BABEL FORMATION PYTHON DJANGO WEEK 4 – LAST 2DAYS

par Emmanuel Sandorfi – e.sandorfi@parleweb.com édition 27/01/2020

LIVRABLE BABEL - TÂCHES

Exercice 1 / Page d'accueil

Exercice 2 / Lecture de fichiers markdown dans la page A propos

Exercice 3 / Rendre la page active dans la barre de navigation

Exercice 4 / Rendre mobile la navigation dewey

Exercice 5 / Rendre cohérente la liste des publications

Exercice 6 / Rendre cohérente la navigation detail / update

Exercice 7 / Donner un aspect reconnu à la page détail et au formulaire par d'update.

Exercice 1 / Page d'accueil

Faire de la home page un ***splash screen*** **avec un CTA**, Call To Action.
Soit une page d'accueil attirante et cohérente pour l'utilisateur afin qu'il clique sur le bouton d'accès au catalogue.

- Personnalisation du composant jumbotron bootstrap
  - Chercher une image sur google grand format et la sauvegarder dans le **static** catalog bg-home.jpg
  - Dans les **templates**
    - Dans le template index.html charger jumbotron-v2.html au lieu de jumbotron.html (compatibilité reste du site)
    - Dupliquer jumbotron.html en jumbotron-v2.html
    - rajouter un bouton au jumbotron pour accéder au catalogue en utilisant des variables à définir dans la **view** home (cta_url et cta_text)
  - Dans la **View** home
    - Importer **reverse** pour récupérer l'url du catalogue
    - passer les variables et une classe **css** .home-jumbotron
  - Dans **catalog\style.css**
    - Définir .home-jumbotron
      - background-image : url('/static/catalog/bg-home.jpg') ;
      - background-color
      - background-repeat
      - background-position
      - background-size
      - pour centrer :
        - display: flex ;
        - justify-content : center ;
        - align-self :center ;
      - pour prendre la hauteur de l'écran :
        - min-height : 90vh ;
    - Définir .home-jumbotron .container
      - background-color
      - color
      - padding
      - border-radius
    - Liens mdn
      - https://developer.mozilla.org/fr/docs/Web/CSS/background
- Personnalisation du **<footer>** via le style css
  - Mise en cohérence avec les couleurs du background
  - Liens à mettre en cohérence
    - footer a { …. }
    - footer a :hover { …. }

Exercice 2 / Lecture de fichiers markdown dans la page A propos

Faire une fonction permettant de lire un fichier markdown du git pour l'afficher dans une page (View et Template).

- Pipenv install markdown
- Dans utils.py
  - Import markdown
  - Fonction read_from_markdown
- Dans views.py
  - Import de la fonction
  - Dans la View about
    - Lire le fichier readme.md dans la variable content1
    - Lire le fichier changelog.md dans la variable content2
    - Mettre les variables dans le context local,
      - content1
      - content1title = Notre readme sur le projet Babel
      - content2
      - content2title = Notre suivi de formation…
- Dans le template about.html
  - Intégrer le composant collapse de bootstrap
  - Ouvrir un container avec les classes mt-5 mb-5 et copier coller l'exemple dans la div : https://getbootstrap.com/docs/4.0/components/collapse/#accordion-example
  - Insérer **local.content1title** et **local.content1** dans le premier accordéon
  - **local.content2title** et **local.content2** dans le deuxième
  - enlever les autres

```
def read_from_markdown(file_md):
    """
    lecture d'un fichier au format markdown à partir du root du projet
    retourne le contenu au format html ou une indication d'erreur en texte
    """
    datafile = settings.BASE_DIR + "/" + file_md
    try:
        with open(datafile, "r", encoding="utf-8") as f:
            data = f.read()
            content = markdown.markdown(
                data, output_format="html5", extensions=["fenced_code"]
            )
    except Exception as e:
        content = f"Cannot read {datafile} ! Error is {str(e)}"
    return content
```

Exercice 3 / Rendre la page active dans la barre de navigation

L'utilisateur doit savoir se repérer dans l'application. Il est nécessaire de lui indiquer où il est dans la barre de navigation.

- Dans le template navbar.html
  - Passer la navbar en plus foncé afin de faire ressortir la page active
    - Changer la classe de navbar, voir : https://getbootstrap.com/docs/4.0/components/navbar/#color-schemes
    - Tester pour chaque nav-item si local.active est égale à home, publication, newsroom ou about et mettre la classe active si oui. {%if local.active == '…' %} ….
- Dans views.py et viewscat.py
  - Ajouter une variable **local.active** dans toutes les Views avec un mot clé pour chacune des pages principales : home, publication, newsroom et about

Exercice 4 / Rendre mobile la navigation dewey

Le jumbotron standard prend trop de place. En intégrant un container tout en conservant la structure du jumbotron, nous pouvons ajouter un composant navbar et un composant collapse à l'intérieur pour rester à l'identique en desktop et afficher un bouton collapse en mobile.

https://getbootstrap.com/docs/4.0/components/navbar/#external-content

- Dans le template publication
  - Intégration d'une navbar dans le jumbotron : nouveau composant dans le block jumbotron
  - dans la navbar,
    - ajouter un bouton retour au catalog mais caché si l'url est /catalog/. Test de {{ request.path }}
    - utiliser un bouton toggle pour un contenu externe pour Filtrer par Dewey
    - ne pas afficher le bouton si desktop, afficher le texte Filtrer par Dewey
    - afficher le nombre de publication
  - vérifier la grille du container ? container-fluid ?

```
{% block content %}
<div class="container-fluid">
  <div class="row">
    <div class="col-sm-12 col-md-4 col-xl-3 left-sidebar">
      <div class="collapse dont-collapse-sm" id="navbarToggleExternalContent">
        {% include 'catalog/dewey-list-v2.html' %}
      </div>
    </div>
    <div class="col-sm-12 col-md-8 col-xl-9 pt-3 dont-padding-sm">
      {% include 'catalog/publication-list-v2.html' %}
    </div>
  </div>
</div>
{% endblock %}
```

Exercice 5 / Rendre cohérente la liste des publications

Un tableau en mobile doit avoir le moins de colonne possible. Il faut donc simplifier la lecture des publications tout en conservant la référence du livre. La date de parution est également une information importante dans le choix d'un ouvrage.

Utiliser 4 colonnes dans la liste : Publication, Parution, Reference, Dewey

Pour une cellule de table clickable avec jquery et data attributes :

```
<td class="width-10 dewey{{publication_record.dewey_number.number}}
clickable-td" data-href="{% url 'publication-
dewey' publication_record.dewey_number.number %}"></td>
```

```
{% block userscript %}
<script>
jQuery(document).ready(function($) {
    $(".clickable-td").click(function() {
        window.location = $(this).data("href");
    });
})
</script>
{% endblock %}
```

Dans la view home, trier les publications par date_publication et ajouter le compte des publications.

Exercice 6 / Rendre cohérente la navigation detail / update

Le cheminement de l'utilisateur doit rester simple entre la consultation d'une page et la modification possible. Une barre de navigation commune à intégrer.

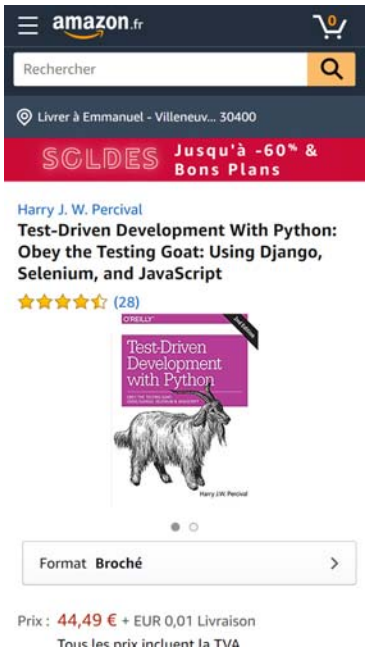
```
    {% block contentactions %}
    <nav class="navbar dewey{{object.dewey_number.number}}">
        <div>
        <a href="{% url 'publication-
dewey' object.dewey_number.number %}" class="btn btn-sm btn-outline-
dark dewey{{object.dewey_number.number}}">< {{object.dewey_number}}</a>
        {%if user.is_authenticated %}
            <a href="{% url 'publication-update-
pk' object.pk %}" class="btn btn-sm btn-danger">Editer</a>
        {% endif %}
        </div>
    </nav>
    {% endblock %}
```

Exercice 7 / Donner un aspect reconnu à la page détail et au formulaire par d'update.

La premier écran comme le dernier sont les plus important, le début et la fin de la démonstration.

Construire un template déjà vu comme celui d'Amazon par exemple..
Utiliser django crispy forms pour les formulaires.

Côté mobile



Côté desktop