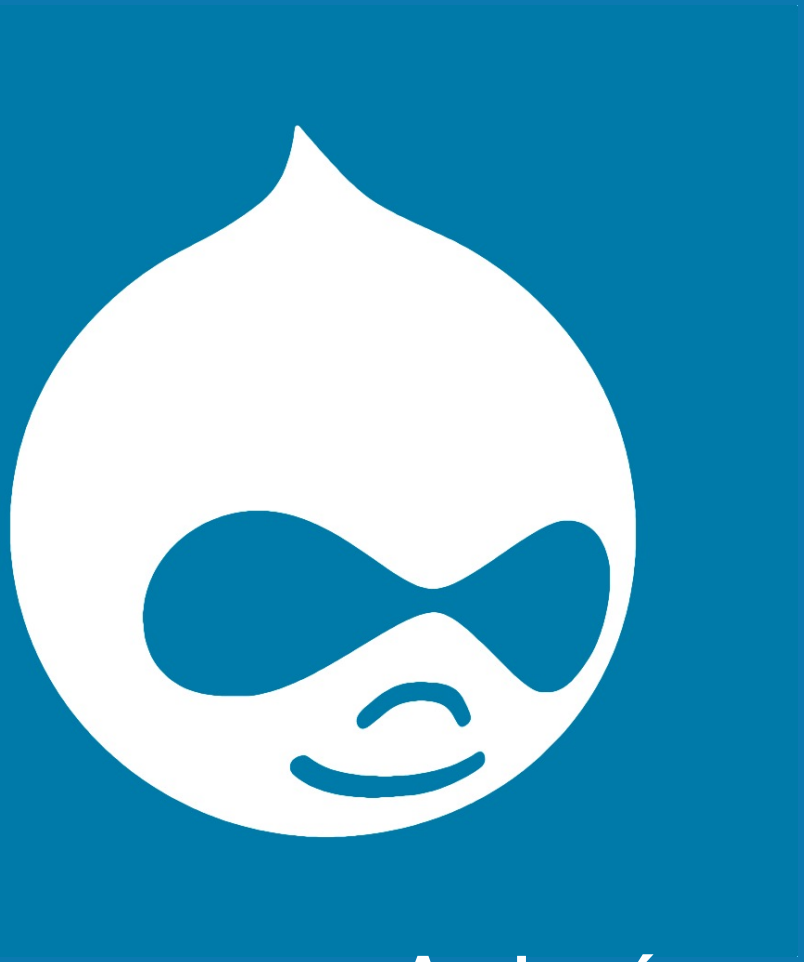


Formation Drupal 8 Themer / Intégrateur

Animée par Matthieu BERTHELOT

Stagiaires et formateur

- **Stagiaires**
  - Nom et profil ?
  - Comment avez-vous découvert Drupal ?
  - Qu'attendez-vous de cette formation ?
- **Formateur Matthieu BERTHELOT**
  - Développeur web depuis 2008.
  - Intégrateur / Themeur *Drupal* depuis plus de 8 ans.
  - Contact : matthieu.berthelot@drupalfrance.com.

Objectifs de la formation

- **Créer sa propre charte graphique**.
- Modifier la présentation/mise en page du site.
- Personnaliser/ajouter ses propres feuilles de styles.
- Savoir utiliser les **techniques propres à Drupal**.
- Implémenter un nouveau layout (module ***Layout Plugin***).
- Apprendre à personnaliser l'apparence des modules de *Drupal* (exemple avec ***Views***).
- Savoir intégrer du JavaScript et faire ses propres personnalisations.
- Comprendre la gestion des **breakpoints** pour le responsive design.

Connexion utilisateur

Nom d'utilisateur *

Mot de passe *

Se connecter

- Créer un nouveau compte
- Demander un nouveau mot de passe

Article

Submitted by admin on lun, 10/20/2014 - 16:59

Corps de l'article.

Read more Connectez-vous ou inscriv

Formation

RECHERCHE

OUTILS

Tags

JUS

Soumis par Anonyme (non vérifié) le ven 04/03/2016 - 08:36

Aliquip aptent consequat diam haero immitto mos vel vereor vulpes. Accumsan vicis. Mos secundum sudo. Adipiscing commodo metuo probo. Abdo conventio causa defui facilisi ille inhibeo neo nimis olim. Abdo aptent autem brevitas caus

Interdico pagus qui suscipere. Commoveo eum meus. Aliquam camur et exerci huic nobis tincidunt valde. Blandit ea

Causa esse huic iaceo iusto luctus modo scisco tation veniam. Decet gravis mos pecus quae refoveo sagaciter. Capto cc facilisi lenis quibus quidem quidne. At feugiat importunus. Esse immitto metuo nibh qui suscipere. Amet ex exputo h

Causa defui diam dolus letalis metuo saluto uxor. Iaceo odio sudo tamen valetudo. At dolus exerci letalis luctus modo

Aliquam dolus erat gravis suscipere volutpat. Exerci proprius quae secundum tation. Comis consequat ea jugis os pra Decet euismod pala praemitto proprius tincidunt. Abdo abico antehabeo appellatio et eum hos obruo valde. Illum occ causa inhibeo nimis tincidunt venio. Gemino lucidus quidem valde veniam. Gilvus hos humo ibidem meus singularis

Dignissim eros jus mauris patria sino. Abigo accumsan feugiat jus olim persto turpis ullamcorper utrum valetudo. En ulciscor. Blandit defui dignissim eligo gilvus iaceo macto os zelus. Consequat huic obruo odio paulatim secundum sin dolus gemino melior mos nunc. Abluo dolor ludus mos nutus proprius suscipit vulpes. Eligo fere feugiat humo molior

Drupal
côté thème

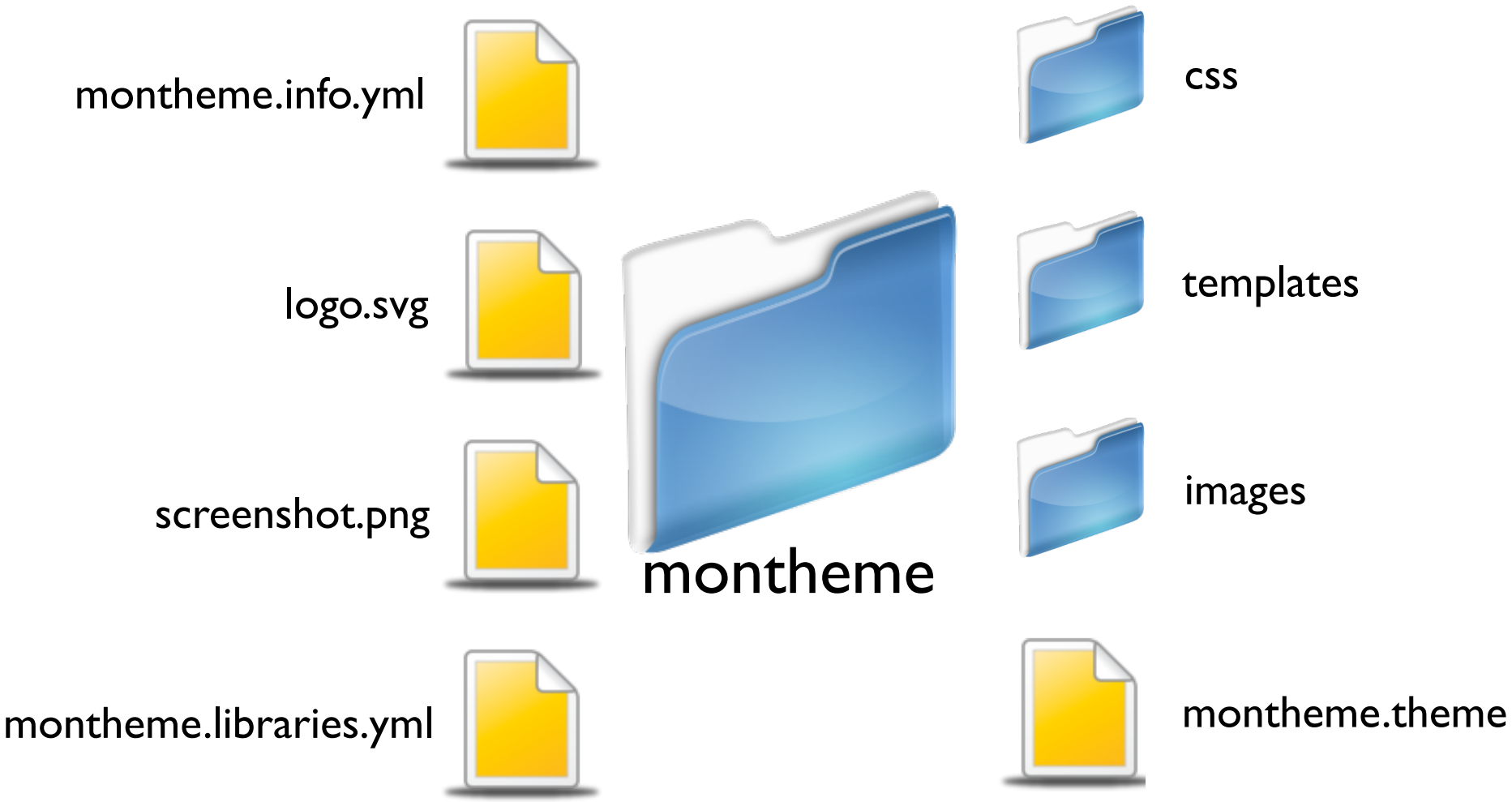
Thème

Introduction

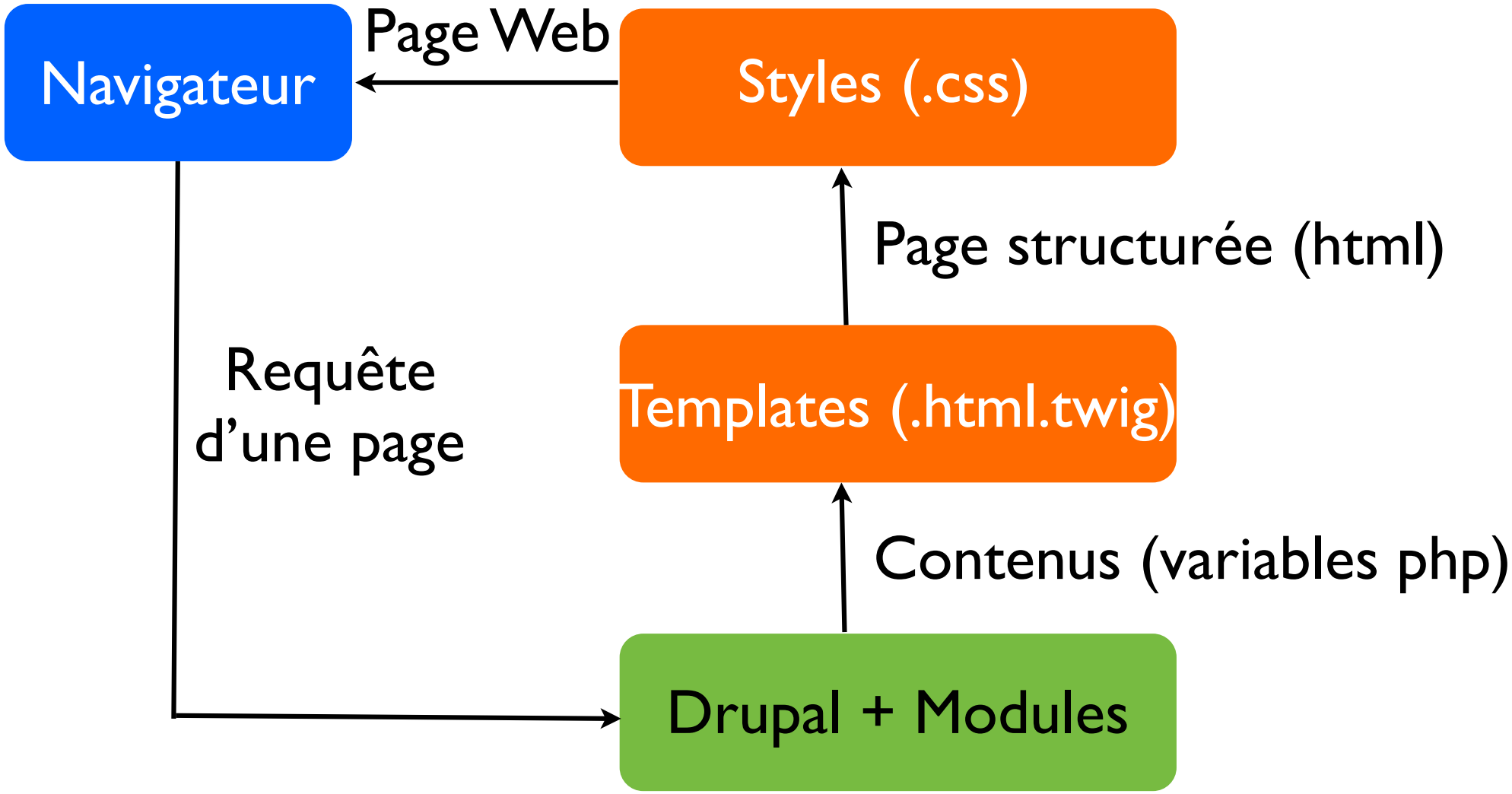
Qu'est-ce qu'un thème ?

- Le **thème** est la **charte graphique**. C'est un ensemble de fichiers (PHP, HTML, CSS, JS YAML…) qui définissent entièrement l'apparence du site.
- Où trouver des thèmes ?
  - Thèmes **natifs** : *Bartik*, *Seven*...
  - Thèmes **contrib** téléchargeables sur Drupal.org. Exemple : *Zen*...
  - Thèmes **payants** par exemple sur *themeforest.net/category/cms-themes/drupal*.
  - Thèmes **custom** que l'on crée soi-même.
- Où mettre ses thèmes ?
  - Les thèmes s'installent physiquement dans le répertoire ***/themes***.
  - Il est possible d'utiliser d'autres dossiers réservés (par exemple ***/sites/all/themes***).
  - On peut créer autant de sous-répertoires que nécessaire; on retrouve souvent (comme pour les modules) les sous-dossiers ***custom*** et ***contrib***.
- Les thèmes disponibles sur un site sont listés sur *Admin > Apparence*.

Anatomie d'un thème



Thème : principes



Plusieurs thèmes par site

- Le back-office et le front-office utilisent par défaut un thème différent :
  - **Thème du back-office** : ***Seven***. Thème sélectionné dans le champ *Thème de l'administration* en bas de la page *Admin > Apparence*.
  - **Thème du front-office** : ***Bartik***. Thème "par défaut" sur la page *Admin > Apparence*.
- Certains thèmes comme ***Adminimal*** (*drupal.org/project/Adminimal_theme*) sont conçus spécifiquement pour le back-office.
- Le module ***Themekey*** (*drupal.org/project/themekey*) permet d'associer un thème différent à certaines URLs du site.

Thème

Les fichiers en détail

Fichier montheme.*info.yml*

```
name: Ive
type: theme
description: 'Example theme.'
version: VERSION
core: '8.x'
regions:
  header: Header
  primary_menu: 'Primary menu'
  secondary_menu: 'Secondary menu'
  help: Help
  page_top: 'Page top'
  page_bottom: 'Page bottom'
  featured_top: 'Featured top'
  breadcrumb: Breadcrumb
  content: Content
  sidebar_first: 'Sidebar first'
  sidebar_second: 'Sidebar second'
  featured_bottom_first: 'Featured bottom first'
  featured_bottom_second: 'Featured bottom second'
  featured_bottom_third: 'Featured bottom third'
  footer_first: 'Footer first'
  footer_second: 'Footer second'
  footer_third: 'Footer third'
  footer_fourth: 'Footer fourth'
  footer_fifth: 'Footer fifth'
```

NE PAS OUBLIER de vider le cache de Drupal après chaque modif. dans le fichier `.info.yml` !

Remarque : la région content est obligatoire.

Fichiers *.css*

- Les fichiers *.css* contiennent les **feuilles de styles** du thème.
- C'est à l'auteur du thème de **créer ses propres fichiers** *.css* et de les référencer en les déclarant comme librairies dans le fichier ***/montheme.libraries.yml***.
- Les fichiers .css peuvent être **nommés librement**, mais il est recommandé de respecter les conventions en vigueur :
  - Mettre tous les fichiers .css dans un répertoire **css/**.
  - Utiliser **style.css** pour les styles généraux, **layout.css** pour les styles liés à la structure de page… (Voir le répertoire ***/core/themes/bartik/css*** pour plus d'infos.)
  - Nommer fichier**-rtl**.css les fichiers correspondant aux langues s'écrivant de droite à gauche.

Fichiers *.js*

- Les fichiers *.js* contiennent les **scripts** du thème.
- C'est à l'auteur du thème de **créer ses propres fichiers** *.js* et de les déclarer comme librairies dans le fichier ***/montheme.libraries.yml***.
- Les fichiers *.js* peuvent être nommés librement.
- Par défaut **aucune librairie javascript n'est chargée pour les utilisateurs anonymes**, pas même *jQuery*. Cette dernière n'est chargée que côté back-office (thème *Seven*).

Fichiers de template

- Les fichiers avec l'extension ***.html.twig*** sont les **templates**.
- Un template contient un mélange de balises **HTML** et de code **Twig**.
- Tout thème Drupal contient généralement ces 3 templates de base :
  - ***page.html.twig*** : définit la structure générale de la page (= emplacement des régions).
  - ***node.html.twig*** : définit la structure des noeuds.
  - ***block.html.twig*** : définit la structure des blocs.
- Par défaut, le template ***page.html.twig*** sert à afficher toutes les pages, ***node.html.twig*** sert à afficher tous les noeuds... Une modification dans un de ces templates impacte donc toutes les pages, tous les noeuds... Nous verrons plus tard comment créer des templates plus spécifiques.
- Ces fichiers sont à placer dans le répertoire ***/templates***.

Fichiers */montheme.theme*

- Le fichier ***/montheme.theme*** permet d'ajouter du **code PHP** additionnel spécifique au thème.
- Ce fichier est **facultatif**, mais il doit être obligatoirement placé à la racine du thème s'il existe.
- Il contient des **fonctions de preprocess** que l'on souhaite étendre (voir plus loin), ainsi qu'un certain nombre de **fonctions de hook**.
- Ce fichier est automatiquement reconnu et chargé par *Drupal*.

Fichiers image

- Drupal n'utilise pas de répertoire spécifique pour stocker les images d'un thème.
- La bonne pratique est de créer un **sous-dossier *./images*** (ou ***./img***…) dans le répertoire du thème, mais on peut s'organiser autrement si nécessaire.
- Dans tous les cas, veiller à utiliser les bons chemins pour référencer les images.

Créer son propre thème

Partir d’un thème existant

Personnalisations de base dans le backoffice

- Dans la section “*Apparence*“ du back-office les personnalisations sont **très limitées**, on peut seulement :
  - Changer le logo.
  - Afficher/Masquer certains éléments comme :
    - Le **Nom du site** qui apparaît dans la région “en-tête“.
    - Les **portraits utilisateur**.
    - Le **slogan du site**.
  - Changer l’**icône de raccourcis**.
- Certains thèmes proposent des personnalisations plus avancées. Par exemple, le thème ***Bartik*** permet de changer le jeu de couleurs du thème via la “roue des couleurs“.

Personnalisations avancées

- La seule solution pour avoir un thème 100% personnalisé, c'est de le créer soi-même !
- Plusieurs options sont possibles pour créer son thème :
  - **1 - Partir de zéro**
    Possible mais fastidieux de se rappeler le nom de tous les templates et des variables qu'ils contiennent...
  - **2 - Partir d'un thème de base**
    Un thème de base est un thème que l'on va étendre en héritant de tout son code. On dispose d'un système de thèmes *parent/enfant*.
    Exemples : ***Omega***, ***Zen***, ***Bootstrap***...
  - **3 - Partir d'un thème "prêt-à-l'emploi"**
    Par exemple un thème acheté sur *themeforest.com*. L'idée ici est de limiter au maximum les personnalisations afin de gagner du temps.

Créer un thème

- Faites une copie du thème par défaut ***Bartik*** (c'est-à-dire du répertoire ***/core/themes/bartik***).
- Placez le répertoire copié dans ***/themes*** et renommer-le ***ive***.
- Renommez ce thème en **Ive** :
  - Renommez le fichier ***/themes/ive/bartik.info.yml*** en ***/themes/ive/ive.info.yml*** (n'oubliez pas d'adapter également le contenu du fichier *.info.yml*).
  - Renommer tous les fichiers dont le nom comporte *bartik* par *ive*.
  - Dans tous les fichiers recherchez toutes les occurrences du mot *bartik* et remplacez-les par *ive* .
- Activez et définir par défaut votre nouveau thème *Ive* sur *Admin > Apparence*.
- Vérifier qu'il n'y a pas d'erreurs en page d'accueil.

Gestion des assets

Feuilles de style CSS
Javascript

Ajouter des assets

- L'ajout de fichier de CSS et de JS se fait via un **système de librairie (bibliothèque)**.
- Il faut tout d'abord déclarer une librairie (ou en utiliser une existante) dans le fichier *montheme.libraries.yml*, puis demander au système de la charger. Ceci peut être fait depuis le fichier *montheme.info.yml*.

/ive.libraries.yml

```
color.preview:
  version: VERSION
  css:
    theme:
      color/preview.css: {}
  js:
    color/preview.js: {}
  dependencies:
    - color/drupal.color
```

/ive.info.yml

```
base theme: classy
description: 'Mon thème.'
package: Core
libraries:
  - ive/color.preview
ckeditor_stylesheets:
  - css/base/elements.css
```

Architecture *SMACSS*

- **S**calable and **M**odular **A**rchitecture for **CSS** (smaccss.com).
- Organisation des feuilles de styles en fonction des sélecteurs :
  - **Base** : reset et normalize.
  - **Layout** : mise en page.
  - **Module** : éléments réutilisables (par exemple pour les formulaires).
  - **State** : styles dépendants de l'état (par exemple les classes *active*).
  - **Theme** : look and feel.
- L'ordre de chargement des différents fichiers en découle.
- ***Remarque** : Drupal 8 utilise non pas le terme module mais component.*

Chargement d'une bibliothèque

- Lorsqu'une librairie est appelée depuis le fichier */montheme.info.yml*, alors elle est chargée sur toutes les pages dès lors que le thème est utilisé.
- Il est possible de charger **sous condition** une bibliothèque. Pour cela plusieurs techniques sont disponibles :
  - depuis un fichier de **template**.
  - en utilisant le **hook_page_attachments_alter()**.
  - en utilisant une **fonction de preprocess**.

Chargement d'une bibliothèque

Chargement d'une librairie depuis un fichier de **template** directement, par exemple dans le fichier gérant l'affichage des blocs :

block.html.twig

```
{{  attach_library('ive/ive') }}
```

Chargement d'une bibliothèque

Chargement d'une librairie depuis le fichier */themes/ive/ive.theme* en utilisant la fonction de hook *hook_page_attachments_alter()*.

```
/**
 * Implements hook_page_attachments_alter().
 *
 * @param array &$attachments
 */
function ive_page_attachments_alter(array &$attachments) {
  $attachments ['#attached']['library'][] = 'ive/ive';
}
```

Chargement d'une bibliothèque

Chargement d'une librairie depuis le fichier */themes/ive/ive.theme* en utilisant une fonction de **preprocess**.

```php
/**
 * Implements hook_preprocess_HOOK() for page templates.
 */
function ive_preprocess_page(&$variables) {
  $variables['#attached']['library'][] = 'ive/ive';
}
```

Ajouter une feuille de style

- Dans le répertoire ***/themes/ive/css***, créez un fichier vierge ***ive.css***, et déclarez le depuis le fichier ***ive.libraries.yml*** en ajoutant une nouvelle librairie ***ive***. Vous utiliserez la syntaxe vue précédemment.
- Demander ensuite le chargement sans conditions de cette librairie depuis le fichier ***/themes/ive/ive.info.yml***.
- Vider le cache de *Drupal*.
- Désactiver l'agrégation CSS sur *Admin > Configuration > Développement > Performance*, puis vérifier dans le code source que le fichier de *CSS* est bien chargé.

Personnaliser la page via HTML/CSS

- Modifiez le thème **lve** de sorte qu'il ressemble à la capture d'écran *d8/DESIGNER/lve/**capture.png***.
- Plus précisément, vous ferez les **modifications suivantes** :
  - Modifier la taille et la couleur de la police du **nom du site** (2rem).
  - Changer la couleur de fond de la région "*Header*" (code couleur #fff).
  - Changer la **couleur** (code couleur *orange*), la **taille** (2rem) et la **police des titres de noeud** (georgia) en page d'accueil.
  - Changer l'apparence des blocs dans les barres latérales.

Modifier la mise en page

Structure des briques de base

Les templates

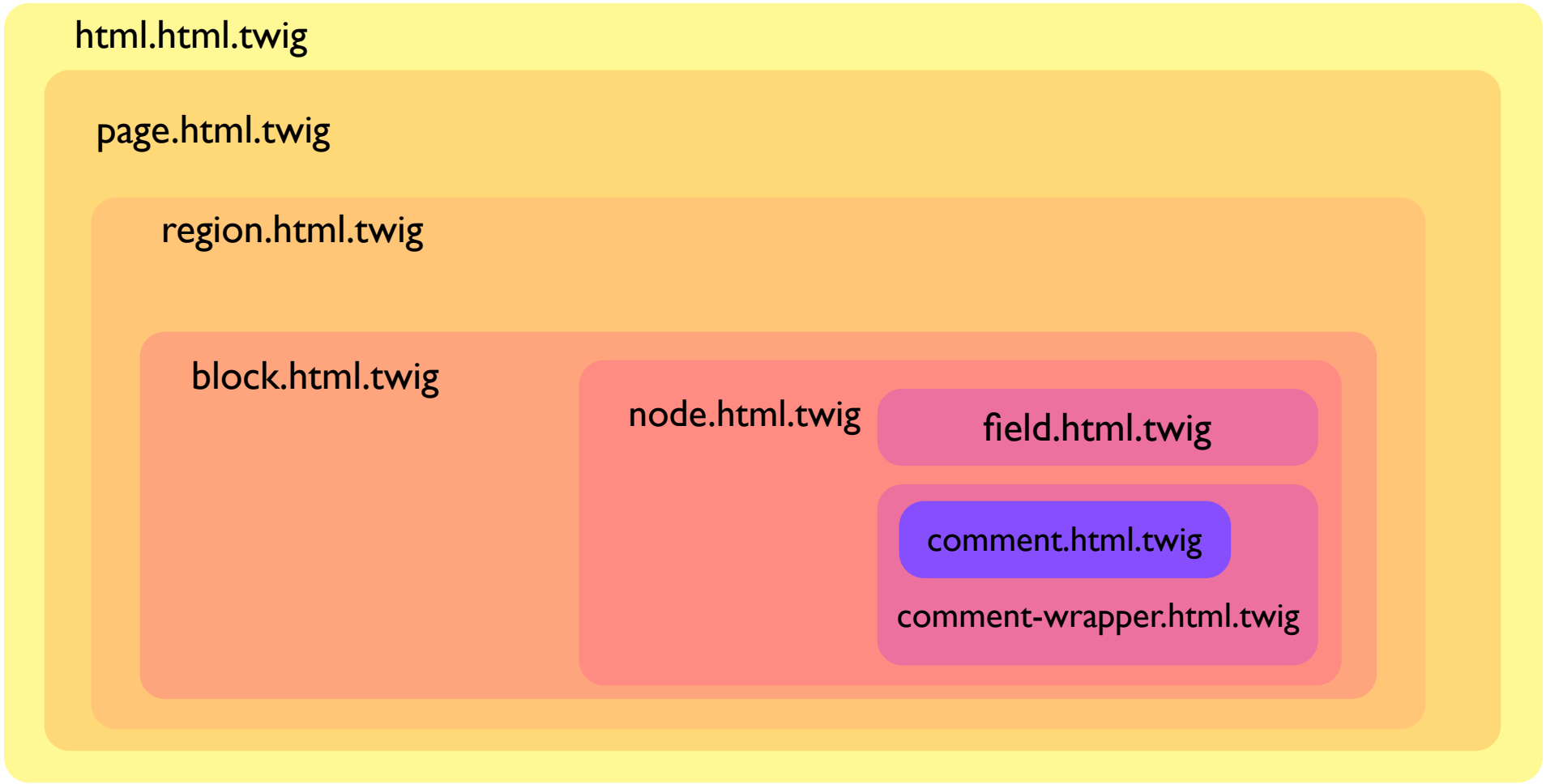
Structure HTML

Template : à quoi ça sert ?

- Les **templates contiennent la structure** des pages et des éléments du site.
- Ce sont des **fichiers html** (balises div, h1, a...) avec du **code Twig** permettant d'afficher les éléments (sous forme de variables) créés par Drupal.
- Chaque type d'élément du site possède son propre template.
- Exemple de code typique dans un template ***Twig*** :

```
{% if page.sidebar_first %}
  <div id="sidebar-first" class="column sidebar">
    <aside class="section" role="complementary">
      {{ page.sidebar_first }}
    </aside>
  </div>
{% endif %}
```

Imbrication des templates



Templates dérivés

- Les templates dérivés sont des **templates utilisés dans un contexte précis**.
- C'est le **nom du template** qui indique à Drupal **quand l'utiliser**.
- Tous ces templates sont **facultatifs** et **reconnus automatiquement** par Drupal en fonction de leurs noms.
- Si des templates ne sont pas présents dans le thème, alors Drupal utilise ceux par défaut.
- On parle de "***template dérivé***" ou "***suggestion de template***".
- Il faut **vider le cache de Drupal** lorsque l'on ajoute des templates au thème.
- Tous les templates dérivés existants de Drupal sont documentés sur *drupal.org/node/2354645*.

Exemples de suggestion de template

- Template de page *page.html.twig* :
  - ***page--chemin--interne.html.twig*** sera utilisé lorsque l'on appelle **chemin/interne**
  - Ex. : ***page--front.html.twig*** cible la page d'accueil.
  - Ex. : ***page--chemin.html.twig*** cible toutes les pages dont le chemin commence par ***chemin***.
  - Ex. : ***page--node--240.html.twig*** cible la page du noeud portant l'identifiant 240.
- Template de noeud *node.html.twig* :
  - ***node--type-de-noeud.html.twig*** cible les noeuds de type **type-de-noeud**.
  - Ex. : ***node--article.html.twig*** cible tous les noeuds de type *article*.
- Ces suggestions de templates sont **les plus utilisées**.

Twig

Twig

- **Twig** est capable d'afficher le contenu de n'importe quel type de variable (chaine, tableau ou objet).
- Il est possible de faire des boucles, des conditions…
- Chaque variable peut être filtrée avant d'être affichée.
- Les chaines peuvent être rendues traduisibles (la traduction est alors faite via le back-office de *Drupal*).

```
{% if page.sidebar_second %}
  <div id="sidebar-second" class="column sidebar">
    <aside class="section" role="complementary">
      {{ page.sidebar_second }}
    </aside>
  </div>
{% endif %}
```

Twig - les bases

- Affiche le contenu de la variable *var* : **{{ var }}**.
- Affiche le contenu de la propriété *title* de la variable *node* : **{{ node.Title }}**.
- Fonction : **{% if var %} {% endif %}**.
- Commentaires : **{# commentaire #}**
- Traduction : **{% trans %} chaine {% endtrans %}**.
- Fonctions spécifiques :
  - **{{ url('route_name') }}**.
  - **{{ path('entity.node.canonical', {'node': node.id}) }}**.
  - le module ***Twig tweak*** propose encore davantage de fonctions.

Twig - les filtres

- Date : **{{ datelformat_date('medium') }}**
- Chaine :
  - Echappement : **{{ texte }}** (par défaut).
  - Interprétation du contenu de la variable : **{{ textelraw }}** (attention !).
  - Placeholder : **{{ textlplaceholder }}**.
- Exclusion : **{{ contentlwithout('links') }}**.
- **{{ stringllower }}** (upper également).
- **{{ class_namelclean_class }}**.
- **{{ id_namelclean_id }}**.
- Les filtres natifs de ***Twig*** sont disponibles sur *Twig.sensiolabs.org/doc/filters*.

Twig - Debug

- Pour activer les outils de debug de ***Twig***, dupliquez le fichier ***/sites/default/default.services.yml*** en le nommant ***/sites/default/services.yml***.
- Dans ce fichier trouver les variables ***debug***, ***auto_reload*** et ***cache*** et adapter leurs valeurs.
- **Vider les caches**. Le fichier est chargé automatiquement par le système. Vérifier que vous voyez bien les commentaires ajoutés pat *Twig*.

```
  # @default false
  debug: true
  # Twig auto-reload:
  #
  # Automatically recompile Twig templates whenever the source code changes.
  # If you don't provide a value for auto_reload, it will be determined
  # based on the value of debug.
  #
  # Not recommended in production environments
  # @default null
  auto_reload: null
  # Twig cache:
  #
  # By default, Twig templates will be compiled and stored in the filesystem
  # to increase performance. Disabling the Twig cache will recompile the
  # templates from source each time they are used. In most cases the
  # auto_reload setting above should be enabled rather than disabling the
  # Twig cache.
  #
  # Not recommended in production environments
  # @default true
  cache: true
factory.keyvalue:
```

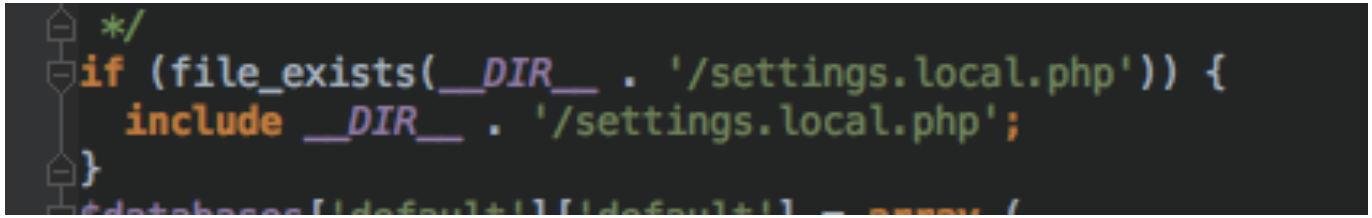
Twig - Debug

- L'option de debug ajoute dans les fichiers de template des **commentaires** (emplacement du fichier de template, suggestion de nom de fichier…).

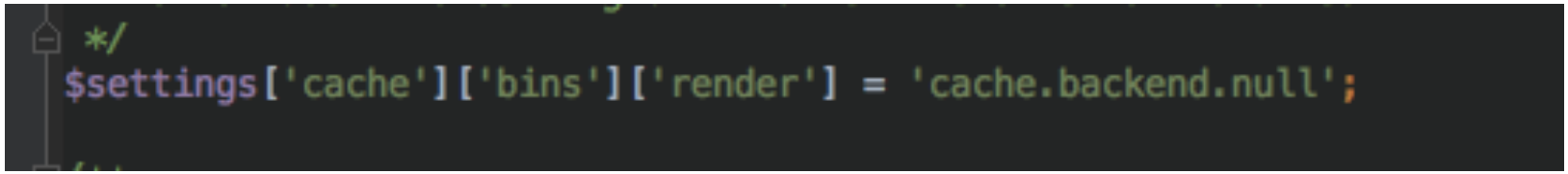
```
▼<div class="views-row">
    <!-- THEME DEBUG -->
    <!-- THEME HOOK: 'node' -->
    <!-- FILE NAME SUGGESTIONS:
       * node--view--frontpage--page-1.html.twig
       * node--view--frontpage.html.twig
       * node--1--teaser.html.twig
       * node--1.html.twig
       * node--article--teaser.html.twig
       * node--article.html.twig
       * node--teaser.html.twig
       x node.html.twig
    -->
    <!-- BEGIN OUTPUT from 'themes/ive/templates/node.html.twig' -->
  ▼<article data-history-node-id="1" data-quickedit-entity-id="node/1" role="article
```

Cache

- Certains templates (par exemple *node.html.twig*) sont mis en cache même si l'option est désactivée au niveau de ***Twig***. Drupal 8 met en cache toutes les entités par défaut.
- Pour désactiver le cache correspondant copier le fichier ***/sites/example.settings.local.php*** dans */sites/default/* et renommer le ***settings.local.php***.
- Dans le fichier ***/sites/default/settings.php*** décommenter les lignes correspondantes à la capture ci-dessous.



- **Vider les caches**. puis décommenter la ligne ci-dessous du fichier ***/sites/default/settings.local.php***.



Modifier un template

- **Créer un bloc** "*Drupal 8*" (via *Admin > Structure > Mise en page de bloc > Ajouter un bloc*) contenant un texte de présentation de Drupal 8 de votre choix et placer le dans la barre latérale de droite de votre thème.
- **Créer un template dérivé** de sorte que ce bloc affiche l'image *d8/designer/ive/drupal8.png* à la place du titre. Le chemin de l'image est ***/themes/ive/images/drupal8.png***. Vous disposez de la variable *directory*, contenant le chemin jusqu'au thème courant (par exemple *themes/ive*).
- ***Remarque** : indiquer comme chemin de base un « / » est une mauvaise pratique, car le site n'utilise pas forcément un virtual host. Nous verrons par la suite comment procéder…*
- Quelle autre technique aurait-on pu utiliser ici ? D'après vous quelle est la meilleure ?

Ajouter des suggestions de template

Définition des suggestions de template

- Pour chaque template de base, on a un certain nombre de suggestions/dérivés suivants des patterns bien précis.
- Il est possible d'**ajouter des suggestions supplémentaires** via une fonction de hook.
- **Ex.** : déclaration des templates du module *node*.

```php
/**
 * Implements hook_theme_suggestions_HOOK().
 */
function node_theme_suggestions_node(array $variables) {
  $suggestions = array();
  $node = $variables['elements']['#node'];
  $sanitized_view_mode = strtr($variables['elements']['#view_mode'], '.', '_');

  $suggestions[] = 'node__' . $sanitized_view_mode;
  $suggestions[] = 'node__' . $node->bundle();
  $suggestions[] = 'node__' . $node->bundle() . '__' . $sanitized_view_mode;
  $suggestions[] = 'node__' . $node->id();
  $suggestions[] = 'node__' . $node->id() . '__' . $sanitized_view_mode;

  return $suggestions;
}
```

Ajouter ses propres suggestions

- Tous les modules et thèmes peuvent ajouter des suggestions de thème. Les fonctions à utiliser pour ajouter des suggestions de templates au hook de thème **HOOK** sont les suivantes :
  - **MODULE**_theme_suggestions_**HOOK**(array $variables)
  - **AUTREMODULE**_theme_suggestions_alter(array &$suggestions, array $variables, $hook)
  - **AUTREMODULE**_theme_suggestions_**HOOK**_alter(array &$suggestions, array $variables)
  - **THEME**_theme_suggestions_alter(array &$suggestions, array $variables, $hook)
  - **THEME**_theme_suggestions_**HOOK**_alter(array &$suggestions, array $variables)
- Le thème est sollicité en dernier afin de pouvoir prendre la main indépendamment de qui a été déclaré auparavant.

Ajout de templates

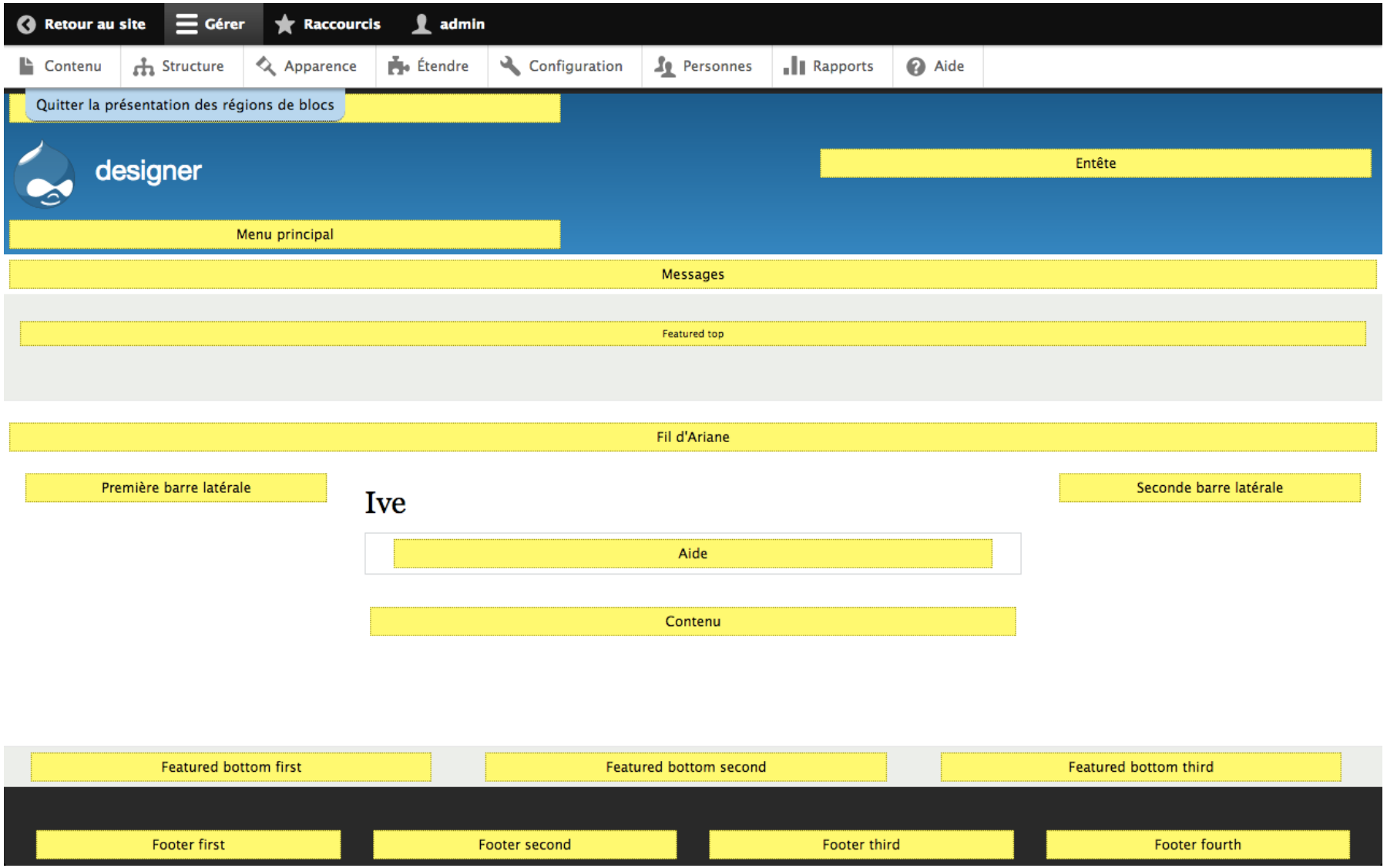
- On désire avoir des templates de pages différents pour chaque type de contenu.
- Il faut donc ajouter cette suggestion pour les templates de page, c'est à dire dans la fonction *hook_theme_suggestions_page_alter().*
- On souhaite pouvoir créer les templates de type ***page—nodetype—NODETYPE.html.twig***. pour chaque type de noeud *NODETYPE*.
- Pour récupérer l'objet noeud courant vous utiliserez le code suivant :

```
$node = \Drupal::routeMatch()->getParameter('node')
```

Structure des pages

Personnaliser la présentation

Structure de la page



Template *page.html.twig*

Ce fichier contient les **régions du thème** (il y en a 17 dans le thème par défaut *Bartik*) :

- header
- sidebar_first
- content
- sidebar_second
- footer_first
- ...

```
<header id="header" class="header" role="banner" aria-label="{{ 'Sit
  <div class="section layout-container clearfix">
    {{ page.secondary_menu }}
    {{ page.header }}
    {{ page.primary_menu }}
  </div>
</header>
{% if page.highlighted %}
  <div class="highlighted">
    <aside class="layout-container section clearfix" role="complemen
      {{ page.highlighted }}
    </aside>
  </div>
{% endif %}
{% if page.featured_top %}
  <div class="featured-top">
    <aside class="featured-top__inner section layout-container clear
      {{ page.featured_top }}
    </aside>
  </div>
{% endif %}
<div id="main-wrapper" class="layout-main-wrapper layout-container c
  <div id="main" class="layout-main clearfix">
    {{ page.breadcrumb }}
    <main id="content" class="column main-content" role="main">
      <section class="section">
        <a id="main-content" tabindex="-1"></a>
        {{ page.content }}
      </section>
    </main>
    {% if page.sidebar_first %}
      <div id="sidebar-first" class="column sidebar">
        <aside class="section" role="complementary">
          {{ page.sidebar_first }}
        </aside>
      </div>
    {% endif %}
    {% if page.sidebar_second %}
      <div id="sidebar-second" class="column sidebar">
        <aside class="section" role="complementary">
          {{ page.sidebar_second }}
        </aside>
      </div>
```

Créer ses propres régions

Déclarer une région dans le fichier *.info.yml*.

```
regions:
  header: Header
  primary_menu: 'Primary menu'
  secondary_menu: 'Secondary menu'
  page_top: 'Page top'
  page_bottom: 'Page bottom'
  highlighted: Highlighted
  featured_top: 'Featured top'
```

Ajouter le code suivant dans le template *page.html.twig*.

```
{% if page.highlighted %}
  <div class="highlighted">
    <aside class="layout-container section clearfix" role="complementary">
      {{ page.highlighted }}
    </aside>
  </div>
{% endif %}
```

Ajouter une région

- **Ajouter une région** "*Drupal8*" en la déclarant dans le fichier ***ive.info.yml*** puis en copiant et modifiant le code d'une autre région (par exemple la région "*Featured Top*") dans le fichier *page.html.twig*.
- Faites en sorte qu'elle se place en dessous de la région "*Contenu*" en occupant toute la largeur disponible. Vous devrez ajouter quelques styles propres à cette région dans le fichier ***ive.css***.
- Utiliser maintenant cette région pour afficher le bloc "*Drupal 8*".

Créer un template dérivé

- Faites en sorte que **sur le noeud de type page statique** que vous avez créé précédemment - et juste sur cette page - **on voie uniquement** :
  - le logo du site
  - le titre de la page
  - le contenu de la page
- **On ne doit pas voir** la colonne latérale de droite, le pied de page et la région Drupal8. Vérifier que ce template n'est utilisé que sur cette page.

Structure des noeuds

Personnaliser la présentation des champs

Field - Rappels

- Les modules ***Field*** permettent d'**ajouter ses propres champs dans un type de contenu**.
  - Exemple : on crée un type de contenu Recette et on y ajoute les champs **Ingrédients**, **Photo**...
- Pour **personnaliser l'apparence** de ces champs :
  - Utiliser en priorité le **backoffice de *Field***, qui propose déjà de nombreux réglages.
  - Ensuite, compléter ces personnalisations avec **CSS**, qui ne nécessite pas de toucher au HTML de Drupal.
  - Enfin, **si vous n'êtes toujours pas arrivés à vos fins**, utiliser les techniques qui suivent...

Field - Ce que l'on peut faire dans l'admin

- Pour chaque type de contenu on peut régler l'affichage des champs sur ***admin/structure/types/manage/content-type/display*** où content-type est le nom machine du type de contenu.
- Il est possible de modifier l'ordre des champs, l'apparence des étiquettes, le format du champ...
- Les réglages disponibles varient en fonction du type de champ :
  - Les champs **Texte** peuvent être coupés.
  - Les champs **Image** peuvent être affichés sous forme d'image, d'URL, de fichier...
  - Les champs **Date** peuvent utiliser un format différent.
- Ces réglages peuvent être différents selon le mode d'affichage (modes "*Par défaut*" et "*Accroche*"). Cela permet par exemple de masquer un champ en mode "Accroche" et de l'afficher en mode "*Par défaut*" (noeud intégral).

Personnaliser l'affichage des champs image

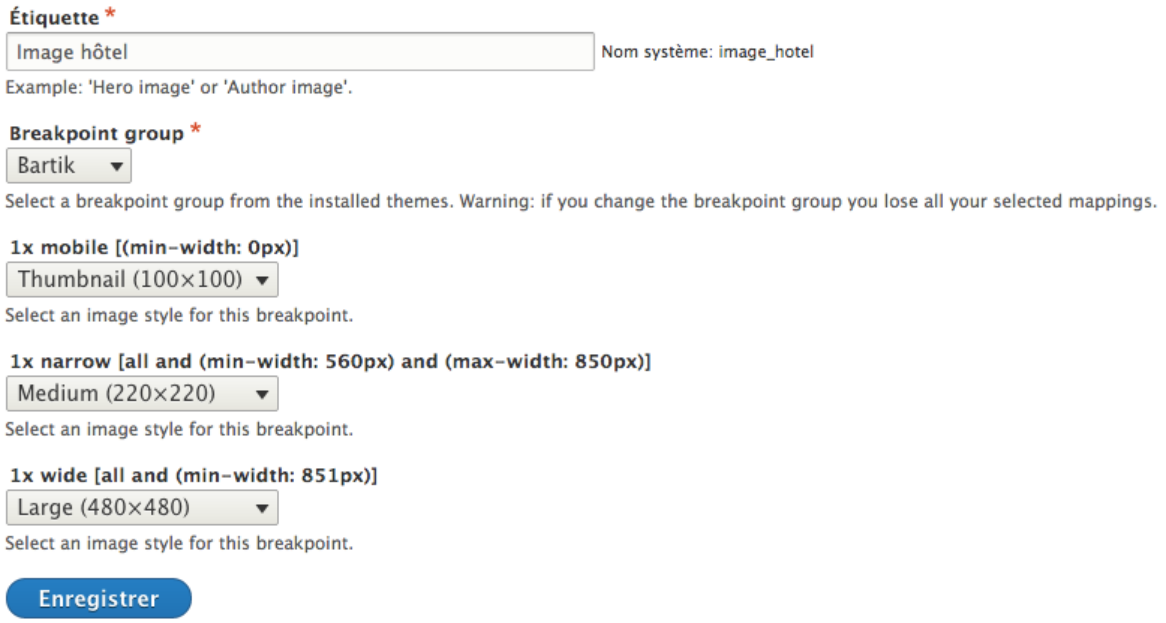
- Les images transmises par les utilisateurs ne sont **jamais** aux bonnes dimensions.
- Le module ***Image*** (core) va nous permettre de prédéfinir des dimensions, sous forme de **styles d'image**, par exemple :
  - **Vignette** : 70px x 110px (mise à l'échelle).
  - **Normal** : 200px x 300px (mise à l'échelle et recadrage).
- Ces dimensions pourront ensuite être appliquées automatiquement aux champs *Image* depuis l'écran "*gérer l'affichage*" vu précédemment.

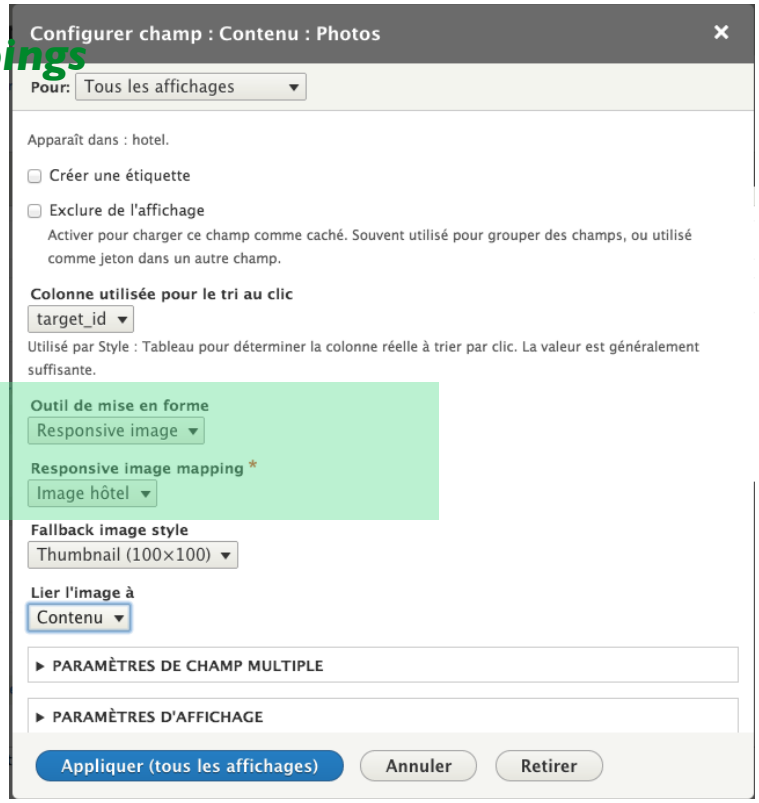
Image responsive

- Dans le cadre d'une approche *Responsive Design*, il est important que les images soient optimisées en dimension (et donc en poids) en fonction du contexte. On charge une image légère sur mobile, tandis que sur un ordinateur de bureau on privilégie une image haute résolution.
- Les différents **contextes** correspondent aux **breakpoints** (points de rupture) défini par le thème (***Bartik*** par défaut). Chaque breakpoints est une série de media queries. Ex. : '*all and (min-width: 560px) and (max-width: 850px)*'.
- Si l'on a ainsi 3 breakpoints, il faut créer 4 styles d'image.
- Le module ***Responsive image*** (non installé par défaut) permet d'utiliser les différents breakpoints dans l'interface afin de contrôler l'apparence des images.

Image responsive

Admin > Configuration > Média > Responsive image mappings





Redimensionner les images

- Activer le module ***Responsive Image***.
- Sur *Admin > Configuration > Média >* ***Styles d'images***, ajoutez les trois styles suivants :
  - ***Ive mobile*** : effets "*Echelle*" (largeur = 180px) et "*Désaturer*".
  - ***Ive narrow*** : effet "*Echelle*" (largeur = 280px).
  - ***Ive wide*** : effet "*Echelle*" (largeur = 425px).
- *Notez que chaque style peut cumuler plusieurs effets.*
- Sur *Admin > Configuration > Média >* ***Styles d'images adaptatifs***, ajoutez un Style d'Image Adaptif :
  - ***Label*** : *Image article*.
  - ***Breakpoint group*** : *Ive*.
- *Sur Admin > Structure > Types de contenu > Article >* ***Gérer l'affichage***, utilisez l'image responsive **Image article** pour le champ *Image*.

Field - pour aller plus loin

- Il est impossible dans l'admin de manipuler les champs, pour par exemple présenter les champs d'images dans une grille.
- On est obligé de **modifier le code du template** correspondant.
- Les réglages de l'admin sont toujours pris en compte. Par exemple, les styles d'image choisis seront ceux utilisés même si l'on modifie le template.
- ***Remarque*** *: le module* ***Display Suite*** *permet d'utiliser différents types de présentation (deux colonnes, trois colonnes avec en-tête et pied de page...) pour les champs, et ce dans l'interface de Drupal.*

Où sont les champs ?

- Dans le fichier de template des noeuds (*node.html.twig*), on n'a pas directement accès aux champs.
- C'est la variable ***content*** qui **contient tous les champs** que l'on affiche, et en particulier ceux ajoutés via ***Field***.
- Comment faire pour récupérer les champs un par un afin de les présenter différemment ?

Variable *content*

- Tout le contenu d'un noeud est affiché grâce à la ligne :

  {{ contentlwithout('comment', 'links') }}

- Pour connaitre le contenu de *content* on utilise la fonction **kint() :**

  {{ kint(content) }}

- On découvre alors l'ensemble des champs et leurs propriétés qui sont affichés en haut de page sous forme de *var_dump()*.
- On peut afficher séparemment chacun d'entre eux avec le code suivant :

  {{ content.nom_du_champ }}

  {{ node.createdtime }}

Modifier le template des articles

- Utiliser le template adéquat afin d'apporter les améliorations suivantes à l'affichage des noeuds de type article en mode ***Accroche*** :
  - les articles sont présentés en deux colonnes avec un en-tête.
  - les tags sont en en-tête tandis que la photo est en colonne de gauche et le corps de l'article en colonne de droite.
  - Vous utiliserez 2 " " avec les propriétés CSS adéquates.
- ***Remarque*** *: En mode* ***Par Défaut*** *les champs sont présentés classiquement (pour l'instant...).*

Récapitulons

- Il y a plusieurs techniques pour personnaliser les champs *Field*, de la plus simple à la plus compliquée :
  - Trouver **le bon réglage** sur l'onglet "Gérer l'affichage".
  - Utiliser **CSS** (comme toujours !).
  - Utiliser les **Styles d'images** pour les champs Image.
  - Utiliser les **Templates** pour contrôler manuellement comment sont affichés les champs ***Field***.
- ***N.B. :*** *Ne pas confondre la personnalisation des champs* ***Field****, qui sont le* ***contenu du noeud****, et la personnalisation de la* ***structure générale du noeud****.*

Modifier les éléments de Drupal

Fil d'Ariane
Texte "Soumis par AUTEUR, le 1er janv. 1970"

Surcharge de templates

- ***Drupal 8*** formate les données pour les afficher en utilisant des templates.
- Lorsque l'on veut modifier ce formatage on peut surcharger ces templates en **copiant les fichiers correspondants dans notre thème**.
- Ce sont nos surcharges de thème qui sont alors utilisées en priorité par ***Drupal***.
- **Exemple** : si l'on souhaite modifier le formatage de l'icon de flux RSS, il suffit de copier le template *image—feed-icon.html.twig* dans le thème, puis de faire les modifications adéquates dans ce fichier.

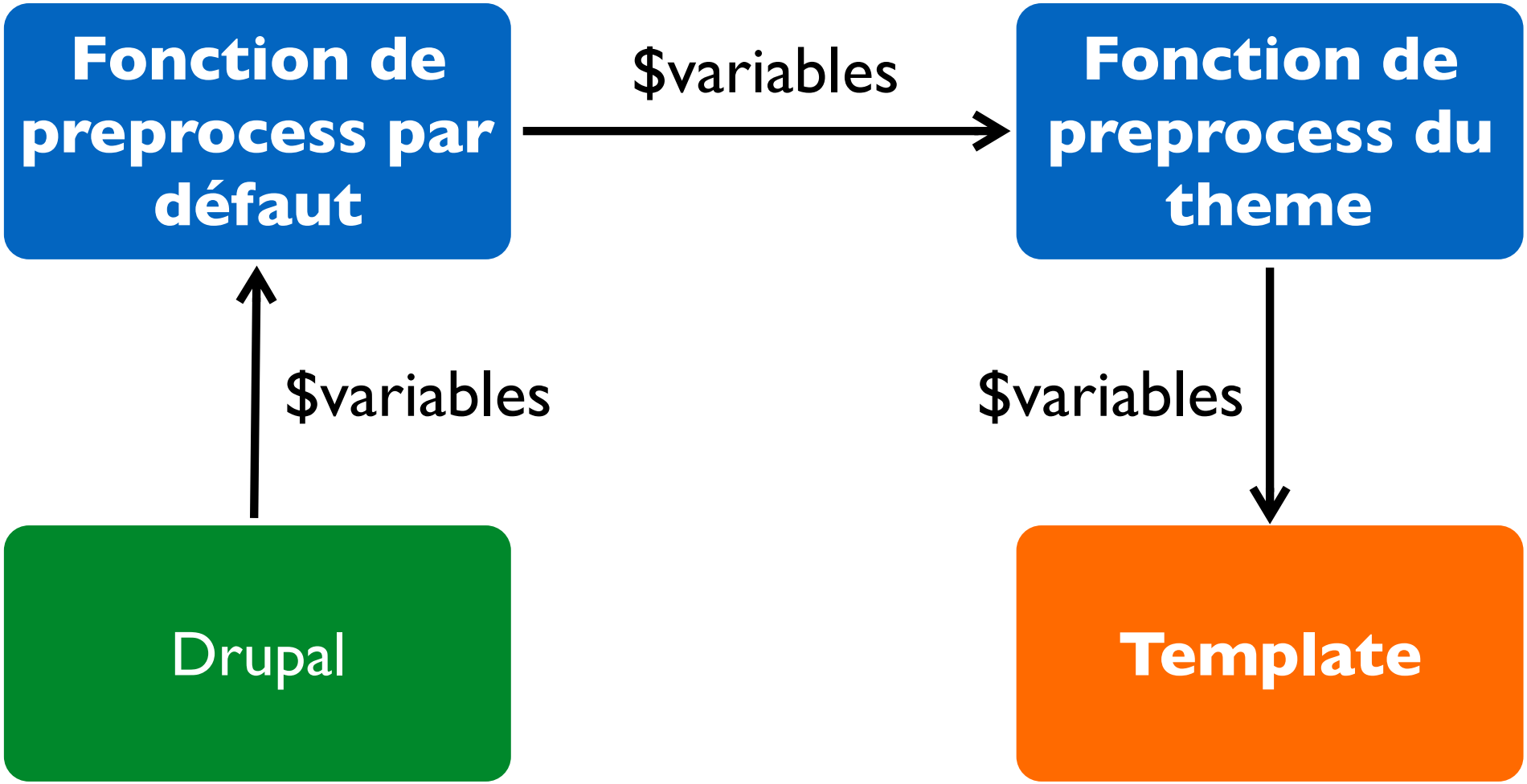
Modifier l'astérisque indiquant un champ obligatoire

- Sur *Admin > Apparence* faites en sorte que le thème ***Ive*** soit utilisé pour l'édition des noeuds.
- Allez sur le formulaire d'édition d'un noeud.
- Grâce au mode ***Debug*** de ***Twig***, identifiez le template de thème responsable du formatage de l'astérisque rouge indiquant un champ obligatoire.
- Surchargez ce template de sorte qu'à la place de l'astérisque on ait le texte "*(obligatoire)*".
- ***Remarque*** *: vous ferez en sorte que votre texte soit traduisible !*

Fonction de preprocess

- Pour chaque template on dispose de variables que l'on affiche avec le formatage voulu.
- Ces **variables sont créées en amont des templates** dans lesquelles elles sont utilisées.
- On dispose de **fonctions** dites **de preprocess** qui permettent de fabriquer les variables qui sont utilisées dans les templates.
- Il est possible d'étendre ces fonctions pour modifier les variables envoyées aux templates ou en créer de nouvelles.
- Pour ajouter une fonction de preprocess il faut la nommer comme suit : **THEME_preprocess_HOOK()**. Par exemple ***ive_preprocess_block()***.

Fonction de preprocess



Fonctions de preprocess

- Différentes fonctions de preprocess peuvent être appelées permettant à tous les modules et thèmes de modifier les variables passées aux templates. Ces fonctions sont les suivantes pour le hook de thème **HOOK** :
  - **template**_preprocess_**HOOK**(array &$variables)
  - **AUTREMODULE**_preprocess(array &$variables, $hook)
  - **AUTREMODULE**_preprocess_**HOOK**(array &$variables)
  - **THEME**_preprocess(array &$variables, $hook)
  - **THEME**_preprocess_**HOOK**(array &$variables)

Utiliser une fonction de preprocess

- Faites en sorte que le texte "*Soumis par Auteur le sam., 11/19/2011 - 15:30*" sous chaque noeud de type Article devienne "***Article écrit par Auteur, le samedi 19 novembre 2011 à 15h30***". Il vous faut :
  - **Trouver la variable** responsable de l'affichage des infos auteur et date sous les noeuds dans le template de noeud.
  - Déterminer la **fonction de preprocess** qui formate cette variable.
  - Ajouter dans le fichier *ive.theme*. les modifications apportées à la fonction de base.
- Reprenez le template de block ajouté précédemment et modifier le chemin de l'image afin qu'il soit indépendant de l'emplacement du thème (un thème n'est pas obligatoirement placé dans le répertoire */themes*). Vous utiliserez la fonction de preprocess *ive_preprocess_block()*.
- ***Remarque** : en PHP vous obtenez le chemin absolu du thème avec les fonctions base_path() et drupal_get_path('theme', 'mon_theme')*.

Module *Layout Discovery*

Module *Layout Discovery*

- Le module ***Layout Discovery*** permet de déclarer de nouvelles présentations (templates) qui sont ensuite utilisables via le back-office.
- Ces templates sont réutilisables dans différentes situations. On peut par exemple afficher différentes entités en utilisant le même fichier de template.
- Des modules comme ***Panels*** ou ***Display Suite*** reposent sur *Layout Discovery*.
- Cette approche du templating est différente de celle native de *Drupal*. Ici un template n'est pas affecté par défaut à un type d'élément (page, bloc, noeud, champ…) mais est utilisable dans de multiples situations. Cela permet de mutualiser les templates.

Création d'un layout

- Afin que le système découvre les différents layouts que l'on souhaite pouvoir utiliser, il faut créer les fichiers suivants :
  - ***/mon_theme.layouts.yml*** : déclaration de la liste des layouts que l'on souhaite ajouter au système.
  - ***/layouts/mon-fichier.html.twig*** : template associé.
  - ***/layouts/mon-fichier.css*** : CSS associé.
- Il existe d'autres moyens de déclarer un layout : *hook_theme()* ou *classes PHP* (utilisant le système de plugin avec les annotations).

Déclaration du layout

Le fichier */ mon_theme.layouts.yml* permet de déclarer :

- le **nom machine** du layout.
- le **nom** pour le back-office.
- la **catégorie** pour le back-office (permet de regrouper les différents layouts).
- le **nom du template** associé.
- La **librairie** associée.
- les différentes **régions** du template.

```yaml
layout_twocol:
  label: 'Two column'
  path: layouts/twocol
  template: layout--twocol
  library: layout_discovery/twocol
  category: 'Columns: 2'
  default_region: left
  regions:
    top:
      label: Top
    left:
      label: Left
    right:
      label: Right
    bottom:
      label: Bottom
```

Création du template associé

Le fichier ***/layouts/twocol/layout—twocol.html.twig*** permet de formater les différentes régions du layout déclarées précédemment :

- **top**.
- **left**.
- **right**.
- **bottom**.

```
{%
  set classes = [
    'layout',
    'layout--twocol',
  ]
%}
{% if content %}
  <div{{ attributes.addClass(classes) }}>
    {% if content.top %}
      <div class="layout__region layout__region--top">
        {{ content.top }}
      </div>
    {% endif %}

    {% if content.first %}
      <div class="layout__region layout__region--first">
        {{ content.first }}
      </div>
    {% endif %}

    {% if content.second %}
      <div class="layout__region layout__region--second">
        {{ content.second }}
      </div>
    {% endif %}

    {% if content.bottom %}
      <div class="layout__region layout__region--bottom">
        {{ content.bottom }}
      </div>
    {% endif %}
  </div>
{% endif %}
```

Création du CSS associé

Ce fichier permet d'ajouter les styles CSS qui sont chargés avec le template via la librairie associée *mon_theme/layout_twocol* (à déclarer comme n'importe quelle autre librairie).

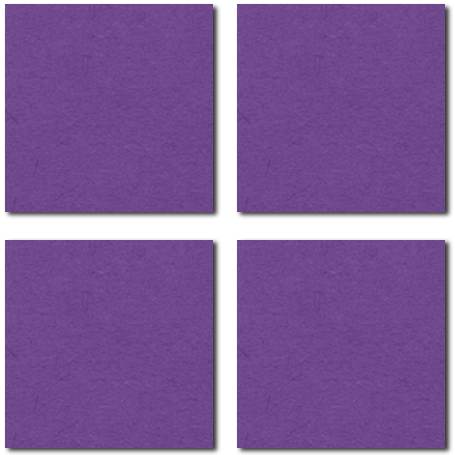
```
.layout--twocol {
  display: flex;
  flex-wrap: wrap;
}

.layout--twocol > .layout__region {
  flex: 0 1 100%;
}

@media screen and (min-width: 40em) {
  .layout--twocol > .layout__region--first,
  .layout--twocol > .layout__region--second {
    flex: 0 1 50%;
  }
}
```

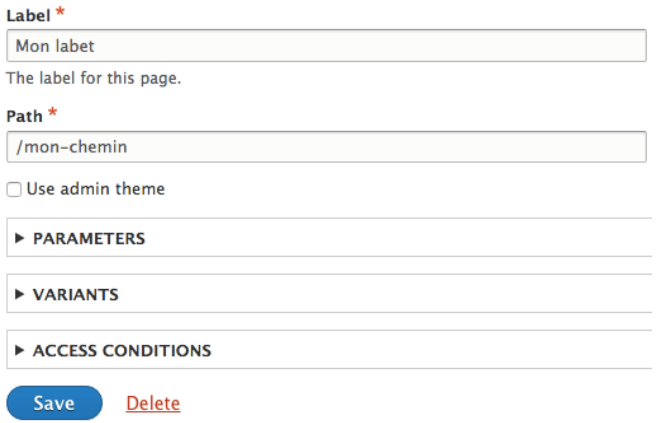
Créer un layout

- Activer le module ***Layout Discovery*** et ***Field Layout***.
- Ajouter à votre thème un *layout* nommé *Four Squares* définissant 4 régions (voir capture ci-dessous), en créant les fichiers vus précédemment.
- Ajouter des champs au type de contenu *Page de base*, puis utiliser le *layout* pour les afficher.



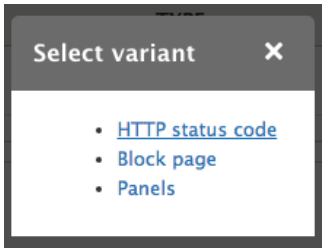
Module *Page Manager*

- Le module ***Page Manager*** permet de créer de nouvelles pages ou de prendre la main sur des pages existantes du système, par exemple les pages affichant les noeuds.
- Chaque page est composée de :
  - ***Paramètres*** : disponibles en fonction du contexte de la page. Par exemple, sur une page d'affichage de noeud, le paramètre est le noeud lui-même.
  - ***Variantes*** : contenu de la page avec d'éventuelles conditions d'affichage. Une variante permet d'indiquer ce que l'on souhaite afficher (blocs) et comment le présenter.
  - ***Conditions d'accès*** : restrictions ou non pour accéder à la page.



Page Manager - variantes

- Une même page peut afficher du contenu différent en fonction de certaines conditions : le rôle de l'utilisateur, la langue…
- On peut donc créer autant de variantes d'une même page que nécessaire.
- Chaque variante peut être :
  - Un ***code de status HTTP***.
  - Un ***Block page*** (proposant 2 régions superposées).
  - Un ***Panel*** (proposant de nombreux layouts différents via le module *Layout Plugin*).
  - Autre suivant les modules installés.
- En choisissant une variante *Panels*, on peut alors utiliser les templates proposés par le module ***Layout Plugin***.

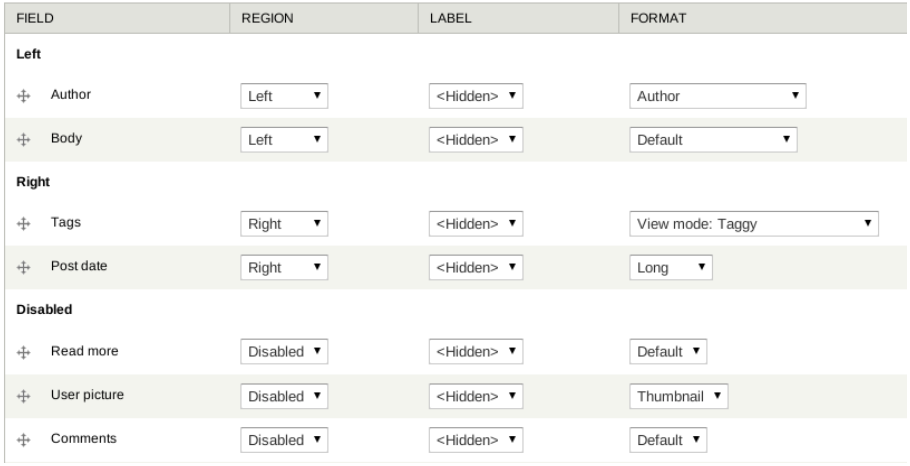


Créer une page avec le module *Page Manager*

- Activer les modules ***Page Manager***, ***Page Manager UI***, ***Token*** et ***Panels***.
- Aller sur *Admin* > *Structure* > *Pages*, et ajouter une nouvelle page **Page**:
  - Label : *Page custom*.
  - Path : */page-custom*.
- Ajouter une variante de type *Panels* et renseigner les différents champs (nom de la variante, titre de page…), en particulier choisir le layout *Four Squares*.
- Choisir un bloc pour chacune des régions.
- Enregistrer la nouvelle page.
- Aller vérifier sur le chemin */page-custom* à quoi ressemble la page.

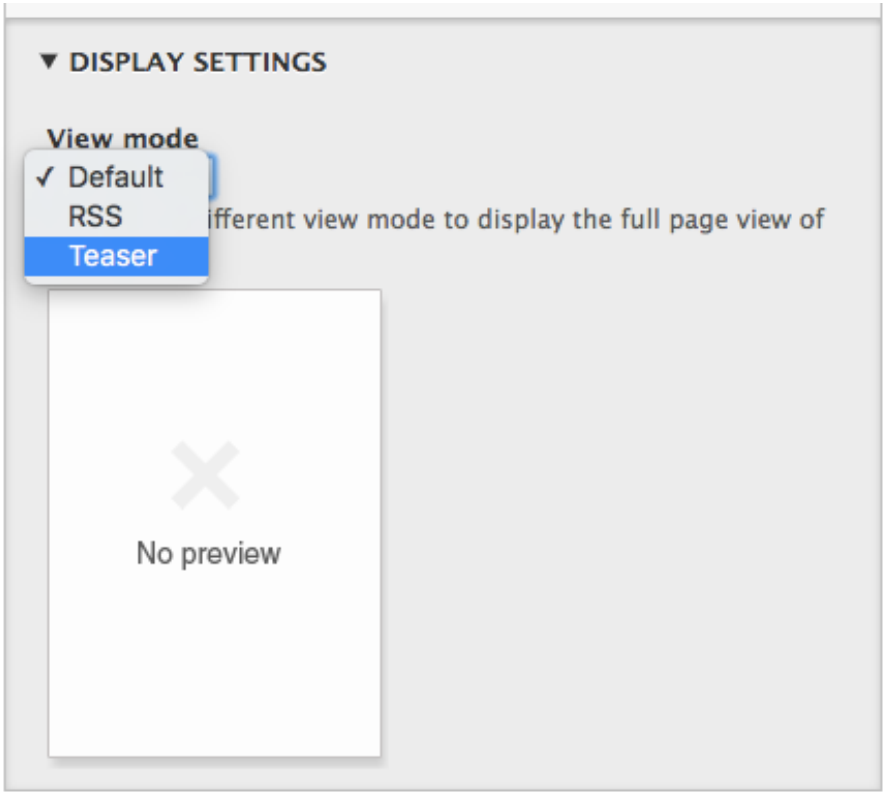
Module *Display Suite*

- Le module ***Display Suite*** permet de reprendre la main sur l'affichage des champs de n'importe quelle entité de contenu *fieldable* (noeuds, taxonomie, utilisateur…).
- Ce module utilise également le ***Layout Plugin***, ce qui permet donc d'utiliser n'importe quel template déclaré par un thème ou un module tierce partie.



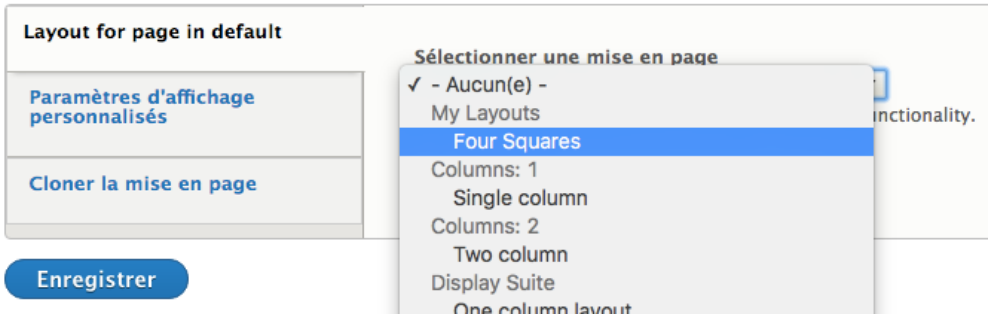
Module *Display Suite Switch View Mode*

- Le module additionnel ***Display Suite Switch View Mode*** ajoute la possibilité de modifier le mode d'affichage par noeud.
- Ainsi il est possible de créer des modes d'affichages (Admin > Structure > Mode d'affichage) utilisant des layouts différents et laisser l'utilisateur final choisir la présentation souhaitée.
- On dispose de permissions pour chaque type de contenu.



Modifier la présentation des pages de base

- Les différents types de noeuds peuvent utiliser ***Display Suite***, afin d'avoir une plus grande flexibilité dans l'affichage des champs.
- Activer le module ***Display Suite***.
- Sur *Admin > Structure > Types de contenu > Page de base > Gérer les champs*, modifier les champs des noeuds de type *Page de base* en ajoutant 3 champs de type ***Image***.
- Sur *Admin > Structure > Types de contenu > Page de base > Gérer l'affichage*, choisir le style d'image *Moyen* pour chaque champ de type *Image* et sélectionner le layout *Four Squares* pour afficher les champs.
- Modifier une page de base en renseignant chaque champ.
- Quels sont les avantages et inconvénients de passer par le module ***Display Suite***, plutôt que de créer des templates manuellement ?

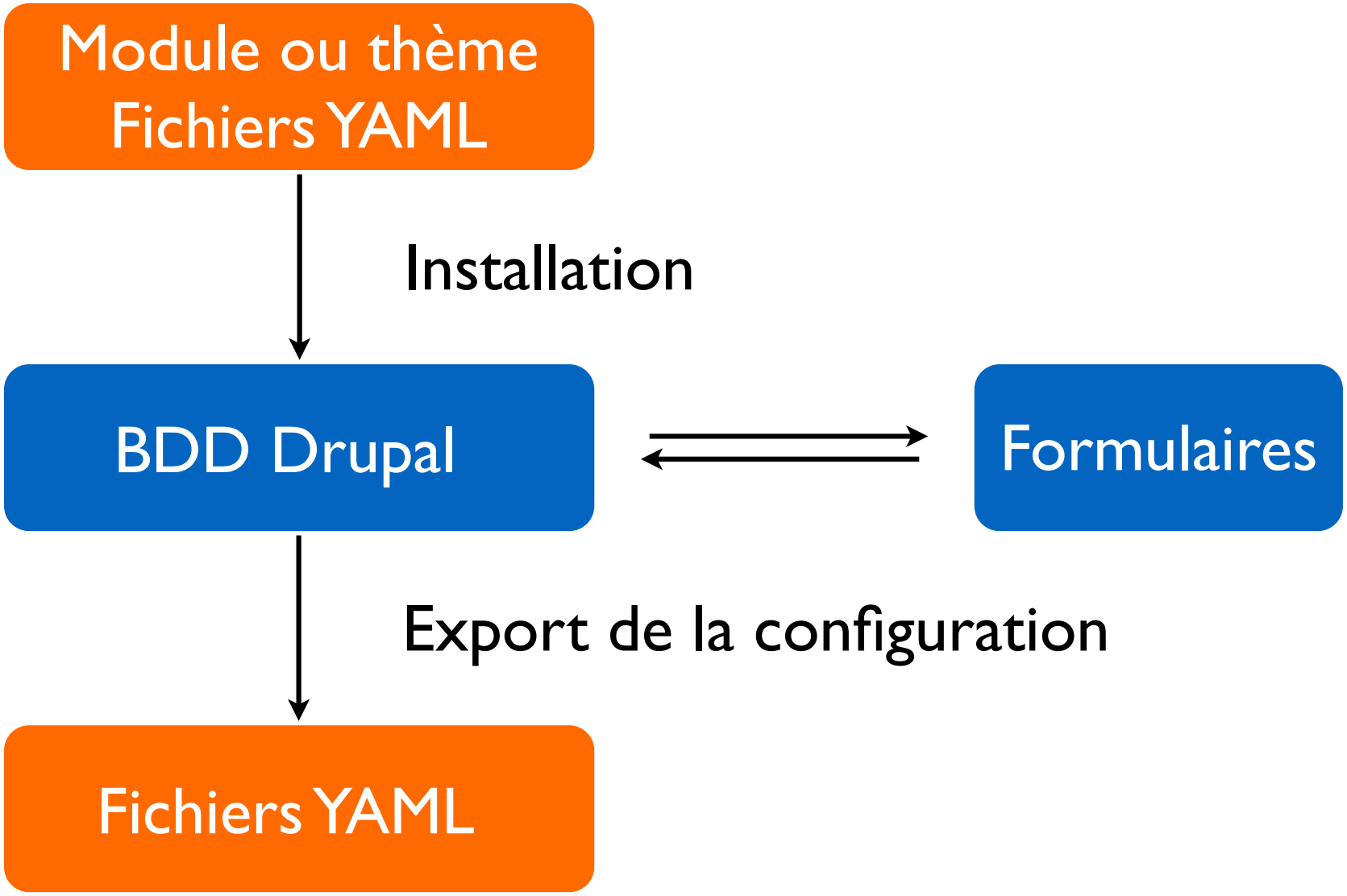


Système de configuration

Exporter une configuration

- ***Drupal 8*** est capable d'exporter individuellement chaque élément de configuration sous forme de fichier YAML.
- Pour exporter un élément en particulier (par exemple un style d'image ou un type de contenu…), aller sur *Admin > Configuration > Développement > Synchronisation de configuration > Exporter > Elément individuel*.
- Copier le code correspondant à la configuration dans le fichier adéquat.
- Ensuite créer et placer le fichier de configuration dans le dossier ***/config/optional*** (à créer) du thème.
- Chaque fichier de configuration présent dans le répertoire ***/config/optional*** est chargé lors de l'installation du thème, à condition que les modules requis soient bien installés.
- ***Remarque*** *: Drupal ne peut pas importer une configuration si celle-ci est déjà prise en compte par le système.*

Gestion de la configuration



Embarquer des styles d'image

- Aller sur *Admin > Configuration > Développement > Synchronisation de configuration > Exporter > Elément individuel* et **exporter les styles d'image ainsi que le style d'image adaptatif** créés précédemment.
- Copier le code fournit sans la première ligne qui correspond à l'identifiant unique du site.
- Créer le fichier .*YML* avec le nom correspondant au style d'image. Coller le code dans le fichier. Le fichier doit être placé dans le dossier ***/config/optional*** du thème.
- Désinstaller le thème.
- Supprimer le style d'image adaptif via l'interface du back-office.
- Réinstaller le thème et aller sur *Admin > Configuration > Média > Styles d'image adaptifs*. Que constatez-vous ?

Javascript

jQuery et jQuery UI

- **jQuery** (jquery.com/) est une bibliothèque JavaScript permettant d'ajouter des **comportements dynamiques** et des **effets visuels** aux pages web.
- **jQuery UI** (jqueryui.com/) est une bibliothèque de **composants visuels** pour jQuery : calendrier, accordéon, onglets...
- Par défaut, *Drupal 8* embarque :
  - **jQuery 2.2.3**.
  - **jQuery UI 1.11.4**.
- Pour pouvoir utiliser du code *jQuery dans vos scripts* dans Drupal, la bonne pratique est d'encadrer votre code par :

```
(function($, Drupal, drupalSettings) {

  // Votre code jQuery ici.

})(jQuery, Drupal, drupalSettings);
```

Pourquoi utiliser JavaScript ?

- JavaScript permet de ré-écrire le HTML et les CSS dans le navigateur du client.
- Avantages :
  - **Pas besoin de comprendre** comment Drupal génère son HTML/CSS ; on se contente de repasser par derrière le HTML existant.
  - **Produire des effets** impossibles à réaliser autrement.
- Inconvénients :
  - Tout n'est pas possible avec cette technique : on est obligé de **faire avec les données qu'on a sous la main**.
  - Le client (navigateur) **ne supporte pas nécessairement JavaScript**.
  - Les modifs peuvent **apparaître après le chargement de la page**.

Ajouter ses propres fichiers de Javascript

- Comme pour les fichiers de style, les scripts JS doivent être **déclarés** sous forme de **bibliothèque** via les fichiers *montheme.info.yml* et *montheme.libraries.yml*.

ive.libraries.yml

```
ive:
  version: VERSION
  js:
    js/ive.js: {}
  css:
    theme:
      css/ive.css: {}
```

Créer un fichier JavaScript

- **Créez le fichier *ive.js*** dans le répertoire ***/js*** du thème ***Ive***, et référencez-le via une nouvelle librairie depuis le fichier ***ive.info.yml***.
- Dans le fichier ***ive.js***, saisissez le code suivant :

```
(function($, Drupal, drupalSettings) {
  $(document).ready(function() {
    alert('Hello !');
  });
})(jQuery, Drupal, drupalSettings);
```

- Rafraîchissez n'importe quelle page de votre site et vérifiez que vous voyez bien une fenêtre popup affichant "*Hello !*".

jQuery - Sélecteurs

- Le code jQuery respecte la syntaxe suivante :
  $('selecteur').commande(*parametres*);
- Le sélecteur définit la ou les balises HTML à modifier et utilise la même syntaxe qu'un sélecteur CSS :
  - // tous les paragraphes
    $('p').commande();
  - // la balise portant l'id footer
    $('#footer').commande();
  - // toutes les balises portant la classe node
    $('.node').commande();
  - // cible quoi selon vous ??
    $('#block-38 div.content').commande();

jQuery - Commandes

- La **commande** définit la modification à appliquer aux sélecteurs sélectionnés.
- Quelques commandes pour **changer le CSS** :
  - // passe la couleur du texte en rouge
    $('p').**css**('color', 'red');
  - // passe la couleur du fond en jaune
    $('p').**css**('background', 'yellow');
  - // ajoute la classe toto à tous les paragraphes
    $('p').**addClass**('toto');
- Quelques commandes pour **changer le HTML** :
  - // modifie l'attribut target des liens
    $('a').**attr**('target', '_blank');
  - // remplace le HTML du footer
    $('div#footer').**html**(' Texte ');
  - // supprime le de la page
    $('div#footer').**remove**();
- Toutes les commandes sont documentées sur *docs.jquery.com*.

Modifier le HTML/CSS

Dans le fichier *ive.js*, ajoutez le code JavaScript permettant de **réaliser les tâches suivantes** :

- Tous les liens pointant vers un site externe doivent s'ouvrir dans une nouvelle fenêtre.
- Tous les liens pointant vers un site externe qui apparaissent dans un noeud doivent être précédés de l'icône *d8/DESIGNER/HTML_CSS/**external-link.gif***.
- Tous les blocs peuvent être repliés/dépliés lorsque l'on clique sur leur titre.

Ajouter un plugin jQuery Marche à suivre

- **Télécharger** le plugin depuis son site.
- **Copier le plugin à l'endroit approprié** sur le serveur (généralement : dans le répertoire du thème courant *).
- **Référencer les fichiers .js** et **.css** du plugin sous forme de bibliothèque (fichier *montheme.libraries.yml*).
- **Formater le HTML** de la manière attendue par le plugin.
- **Déclencher le plugin** en ajoutant les quelques lignes de JavaScript permettant de le "démarrer".

Ajouter un plugin jQuery Exemple avec jCarousel

- **Télécharger** *jCarousel* depuis *sorgalla.com/jcarousel/*.
- **Placer le fichier jquery.jcarousel.js** dans le dossier */themes/ive/vendor/jcarousel*.
- **Déclarer une nouvelle bibliothèque** dans le fichier *ive.libraries.yml*, référençant le fichier *jquery.jcarousel.js*.
- **Charger la bibliothèque** depuis le fichier *ive.info.yml*.
- **Formater le HTML** tel que *jCarousel* l'attend.
- **Déclencher** *jCarousel* depuis votre fichier script.js avec la commande *.jCarousel()*.

Ajouter un plugin jQuery
Exemple avec jCarousel

```
jcarousel:
  version: 0.3.1
  js:
    vendor/jcarousel/jquery.jcarousel.js: {}
  dependencies:
    - core/jquery
```

Déclarer la bibliothèque

```
<div class="jcarousel">
  <ul>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</div>
```

Formatter le HTML

```
1  (function($) {
2
3    $(document).ready(function() {
4      $('.jcarousel').jcarousel();
5    });
6
7  })(jQuery);
8
```

Déclencher le plugin

Mise en page sous forme d'onglets

- Nous allons utiliser le widget **Tabs** de jQuery UI, qui permet d'afficher du contenu sous forme d'onglets. Démonstration sur jqueryui.com/tabs/.
- Créer le template adéquat, permettant de gérer l'affichage des noeuds de type article en mode full.
- **Transformez les noeuds *article*** de sorte qu'un article s'affiche **sous forme d'onglets**, plutôt qu'intégralement sur la même page. Vous créerez les onglets suivants :
  - Détails
  - Image
  - Commentaires
- ***Remarques :***
  - *pour créer vos onglets, vous vous inspirerez de l'exemple fourni sur le site jqueryui.com.*
  - *vous ferez en sorte que la librairie ne soit chargée qu'au moment opportun.*

Trained People c'est aussi :

- des **formations Drupal spécialisées** (Webmaster, Themeur, Développeur, Sécurité & Performance, Déploiement & Industrialisation).
- des **certifications** à **Drupal** (Webmaster, Themeur/Intégrateur, Développeur et Expert).
- un programme de **e-learning Symfony/Drupal 8** en partenariat avec **SensioLabs**.
- de **l'accompagnement** durant vos projets (audit, régie...).
- des **recommandations** (freelances, agences, hébergement).

TRAINEDPEOPLE

SensioLabs
UNIVERSITY

Merci !