TP n°2 *Héritage de templates, Bootstrap*

Exercice 1. Héritage de templates

A l'intérieur d'un même site, il est préférable que toutes les pages utilisent la même charte de présentation. Pour ce faire, on utilise un template de base qui contient le squelette de la charte de présentation, puis on spécialise ce template pour chaque type de pages du site.

Considérons notre template `home.html` :

```
<! doctype html>
<html>
  <head>
    < title>{{ title }}</ title>
    <link rel=" stylesheet"
          href="{{" url_for('static ',"  filename='home.css ')"}}"/>
  </ head>
  <body>
    <h1 >{{ title }}</ h1 >
    <ul>
      {% for x in names  %}
      <li>{{ x }}</ li>
      {% endfor %}
    </ ul>
  </ body>
</ html>
```

Nous allons le scinder en 2 parties : `base.html` contenant le template de base, et `home.html` qui spécialise ce template de base pour notre page home.

`base.html`

```
<! doctype html>
<html>
  <head>
    < title>{{ title }}</ title>
    <link rel=" stylesheet"
           href="{{"url_for('static ',"  filename='home.css ')"}}"/>
  </ head>
  <body>
    <h1 >{{ title }}</ h1 >
{% block main %}{% endblock  %}
  </ body>
</ html>
```

home.html

```
{% extends " base. html" %}
{% block main  %}
    <ul>
      {% for x in names  %}
      <li>{{ x }}</ li>
      {% endfor %}
    </ ul>
{% endblock %}
```

Vérifiez que ça marche comme avant ; ajoutez et commitez !

Exercice 2. Utilisation des données scrapées sur Amazon

Nous supposerons que le fichier `data.yml` est placé dans le sous-répertoire `myapp` et que les images sont dans `myapp/static/images`. Les données alimentent les *modèles* utilisés par notre app que nous plaçons conventionnellement dans le fichier `myapp/models.py` :

models.py

```
import yaml, os.path

Books = yaml.safe_load(
    open(
        os.path.join(
            os.path.dirname(__file__),
            "data.yml"
            )
        )
    )
# Pour avoir un id
i = 0
for book in Books:
    book['id'] = i
    i += 1

def get_sample():
    return Books[0:10]
```

views.py

Modifions notre vue en conséquence :

```
from .app import app
from flask import render_template
from .models import get_sample


@app.route("/")
def home():
    return render_template(
        "home.html",
        title="My Books !",
        books=get_sample()
```

puis notre template :

templates/home.html

```
{% extends "base.html" %}
{% block main %}
        <div class="sample">
            {% for x in books %}
            <img src="{{ url_for('static', filename='images/'+x.img)
}}" alt="{{x.title}}" title="{{x.title}}">
            {% endfor %}
        </div>
{% endblock %}
```

et enfin le css pour faire moins moche :

static/home.css

```
h1      {color: red;}

div.sample{
    text-align: center;
    margin-left: 25%;
    margin-right: 25%;
}
```

Exercice 3. Utilisation de Bootstrap

Bootstrap (`http://getbootstrap.com/`) est un Framework HTML+CSS+JS pour créer des sites web "responsive." Il vous appartient de vous former vous-même à l'usage de Bootstrap. Cet exercice vous montre simplement comment installer Bootstrap et adapter le template de base à son usage.

Installation du plugin `bootstrap-flask` :

```
( venv) $ pip install bootstrap-flask
```

Par défaut, le plugin sert les fichiers de Bootstrap en renvoyant aux sites officiels. Le plugin contient aussi des copies de ces fichiers. Si on veut qu'il nous serve ces copies locales (par exemple si on n'a pas accès au net), il faut rajouter la config suivante dans `app.py` :

```
app. config['BOOTSTRAP_SERVE_LOCAL'] = True
```

Activation du plugin : dans le fichier `app.py` nous avons au total :

```
from flask import Flask
from flask_bootstrap import Bootstrap5
app = Flask(__name__)
app. config[ 'BOOTSTRAP_SERVE_LOCAL'] = True
bootstrap = Bootstrap5(app)
```

Modifions maintenant le template de base.
Pour cela nous pouvons nous appuyer sur le code des exemples proposés sur http://getbootstrap.com/getting-started/.
Seuls 2 fragments nous intéressent : (1) la barre de navigation, (2) le div du contenu principal.
On peut faire du copier/coller du code correspondant dans les blocks appropriés en partant du template proposé sur le site : https://getbootstrap.com/docs/5.2/examples/sticky-footer-navbar/

templates/base.html

```
<!doctype html>
<html lang="fr">
    <head>
        {% block head %}
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport"
              content="width=device-width, initial-scale=1, shrink-to-fit=no">

        {% block styles %}
            <!-- Bootstrap CSS -->
            {{ bootstrap.load_css() }}
        {% endblock %}

        <title>Ma bibliothèque SF</title>
        {% endblock %}
    </head>

    <body>
        <header>
            <!-- Fixed navbar -->
```

```
            <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
              <div class="container-fluid">
                <a class="navbar-brand" href="#">My Books</a>
                 <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarCollapse"
aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle
navigation">
                    <span class="navbar-toggler-icon"></span>
                  </button>
                <div class="collapse navbar-collapse" id="navbarCollapse">
                  <ul class="navbar-nav me-auto mb-2 mb-md-0">
                    <li class="nav-item">
                      <a class="nav-link active" aria-current="page"
href="#">Home</a>
                    </li>
                    <li class="nav-item">
                      <a class="nav-link" href="#">Link</a>
                    </li>
                    <li class="nav-item">
                      <a class="nav-link disabled">Disabled</a>
                    </li>
                  </ul>
                  <form class="d-flex" role="search">
                    <input class="form-control me-2" type="search"
placeholder="Search" aria-label="Search">
                    <button class="btn btn-outline-success"
type="submit">Search</button>
                  </form>
                </div>
              </div>
            </nav>
          </header>

        <!-- Your page content -->
        <main class="flex-shrink-0">
            <div class="container">

            {% block content %}

            {% endblock %}
            </div>
        </main>

        <footer class="footer mt-auto py-3 bg-light">
            <div class="container">
              <span class="text-muted">Place sticky footer content here.</span>
            </div>
          </footer>

        {% block scripts %}
            <!-- Optional JavaScript -->
            {{ bootstrap.load_js() }}
        {% endblock %}
```

```
    </body>
</html>
```

Finalement, on peut aussi modifier le template `home.html` pour y préciser le h1 :

templates/home.html

```
{% extends 'baseBS.html' %}
{% block styles %}
{{ super() }}
<link rel="stylesheet" href="{{ url_for('static',
filename='home.css')}}">
{% endblock %}
{% block content %}
<h1>{{title}}</h1>
<div class="sample">
    {% for x in books %}
    <a href="{{url_for('detail', id=x.id)}}">
    <img src="{{ url_for('static', filename='images/'+x.img) }}"
alt="{{x.title}}" title="{{x.title}}">
    </a>
    {% endfor %}
</div>
{% endblock %}
```

Vérifions si ça marche : . . . en fait le h1 est caché derrière la barre de navigation. Il faut régler le décalage vertical dans le CSS pour l'élément `<main>` et la position du `<footer>`.

Dans `static/home.css` nous mettons par exemple :

```
h1 {color: red;}

div.sample{
    text-align: center;
    margin-left: 25%;
    margin-right: 25%;
}

main{
    margin-top: 8em;
}

footer{
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

```
main > .container {
    padding: 60px 15px 0;
}
```

Vérifiez que tout fonctionne ; commitez !
Bien entendu l'exemple de barre de navigation n'est pas adapté à notre application. Il faudra la modifier pour faire apparaitre les liens et fonctionnalités pertinentes.

Ajoutez maintenant une route de détail dans views.py :

```
@app.route("/detail/<id>")
def detail(id):
    books = get_sample()
    book = books[int(id)]
    return render_template(
        "detail.html",
         book=book)
```

A vous de jouer pour le template detail.html !