TRAVAUX DIRIGÉS
DJANGO : INSTALLATION ET PREMIÈRES PAGES

I. INTRODUCTION

le but de ce TD est de procéder à l'installation de Django dans votre environnement et à créer vos premières pages web dynamiques

II. INSTALLATION DANS PYCHARM

Nous allons procéder dans un premier temps à l'installation de Django dans votre nouveau projet. Créer un nouveau projet avec un environnement virtuel comme indiqué sur la figure 1

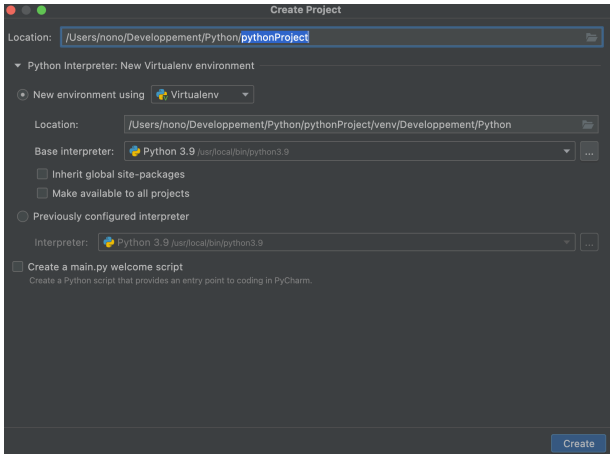

FIGURE 1 – fenêtre du nouveau projet. la partie environnement virtuel doit être renseignée.

L'intérêt de l'environnement virtuel c'est que vous n'avez pas à tenir compte des droits d'installation de logiciel sur la plateforme, et vous pouvez gèrer plusieurs versions d'un soft au travers de plusieurs projet. Maintenant, dans l'onglet terminal du projet, vous allez taper la commande d'installation de Django

Console pyCharm 1: installation de Django

```
1  pip install django
```

cette commande installe la librairie Django dans votre projet. Pour démarrer Django, il faut taper la commande

Console pyCharm 2: création du projet Django

```
1  django-admin startproject firstproject
```

ce qui vient créer un nouveau répertoire *firstproject* (figure 2) qui contiendra l'administration de Django et les différentes applications que vous allez créer.

Le fichier *__init.py__* indique que le répertoire est un package python. Le fichier *url.py* va indiquer comment interpréter les url de votre site web et à quels contrôleurs elles vont être adressées. le fichier *setting.py* contient les paramètres de configuration de votre projet, entre autre la déclaration des applications, la base de données utilisée, la langue par défaut, ... Les deux fichiers *wsgi.py* et. *asgi.py*

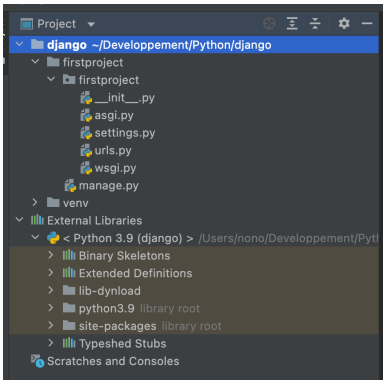

FIGURE 2 – Arborescence du projet.

correspond aux modes de déploiement d'application en fonction du type de serveur web. Enfin, *manage.py* correspond au code python qui va nous permettre de démarrer une instance, créer une nouvelle application,...

afin de vérifier que l'installation est correcte, nous allons démarrer le serveur web intégré à Django (serveur qui ne doit absolument pas être utilisé dans le cadre de la production, mais uniquement du développement.

Console pyCharm 3: création du projet Django

```
cd "nom de votre projet sans les guillemets" #répertoire contenant manage.py
python manage.py  runserver
```

Cette commande démarre le serveur en localhost sur le port 8000. Vous pouvez indiquer à la suite de runserver, soit un autre numéro de port que celui par défaut, voir une adresse IP et un numéro de port au format

```
python manage.py runserver IP:port
```

III. CRÉATION D'UNE PREMIÈRE APPLICATION

Pour créer une application, vous aller devoir taper la commande

Console pyCharm 4: création du projet Django

```
python manage.py  startapp myfirstapp
```

ou *myfirstapp* est le nom de votre application (indépendant du nom du projet, un projet pouvant contenir plusieurs applications partageant du code commun. Cette commande a créer un nouveau sous dossier nommé *myfirstapp* contenant l'arborescence de votre application comme illustré sur la figure 3

Pour rappel, nous sommes dans un environnement permettant de créer une application sur le modèle MVC. Le fichier *models.py* contiendra les différentes classe de modèles pour les données. Le fichier *views.py* contiendra les contrôleurs de votre application (*Remarques : Django a eu la merveilleuse idée de nommer* ***view*** *ou vues en français les contrôleurs, et* ***templates*** *ou gabarit en français les vues.*) Nous verrons plus tard comment ajouter des gabarits (vues) à notre application pour la gestion du rendu par votre navigateur. Ensuite, il faut ajouter l'application à la liste des applications dans le fichier *settings.py* de la partie administration

Code Python 1: ajouter dans le code de Settings.py

```
INSTALLED_APPS = [
    'myfirstapp.apps.MyfirstappConfig', # ajouter cette ligne, MyfirstappConfig est le
    nom de la classe dans le fichier app.py de l'application
    'django.contrib.admin',
    'django.contrib.auth',
```

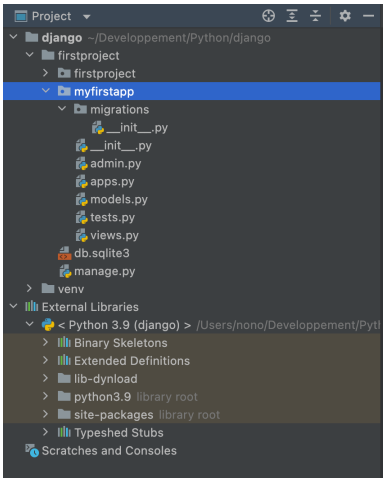

FIGURE 3 – Arborescence du projet avec la nouvelle application myfirstapp

cette étape n'est à faire qu'une seule fois pour ajouter l'application à la liste des applications.
Afin d'ajouter une page web statique à notre site, nous allons ensuite effectuer les étapes suivantes :

(1) Créer le répertoire *templates* dans le répertoire de votre application puis dans celui ci, créer un répertoire *myfirstapp* qui contiendra les gabarits de votre application. *remarques : par défaut les gabarit sont cherchés dans tous les répertoires templates de votre projets, donc dans toutes les applications. Rajouter le nom de l'application dans le répertoire permet de lever l'ambiguïté sur un nom de template qui pourrait exister dans plusieurs applications différentes.*

(2) Créer le fichier index.html dans le répertoire *templates/myfirstapp*. Ce fichier contiendra uniquement le code

Code html 1: page web index.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>index</title>
</head>
<body>
    <h1> Hello World </h1>
</body>
</html>
```

(3) ajouter un controleur dans le fichier *views.py*

Code Python 2: ajouter dans le code de views.py

```
def index(request):
    return render(request, 'myfirstapp/index.html')
```

toutes les fonctions de votre contrôleur doivent renvoyer un élément de type HttpResponse. La fonction **render()** qui est importé au début de votre fichier, permet de générer cette réponse à l'aide du gabarit désigné.

(4) Créer un fichier *urls.py* dans le répertoire de l'application et y ajouter une entrée pour le chemin index

Code Python 3: ajouter dans le code de myfirstapp/urls.py

```python
from django.urls import path

from . import views

urlpatterns = [
    path('index/', views.index),
]
```

(5) Modifier le fichier *urls.py* du projet pour y ajouter l'accés à *urls.py* de l'application

Code Python 4: ajouter dans le code de monprojet/urls.py

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myfirstapp/', include('myfirstapp.urls')),
]
```

il faut ajouter include dans les import puis la dernière ligne de la liste.

(6) enfin relancer le serveur *(*python manage.py runserver)

Il ne reste alors plus qu'à ouvrir votre navigateur et taper l'url

`http://127.0.0.1:8000/myfirstapp/index/`

IV. Les fichiers statiques (CSS, images, ...)

Pour accéder à des fichiers statiques dans la conception de vos pages web il faut les mettre dans le répertoire *static/myfirstapp/* de votre application. c'est le répertoire qui est utilisé par défaut et qui correspond à la valeur de configuration STATIC_URL dans le fichier *settings.py*. Vous devez ensuite modifier le code du fichier html pour ajouter le lien comme suit

Code html 2: page web index.html

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    {% load static %}
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="{%static 'myfirstapp/index.css' %}">
    <title>index</title>
</head>
<body>
    <h1> Hello World </h1>
</body>
</html>
```

V. Récupérer un champ de formulaire

Nous allons créer un formulaire simple avec un champ de saisie texte sous la forme du fichier html suivant

Code html 3: page web formulaire.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>formulaire</title>
</head>
<body>
    <form action="/myfirstapp/bonjour" method="get">
    {% csrf_token %}
        <label for="nom"> Nom :</label>
        <input type="text" name="nom"/>
        <input type="submit" value="envoyer"/>
    </form>
</body>
</html>
```

Pensez à ajouter un contrôleur dans *views.py* et une entrée dans *urls.py*. Ensuite, vous allez créer un autre contrôleur qui sera lié à la page de traitement des données du formulaire. Ce contrôleur va récupérer les données du formulaire et les transmettre à la vue d'affichage. La récupération des données se fait au travers de la requête (**request**)

Code Python 5: controleur de récupération des données de formulaire

```
def bonjour(request):
    nom=request.GET["nom"] # récupère la valeur du paramètre nom du formulaire
    return render(request,'myfirstapp/bonjour.html',{"nom":nom}) # passe cette valeur à
    la vue au travers du dictionnaire de contexte
```

Enfin, il faut créer le vue qui utilise le dictionnaire de contexte pour afficher les données.

Code html 4: code à insérer dans page web d'affichage des données du formulaire

```
<body>
{% if nom is not None %}
    <h1> Hello {{nom}} </h1>
{% else  %}
     <h1> Hello world </h1>
{% endif %}
</body>
```

Vous pouvez voir ainsi qu'on utilise la variable de contexte identifiée par son nom entre 2 paires d'accolade, ici *{{ nom }}* . Le code python est situé entre des balises {% %}, ici un test pour voir si une valeur a été saisie dans le champs de formulaire. Le routage des fichiers créés est aussi nécéssaire.

*Exercice :*améliorer le formulaire pour saisir aussi le prénom et l'âge de la personne et qui modifie l'affichage avec ces nouvelles données. Attention, les données doivent être transmises au template au travers d'un unique dictionnaire python correspondant au contexte.

VI. code html pour un ensemble de pages communes.

nous allons à présent voir comment créer un cadre de page réutilisable pour toute les pages (bandeau de titre, pied de pages, ...). En Django, on peut hériter d'un gabarit, c'est à dire indiquer au début de notre fichier html qu'il faut utiliser un patron de site dont le code est indiqué dans une balise django, puis indiquer pour chaque block à remplir le contenu qui doit y arriver. Voyons cela sur notre page de formulaire.

Nous allons construire le gabarit squelette suivant

Code html 5: gabarit squelette main.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
{% load static %}
    <link rel="stylesheet" href="{%static 'myfirstapp/index.css' %}">
    <title>{% block title %}Mon Site {% endblock %} </title>
</head>

<body>
    <aside id="sidebar">
        {% block sidebar %}
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/myfirstapp/index">Index</a></li>
        </ul>
        {% endblock %}
    </aside>

    <div id="content">
        {% block content %}
          <h1> bloc principal </h1>
        {% endblock %}
    </div>
</body>
</html>
```

ce cadre sera importé dans les autres gabarits qui définiront les contenus des blocs identifié par leur id (content par exemple) et les balises (*{% block content%}* et *{% endblock %}*). Le gabarit squelette doit être placé dans le répertoire templates de l'application pour la suite de l'exemple On peut voir ici un exemple de gabarit héritant du squelette défini précédemment

Code html 6: gabarit héritant de main.html

```
<!-- héritage du contenu du gabarit squelette main.hml -->
{% extends "main.html" %} <!-- si le gabarit n'est pas dans templates mais dans le
    répertoire templates/myfirstapp/, il faudra ajouter le chemin d'accès avant le nom
    du fichier donc extends "myfirstapp/main.html" -->

<!-- définition de ce que contiendra la balise title du head -->
{% block title %}
  Mon formulaire
{% endblock %}
{% block content %}
  <!-- les champs de formulaires -->
   <form action="/myfirstapp/bonjour" method="get" />
    {% csrf_token %}
        <label for="nom"> Nom :</label>
        <input type="text" name="nom"/>
        <input type="submit" value="envoyer"/>
    </form>

{% endblock %}
```

Le contenu des box n'est pas forcément à remplir dans un gabarit. s'il n'apparait pas, on utilise le contenu défini par défaut dans le squelette. Si on souhaite enrichir le contenu par défaut sans le perdre,

il faut utiliser le code *{{block.super}}* dans le bloc en question. ce code sera remplacé par le contenu par défaut du block et enrichi de ce qui sera rajouté.

Code html 7: gabarit héritant et surchargeant le squelette main.html

```
1
2 {% extends "mysfirstapp/main.html" %}
3
4 {% block title %} Mon formulaire {% endblock %}
5 {% block content %}
6   <!-- les champs de formulaires -->
7   {{block.super}} <!-- récupére les données par défaut du bloc content du squelette -->
8    <form action="/myfirstapp/bonjour" method="get" />
9     {% csrf_token %}
10    ...        <input type="submit" value="envoyer"
11    <\form>
12
13 {% endblock %}
```

Exercice : Créer un squelette de page présentant un bandeau de titre, un menu avec les liens le formulaire et la page d'accueil et utiliser le pour le formulaire et la page d'affichage des résultats.