T5 - Front-end web development

T-WEB-501

Bootstrap

UI / UX

1.0

Bootstrap

delivery method:	Github
repository name:	WEB_my_gohugo_bootstrap_*$ACADEMICYEAR*
repository rights:	ramassage-tek
language:	Hugo

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Introduction

Hugo is a framework developed in Go. It is currently considered as the biggest framework for the creation of static sites.
This framework has several advantages, because it is written in Go, but also thanks to its static aspect :

- It is multiplatform (Windows, macOS, Linux, ...)
- It has many themes developed by the community
- It does not have any security flaws, since it does not have a database
- It has almost no server-side processing, which allows an extremely fast sending of pages of the site

In this bootstrap, you will learn to :

- Generate a static site
- Create a simple blog with *“Hugo”*
- Create a theme for *“Hugo”*
- Add content to the site

This bootstrap was made with the Windows version “0.55.6” of *"Hugo*.

Step 1 : Installation

To install *"Hugo"* on your computer we invite you to visit Hugo's installation page.
After the installation is complete, you can verify that the `hugo` executable is installed by entering the following command in your terminal :

Terminal

```
∼/T-WEB-501> hugo version
```

The current version of *"Hugo"* should be displayed in your device.

Step 2 : Creating the site

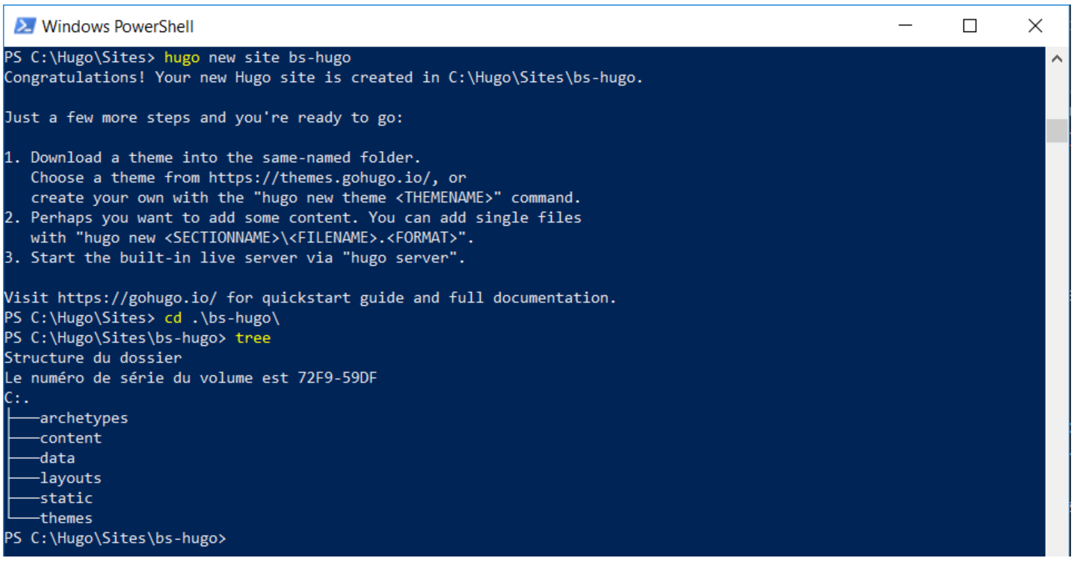
Thanks to the executable `hugo`, you can very simply generate the "base" of your site. This allows you to create the folder architecture of your project.
So use the following command to build your architecture :

Terminal

```
∼/T-WEB-501> hugo new site NAME_OF_YOUR_SITE
```

This command will create a folder "NAME_OF_YOUR_SITE" as well as the classic sub-folders imposed by *"Hugo"*.

Once the command is executed, you should have an architecture close to this one :



```
PS C:\Hugo\Sites> hugo new site bs-hugo
Congratulations! Your new Hugo site is created in C:\Hugo\Sites\bs-hugo.

Just a few more steps and you're ready to go:

1. Download a theme into the same-named folder.
   Choose a theme from https://themes.gohugo.io/, or
   create your own with the "hugo new theme <THEMENAME>" command.
2. Perhaps you want to add some content. You can add single files
   with "hugo new <SECTIONNAME>\<FILENAME>.<FORMAT>".
3. Start the built-in live server via "hugo server".

Visit https://gohugo.io/ for quickstart guide and full documentation.
PS C:\Hugo\Sites> cd .\bs-hugo\
PS C:\Hugo\Sites\bs-hugo> tree
Structure du dossier
Le numéro de série du volume est 72F9-59DF
C:.
├───archetypes
├───content
├───data
├───layouts
├───static
└───themes
PS C:\Hugo\Sites\bs-hugo>
```

Each folder has a particular function. You will find here the link to the official documentation, and below a summary of this folder structure :

- `'archetypes'` : Allows you to define Front Matter by category. A file corresponds to a category
- `'content'` : Contains all the content of your site. A subfolder is a category
- `'data'` : Contains data for your site that you can then reuse in your pages. This data can be stored in YAML, TOML or JSON
- `'layouts'` : This folder contains the *"templates"* of your site. Like the "content" folder, a subfolder is a category. The *"templates"* correspond to HTML code ; they define the layout of a category
- `'static'` : Contains the static content of your site, such as images, CSS, or JavaScript
- `'themes'` : Contains the theme of your site. We will come back to it a little later

> We strongly recommend you to watch the videos available from the links to the documentation!

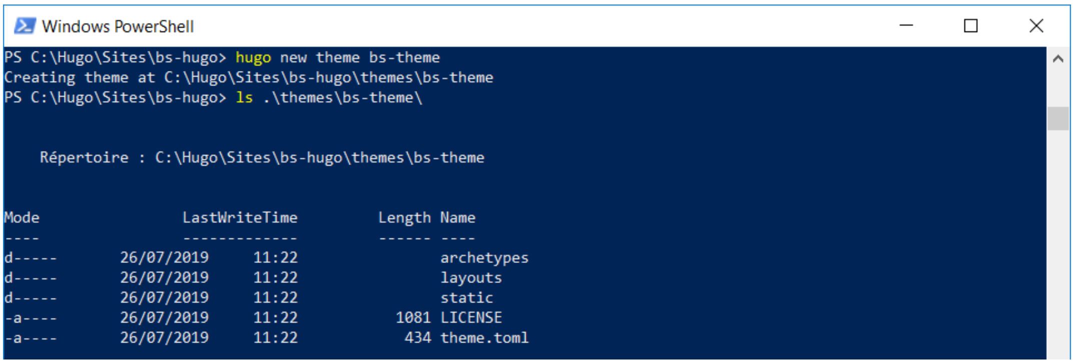
STEP 2 : CREATING THE THEME

"Hugo" uses a theme system. The latter consists of defining *"templates"* for specific categories, which can then be used in any site developed with the help of *"Hugo"* and which includes your theme.
As with creating your site, the `hugo` executable allows you to simply create your theme using the following command :

Terminal

```
~/T-WEB-501> hugo new theme NOM_DE_VOTRE_THEME
```

Once the command is executed, you should get an architecture close to this one :



```
PS C:\Hugo\Sites\bs-hugo> hugo new theme bs-theme
Creating theme at C:\Hugo\Sites\bs-hugo\themes\bs-theme
PS C:\Hugo\Sites\bs-hugo> ls .\themes\bs-theme\


    Répertoire : C:\Hugo\Sites\bs-hugo\themes\bs-theme


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       26/07/2019     11:22                archetypes
d-----       26/07/2019     11:22                layouts
d-----       26/07/2019     11:22                static
-a----       26/07/2019     11:22           1081 LICENSE
-a----       26/07/2019     11:22            434 theme.toml
```

We find three folders : `'archetypes'`, `'layouts'` and `' static'`, which have the functions described above.
In addition, `hugo` also generates two other files :

- `'LICENSE'` : Contains the license for your theme (MIT license)

- `'theme.toml'` : Contains information about your theme

Finally, you have one more thing to do in order to start using your theme. Indeed, it is necessary to indicate to *“Hugo”* what theme to use, and in what file it can find it.
To do this, modify the `'config.toml'` file to the root of your project and add these lines :

```
theme = "NOM_DE_VOTRE_THEME"
themesDir = "./themes"
```

Now that you have a theme, *“Hugo”* will automatically load it if you have not overloaded it.
Here is an example :
You have the file *“content”* following `'/content/posts/first.md'`. When compiling, *“Hugo”* will look for the files *“archetypes”* in the following order :

1. `'/archetypes/posts.md'` : You can “overload” your theme for a specific category (here, “posts”)
2. `'/archetypes/default.md'` : You can “overload” your theme for all categories
3. `'/themes/bs-theme/archetypes/posts.md'` : You can use the theme for a specific category (here, “posts”)
4. `'/themes/bs-theme/archetypes/default.md'` : You can use the theme for all categories

This file loading order is valid for **any** architecture *“Hugo”*.

You should keep in mind that a theme is generic : it does not have any “hard” content, so it’s better to “overload” the theme according to your own requirements than to edit the theme files directly.

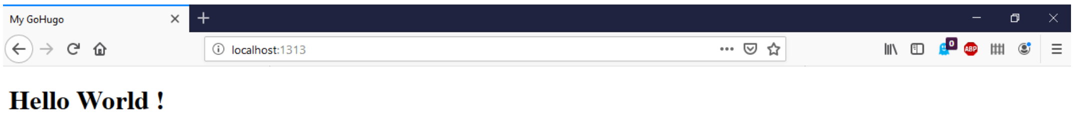
Also modify the file `'/themes/YOUR_THEME_NAME/layouts/index.html'` and add the following html code :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>My GoHugo</title>
</head>
<body>
    <h1>Hello World !</h1>
</body>
</html>
```

Finally launch your server using the following command (you must be at the root of your project) :

```
Terminal
~/T-WEB-501> hugo server -D
```

Then visit the address `http://localhost:1313/` ; you should see this :



Hello World !

Learn about the static files in Hugo to add CSS to your page.

Step 3 : Create an article

+ Part 1 : Create an "archetype"

To begin, we will create a `archetype` in our theme. This' archetype`will be used to initialize the _"Front Matter"_ of your posts. Create the file`'/themes/NOM_DE_VOTRE_THEME/archetypes/posts.md" which will serve as a reference for all your created posts.

Take back the file `'/archetypes/default.md'`.
However, replace the "draft" field with the "Description" field. It will be initialized to an empty string.

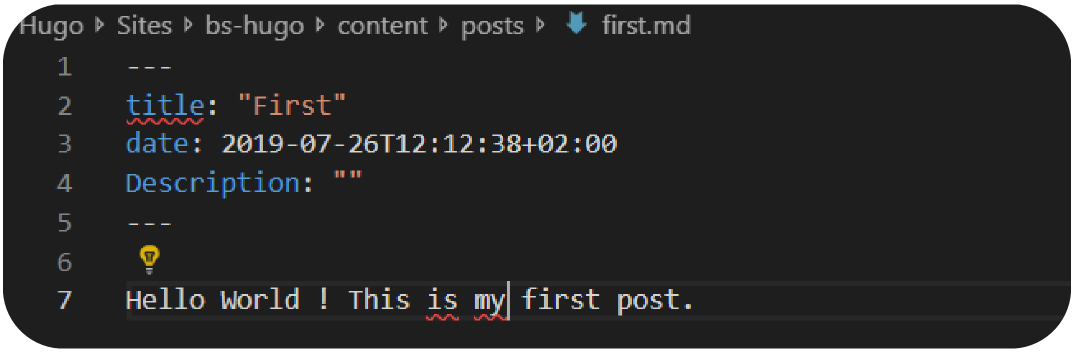
+ Part 2 : Create the content

Now that you have your `archetype`, you will be able to create your first "post".
To do this, use the following command at the root of your project :

```
Terminal
~/T-WEB-501> hugo new posts/TITRE_DE_VOTRE_POST.md
```

This command will create a new *"post"* of the name "TITRE_DE_VOTRE_POST". Since you have defined your `archetype`, its *"Front Matter"* will be initialized with it.

Add content to your post now by editing the `'/content/posts/TITRE_DE_VOTRE_POST.md'` file :



```
---
title: "First"
date: 2019-07-26T12:12:38+02:00
Description: ""
---

Hello World ! This is my first post.
```

Create two more “posts” using the same command, and add them to the content.

STEP 4 : VIEW ARTICLES

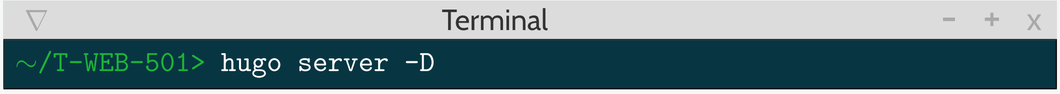
“Hugo” uses the template engine of “Go”. This engine scans templates to find commands that are wrapped between ‘{{’ and ’}}’.
In order to display your articles, you will use the commands :

- `'range'` : allows you to loop (while)
- `'first X'` : allows to display only the first X elements
- `'.Site.RegularPages'` : This variable refers to all the’ content ’of your site
- `'if eq'` : Allows to perform an equal condition (if XX == XX)
- `'.Type'` : Contains the type of the page
- `'.Title'` : Contains the value of the ‘Title’ attribute of the *“Front Matter”*
- `'.Permalink'` : Contains a single route to display the details of an item
- `'end'` : Delimit the end of a loop

So add the following HTML code in your `'/themes/NOM_DE_VOTRE_THEME/layouts/index.html'` file :

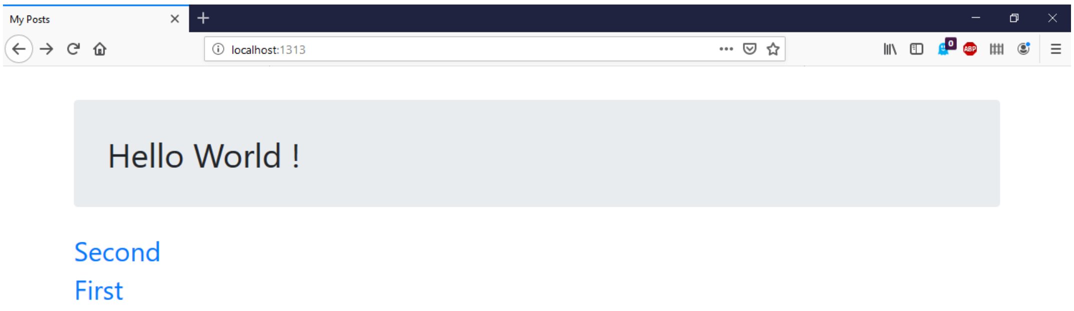
```
<body>
    <h1>Hello World !</h1>
    <!-- Effectue une boucle sur TOUTES les pages du site -->
    {{ range first 10 .Site.RegularPages }}
        <!-- SI la page est un post -->
        {{ if eq .Type "posts" }}
            <h2>
                <a href="{{ .Permalink }}">{{ .Title }}</a>
            </h2>
        {{ end }}
        <!-- Fin if -->
    {{ end }}
    <!-- Fin range -->
</body>
```

“Hugo” uses an automatic “compile & reload” system, so you do not need to restart your server in order to see your actual changes, but if a problem persists despite your changes, you can :
- Stop the server with the shortcut ‘Ctrl + c’ (in your terminal)
- Start it with :



```
~/T-WEB-501> hugo server -D
```

Once your server is restarted, you can refresh your browser to see the changes.
You should get something similar to this (with a little CSS of course) :



Hello World !

Second

First

Nevertheless, if you click on a *"post"*, you will be hit with a "404 page not found" error. You will therefore define the template of your page which will display the details of a *"post"*.
For this, *"Hugo"* has a precise file and folder architecture :

- `'layouts/NOM_DE_DOSSIER'` : The name of the folder must be identical to the name of the folder in `'content'` (in our case, it is `'posts'`)
- `'layouts/posts/single.html'` : Corresponds to the template for displaying the details of a post
- `'layouts/posts/list.html'` : Corresponds to the template for displaying the list of posts

You will therefore create the file for displaying the details of a post.
Like earlier on, the "values" of your article are available by the name of their property (ex: the property *"title"* is available using `'{{.Title}}'`).

The *"content"* of your post is available using `'{{ .Content }}'`.

You can format the display of the date by specifying the graphical representation (in English) that should have the date of "January 2, 2006 at 15h04m05s" :

```
<h3>{{ .Date.Format "02 January 2006 15:04" }}</h3>
```

For a date set on 28 July 2018 at 20h30m30s, the code above will show "28 July 2018 20:30"

Remember that we're building a "theme", so be sure to edit the `'/themes/layouts'` et non `'/layouts'`

Step 5 : Using the "partials"

Take as an example two of your files :

- `'/themes/NOM_DE_VOTRE_THEME/layouts/index.html'` : Homepage of your site (shows all posts)
- `'/themes/NOM_DE_VOTRE_THEME/layouts/posts/single.html'` : Post Details Page

The current problem is that you have a duplicate of your html code.
Indeed, opening and closing the `<html>`, `<head>` and `<body>` tags can be centralized in specific files.
For this we will use the [partials]
(https://gohugo.io/templates/partials/ "Partials - Documentation") of *"Hugo"*.

Here is the list of *"partial files"* to modify (in the folder `'/themes/NOM_DE_VOTRE_THEME'`) :

- `'layouts/partials/head.html'` : Contains the *"head"* of your site (opening of tags, CSS inclusions, ...)
- `'layouts/partials/header.html'` : Contains the *"header"* of your site (navbar, ...)
- `'layouts/partials/footer.html'` : Contains the *"footer"* of your site (copyright, closing tags, ...)

Once the *"fichiers partiels"* have been created, it is necessary to specify to the other pages of your site to use these files.
For this, we will modify the two current templates of your site (always in the `'/themes/NOM_DE_VOTRE_THEME'` folder) :

- `'layouts/index.html'`
- `'layouts/posts/single.html'`

With the following code :

```
<!-- Inclusion de la "head" (ouverture du tag body, ...) -->
{{ partial "head.html" . }}

<!-- Inclusion du "header" (navbar, ...) -->
{{ partial "header.html" . }}

<!-- Le "Contenu" de votre page -->

<!-- Inclusion du "footer" (fermeture du tag body, ...) -->
{{ partial "footer.html" . }}
```

+ Conclusion

In this Bootstrap, you have learned to :

- Create a static site with *"Hugo"*
- Create and use a theme with *"Hugo"*
- Use the `hugo` executable provided by *"Hugo"*
- Create "layouts" (templates) for your displays
- Use "partials" to avoid code duplication

With these new skills, you should have all the tools to create your own static site and start the project associated with this bootstrap.