PYTHON PAR LA PRATIQUE

Présenté par :

Xavier TABUTEAU

17. Django

Django

C’est un framework web python permettant de créer un site web ou une API web.

Django est ordonné de façon MVT (Model View Template) plutôt que MVC (Model View Controller).

Avec Django le view correspond au controller du MVC et les templates correspond au view du MVC.

Aide sur django : https://docs.djangoproject.com/

17. Django

Etapes pour créer une application Django

Voici les étapes permettant de créer une application Django :

- Création du projet
- Création de l’application principale
- Création du modèle
- Migration du modèle
- Activation de l’admin
- Views
- Urls
- Templates
- Pages dynamiques

17. Django

Création d'un projet Django

Créer un environnement virtuel dédié à partir d'une console python dans un dossier :
venv envdjangotest *# ou python -m venv envdjangotest* dans la console Windows

Activation de l'environnement virtuel :
cd envdjangotest\Scripts
Activate

Remarque : le nom de l'environnement virtuel doit s'afficher avec des () en début de ligne.

Installer Django dans l'environnement virtuel et créer un fichier requirements.txt :
cd ..
pip install django
pip freeze > requirements.txt

Création d'un projet wisdompets :
django-admin startproject wisdompets
cd wisdompets

Démarrer le serveur local pour vérifier le fonctionnement en allant sur localhost:8000
python manage.py runserver

Pour arrêter le serveur local, faire *Ctrl+C*

17. Django

Création d’un projet Django

Créer une nouvelle application "adoption" du projet wisdompets :
python manage.py startapp adoption

Editer le fichier settings.py situé dans le sous dossier de l’application principale nommée aussi wisdompets.
Ajouter adoption dans la partie INSTALLED_APP pour donner accès à cette application.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'adoption'
]
```

17. Django

Modèle

Un modèle est une classe qui hérite de django.db.models.Model, et qui définit les champs comme des attributs de classe.

Il y a différent type de champ que Django utilise :

Champ	Exemples
CharField	"Parent", "Elève"
TextField	"Ceci est un texte très très long…."
EmailField	mon.email@exemple.fr
URLField	www.example.com
IntegerField	-1, 245, 0
DecimalField	0.5, 3.14
BooleanField	True, False
DateTimeField	Datetime(1960, 1, 1, 8, 0, 0)
ForeignKey	1 (lien vers un identifiant d’une autre table)
ManyToManyField	models.ManyToManyField(classe)

17. Django

Modèle

Mettre à jour le modèle dans le fichier models.py dans le dossier adoption :

```
from django.db import models

# Create your models here.
class Submitter(models.Model):
    firstname = models.CharField(max_length = 100)
    lastname = models.CharField(max_length = 100)
    is_active = models.BooleanField(default = True)

    def __str__(self):
        return f"{self.firstname} {self.lastname}"
```

17. Django

```
class Pet(models.Model):
    SEX_CHOICES = [('M', 'Male'), ('F', 'Female')]
    name = models.CharField(max_length = 100)
    species = models.CharField(max_length = 30)
    breed = models.CharField(max_length = 30, blank = True)
    description = models.TextField()
    sex = models.CharField(max_length = 1, choices = SEX_CHOICES, blank = True)
    submission_date = models.DateTimeField()
    age = models.IntegerField(null = True)
    submitter = models.ForeignKey(Submitter, on_delete = models.SET_NULL, blank =
True, null = True)
    vaccinations = models.ManyToManyField('Vaccine', blank = True)

    def __str__(self):
        return f"{self.name} {self.species}"


class Vaccine(models.Model):
    name = models.CharField(max_length = 50)
    description = models.CharField(max_length = 500, blank = True)

    def __str__(self):
        return f"{self.name}"
```

17. Django

Migrations

```
python manage.py showmigrations    # indique la liste des migrations en attente d’exécution
                                   (création/modification des tables pour les applications)

python manage.py migrate           # exécute les migrations disponible.

python manage.py makemigrations    # regarde les éléments dans le fichier models.py
                                   # et créer un fichier de migration si nécessaire qu'il
                                   # faudra exécuter
```

Chaque fois qu'on modifie le modèle on doit faire un makemigrations et un migrate pour valider les scripts crée.

Nous pouvons vérifier dans la BDD que cela est effectué.

17. Django

Admin

Copier le code suivant dans le fichier admin.py situé dans le dossier adoption :

```
from django.contrib import admin
from . import models

# Register your models here.
admin.site.register(models.Pet)
admin.site.register(models.Submitter)
admin.site.register(models.Vaccine)
```

Puis dans l'invite de commande
python manage.py runserver

Ensuite verifier en ouvrant la page web localhost:8000/admin qu'une page web s'ouvre, demandant un utilisateur et un mot de passe.

17. Django

Création d’un utilisateur admin (super user)

Dans l’invite de commande exécuter :
python manage.py createsuperuser

Permet de creer l'admin et son mot de passe.
Saisir le nom du user, l'email, le mot de passe et le confirmer.

Ensuite tester en ouvrant la page web localhost:8000/admin et se logger.

Mantenant nous pouvons enregistrer, modifier, supprimer des valeurs dans la bdd via la page admin.

Entrer des infos pour tester.

17. Django

Shell

Cela permet d'exécuter des requêtes dans une console Django.

- Entrer dans le shell Django

python manage.py shell

- Afficher les animaux

```
from adoption.models import Pet
results = Pet.objects.all()
for res in results:
    print(res)
```

Puis appuyer sur entré pour valider et afficher le résultat du code.

- Afficher le premier animal

```
pet = Pet.objects.get(id=1)
print(pet)
```

- Quitter le shell

```
exit()
```

17. Django

View

View sert à définir quelles sont les données qui sont renvoyés.
Editer views.py dans le dossier adoption

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def home(request):
    return HttpResponse('<h1>Menu Accueil</h1>')
```

17. Django

Urls

Editer urls.py dans le dossier principal (au même niveau que settings.py).
Ajouter les lignes suivantes :

from adoption import views

Et

path('', views.home, name = "accueil")

17. Django

Templates

Créer le dossier templates dans le dossier adoption, puis un fichier home.html dans ce dossier.

- Ecrire dans ce fichier la ligne suivante :

<h1>Menu Accueil</h1>

- Aller dans le fichier views.py et modifier le par :

```
from django.shortcuts import render
# Create your views here.

def home(request):
    context = {}
    return render(request, 'home.html', context)
```

17. Django

Pages dynamiques

- Modifier le fichier views.py par :

```
from django.shortcuts import render
from .models import Pet
# Create your views here.
def home(request):
    pets = Pet.objects.all()
    context = {
        "pets": pets,
        "titre": "accueil"
        }
    return render(request, 'home.html', context)
```

17. Django

Pages dynamiques

- Aller le fichier home.html et modifier le par :

<!-- code permettant les pages dynamiques s'appel jinja
et il est utilisé dans d'autre framework. -->
<h1>{{ titre }}</h1>

{% for pet in pets %}
<h1>{{pet.name}}</h1>
<h1>{{pet.species}}</h1>
<h1>{{pet.desription}}</h1>
{% endfor %}

PYTHON INITIATION

Présenté par
Xavier TABUTEAU