoration: none;}
```

Ce n'est qu'une partie des règles CSS du thème, c'est juste assez pour afficher le style de la première page lorsque l'utilisateur n'est pas connecté. Rappelez-vous ce thème étend le thème 'base', une partie du CSS est hérité, de sorte qu'il y a déjà du CSS pour la mise en page.

Note:

- Les règles CSS sont écrites sur une seule ligne. Les thèmes de base de Moodle sont écrit ainsi. Les sélecteurs sont plus rapides à lire et à voir où le style s'applique.
- J'ai écrit mes sélecteurs en prenant en compte la structure du HTML (plus que l'id de la balise que je veux décorer). Cela permet en outre de réduire les conflits.
- J'utilise des classes génériques comme. *sideblock* si je veux être générique, dès que je veux être précis, j'utilise les classes uniques, telles que. *block_calendar_month*

L'utilisation d'images dans les règles CSS.

J'ai créé deux images pour mon thème, l'image de fond : background.png et un gradient pour l'entête et les titres : gradient.png. Ces deux images sont dans le répertoire 'pix' de mon thème.

```
html {background-image:url([[pix:theme|background]]);}

h2.headingblock,
#page-header,
.sideblock,
.block_calendar_month .content .minicalendar .day {background-
image:url([[pix:theme|gradient]]);background-repeat:repeat-x;background-color:
#0273C8;}
```

Ci-dessus, les deux nouvelles règles a écrire pour utiliser mes images.
La première règle définit background.png comme image de fond.
La deuxième règle définit gradient.png comme image de fond pour les titres et les blocs latéraux. Vous remarquerez que je n'ai pas besoin d'écrire le chemin complet vers l'image. Moodle remplace 'url([[pix:theme|gradient]])' par l'url de l'image lorsque il compile les fichiers CSS.

D’autres designers de thèmes peuvent choisir d’étendre votre thème; si vous utilisez cette syntaxe, tout ce qu'ils doivent faire pour remplacer l'image, est de créer une image avec le même nom dans le répertoire ‘pix’ de leur thème.
Vous remarquerez également que je n'ai pas besoin d'ajouter l'extension des fichiers image. C'est parce que Moodle est assez intelligent pour savoir quel est l'extension du fichier utilisé. Cela permet également de passer outre de différents formats d’image lorsque l’on surcharge les images.

Le CSS complet du thème ‘excitment’

```
a {text-decoration: none;}
.addcoursebutton .singlebutton {text-align: center;}

h1.headermain {color: #fff;}
h2.main {border-bottom: 3px solid #013D6A;color: #013D6A;text-align: center;}
h2.headingblock {font-size: 18pt;margin-top: 0;background-color: #013D6A;color:
#FFF;text-align: center;}
#page-header {background-color: #013D6A;border-bottom:5px solid #013D6A;}
#page-header .headermenu  {color: #FFF;}
#page-header .headermenu a {color: #FDFF2A;}

.sideblock {background-color: #013D6A;}
.sideblock .header .title {color: #FFF;}
.sideblock .header .title .block_action input {background-color: #FFF;}
.sideblock .content {border: 1px solid #000;padding: 5px;background-color: #FFF;}
.sideblock .content .block_tree p {font-size: 80%;}

.block_settings_navigation_tree .content .footer {text-align: center;}
.block_settings_navigation_tree .content .footer .adminsearchform {margin-left:
5%;width: 90%;font-size: 9pt;}
.block_settings_navigation_tree .content .footer .adminsearchform #adminsearchquery
{width: 95%;}

.block_calendar_month .content .calendar-controls a {color: #013D6A;font-weight: bold;}
.block_calendar_month .content .minicalendar td {border-color: #FFF;}
.block_calendar_month .content .minicalendar .day {color: #FFF;background-color:
#013D6A;}
.block_calendar_month .content .minicalendar .day a {color: #FFF000;}
.block_calendar_month .content .minicalendar .weekdays th {border-width: 0;font-weight:
bold;color: #013D6A;}
.block_calendar_month .content .minicalendar .weekdays abbr {border-width: 0;text-
decoration: none;}

html {background-image:url([[pix:theme|background]]);}
```

```
h2.headingblock,
#page-header,
.sideblock,
.block_calendar_month .content .minicalendar .day {background-
image:url([[pix:theme|gradient]]);
   background-repeat:repeat-x;background-color: #0273C8;}
```

Ajout d'une capture d'écran et de favicon

La dernière chose à faire à ce stade est d'ajouter à la fois une capture d'écran du thème et un favicon. La capture d'écran sera montrée dans l'écran de sélection des thèmes et devrait être nommé '*screenshot.jpg*'. Le favicon sera utilisé lorsque quelqu'un place un signet sur la page. Les deux images doivent être situées dans votre répertoire 'pix' comme suit:

- /theme/excitement/pix/screenshot.jpg
- /theme/excitement/pix/favicon.ico

Annexe

Pour aller plus loin (document en anglais) :

Surcharger un renderers : Un tutoriel sur la création d'un moteur de rendu personnalisé permettant de changer le code HTML produit par Moodle.
http://docs.moodle.org/en/Development:Themes_2.0_overriding_a_renderer

Comment utiliser les images dans votre thème : Explique comment utiliser et remplacer des images dans votre thème. http://docs.moodle.org/en/Development:Themes_2.0_How_to_use_images_within_your_theme

Ajouter une page de configuration : Comment ajouter une page de configuration pour facilement personnaliser votre thème. http://docs.moodle.org/en/Development:Themes_2.0_adding_a_settings_page

Personnaliser le menu : Comment personnaliser le menu Moodle.
http://docs.moodle.org/en/Development:Themes_2.0_extending_the_custom_menu

Le style et la personnalisation du dock : Comment décorer et personnaliser le dock de Moodle.
http://docs.moodle.org/en/Development:Styling_and_customising_the_dock

Utilisation de jQuery avec Moodle. http://docs.moodle.org/en/Development:Using_jQuery_with_Moodle_2.0