Theming : c'est compliqué mais on vous explique

Romain JARRAUD

DRUPALCAMP PARIS 2013

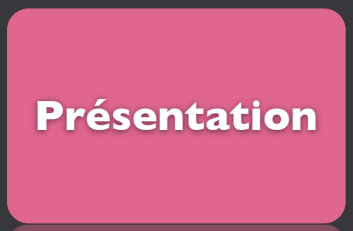
actency Acquia adyax Open Source Experts commerce guys OWS Windows Azure

Romain JARRAUD
Formateur/consultant
Trained People - drupalfrance.com
@romainjarraud

Qu'est-ce que le theming ?

Theming ?

Drupal et modules



Thème

Theming ?

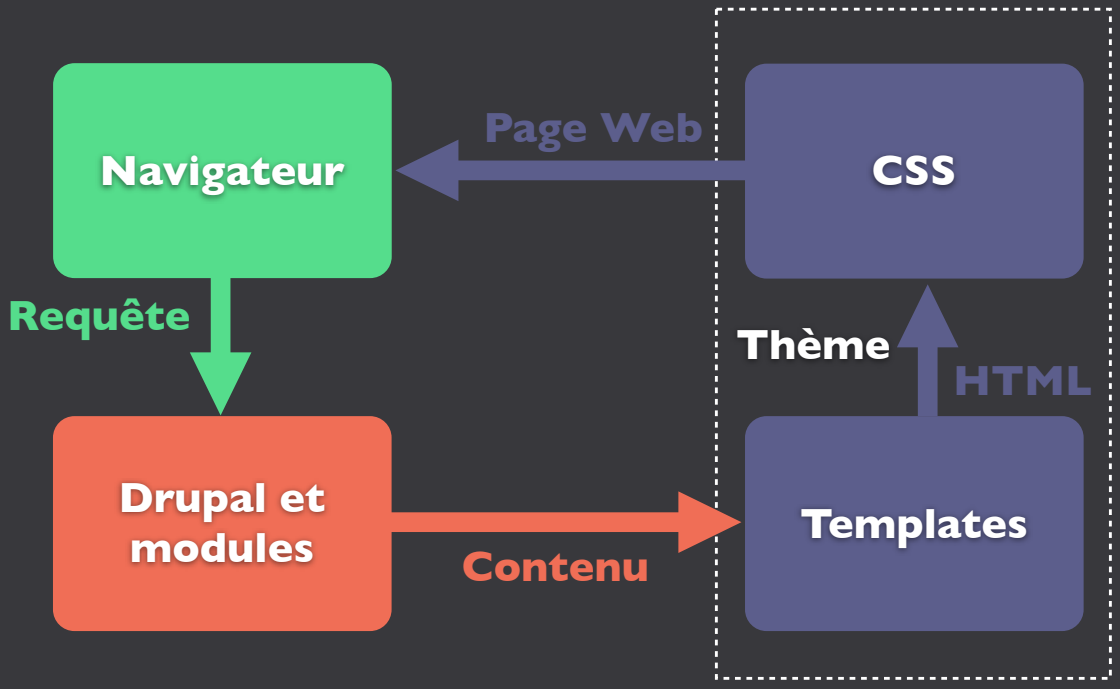
- Produire le **code html** de chaque page.
- Habiller les balises de **styles** : tailles, couleurs, images...
- Ajouter des **effets** à l'aide de javascript.

Moteur de thème PHPTemplate

- Drupal 7 utilise le moteur de thème **PHPTemplate**.
- PHPTemplate a été créé par **Adrian Rossouw**.
- Il offre un système de **templates** (*template.tpl.php*) et de **fonctions de thème** (*theme_function()*) pour générer le code HTML.
- La partie dynamique est obtenue grâce à du **code php** inséré dans le HTML.

Principe de base

Principe


Navigateur
Page Web
CSS
Requête
Thème
HTML
Drupal et modules
Contenu
Templates

CSS

CSS

- Les fichiers CSS peuvent provenir de 3 sources :
  - Coeur de **Drupal**
  - **Modules** contrib
  - **Thème**
- L'ordre de chargement de ces fichiers est primordial; le thème doit pouvoir surcharger tous les styles et ainsi reprendre la main.

CSS - chargement

```
// Fichier mon_theme.info.

stylesheets[all][] = messtyles.css
```

```
// Fonction drupal_add_css().

$data = drupal_get_file('theme', 'mon_theme') . '/messtyles.css';
$options = array(
  'group'      => CSS_THEME,
  'weight'     => '12',
  'media'      => 'all',
  'preprocess' => FALSE,
  'every_page' => FALSE,
  'browser'    => array('IE' => TRUE, '!IE' => FALSE),
);
drupal_add_css($data, $options);
```

Intercepter les CSSs

```
function mon_theme_css_alter(&$css) {
  // Afficher le contenu de $css.
  dpm($css);
  // Ne pas charger le fichier file.css du module nom_du_module.
  unset($css[drupal_get_path('module', 'nom_du_module') . '/file.css']);
}
```

Javascript

Javascript

- Les fichiers JS peuvent provenir de 3 sources :
  - Coeur de **Drupal**
  - **Modules** contrib
  - **Thème**
- L'ordre de chargement de ces fichiers est primordial; par exemple le thème ne peut utiliser une librairies qu'à partir du moment où elle a déjà été chargée.

Javascript

```
// Fichier mon_theme.info.

scripts[] = monscript.js
```

```
// Fonction drupal_add_js().

$data = drupal_get_file('theme', 'mon_theme') . '/monscript.js';
$options = array(
  'group'      => JS_THEME,
  'weight'     => '12',
  'defer'      => TRUE,
  'preprocess' => FALSE,
  'every_page' => FALSE,
  'scope'      => 'footer',
);
drupal_add_js($data, $options);
```

Intercepter les JSs

```
function mon_theme_js_alter(&$js) {
  // Afficher le contenu de $js.
  dpm($js);
  // Ne pas charger le fichier file.js du module nom_du_module.
  unset($js[drupal_get_path('module', 'nom_du_module') . '/file.js']);
}
```

Templates et fonctions de thème

Formatage HTML

- Drupal propose deux emplacements pour formater le code HTML :
  - **fonction de thème**
  - **template**
- Les fonctions de thème sont plus performantes, mais restent souvent moins lisibles et plus complexes que les fichiers de template.
- Dans les deux cas, on invoque un hook de thème avec la fonction theme() : `theme('hook_de_theme', $variables)`.
- Cette fonction permet de mettre en oeuvre le mécanisme de surcharge du thème.

Fonction de thème

- Toutes les fonctions de thème sont de la forme : `theme_hook_de_theme($variables)`.
- `$variables` est un tableau associatif contenant les données à formater.
- Une fonction de thème doit retourner du code HTML.

Fonction *theme_image()*

```
function theme_image($variables) {
  $attributes = $variables['attributes'];
  $attributes['src'] = file_create_url($variables['path']);

  foreach (array('width', 'height', 'alt', 'title') as $key) {
    if (isset($variables[$key])) {
      $attributes[$key] = $variables[$key];
    }
  }

  return '<img' . drupal_attributes($attributes) . ' />';
}
```

Template

- Tous les templates ont l'extension *.tpl.php*.
- Chaque fichier de template reçoit, non pas le tableau $variables, mais autant de variables que le tableau $variables contient de clés. Ex. : si $variables = array('data1' => $data_1, 'data2' => $data_2) est passé au template, ce dernier reçoit $data1 et $data2.
- Un template contient le **code HTML** avec du **php** qui principalement affiche le contenu des variables.

Template *block.tpl.php*

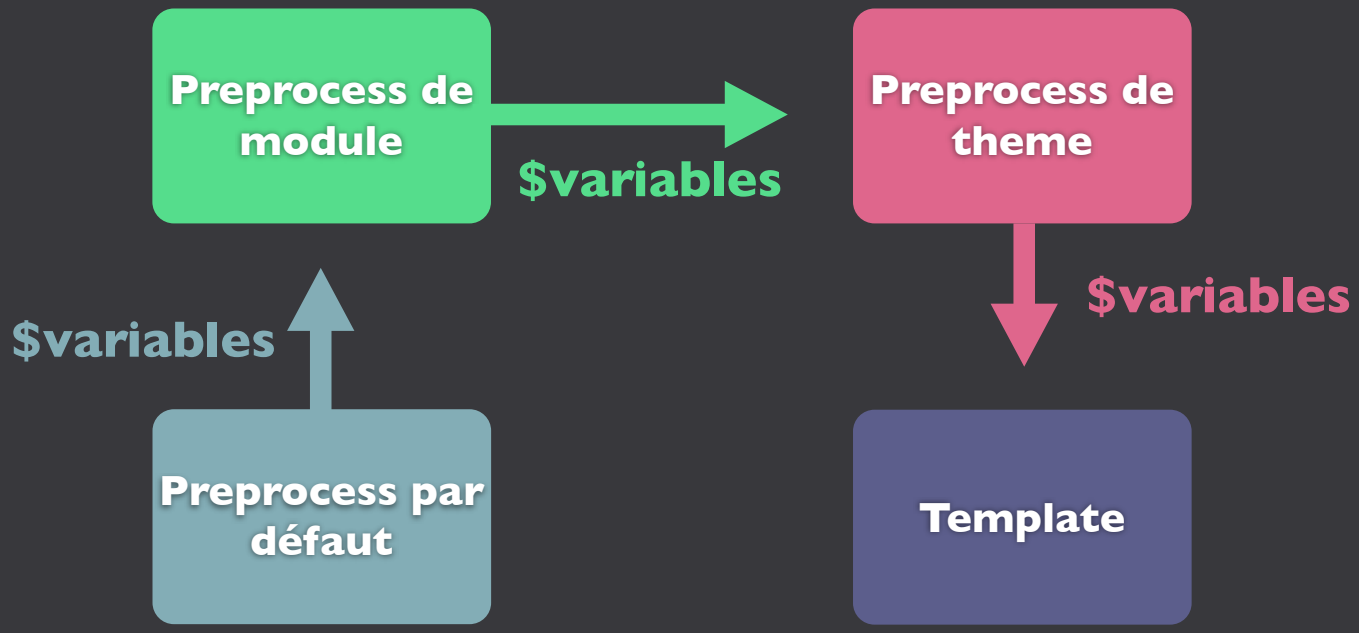
```
  <div id="<?php print $block_html_id; ?>" class="<?php print $classes; ?>"<?php print
$attributes; ?>>

    <?php print render($title_prefix); ?>
    <?php if ($block->subject): ?>
      <h2<?php print $title_attributes; ?>><?php print $block->subject ?></h2>
    <?php endif;?>
    <?php print render($title_suffix); ?>

    <div class="content"<?php print $content_attributes; ?>>
      <?php print $content ?>
    </div>
  </div>
```

Fonctions de preprocess et de process

Fonction de preprocess



Fonctions *template_preprocess()* et *template_preprocess_HOOK()*

- **Initialisation** de *$variables*.
- *$variables['classes_array']* : contient un nom de classe correspondant au nom du hook de thème invoqué.
- *$variables['theme_hook_suggestions']* : liste des noms de hook dérivés.

Fonction template_preprocess()

- Cette fonction de preprocess n'est **pas utilisée** dans le cas d'une **fonction de thème**.
- C'est donc au développeur/themeur de définir ses propres variables.

Fonction de process

- Les fonctions de process interviennent **après** celles de **preprocess** et permettent de finaliser le formattage des données **avant** de les passer aux **fonctions de thème**/ **templates**.
- Il s'agit essentiellement de transformer des tableaux en chaine :
  - *$variables['classes_array'] => $variables['classes']*
  - *$variables['attributes_array'] => $variables['attributes']*
  - *$variables['title_attributes_array'] => $variables['title_attributes']*
- Ces fonctions peuvent **être étendues**, comme celles de preprocess.

Surcharge

Surcharge

- L'un des rôles du thème est de **surcharger** le formatage par défaut afin de l'**adapter** aux besoins.
- Drupal propose un mécanisme de surcharge, aussi bien pour les fonctions de thème que pour les templates.
- Dès qu'un hook de thème est invoqué, **le thème est sollicité en premier**. Dans le cas où aucune surcharge n'est proposée, c'est l'élément par défaut qui est appelé.

Surcharge

- **Fonction de thème**
  - **Copier le code** de la fonction originale dans le fichier *template.php*.
  - **Renommer** cette dernière : ***theme***_*hookdetheme()* => ***mon_theme***_*hookdetheme()*.
  - **Vider** le registre du thème.
- **Template**
  - **Copier le fichier** original dans le dossier de son thème.
  - **Vider** le registre du thème.

Templates dérivés

Templates dérivés

- Par **défaut** chaque fois qu'un hook de thème est invoqué, le **candidat de base** est appelé.
- Il est possible d'appeler d'**autres candidats** proposant un formatage adapté au **contexte**.
- Par exemple pour afficher un nœud, le template utilisé par défaut est ***node.tpl.php***. Si ce nœud est de type *article*, Drupal peut potentiellement utiliser le template ***node--article.tpl.php*** (s'il existe!).

Déterminer le candidat

- La **liste** des candidats pour un hook de thème donné est définie par les **fonctions de preprocess**.
- Le tableau ***$variables['theme_hook_suggestions']*** contient la **liste** des noms des candidats potentiels.
- Drupal définit dans ses diverses fonctions de preprocess (*template_preprocess_HOOK()*) des suggestions par défaut, qui peuvent être modifiées ou étendues.
- Les **derniers** éléments ajoutés au tableau *$variables['theme_hook_suggestions']* sont appelés en **premier** : FILO.

Exemple : *node.tpl.php*

```
// Fonction template_preprocess_node().

...
$variables['theme_hook_suggestions'][] = 'node__' . $node->type;
$variables['theme_hook_suggestions'][] = 'node__' . $node->nid;
...
```

Exemple : *page.tpl.php*

```
// Fonction template_preprocess_page().

...
if ($suggestions = theme_get_suggestions(arg(), 'page') {
  $variables['theme_hook_suggestions'] = $suggestions;
}
...


// Fonction mon_theme_preprocess_page().

...
if ($node = menu_get_object()) {
  $node_type = $variables['node']->type;
  $variables['theme_hook_suggestions'][] = 'page__node_' . $node_type;
}
...
```

Registre de thème

Registre du thème

- Le **registre du thème** est un **cache** local au thème.
- Pour une page il peut y avoir des dizaines de fonctions de thème et de templates qui sont impliqués.
- Pour chaque hook de thème Drupal doit détermier les candidats appropriés.
- Ce processus est **extrêmement coûteux** en ressources.
- La liste de tous les fichiers et fonctions à utiliser est donc mise en cache.

Registre du thème

- Fonctions possibles pour un hook de thème donné :
  - template_preprocess()
  - template_preprocess_HOOK()
  - MODULE_preprocess()
  - MODULE_preprocess_ HOOK()
  - THEME_preprocess()
  - THEME_preprocess_ HOOK()
  - template_process()
  - template_process_ HOOK()
  - MODULE_process()
  - MODULE_process_ HOOK()
  - THEME_process()
  - THEME_process_ HOOK()

Registre du thème

- Ce que contient le registre :
  - Pour chaque hook de thème : template ou fonction ?
  - Liste des fonctions de preprocess.
  - Liste des fonctions de process.
  - Chemins vers les fichiers de template.
  - Noms des fonctions de thème.
  - Liste des variables passées à la fonction *theme()*.
  - Liste des éléments à rendre (le cas échéant) à la fonction *theme()*.

Hooks de thème

Déclarer un hook de thème

- Un hook de thème entraîne l'utilisation soit d'une **fonction de thème** soit d'un **template**.
- C'est à **vous** de décider.
- Pour chaque hook de thème que vous déclarer, Drupal utilise potentiellement la fonction de preprocess associée : `template_preprocess_hook_de_theme()`.
- Cette dernière peut également être étendue.

Déclarer une fonction de thème

```
// Fichier template.php

function mon_theme_theme($existing, $type, $theme, $path) {
  return array(
    ´mon_theme´ => array(
      ´variables´ => array('data' => NULL),
    ),
  );
}

function theme_mon_theme($variables) {
  $data = $variables['data'];
  $output = '<p>' . $data . '</p>';
  return $output;
}
```

Déclarer un template

```
// Fichier template.php

function mon_theme_theme() {
  return array(
    ´mon_theme´ => array(
      ´variables´ => array('data' => NULL),
      ´template´ => 'mon-theme',
    ),
  );
}

// Fichier mon-theme.tpl.php

<p><?php print $data; ?></p>
```

Intégrer un plugin

Ce qu'il nous faut

- **Charger** les fichiers nécessaires, CSS et Javascript.
- **Formater** le code HTML comme attendu par le plugin.
- **Déclencher** l'effet.

Exemple - Nivo Slider

```
// Fichier .info

stylesheets[all][] = nivo-slider.css
scripts[] = jquery.nivo.slider.js
scripts[] = script.js


// Fichier de template

<div id="slider">
  <img src="image1.jpg" />
  <img src="image2.jpg" />
  <img src="image3.jpg" />
</div>


// Fichier script.js

$("#slider").nivoSlider();
```

Configuration Backoffice

Configuration Backoffice

- Comment **paramétrer le thème** depuis le backoffice de Drupal ?
- Il existe une page avec un **formulaire** par défaut pour chaque thème.
- On peut le modifier avec *hook_form_system_theme_settings_alter()* en utilisant la **Form API**.
- En pratique il faut :
  - Créer le fichier *theme-settings.php* à la racine du thème.
  - Dans ce fichier ajouter la fonction ***mon_theme***_*form_system_theme_settings_alter()*.
  - Dans le fichier *mon_theme.info* ajouter *settings['mon_theme_mon_champ'] = 1* afin d'initialiser le champ.

Exemple

- Notre thème doit permettre d'**afficher/masquer une région** depuis l'interface d'administration de Drupal.
- Il faut :
  - **Déclarer** une région et un réglage (à initialiser) dans le fichier *mon_theme.info*.
  - **Insérer** une nouvelle case à cocher dans le formulaire de configuration du thème (*admin/appearance/settings/mon_theme*).
  - **Adapter** le template de page (*page.tpl.php*) afin que la région ne soit affichée que sous conditions.

Exemple

```
// Fichier mon_theme.info
...
regions[ma_region] = My region
settings[toggle_mon_theme_ma_region] = 1
...

// Fichier theme-settings.php
...
function mon_theme_form_system_settings_alter(&$form, $form_state) {
  $form['theme_settings']['toggle_mon_theme_ma_region'] = array(
    '#type'          => 'checkbox',
    '#title'         => t('My region'),
    '#default_value' => theme_get_setting('toggle_mon_theme_ma_region'),
  );
}
...
```

Exemple

```
// Fichier page.tpl.php
...
<?php
  if ($page['ma_region'] && theme_get_setting('toggle_mon_theme_ma_region')):
?>

  <div id="ma-region">
    <?php print render($page['ma_region']); ?>
  </div>

<?php endif; ?>
...
```

Et Drupal 8 ?

- On oublie tout et on recommence...
- TWIG !

Merci à vous !

Questions ?