Cleverinstitut

Magento 2 - Développement Front-End

03 - 04 - 05 Juillet 2017

CLEVER AGE

Votre formateur Clever Institut :
Claude Lecomte

VOTRE FORMATEUR

Objectifs de la formation

- Créer un thème
- Modifier / surcharger / étendre un layout
- Modifier / surcharger un template
- Modifier / surcharger / étendre les CSS
- Ajouter un fichier javascript
- Comprendre et maîtriser les outils de build Javascript
- Comprendre le data-binding
- Comprendre et maîtriser KnockoutJS
- Ajouter un widget Magento 2
- Personnaliser un email transactionnel

SOMMAIRE

- Généralités Magento 2
- Développement d'un thème
- Gestion des statiques
- La librairie graphique Magento 2
- Les widgets Magento 2
- Le responsive design sous Magento 2
- Javascript sous Magento 2
- KnockoutJS
- Les tests sous Magento 2
- Les traductions
- Emails transactionnels

Cleverinstitut

Généralités Magento 2

Les pré-requis

Connaître et maîtriser les technologies suivantes :

- CSS 3
- HTML 5
- XML
- Javascript
- Responsive Web Design
- PHP/Mysql

Navigateurs supportés :

- Front : IE11+ et les autres navigateurs en version “latest-1”
- Back : IE11+ et les autres navigateurs en version “latest-1”

Documentation officielle frontend

http://devdocs.magento.com/guides/v2.1/frontend-dev-guide/bk-frontend-dev-guide.html

Les modes 1/2

Trois modes différents :

- default
  - Non optimisé pour la production
  - Mise en cache des fichiers statiques
  - Exceptions loguées mais non affichées à l'internaute
- developer
  - Pas de mise en cache des fichiers statiques
  - Logs détaillés
  - Active la compilation automatique
  - Débug facilité
  - Performances moindres
- production
  - Exceptions non affichées
  - Fichiers statiques en cache

Les modes 2/2

Passer en mode développeur

Se placer dans /var/www/magento

- cd /var/www/magento

Vérifier le mode :

- bin/magento deploy:mode:show

Si la réponse n'est pas "developer" :

- Vider les dossiers var/generation et var/di :
  - rm -R var/generation/*
  - rm -R var/di/*
- Taper :
  - bin/magento deploy:mode:set developer

Arborescence des fichiers

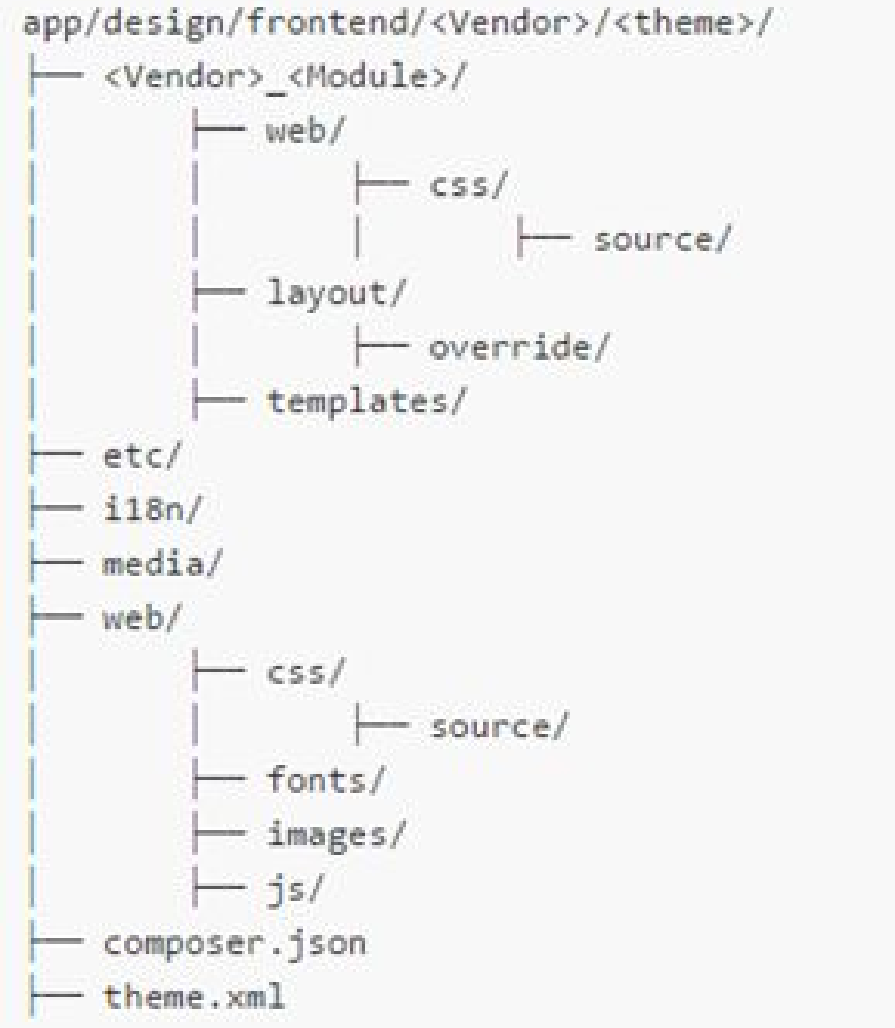
Ce sera plus simple dans un IDE :-)

Arborescence d'un thème

Règle de base : on ne modifie jamais directement les thèmes standard Magento

Structure :

- adminhtml : pour le back office
- frontend : pour le front
- base : pour les 2



Thèmes disponibles de base

Thèmes Blank et Luma

- vendor\magento\theme-frontend-blank
- vendor\magento\theme-frontend-luma

=> luma hérite de blank

Fichiers css

- Magento_Theme/layout/default_head_blocks.xml

Mises en page standard : 1 colonne, 2 colonnes, 3 colonnes

- vendor\magento\module-theme\view\frontend\page_layout

Cleverinstitut

Développement d'un thème

Les lignes de commandes utiles 1/2

A lancer depuis le dossier magento de la VM

Vérifier si le mode est bien “developer” :

➢ bin/magento deploy:mode:show

Si la réponse n'est pas "developer", taper :

➢ bin/magento deploy:mode:set developer

Vider le dossier des statiques sauf le .htaccess :

➢ rm -R pub/static/*

Vider le dossier var/ (attention le dossier var qui est dans var/www/magento) :

➢ rm -Rf var/*

Deploy les statics contents à chaque ajout de nouveau fichier .less :

➢ bin/magento setup:static-content:deploy fr_FR

Plusieurs langues à la fois :

➢ bin/magento setup:static-content:deploy fr_FR es_ES

Vider le cache

➢ bin/magento cache:flush

Commandes Grunt -> grunt refresh après avoir déployé les statics (--force à la fin si nécessaire)

Les lignes de commandes utiles 2/2

A lancer depuis le dossier magento de la VM

Pour voir les commandes de l'instance de magento

- bin/magento

Aide de la commande en question

- bin/magento ma_commande --help

Lancer les installeurs (suite à des modifications de modules par des développeurs back-end) :

- bin/magento setup:upgrade

Compiler les classes Magento (si demandé après avoir lancé les installeurs) :

- bin/magento setup:di:compile

Changer les droits d'écriture

- chmod -R g+w var/ app/etc/ pub/ pub/media/
- find . -type d -exec chmod 755 {} \; && find . -type f -exec chmod 644 {} \; && chmod u+x bin/magento

Réindexer la base de données (après un import) :

- bin/magento indexer:reindex

Création d'un thème

Création arborescence et déclaration

1. Créer le dossier du thème
   - ➢ app/design/frontend/<your_vendor_name>/<your_theme_name>

2. Déclarer le nouveau thème et sa dépendance (thème parent)

3. Ajouter un fichier theme.xml à la racine du thème

4. Enregistrer le nouveau thème
   - ➢ Ajouter un fichier registration.php à la racine

5. Le thème apparaît désormais dans le backend
   - ➢ Content > Design > Configuration

Note : le noeud preview doit être enlevé (ou bien mettre une image)

A vos claviers

1. Créer le dossier du thème
   – app/design/frontend/<your_vendor_name>/<your_theme_name>

2. Déclarer le nouveau thème et sa dépendance (thème parent)
   – Ajouter un fichier theme.xml à la racine du thème

3. Enregistrer le nouveau thème
   – Ajouter un fichier registration.php à la racine

4. Le thème apparaît désormais dans le backend :
   – Content > Design > Configuration

Note : le noeud preview doit être enlevé (ou bien mettre une image)

Déclaration du logo

Nom par défaut du fichier logo dans Magento 2 : logo.svg

Deux possibilités :

- Notre thème n'a pas de thème parent
  - Si le logo utilise un nom identique au nom par défaut, rien à faire
  - Si nom du logo ou format différent, déclaration du logo à faire
- Notre thème a un thème parent
  - Si le logo utilise un nom identique au parent, rien à faire
  - Sinon, il faut le déclarer

Pour déclarer un logo : <theme_dir>/Magento_Theme/layout/default.xml

Note : si le layout.xml n'est pas interprété, vérifier que le fichier xml est en utf-8, vérifier en base de données que le type du thème est bien à 0 (un bug le met à 1 qui est un type virtuel et de ce fait layout.xml n'est pas interprété)

A vos claviers

- Afficher un autre logo à la place du logo luma
- Modifier les dimensions du logo

Configurer les images du thème

Les images produits sont redimensionnées automatiquement selon les instructions présentes dans le layout.

- etc/view.xml : obligatoire, sauf si on a un thème parent

- Copier le view.xml d'un thème standard (Luma, Blank)
  - ➢ exemple : vendor\magento\theme-frontend-luma\etc\view.xml

- Modifiez-le à votre convenance…

Pour régénérer les images : Système > Cache management > Flush catalogue images cache (à la fin)

Les fichiers du thème

Ces fichiers se trouvent dans le dossier web qui se trouve à la racine du thème

- app/design/frontend/<your_vendor_name>/<your_theme_name>/web

```
app/design/<area>/<Vendor>/<theme>/
├── web/
│   ├── css/
│   │   ├── source/
│   ├── fonts/
│   ├── images/
│   ├── js/
```

Un thème peut également contenir des fichiers statiques spécifiques à un module, comme ceci :

- .../<theme>/<Namespace_Module>/web/css

Création d'un thème back-office

Fichiers requis

- Doit comporter les mêmes fichiers qu'un thème front
  - ➢ theme.xml
  - ➢ registration.php
- Doit être placé dans app/design/adminhtml/<Vendor>/<admin_theme>

- Thème back-office disponible dans Magento par défaut
  - ➢ Magento/Backend

- Le theme.xml est le même que pour un thème front

Exemple de registration.php pour un thème back-office

Copy

```php
<?php
/**
 * Copyright © 2016 Magento. All rights reserved.
 * See COPYING.txt for license details.
 */
\Magento\Framework\Component\ComponentRegistrar::register(
    \Magento\Framework\Component\ComponentRegistrar::THEME,
    'adminhtml/%vendor_dir/your_theme_dir%', // Example: 'adminhtml/Magento/backend'
    __DIR__
);
```

Module.xml

Une fois les fichiers requis créés il faut :

- ➢ Créer un nouveau module custom
- ➢ Configurer le module pour qu'il se charge après le module Magento_Theme

Pour cela, créer le fichier <votre_module_custom>/etc/module.xml avec le contenu suivant :

```
<module name="%YourVendor_YourModule%" setup_version="2.0.1"> <!-- Example: "Magento_Backe
    <sequence>
        <module name="Magento_Theme"/>
    </sequence>
</module>
```

Copy

Appliquer le thème

Pour appliquer le thème il faut :

- Créer le fichier <votre_module_custom>/etc/adminhtml/di.xml
- Configurer le thème à appliquer en back-office

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=
    <!-- Admin theme. Start -->
    <type name="Magento\Theme\Model\View\Design">
        <arguments>
            <argument name="themes" xsi:type="array">
                <item name="adminhtml" xsi:type="string">%Your_vendor_dir%/%your_theme_code%</item>
            </argument>
        </arguments>
    </type>
    <!-- Admin theme. End -->
</config>
```

Copy

Afin que le changement soit pris en compte, lancer les commandes suivantes :

- bin/magento setup:upgrade
- bin/magento setup:di:compile

A vos claviers

- Créer un nouveau thème pour le back-office
- Créer les fichiers requis
- Créer le module custom requis
- Créer la configuration nécessaire pour que le thème soit pris en compte
- Appliquer le thème

Les layouts

Introduction

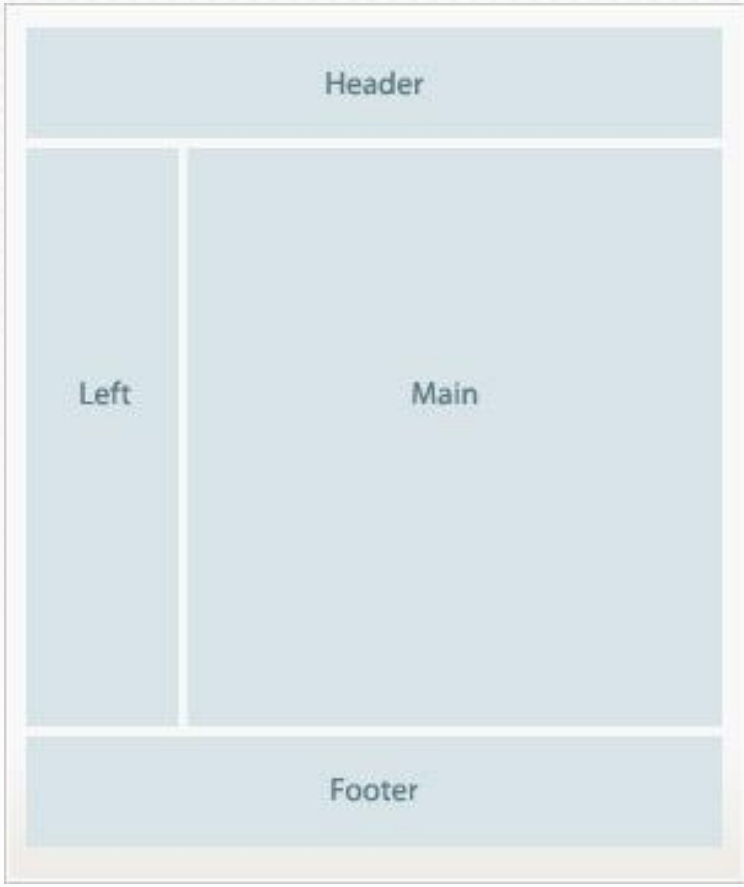
Partie la plus importante de la brique View du pattern MVC utilisé par Magento 2

- Fichier au format XML
- Définit la structure des pages et la hiérarchie des éléments de la page
- Déclare et manipule deux types d'éléments :
  - ➢ Container
  - ➢ Block
- On distingue les layouts selon leur provenance :
  - ➢ Base layouts : Layout provenant d'un module
  - ➢ Theme layouts : Layout provenant d'un thème

Types d'éléments - Container

- Uniquement pour définir la structure de la page
- N'a pas de contenu, si ce n'est celui des "blocks" qu'il possède

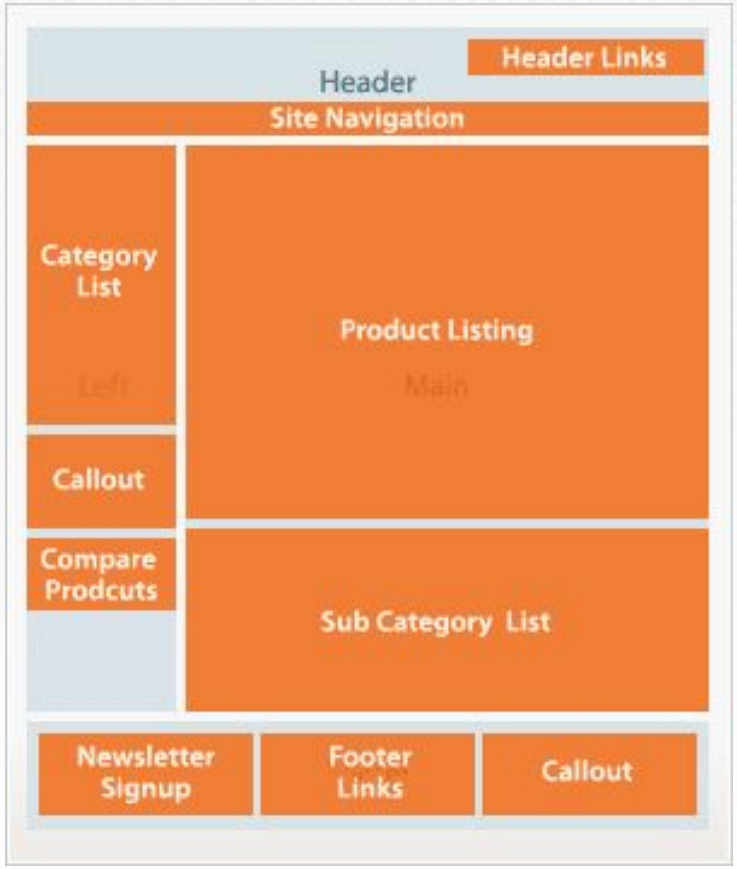
Exemples : header, left column, main column, et footer



Types d'éléments - Block

- Représente une fonctionnalité de la page
- Peut utiliser un template
- Est enfant d'un container

Exemples : category list, mini cart, product tags, product listing



Types de layouts

Il existe **plusieurs types de fichiers de layout**, chacun ayant un **rôle défini.**

- Page layout
  - ➢ Déclare les squelettes de pages à l'intérieur de la balise body (1column, 2column-left, etc.)
- Page configuration
  - ➢ Ajoute les "blocks" dans les squelettes définis par les "page layout"
  - ➢ Ajoute également les meta-information d'une page et le contenu des balises head, html et body
- Generic Layout
  - ➢ Défini le contenu et la structure détaillée de la balise body
  - ➢ Est utilisé pour les retours ajax, les emails, les snippets HTML

Page layout

- Instruction autorisées :
  - <container>
  - <referenceContainer>
  - <move>
  - <update>
- Emplacement des fichiers :
  - Module : <module_dir>/view/frontend/page_layout
  - Theme : <theme_dir>/<Namespace>_<Module>/page_layout

Exemple :

<Magento_Theme_module_dir>/view/frontend/page_layout/2columns-left.xml

```
<layout xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_layout.xsd">
    <update handle="1column"/>
    <referenceContainer name="columns">
        <container name="div.sidebar.main" htmlTag="div" htmlClass="sidebar sidebar-main" after="main">
            <container name="sidebar.main" as="sidebar_main" label="Sidebar Main"/>
        </container>
        <container name="div.sidebar.additional" htmlTag="div" htmlClass="sidebar sidebar-additional" after="div.sidebar.main">
            <container name="sidebar.additional" as="sidebar_additional" label="Sidebar Additional"/>
        </container>
    </referenceContainer>
</layout>
```

Déclaration de page layout

Chaque “page layout” doit être déclaré dans un fichier “**layouts.xml**”

Emplacement des fichiers “layouts.xml” :

- Module : <module_dir>/view/frontend/layouts.xml
- Thème : <theme_dir>/<Namespace>_<Module>/layouts.xml

Ex : <Magento_Theme_module_dir>/view/frontend/layouts.xml

```
<page_layouts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/PageLayout/etc/layouts.xsd">
    <layout id="1column">
        <label translate="true">1 column</label>
    </layout>
    <layout id="2columns-left">
        <label translate="true">2 columns with left bar</label>
    </layout>
    <layout id="2columns-right">
        <label translate="true">2 columns with right bar</label>
    </layout>
    <layout id="3columns">
        <label translate="true">3 columns</label>
    </layout>
</page_layouts>
```

Page configuration

Instructions autorisées :

- <page></page>
- <html></html>
- <head></head>
- <body></body>
- <attribute>
- <title>
- <meta>
- <link>
- <css>
- <script>

Emplacements des fichiers :

- Module : <module_dir>/view/frontend/layout
- Theme : <theme_dir>/<Namespace>_<Module>/layout

Generic layout

Instructions autorisées :

- <layout></layout>
- <update>
- <container>

Emplacements des fichiers :

- Module : <module_dir>/view/frontend/layout
- Theme : <theme_dir>/<Namespace>_<Module>/layout

```
<layout xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/layout_generic.xsd">
    <update handle="formkey"/>
    <update handle="adminhtml_googleshopping_types_block"/>
    <container name="root">
        <block class="Magento\Backend\Block\Widget\Grid\Container" name="googleshopping.types.container" template="..."/>
    </container>
</layout>
```

Étendre un layout

Avec Magento 1 il était impossible d'étendre un layout

Créer un fichier du même nom que le fichier à étendre, comme ceci :

```
<theme_dir>
 |__/<Namespace>_<Module>
   |__/layout
     |--<layout1>.xml
     |--<layout2>.xml
```

OU

```
<theme_dir>
 |__/<Namespace>_<Module>
   |__/page_layout
     |--<layout1>.xml
     |--<layout2>.xml
```

Ex. :

Pour étendre le fichier

<Magento_Catalog_module_dir>/view/frontend/layout/catalog_product_view.xml

Ajouter ce fichier

<theme_dir>/Magento_Catalog/layout/catalog_product_view.xml

Surcharger un layout

Avec Magento 1 on ne pouvait que surcharger les fichiers xml de layout

Ceci n'est pas recommandé, mais on surcharge un fichier de layout pour différentes raisons :

- Suppression de l'appel à une méthode (si le "block" ne possède pas de méthode permettant d'annuler l'effet de la première méthode appelée)
- Modification des arguments d'une méthode
- Ajout d'attributs XML sur un block/container
- Suppression d'arguments d'un bloc
- Modification/suppression de l'inclusion d'un handle (update handle="..." />)

De manière générale, on l'utilise donc pour annuler une instruction qui ne peut pas l'être par une simple extension de layout.

Surcharger un layout, comment ?

On peut surcharger deux "types" de layout

Base layout (layout provenant d'un module)
Créer un fichier de layout avec le même nom mais dans un dossier "override", comme ceci :

```
<theme_dir>
  |__/<Namespace_Module>
    |__/layout
      |__/override
        |__/base
          |--<layout1>.xml
          |--<layout2>.xml
```

Fichiers surchargés :
<module_dir>/view/frontend/layout/<layout1>.xml
<module_dir>/view/frontend/layout/<layout2>.xml

Surcharger un layout, comment ?

On peut surcharger deux "types" de layout

Theme layout (layout d'un thème parent)
Créer un fichier de layout avec le même nom, comme ceci :

```
<theme_dir>
  |__/<Namespace_Module>
    |__/layout
      |__/override
        |__/theme
          |__/<Parent_Vendor>
            |__/<parent_theme>
              |--<layout1>.xml
              |--<layout2>.xml
```

Fichiers surchargés :
<parent_theme_dir>/<Namespace>_<Module>/layout/<layout1>.xml
et <parent_theme_dir>/<Namespace>_<Module>/layout/<layout2>.xml

A vos claviers

- Déplacer le logo du site dans le footer
- Supprimer le copyright

Les templates

Introduction

Font partie de la couche View de Magento

Sont instanciés via le layout (en grande majorité)

Sont des fichiers .phtml ou .html (pour knockout.js)

Les templates ne sont pas censés contenir de logique, qu'ils soient affichés ou non.

Se trouvent à divers endroits :

- Modules :
  <module_dir>/view/frontend/templates/<path_to_templates>
- Thèmes :
  <theme_dir>/<Namespace>_<Module>/templates/<path_to_templates>

Initialisation via le layout

- Les templates sont initialisés via le layout.
- Chaque block a un template associé
- Ce template est défini grâce à l'attribut "*template*"

Exemple pour un template d'image de catégorie :

```
<block class="Magento\Catalog\Block\Category\View" name="category.image"
template="Magento_Catalog::category/image.phtml"/>
```

Le template se trouve donc dans le dossier "category" du module Magento_Catalog. Il se trouve donc ici :

```
<Magento_Catalog_module_dir>/view/frontend/templates/category/image.phtml
```

Note : pour un template de bloc cms, la class du block sera Magento\Framework\View\Element\Template

Root

- Template de base, appelé sur TOUTES les pages
- Contient la structure basique HTML (balises html, head et body)
- Comme les autres templates, il peut être surchargé

Il se trouve dans :
\vendor\magento\module-theme\view\base\templates\root.phtml

Magento et le XSS 1/2

Le XSS ? Cross-site Scripting, faille de sécurité touchant les sites web permettant d'injecter du contenu sur la page d'un site afin de provoquer des actions sur le navigateur de l'internaute

Magento fournit quelques recommandations en ce qui concerne l'échappement des contenus à afficher :

- Si le nom de la méthode indique que le contenu est déjà échappé, alors pas besoin de l'échapper dans le template. Ex : getTitleHtml() ou getHtmlTitle() indiquent que le contenu est déjà prêt pour une sortie HTML
- Quelques méthodes d'échappement :
  - $block->escapeHtml()
  - $block->escapeQuote()
  - $block->escapeUrl()
  - $block->escapeXssInUrl()

Magento et le XSS 2/2

Les conversions de type et la méthode count() n'ont pas besoin d'être échappés :

- echo (int)$var
- echo (bool)$var
- echo count($var)

Les string simple quote n'ont pas besoin d'être échappés : echo 'some text'

Les contenus en double quote sans variable n'ont pas besoin d'être échappés

Note : Il existe une classe pour tester la sécurité vis-à-vis des injections XSS dans Magento 2. Ce test cherche tous les "echo" dans les fichiers .phtml et vérifie qu'ils sont correctement échappés ou non.

Elle se situe ici :

dev\tests\static\testsuite\Magento\Test\Php\XssPhtmlTemplateTest.php

$this vs $block

Pour accéder aux méthodes du block courant, une nouvelle variable (par rapport à Magento 1) fait son apparition : $block !

Attention : var_dump($this) ou var_dump($block) plante car trop d'infos à afficher

Pour afficher les handles :

```
var_dump($this->getLayout()->getUpdate()->getHandles());
```

Pour afficher les infos du bloc :

```
<?php Zend_Debug::dump($this->getData()); ?>
```

Par rapport à Magento 1, on ne peut plus appeler les méthodes *protected* du block dans les templates .phtml comme c'était le cas.

$this ne fait plus référence à l'instance du block courant, mais au moteur de templates : Magento\Framework\View\TemplateEngine\Php

A vos claviers

- Trouver le nom des templates à modifier
- Surcharger le template de la page d'accueil afin d'ajouter un message de bienvenue
- Modifier le copyright afin de mettre un copyright API&You à la place de celui de Magento

Les CSS

Introduction

Magento 2 utilise un pré-processeur nommé LESS. Les dernières versions de Magento 1 (CE1.9+ et EE1.14+) intégraient SASS / Compass

On peut quand même faire du CSS classique sans pré-processeur

On peut ajouter nos propres fichiers LESS

On peut utiliser un autre pré-processeur pour générer nos CSS qui seront ensuite intégrés aux pages Magento si on le souhaite

On intègre les CSS via le layout, comme sur Magento 1

Inclure des fichiers CSS

Il n'est pas recommandé de charger les CSS directement dans un template, bien qu'il soit possible techniquement de le faire.

Par convention, les fichiers CSS et LESS ne se trouvent que dans les thèmes : les modules ne contiennent aucun fichier de style (sauf les modules custom spécifiquement développés pour le projet).

Inclusion de fichiers CSS :
<Magento_Blank_theme_dir>/Magento_Theme/layout/default_head_blocks.xml

```
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_configuration.xsd">
    <head>
        <css src="css/styles-m.css" />
        <css src="css/styles-l.css" media="screen and (min-width: 768px)"/>
        <css src="css/print.css" media="print" />
    </head>
</page>
```

Le fichier default_head_block.xml, originellement présent dans le module "Magento_Theme" génère la section "head" de TOUTES les pages du site.

Pour ajouter un css supplémentaire dans un module, il faut donc mettre dans le xml un noeud <head> et mettre à l'intérieur l'appel vers le css <css src="" /> (le système va alors chercher automatiquement le .less correspondant)

Surcharger ou étendre le CSS

Dans les modules se trouvent des fichiers _module.less présents dans le dossier source, Magento passe automatiquement dans ces fichiers sans déclaration préalable.

Il est possible de surcharger ou d'étendre ces fichiers.
Il est conseillé de surcharger si beaucoup de modifications sont à faire.
Il est conseillé d'étendre si peu de modifications sont à apporter.

Surcharger :

➢ Copier le fichier _module.less et le placer dans le dossier du thème:
.../design/nom du module/web/css/source

Étendre :

➢ Placer un fichier _extend.less dans le dossier du thème :
.../design/nom du module/web/css/source

Les fonts custom

Magento fournit quelques polices de caractères en standard (opensans + des fonts d'icônes). On peut ajouter les nôtres facilement (localement ou via CDN)

Mettre les fonts dans le dossier <theme_dir>/web/fonts

Déclarer la police de caractère dans le fichier <theme_dir>/web/css/source/_typography.less grâce au mixin .font-face() :

```
.font-face(
    @family-name:'<any_font_name>',
    @font-path: '@{baseDir}fonts/<path_to_font_file>',
    @font-weight: <font_weight>,
    @font-style: <font_style>
)
```

Gestion des statiques

Les process de build

Introduction

Le web moderne implique des ressources statiques de plus en plus complexes.

Afin de profiter des dernières technologies Web il est nécessaire de mettre en place des outils capable de prendre en charge les statiques et de les rendre compréhensibles par un navigateur.

Ils sont entre autre nécessaires pour :

- Sass / Less
- JS ES6
- JSX
- TypeScript

Ils peuvent également être une aide comme :

- Minifier
- Uglifyer
- Autoprefixer

Les outils de build

Gulp

- Simple et intuitif
- Permet à l'aide de fonctions Javascript de décrire comment doit se dérouler la prise en charge des statiques
- Configuration stockée dans gulpfile.js
- Permet de gérer tout type de statique grâce à une large bibliothèque de plugins
- Installable via npm

Exemple :

https://github.com/gulpjs/gulp/blob/master/docs/API.md

Grunt

- Similaire à Gulp
- Décrit les interactions dans des objets de configuration Javascript
- Configuration stockée dans gruntfile.js
- Peut également être étendu grâce à divers plugins
- Installable via npm

Exemple :

https://gruntjs.com/getting-started#an-example-gruntfile

Webpack

- Nouvelle génération d'outil de build
- Adapté aux applications single page
- Adapté aux frameworks JS modernes comme React, Angular2 ou VueJs
- Optimisé pour générer des bundles contenant toutes les ressources
- Configuration dans webpack.config.js
- Peut-être étendu grâce aux plugins disponibles
- Installable via npm
- Intègre un serveur de développement

Exemple :
https://webpack.js.org/configuration/#options

Cleverinstitut

Et sous Magento ?

Pré-processeur CSS

Plusieurs mode de compilation :

- Compilation côté serveur (par défaut), c'est la seule option en production
- Compilation côté client (hors mode production)

Pour modifier le mode de compilation :

- Admin > Stores > Configuration > ADVANCED > Developer > Front-end developer workflow > Workflow type.
- -> Vérifier que "Server side less compilation" est sélectionné

Note : @magento_import

- Cette directive LESS est spécifique à Magento.
- Permet de charger plusieurs fichiers selon un pattern
- Elle doit être précédée de deux slashes : //@magento_import '_example.less'
- A la compilation, Magento cherche toutes les occurrences de "@magento_import" et les transforme en directive "@import" standard less.

Installation de Grunt

Installation de NodeJS

- sudo apt-get install nodejs -y

Installation de l'utilitaire en ligne de commande de Grunt via npm

- npm install -g grunt-cli

Configuration pour un projet

Se positionner sur le répertoire magento, taper : cd /var/www/magento

Attention : il faut package.json et Gruntfile.js dans le dossier d'installation, sinon renommer :

- cp package.json.sample package.json
- cp Gruntfile.js.sample Gruntfile.js

Exécuter les commandes suivantes pour installer grunt sur ce projet, ainsi que ses dépendances, et enfin mettre le tout à jour :

- npm install grunt --save-dev
- npm install
- npm update

Option : refaire l'étape "Configuration pour un projet" dans un dossier app/local/ pour pouvoir configurer gruntfile.js spécifique pour le projet : copier dev/tools/grunt/configs/theme.js + package.json et gruntfile.js à la racine de app/local

Déclarer les thèmes

Pour que Grunt reconnaisse votre thème, vous devez le déclarer dans sa configuration, qui se trouve ici :

- .../dev/tools/grunt/configs/themes.js

Quelques commandes grunt :

- grunt exec:<theme>
- grunt clean:<theme>
- grunt less:<theme>
- grunt refresh:<theme>
- grunt watch

```
module.exports = {
    <theme>: {
        area: 'frontend',
        name: '<Vendor>/<theme>',
        locale: '<language>',
        files: [
            '<path_to_file1>', //path to root source file
            '<path_to_file2>'
        ],
        dsl: 'less'
    },
```

A vos claviers

- Déclarer le thème pour que Grunt le prenne en compte
- Créer une nouvelle feuille de style
- Changer le style d'un élément de la page d'accueil
- Vérifier que grunt watch bien les fichiers

- Créer une instance de Grunt spécifique au projet
- Ecrire une tâche afin qu'il transfère les fichiers Javascript modifiés dans pub/static/test

- Faire le même exercice avec Gulp

Problèmes courants

Vide ton cache ! :-)

Les fichiers statiques du site sont automatiquement enregistrés dans le dossier pub/static/

Dans ce dossier on ne trouvera jamais le sous dossier base/ même s'il est rempli, le contenu de ce dossier ira automatiquement dans frontend/ ou dans adminhml/ selon l'utilisation des fichiers

Très utile !! Effacer tous les fichiers statiques sauf le .htaccess

- rm -R pub/static/*

Le cache

- bin/magento cache:flush

Note : Les fichiers less

Après avoir lancé la commande bin/magento setup:static-content:deploy lang_LANG, ils se mettent dans var/view_preprocessed (les .css compilés se trouvent dans le sous dossier source/)

Le dossier var/

Contient toutes les données générées par Magento.

Les données sont :

- Classes PHP générées
- Vues pré-compilées
- Données mise en cache

Si le vidage de cache n'a pas suffit, vider le dossier var :

➢ rm -rf var/*/*

Puis régénérer les classes nécessaires :

➢ bin/magento setup:upgrade

Si Grunt ne watch pas…

- Vérifier que le mode de compilation en back office est bien “server side compilation” (Admin > Stores > Configuration > ADVANCED > Developer > Front-end developer workflow > Workflow type)
- Regarder dans pub/static si le fichier est un lien symbolique (barré)

Si le fichier n’est pas un lien symbolique :

- lancer la tâche **grunt clean** et vérifier que la tâche vide les dossiers pub/static/frontend et var/view_preprocessed
- lancer la tâche **grunt exec**

Résultat attendu : Le fichier .less apparaît sous forme de lien symbolique dans pub/static et la tâche grunt watch sur ce fichier fonctionne.

Problèmes de permissions

Il peut arriver lors de certaines opérations que Magento 2 lève une erreur concernant des problèmes de permissions (en général en écriture)

Pour rétablir les bonnes permissions et éviter ces erreurs, lancer depuis la racine de Magento :

- chmod -R g+w var/ app/etc/ pub/ pub/media/

Si cette commande ne suffit pas, lancer depuis la racine de Magento :

- find . -type d -exec chmod 755 {} \; && find . -type f -exec chmod 644 {} \; && chmod u+x bin/magento

Mauvais placement de règles CSS

Attention, ne jamais mettre autre chose que des variables dans _theme.less sinon les règles concernées n'apparaîtront pas dans le CSS généré.

A la place, ces règles doivent être placées dans le fichier de surcharge correspondant.

La librairie graphique Magento 2

Introduction

Magento UI est une librairie basée sur LESS. Elle fournit un ensemble de mixins pour créer des éléments de base HTML : (http://devdocs.magento.com/guides/v2.1/frontend-dev-guide/css-topics/theme-ui-lib.html)

- action-toolbar
- breadcrumbs
- buttons
- drop-downs
- forms
- icons
- layout
- loaders
- typography
- messages
- pagination
- popups
- ratings
- tabs
- accordions
- tables
- tooltips
- variables de thème

Emplacement des mixins

```
lib/web
    ├─ css/
    │    ├─ docs/ (Library documentation)
    │    ├─ source/
    │    │    ├─ lib/ (Library source files)
    │    │    │    ├─ variables/ (Predefined variables for each mixin)
    │    │    │    ├─ _actions-toolbar.less
    │    │    │    ├─ _breadcrumbs.less
    │    │    │    ├─ _buttons.less
    │    │    │    ├─ _dropdowns.less
    │    │    │    ├─ _forms.less
    │    │    │    ├─ _grids.less
    │    │    │    ├─ _icons.less
    │    │    │    ├─ _layout.less
    │    │    │    ├─ _lib.less
    │    │    │    ├─ _loaders.less
    │    │    │    ├─ _messages.less
    │    │    │    ├─ _navigation.less
    │    │    │    ├─ _pages.less
    │    │    │    ├─ _popups.less
    │    │    │    ├─ _rating.less
    │    │    │    ├─ _resets.less
    │    │    │    ├─ _responsive.less
    │    │    │    ├─ _sections.less
    │    │    │    ├─ _tables.less
    │    │    │    ├─ _tooltips.less
    │    │    │    ├─ _typography.less
    │    │    │    ├─ _utilities.less
    │    │    │    └─ _variables.less
    │    │    └─ _extend.less
    │    │    └─ _theme.less
    │    │    └─ _variables.less
    │    └─ styles.less
    ├─ fonts/
    │    └─ Blank-Theme-Icons/ (Library custom icons font)
    ├─ images/
    │    └─ blank-theme-icons.png (Library icons sprite)
    └─ jquery/ (Library javascript files)
```

Les mixins sont situés dans le dossier :

➢ lib/web/css/source/lib

Pour obtenir une liste exhaustive des mixins disponibles :

➢ Dans le dossier docs/, glisser-déposer le fichier index.html dans un navigateur

Les variables prédéfinies

Elles se trouvent dans le dossier : lib/web/css/source/lib/variables

Dans ce dossier, on retrouve les mêmes noms de fichiers que pour les mixins. C'est parce que les variables sont divisées par mixin.

Il est possible de surcharger les valeurs des variables prédéfinies grâce au fichier _theme.less du dossier <theme_dir>/web/css/source/_theme.less

On peut ajouter nos propres variables en suivant la convention de nommage Magento pour les variables LESS. Quelques exemples de recommandation :

- noms de variables en minuscule et significatifs
- toujours préfixé d'un @ ou @_
- @property-name pour les variables de valeur
- @component-element__state__property__modifier pour les variables de paramètre (ex. @link__hover__color, @nav-element__background-color)

Utiliser les mixins

Les mixins utilisent les variables prédéfinies en paramètres. On peut donc appeler un mixin avec ou sans arguments en paramètre.

- Sans arguments, le mixin utilise les variables prédéfinies
- Avec arguments, le mixin utilise en priorités les valeurs transmises

```
.breadcrumbs {
    .breadcrumbs();
}
```

```
.example-button {
    .button(
        @_button-padding: @button-padding,
        @_button-color: #fff,
        @_button-color-hover: #ccc
    );
}
```

Permet de modifier l'affichage d'un élément DOM sans modifier le mixin associé.

Pour plus d'infos :

- lib/web/css/docs/source/README.md
- lib/web/css/docs

Les widgets Magento 2

Création depuis le back-office

Un widget se crée en back office dans le menu : Content > Widgets

1. Choisir un widget
2. Choisir le thème du site
3. Remplir les paramètres
4. Layout updates > permet d'indiquer dans quelle(s) page(s) afficher le widget et dans quel bloc

Lorsqu'un widget est créé, il n'est pas possible de modifier le type de widget ni le site. Pour changer ces informations, il faut donc supprimer le widget et le créer de nouveau.

Il est possible d'afficher un widget dans plusieurs pages en utilisant le bouton "Add layout update".

Création depuis le code

Il s'agit d'une méthode plutôt destinée aux développeurs back-end.

Pour créer un widget, nous avons besoin de deux choses :

- Un module déclaré et fonctionnel (composer.json, registration.php et etc/module.xml)
- un fichier de déclaration de widgets appelé widget.xml

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>

<widgets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../../Magento/Widget/etc/widget.xsd">
        <widget id="ves_customwidget" class="Ves\CustomWidget\Block\Widget\ContactInformations">
                <label translate="true">Contact Informations Widget</label>
                <description>Widget in Magento2</description>
                <parameters>
                        <parameter name="fullname" xsi:type="text"  visible="true" sort_order="0" >
                                <label translate="true">Full Name</label>
                        </parameter>
                        <parameter name="age" xsi:type="text"  visible="true" sort_order="10" >
                                <label translate="true">Age</label>
                        </parameter>
                        <parameter name="gender" xsi:type="select" source_model="Ves\CustomWidget\Model\Config\Source\Gender" visible="true" sort_order="10" >
                                <label translate="true">Gender</label>
                        </parameter>
                </parameters>
        </widget>
</widgets>
```

A vos claviers

- Créer un bloc contenant un message de bienvenue
- Créer un widget utilisant ce bloc
- Insérer ce widget sur les pages dans la zone de contenu

Cleverinstitut

Le responsive design sous Magento 2

Mobile first

Deux fichiers de styles par défaut :

- styles-m.css : pour les mobiles et styles communs
- styles-l.css : pour le desktop (> 768px)

Regarder ce fichier :

- vendor\magento\theme-frontend-blank\Magento_Theme\layout\default_head_blocks.xml

On voit que le fichier styles-l.css n'est chargé que si la largeur du viewport est supérieure à 768px. Grosse économie de poids pour le mobile !

Les outils à disposition

Media queries : mixin .media-width()

➢ Se situe dans /lib/web/css/source/lib/_responsive.less

Breakpoints

➢ Se situent dans /lib/web/css/source/lib/variables/_responsive.less

@screen__m: 768px; ⇒ en standard Magento 2 (thèmes blank et luma) c'est le breakpoint de basculement desktop/mobile

Mobile first

Magento a choisi une approche mobile first pour ses thèmes blank et luma. ⇒ Sur mobile, les styles 'desktop' (styles-l.css) ne sont pas chargés.

Le mixin .media-with() 1/3

Au moment de la compilation, less va mettre les media queries mobile uniquement dans le fichiers styles-m.css et les media queries desktop dans styles-l.css.

Cela évite d'avoir une css mobile très lourde et dont la moitié des instructions ne servent pas.

Les fichiers css générés ne font qu'un seul et unique appel à chaque media query.

Pour info : Sur magento 1, on avait de multiples appels aux media queries et elles n'étaient pas regroupés. Maintenant elles le sont.

Le mixin .media-with() 2/3

.media-width(<@extremum>, <@break>);

Par exemple :

```
.media-width(@extremum, @break) when (@extremum = 'max') and (@break = @screen__s) {
    your styles
}
```

Génère ->

```
@media only screen and (max-width: 640px) {
    your styles
}
```

@extremum : min|max
@break : value ⇒ breakpoint

@media-target: all|desktop|mobile ⇒ défini le type de device

Le mixin .media-with() 3/3

Dans les fichiers .less on rencontre plusieurs cas :

& **when** (**@media-common** = **true**) { … }

➢ commun à tous les formats d'affichage, les css s'inscrivent dans styles-m.css (car mobile first)

Si cette notation n'est pas utilisée, les css s'écrivent en doublon, dans les 2 feuilles de style.

.media-width(**@extremum**, **@break**) **when** (**@extremum** = **'min'**) **and** (**@break** = **@screen__m**) { … }

➢ les css iront dans la feuille de style styles-l.css (> 768px par défaut)

.media-width(**@extremum**, **@break**) **when** (**@extremum** = **'max'**) **and** (**@break** = **@screen__m**) { … }

➢ les css iront dans la feuille de style styles-m.css (< 768px par défaut)

Javascript RWD

Les thèmes standards (blank et luma) utilisent les script suivants pour gérer le déplacement d'éléments en fonction des breakpoints :

- responsive.js
- menu.js
- matchMedia.js
  (utilisé par les deux précédents)

```
├─ app/design/frontend/Magento/blank/web/js/
    ├─ responsive.js
├─ lib/web/
    ├─ matchMedia.js
    ├─ mage/
            ├─ menu.js
```

matchMedia.js fournit une méthode fort pratique pour faire du RWD :

➢ mediaCheck()

```
/*...*/
    mediaCheck({
        media: '(min-width: 768px)',
        // Switch to Desktop Version
        entry: function () {
            /* The function that toggles page elements from desktop to mobile mode is called here */
        },
        // Switch to Mobile Version
        exit: function () {
            /* The function that toggles page elements from mobile to desktop mode is called here*/
        }
    }); /*...*/
```

Javascript sous Magento 2

Les widgets JQuery

Magento propose un ensemble de widgets/plugins jQuery dont les principaux sont :

- accordion
- alert
- calendar
- collapsible
- confirm
- loader
- modal (popup)
- menu
- prompt
- tabs

Les sources de ces widgets sont disponibles ici : lib/web/mage/

Initialisation de widgets JQuery

Trois manières d'instancier un composant jQuery :

- via l'attribut data-mage-init
- via la balise <script type="text/x-magento-init">
- via un fichier js classique

Doc Magento 2 :
http://devdocs.magento.com/guides/v2.1/javascript-dev-guide/bk-javascript-dev-guide.html

Plus d'informations sur chaque widget js :
http://devdocs.magento.com/guides/v2.1/javascript-dev-guide/widgets/jquery-widgets-about.html

RequireJS

- Déclaration des fichiers dans requirejs-config.js
- Appel dans le phtml
- Chargement au début du fichier JS

Exemple du module catalogaddtocart dans :

- vendor\magento\module-catalog\view\frontend\web
- vendor\magento\module-catalog\view\frontend\templates\product\list.phtml

A vos claviers

- Créer un widget JQuery qui prend en options
  - un nom
  - un prénom
  - l'id d'un div
  - l'id d'un bouton
  - un message
- Créer la méthode pour qu'au clic sur le bouton, le widget insère le message dans le div
- Surcharger le template de la page d'accueil avec la structure nécessaire
- Initialiser le widget selon la méthode de votre choix

KnockoutJS

Qu'est-ce que KnockoutJS ?

Caractéristiques

- Librairie écrite en Javascript pur (sans dépendances)
- A pour but d'implémenter un mécanisme simple d'utilisation du data-binding
- Grande compatibilité (IE6+)
- Très légère (~13kb)
- Peut être utilisée avec n'importe quelle technologie back-end

Utilisation

- Binding écrit de manière déclarative
- Les nouveaux comportements peuvent être écrit très simplement
- Peut être utilisée comme moteur de template

Qu'est-ce que le data-binding ?

- Permet de lier la vue au modèle
- Permet le rafraîchissement de la vue lors d'un changement dans le modèle
- Permet le rafraîchissement du modèle lors d'une interaction utilisateur

Pour implémenter ce mécanisme, trois composants logiciel sont nécessaires

- view-model
- binding
- observable

Le view-model

Le view-model est un objet Javascript décrivant le comportement liant la vue au modèle.

Exemple :

- Je saisis mon nom dans un champ input
  - Le modèle doit se mettre à jour pour enregistrer ma saisie
  - La vue doit se mettre à jour si nécessaire pour rendre les éléments liés à cette saisie

Le view-model doit être créé dans le thème ou dans un module spécifique. Il doit se trouver dans .../<theme_dir>/web/js ou dans le répertoire js du module spécifique.

Les bindings

Fonctions réagissant sur des évènements afin de synchroniser le modèle et la vue.

Trois fonctions possibles :

1. Contrôler le texte et l'apparence d'un élément
   - text
   - html
   - attr
2. Générer la structure HTML à partir d'itérables ou suivant des conditions
   - foreach
   - if
   - with
3. Interagir avec des éléments de formulaires
   - click
   - submit
   - checked

Il est également possible de définir ses propres bindings facilement.

Les observables

Les observables sont la brique de base de knockoutJS. Le modèle est capable de surveiller leur changement et de réagir en conséquence.

Trois types possible :

- observable
  - Permet de surveiller une valeur non itérable comme une chaîne de caractère
- observableArray
  - Permet de surveiller le contenu d'un array
- computed
  - Permet de générer une valeur de retour à partir de n observables surveillés

Les components

Permettent d'isoler un comportement afin de le rendre réutilisable facilement.

Composés de :

- Un template HTML
- Un view-model

Le templating

KnockoutJS autorise à définir des templates qui seront ensuite utilisés via un binding afin de les insérer dans l'élément.

Les données passées au template peuvent être soit celles du view-model courant soit celles passées en argument.

Knockout et les scopes Magento 2

KnockoutJS n'autorise qu'un seul view-model par page.

Afin de pouvoir multiplier le nombre de view-model par page Magento 2 a :

- Surcharger le moteur knockout
- Introduit la notion de scope
- Changer la méthode d'initialisation des view-model

A vos claviers

- Mettre deux inputs (nom et prénom)
- Mettre un label « Bonjour ${prénom} ${nom} »
- A chaque modification d'une input le label doit se mettre à jour

Les tests sous Magento 2

Les tests unitaires

Pour information.

Magento 2 propose des outils de tests unitaires :

- JsTestDriver test library
- JsTestDriver API
- jsunit.requirejsUtil framework

Les tests peuvent être lancés depuis l'IDE PhpStorm

JSTestDriver sur PHPStorm

Pour installer le plugin :

- Dans PhpStorm, ouvrir les “Settings” puis aller dans l’onglet “Plugins”
- Cliquez sur “Browse repositories…”
- Chercher dans la barre de recherche le terme “JSTestDriver”
- Faire un clic droit sur le plugin et cliquez sur “Download and install”
- Redémarrer PhpStorm

Lancer le serveur JsTestDriver :

- Cliquer sur “Run” dans la barre de menu puis cliquer sur “Edit configuration…”
- Dans le panel de gauche, sélectionner JsTestDriver
- Dans le champs “Configuration file” mettre le chemin vers le fichier

CasperJS avec KnockoutJS sous M2

- Magento 2 a surchargé le moteur de KnockoutJS
- Incompatibilité entre le moteur surchargé et CasperJS
- Devrait être corrigé dans une future version de Magento 2

Solution trouvée

- Utilisation de wd.js (wrapper selenium)
- Utilise la puissance de NodeJS
- Basé sur le système de promise
- Utilisation du package selenium-standalone
- Compatible avec le test runner mocha.js et chai.js

Exemple :

http://admc.io/wd/

Cleverinstitut

Les traductions

Introduction

Lorsque Magento 2 rencontre un texte à traduire, il va chercher dans les dossiers suivants :

1. Traductions de modules : code/<vendor>/<module_dir>/i18n/
2. Traductions de thèmes :
   a. design/.../<parent_theme_dir>/i18n/ (itère sur tous les thèmes parents)
   b. <current_theme_dir>/i18n/
3. Package de traduction : app/i18n/
4. Traductions en base de données

Si plusieurs traductions sont trouvées pour un même texte, alors Magento donne la priorité à la traduction du thème courant, puis ses ancêtres, etc.

Clés de traduction

Déclarer une clé de traduction dans un template :

- <?php echo __('mon texte à traduire'); ?>

Code pays :

- Store > Settings > Configuration > General > Locale options
  (examiner le code pour trouver le bon code)

Traduire une chaîne de layout :

- Ajouter translate="true" sur le noeud xml de la chaîne de caractères

Dictionnaires de traduction

Qu'est-ce qu'un dictionnaire de traduction ?

- Fichier .csv
- Deux colonnes au moins, séparées par une virgule
- Première colonne en anglais "en_US"
- Deuxième colonne contient la traduction dans la locale voulue
- % est utilisé quand la chaîne de traduction qui comporte une variable (%1, %2 etc dans l'ordre des variables)

Exemple :

```
"Add to Cart","Zum Warenkobrn hinzufügen"
"Add to Compare","Hinzufügen um zu vergleichen"
"Add to Wishlist","Zum Wunschzettel hinzufügen"
"Additional Product Info","Zusätzliche Angaben zum Produkt"
"Address","Adresse"
"Address %1 of %2","Adresse %1 von %2"
```

Package de traduction

Un package de traduction est un ensemble de dictionnaires de traductions pour une locale donnée. Il contient également des meta-informations.

Un package de traduction peut se faire de deux façon différentes :

- Un ensemble de fichiers .csv pour les modules et les thèmes
- Un seul dossier contenant l'ensemble des traductions dans un seul dossier

Les packages de traductions peuvent être gérés via composer.

Cleverinstitut

Emails transactionnels

Création

Les templates d'emails sont stockés dans le dossier :
<module_dir>/view/<area>/email
Exemple : vendor\magento\module-customer\view\frontend\email

Ils se présentent sous la forme de fichiers .html.

Rappel : ON NE TOUCHE PAS AU TEMPLATES EMAILS DU CORE MAGENTO.

➢ Créer un nouveau template dans le thème et passer par le backend Magento pour l'utiliser à la place du template par défaut ou bien surcharger le template existant.

Deux approches sont possibles :

- On crée un nouveau fichier .html dans notre thème
- On crée un nouveau template par le backend Magento

Exemple via le thème

Les templates emails supporte le mécanisme de fallback des templates.

Exemple :

Je veux surcharger le template email pour les nouvelles commandes qui se trouve normalement ici :

- <Magento_Sales_module_dir>/view/frontend/email/order_new.html

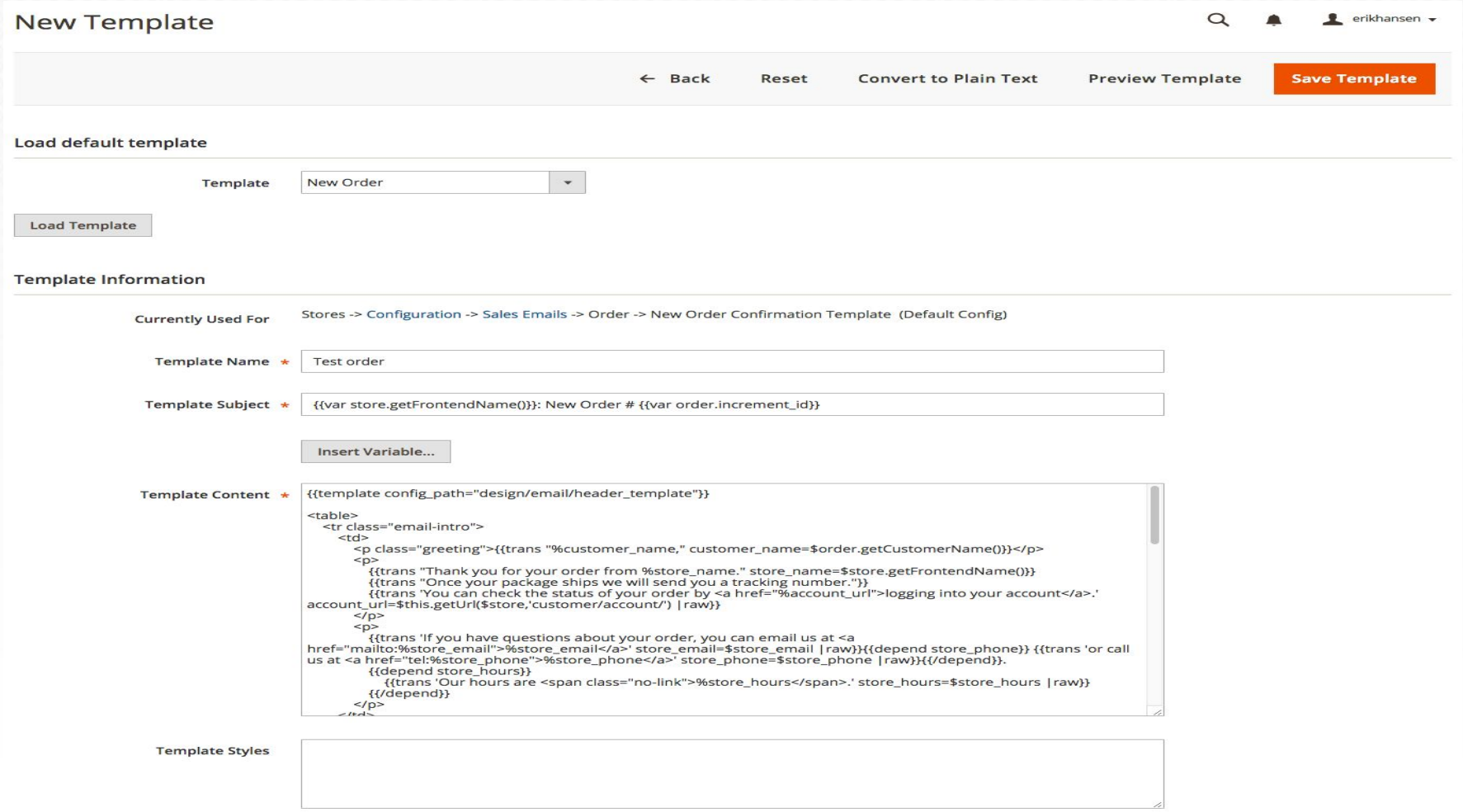
Pour cela, je crée le même fichier ici :

- <theme_dir>/Magento_Sales/email/order_new.html

Notre nouveau template d'email prend la place de l'ancien.

Exemple via le back-office

Les templates configurés via l'admin de Magento prennent le dessus sur les templates par défaut (core de Magento) et les templates surchargés dans les thèmes.



THE END

- Clever Institut vous remercie de votre écoute et espère que cette formation a répondu à vos attentes.

- Merci de penser à signer la feuille de présence et renseigner le formulaire d'évaluation.

A très bientôt

L'équipe Clever Institut