Modifier le thème CSS

Cette page concerne POD à partir de la **version 2.6**.

Le thème de POD s'appuie sur Bootstrap 4 ainsi qu'une feuille de style spécifique à POD. Vous pouvez aussi ajouter votre propre feuille de style. Bootstrap est un toolkit pour développer rapidement en HTML, CSS et JS. Pour en savoir plus, consulter le site GetBootstrap.

Les modifications des styles se font principalement à deux endroits :

- vous pouvez surcharger les styles existants dans votre propre feuille de style
- vous pouvez fournir un css bootstrap personnalisé pour remplacer celui par défaut

Voyons comment faire ...

Choisir un thème de départ

Dans POD, vous disposez de 3 thèmes, configurables dans votre fichier **settings_local.py**. Choisissez le plus proche de ce que vous voulez faire. Pour cela modifiez la variable USE_THEME :

```
###
# Choose a theme for your pod website
# 'default' is the simpliest, bootstrap $enable_rounded is true
# 'green' is with a dark green for primary color, $enable_rounded is false
# 'dark' is black and red, without grey background, $enable_rounded is false
USE_THEME = 'default'
```

default est le thème le plus simple, aux tons clairs, proche des versions précédentes de POD, avec des coins arrondis.

green ajoute une couleur de fond dans la barre de navigation en entête de page et la couleur primaire est un vert sombre. La couleur primaire est mise sur ce fond, les boutons et les liens. Les coins sont droits.

dark ressemble à green, mais en noir. La couleur de fond est blanche (pas de background sur la class jumbotron). Il y a deux couleurs primaires pour différencier boutons et liens. Les coins sont droits.

Rappel

après chaque modification du fichier **settings_local.py**, il faut redémarrer Pod :

```
[root] systemctl restart uwsgi-pod
```

Modifier juste les couleurs

POD s'appuie sur Boostrap. Les styles CSS utilisés sont donc ceux de bootstrap.min.css. Ce fichier contient en entête des variables pour les couleurs. En particulier, POD utilise surtout deux couleurs primaires (primary et primary2) et une couleur secondaire. Et ponctuellement d'autres couleurs (info, danger, ...). Il suffit donc de modifier la variable de la couleur primaire pour que tous les boutons et zones basées sur cette couleur soient modifiés !

Procédure :

Dupliquer le theme de votre choix (default, green ou dark) dans votre dossier custom :

```
mkdir pod/custom/static/custom/css -p
cp pod/main/static/bootstrap-4/css/bootstrap-default.min.css pod/custom/static/custom/css/
```

Editer le ficher et changer les valeurs des variables qui vous intéressent (à priori $primary)

Modifier la variable BOOTSTRAP_CUSTOM dans le fichier settings_local.py :

```
BOOTSTRAP_CUSTOM = 'custom/css/bootstrap-default.min.css'
```

Voilà, c'est tout, votre site a déjà une nouvelle allure 🙂

Vous voulez aller plus loin ? Ajouter une feuille de style !

Bon, en général, cela ne suffit pas de changer les couleurs, on veut aussi aménager son appartement 😉

Pour ajouter votre propre feuille de style, modifier le fichier settings_local.py :

```
TEMPLATE_VISIBLE_SETTINGS = {
...
'CSS_OVERRIDE': 'custom/css/custom.css',
...
}
```

Votre feuille de styles doit être dans le dossier pod/custom/static/custom/css/

Voir cette page de configuration pour plus d'infos.

Vous voulez aller encore plus loin ? Customiser Bootstrap !

Il est intéressant de customiser Bootstrap et c'est très simple pour les modifications de base. En entête du fichier bootstrap.css, vous verrez une petite trentaine de variables CSS. Si vous changez par exemple la couleur "primary" dans ces variables, tous les éléments de bootstrap (et de POD) basés sur cette couleur changeront. En une seule ligne vous changez ainsi la couleur de nombreux éléments.
Ces variables étant en début de fichier, elles sont faciles à changer, même dans la version minifiée de bootstrap (bootstrap.min.css).

Pour aller plus loin dans la customisation, vous aurez besoin de vous intéresser à SASS. Bootstrap contient de très nombreuses variables à customiser dans des fichiers scss. Ces fichiers doivent être compilés avant de pouvoir être utilisés et nécessitent donc d'installer un environnement de développement. Vous trouverez de nombreux tutoriels qui expliquent cela, mais attention, de nombreux sont périmés. Attention d'utiliser ceux qui parlent de bootstrap 4 et gulp 4.

Après de nombreux tatônnements, ce qui fonctionne le mieux chez moi, c'est d'utiliser gulpjs pour compiler les fichiers scss de bootstrap.

Voici une tentative de tuto ... mais après de nombreux essais / erreurs, je ne suis plus sur de ce qui marche ou pas ! J'ai fait l'installation sous Ubuntu, mais cela devrait être similaire sous d'autres OS, y compris Windows. En gros, télécharger les sources de bootstrap, installer les dépendances avec npm et utiliser gulpjs pour compiler les fichiers scss en css. Donc n'hésitez pas à modifier ce tuto s'il y a des erreurs ...

Méthode 1 : installation automatique avec Ansible

Cette méthode automatise complètement l'installation de votre environnement de développement SASS. Elle utilise les playbooks de Ansible. Ce logiciel permet d'automatiser des tâches. Ici, le playbook va installer Node.js, télécharger boostrap et ses dépendances, créer les dossiers scss et dist nécessaires pour customiser, installer gulp, gulp-sass et autres extensions, mettre en place un fichier de configuration pour gulp et vous préparer délicatement un fichier bootstrap à customiser.

Installer Ansible

Si vous avez installé Python (à installer dans un environnement virtuel si vous êtes à l'aise avec ça) :

```
pip install ansible
```

Sinon, des paquets existent pour votre distribution favorite. Pour ubuntu / debian :

```
sudo apt install ansible
```

Télécharger le playbook

```
git clone https://github.com/halbo5/ansible-bootstrap-css.git
```

ou si vous n'avez pas git, télécharger les fichiers depuis github

Lancer l'installation

Depuis le dossier ansible-bootstrap-css :

Vérifier les fichiers de configuration dans les dossiers *roles/bootstrap/defaults* et *roles/nodejs/defaults*. En particulier, vous aurez peut-être envie de modifier le dossier dans lequel installer votre dossier de travail. Il faut modifier la variable '**bootstrap_directory**'.

```
ansible-playbook install-bootstrap.yml -K
```

On vous demandera un "BECOME password". Cela correspond au mot de passe sudo pour installer des applications.

Si tout se passe bien, vous n'avez qu'à regarder défiler les lignes dans votre console ... Ce script a été très peu testé ... signalez les erreurs sur github.

Si tout est bien installé, passer au point "Générer un fichier CSS", sinon, il vous faudra passer par l'installation manuelle.

Méthode 2 : installation manuelle

Si la version automatisée ne fonctionne pas, ou si vous voulez comprendre ce qui se passe, voici une procédure manuelle.

Installation des sources de bootstrap :

Télécharger et installer node.js (pour Ubuntu : https://doc.ubuntu-fr.org/nodejs)

Créer un dossier pour votre projet.

Télécharger les fichiers sources de bootstrap, les décompresser dans votre dossier projet.

Aller dans le dossier "bootstrap"

Télécharger les dépendances de bootstrap : npm install

Vous devriez avoir la structure de fichiers ci-dessous. Créer les dossiers manquants (dist et scss) :

|-Votre-projet-css-pour-pod
|--bootstrap-4
|--dist
|--scss
|----custom.scss (qu'on renommera bootstrap.min.scss plus tard)

Le dossier dist contiendra le fichier css compilé. Le dossier scss contient votre fichier custom.scss. Vous pouvez l'appeler comme vous voulez

Installation des outils pour compiler :

Installer gulpjs 4 :

Pour les détails, voir ce site pour une démarche très complète et qui permet de bien comprendre le fonctionnement de gulp :

https://www.goede.site/setting-up-gulp-4-for-automatic-sass-compilation-and-css-injection)

En résumé :

sudo npm install -g gulp
sudo npm install -g gulp-cli
npm install --save-dev gulp-sass
npm install --save-dev gulpjs/gulp
npm install --save-dev gulp-postcss autoprefixer cssnano gulp-sourcemaps

Créer le fichier gulpfile.js à la racine de votre projet et l'adapter à votre structure de dossier :

```
// gulpfile.js
var gulp = require("gulp"),
    sass = require("gulp-sass"),
    postcss = require("gulp-postcss"),
    autoprefixer = require("autoprefixer"),
    cssnano = require("cssnano"),
    sourcemaps = require("gulp-sourcemaps");

var paths = {
    styles: {
        // By using styles/**/*.sass we're telling gulp to check all folders for any sass file
        src: "scss/**/*.scss",
        // Compiled files will end up in whichever folder it's found in (partials are not compiled)
        dest: "dist"
    }
};

// Define tasks after requiring dependencies

function style() {
    return (
        gulp
            .src(paths.styles.src)
            .pipe(sourcemaps.init())
            .pipe(sass())
            .on("error", sass.logError)
            .pipe(postcss([autoprefixer(), cssnano()]))
            .pipe(sourcemaps.write())
            .pipe(gulp.dest(paths.styles.dest))
    );
}
exports.style = style;

function watch(){
    // gulp.watch takes in the location of the files to watch for changes
    // and the name of the function we want to run on change
    gulp.watch(paths.styles.src, style)
}
exports.watch = watch;
```

Générer un fichier css

Lancer la commande : **gulp watch** dans le dossier qui contient gulpjs. Ce script surveille vos modifications et compile à la volée, donc ne l'arrêtez pas.

Modifier votre fichier bootstrap-default.min.scss comme expliqué sur la page Theming de bootstrap.

Votre fichier css est automatiquement compilé et généré dans le dossier **dist** à chaque enregistrement de bootstrap-default.min.scss.